ADSP-SC596/ADSP-SC598 SHARC+ Processor

Hardware Reference

Revision 0.3, July 2023

Part Number
82-100149-01

,

ANALOG DEVICES

Notices

Copyright Information

© 2023 Analog Devices, Inc., ALL RIGHTS RESERVED. This document may not be reproduced in any form without prior, express written consent from Analog Devices, Inc.

Printed in the USA.

Disclaimer

Analog Devices, Inc. reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent rights of Analog Devices, Inc.

Trademark and Service Mark Notice

The Analog Devices logo, Blackfin, Blackfin+, CrossCore, EngineerZone, EZ-Board, EZ-KIT Lite, EZ-KIT Mini, EZ-Extender, SHARC, SHARC+, Blackfin+, A^2B, SigmaStudio, SigmaStudio+, and VisualDSP++ are registered trademarks of Analog Devices, Inc.

All other brand and product names are trademarks or service marks of their respective owners.

Contents

Arm Cortex-A55 Subsystem

Cortex-A55 Features 1–2
Functional Description 1–2
Cortex-A55 Block Diagram 1–2
Instruction Fetch Unit (IFU) 1–3
Data Processing Unit (DPU) 1–3
Performance Monitor Unit (PMU) 1–3
Advanced SIMD and Floating-point support 1–3
Cryptographic Extension 1–4
Memory Management Unit (MMU) 1–4
L1 Memory System 1–6
L2 Memory System 1–7
Reliability, Availability, and Serviceability (RAS) 1–7
Generic Interrupt Controller (GIC) 1–7
A55 Core in DynamIQ Shared Unit (DSU) 1–8
System Control 1–8
Generic Timer 1–8
ADSP-SC59x GIC Interrupt List 1–9
A55 Configurations 1–9
DynamIQ Shared Unit Interfaces 1–10
Performance Monitor Unit (PMU) 1–13

Clock Generation Unit (CGU)

CGU Features 2–1
CGU Functional Description 2–1
ADSP-SC59x CGU Register List 2–2
ADSP-SC59x CGU Interrupt List 2–3

ADSP-SC59x CGU Trigger List 2–3
CGU Definitions 2–3
CGU PLL Block Diagram 2–4
CLKOUT Selections 2–5
CGU Operating Modes 2–6
CGU Power-up Sequence 2–7
CGU Event Control 2–7
Oscillator Watchdog 2–8
CGU Programming Model 2–9
Configuring CGU Modes 2–9
Changing the Clock Frequencies 2–10
Changing the PLL Clock Frequency 2–10
Changing the CCLKn or SYSCLK Frequency Without Modifying the PLLCLK Frequency 2–11
Changing the OCLK Frequency 2–11
Selecting SCLK1_X and SCLK0_X Source 2–12
Changing SCLK1_X or SCLK0_X Frequencies 2–12
Aligning All Clocks 2–12
Shutting Off CCLKn from Another Controller 2–13
Valid Clock Multiplier Settings 2–14
PLL Bypass and PLL Disable 2–14
ADSP-SC59x CGU Register Descriptions 2–15
Core Clock Buffer Disable Register 2–16
Core Clock Buffer Status Register 2–17
CLKOUT Select Register 2–18
Control Register 2–20
Clocks Divisor Register 2–22
DIV Register Extension 2–25
Oscillator Watchdog Register 2–26
PLL Control Register 2–28
Revision ID Register 2–30

System Clock Buffer Disable Register 2–31
System Clock Buffer Status Register 2–33
Status Register 2–35
Time Stamp Counter 32 LSB Register 2–38
Time Stamp Counter 32 MSB Register 2–39
Time Stamp Control Register 2–40
Time Stamp Counter Initial 32 LSB Value Register 2–41
Time Stamp Counter Initial MSB Value Register 2–42
ADSP-SC59x MISCREG Register Descriptions 2–42
CANFD Low Power Selection Mode 2–43
Error Register 2–45
PLL2 Control Register 2–48
SH0 Prefetch Range Selection Register 2–50
SH1 Prefetch Range Selection Register 2–51
SPORT Direct Interrupt Enable for SH0 and SH1 2–52

Clock Distribution Unit (CDU)

CDU Features 3–1
CDU Functional Description 3–1
CDU Block Diagram 3–1
CDU Definitions 3–3
ADSP-SC59x CDU Register List 3–3
CDU Clock Configuration Options 3–4
CDU Programming Model 3–4
Changing the PLL and Clock Frequency 3–4
Changing the Clock Frequency 3–5
ADSP-SC59x CDU Register Descriptions 3–6
CDU Configuration 3–7
CLKIN Select 3–8
CDU Revision ID 3–9

CDU Status 3–10

Dynamic Power Management (DPM)

DPM Features 4–1
DPM Functional Description 4–1
ADSP-SC59x DPM Register List 4–1
DPM Definitions 4–2
DPM Operating Modes 4–2
Reset State 4–2
Full-on Mode 4–2
DPM Event Control 4–3
DPM Programming Model 4–3
ADSP-SC59x DPM Register Descriptions 4–4
Control Register 4–6
Peripherals Disable Register 0 4–7
Peripherals Disable Register 1 4–9
Revision ID 4–11
Status Register 4–12

Reset Control Unit (RCU)

RCU Features 5–1
RCU Functional Description 5–2
ADSP-SC59x RCU Register List 5–2
ADSP-SC59x RCU Trigger List 5–2
RCU Definitions 5–3
RCU Architectural Concepts 5–3
RCU Status and Error Signals 5–4
Resetting the Arm Core through Another Core/System Requester 5–4
Resetting a SHARC+ Core Through Another Core 5–5
ADSP-SC59x RCU Register Descriptions 5–6
Boot Code Register 5–8

Core Reset Outputs Control Register 5–11
Core Reset Outputs Status Register 5–12
Control Register 5–13
Message Register 5–15
Message Clear Bits Register 5–18
Message Set Bits Register 5–19
System Interface Disable Register 5–20
System Interface Status Register 5–21
System Reset Request Status Register 5–22
Status Register 5–23
Software Vector Register 0 5–25
Software Vector Register 1 5–26
Software Vector Register 2 5–27
SVECT Lock Register 5–28

System Event Controller (SEC) and Generic Interrupt Controller (GIC)

SEC Features 6–1
SEC Functional Description 6–2
ADSP-SC59x SEC Register List 6–2
ADSP-SC59x SEC Interrupt List 6–3
ADSP-SC59x SEC Trigger List 6–3
Combined SEC and GIC Interrupt List 6–4
SEC Definitions 6–18
SEC System Event Aggregator (SEA) 6–19
SEC Block Diagram 6–19
SEC Fault Interface (SFI) 6–20
SEC Core Interface (SCI) 6–21
SEC Source Interface (SSI) 6–22
SEC Architectural Concepts 6–22
System Interrupt Acknowledge 6–22
System Interrupt Groups 6–23

System Interrupt Flow 6–23
System Interrupt Priorities 6–24
SEC Error 6–24
SEC Programming Model 6–24
Programming Concepts 6–25
Programming Examples 6–25
Fault Management Interface Programming Model 6–25
Configuring a System Source to Interrupt a Core 6–26
Core/SEC Handshake Requirements to Ensure Proper Interrupt Handling 6–27
Configuring a System Source as a Fault 6–28
Configuring the WDOG Expiry Event to Issue a System Reset 6–28
SEC Programming Restrictions 6–28
ADSP-SC59x SEC Register Descriptions 6–29
SCI Active Register n 6–31
SCI Control Register n 6–32
SCI Group Mask Register n 6–34
SCI Priority Level Register n 6–36
SCI Priority Mask Register n 6–38
Core Pending Register n 6–39
SCI Source ID Register n 6–40
SCI Status Register n 6–41
Global End Register 6–43
Fault COP Period Register 6–44
Fault COP Period Current Register 6–45
Fault Control Register 6–46
Fault Delay Register 6–49
Fault Delay Current Register 6–50
Fault End Register 6–51
Fault Source ID Register 6–52
Fault System Reset Delay Register 6–53

Fault System Reset Delay Current Register 6–54
Fault Status Register 6–55
Global Control Register 6–57
Global Status Register 6–58
Global Raise Register 6–60
Source Control Register n 6–61
Source Status Register n 6–64
GIC Overview 6–65
GIC Functional Description 6–66
GIC Block Diagram 6–67
GIC Performance Monitoring Unit (GIC PMU) 6–67
ADSP-SC59x GIC Interrupt List 6–68

Trigger Routing Unit (TRU)

TRU Features 7–1
TRU Functional Description 7–1
ADSP-SC59x TRU Register List 7–2
ADSP-SC59x TRU Interrupt List 7–2
ADSP-SC59x Trigger List 7–3
TRU Definitions 7–17
TRU Block Diagram 7–18
TRU Architectural Concepts 7–18
TRU Programming Model 7–19
Programming Concepts 7–19
Programming Examples 7–19
Configuring a Simple Trigger Sequence 7–19
TRU Event Control 7–20
TRU Status and Error Signals 7–20
ADSP-SC59x TRU Register Descriptions 7–20
Error Address Register 7–21

Global Control Register 7–22
Generator Trigger Register 7–24
Receiver Select Register 7–25
Status Information Register 7–26

L2 System Memory

L2 System Memory Features 8–1
L2 System Memory Functional Description 8–1
ADSP-SC59x L2CTL Register List 8–1
ADSP-SC59x L2CTL Interrupt List 8–3
ADSP-SC59x L2CTL Trigger List 8–3
L2 System Memory Block Diagram 8–3
L2 System Memory Architectural Concepts 8–4
Access Characteristics 8–4
Read/Write Latency and Throughput 8–4
L2 Memory Controller Block Diagram (Instance) 8–4
Arbitration and Priority 8–5
Data Integrity 8–7
ECC Algorithm 8–7
ECC Hardware Control 8–9
ECC Error Management 8–9
Memory Refresh 8–10
Power Modes 8–10
L2 System Memory Event Control 8–11
ECC Error Interrupt 8–11
ADSP-SC59x L2CTL Register Descriptions 8–11
Control Register 8–13
Error Type 0 Address Register 8–16
Error Type 1 Address Register 8–17
Error Type 2 Address Register 8–18
Error Type 3 Address Register 8–19

Error Type 4 Address Register **8–20**
ECC Error Address 0 Register **8–21**
ECC Error Address 1 Register **8–22**
ECC Error Address 2 Register **8–23**
ECC Error Address 3 Register **8–24**
ECC Error Address 4 Register **8–25**
ECC Error Address 5 Register **8–26**
ECC Error Address 6 Register **8–27**
ECC Error Address 7 Register **8–28**
ECC Error Address 8 Register **8–29**
Error Type 0 Register **8–30**
Error Type 1 Register **8–32**
Error Type 2 Register **8–33**
Error Type 3 Register **8–34**
Error Type 4 Register **8–35**
Initialization Register **8–36**
Initialization Status Register **8–38**
Power Control Register **8–40**
Revision ID Register **8–42**
Read Priority Count Register **8–43**
Read Priority Count Register **8–44**
Scrub Start Address Register **8–45**
Scrub Count Register **8–46**
Scrub Control Register **8–47**
Status Register **8–49**
L2 Port Error Status Register **8–52**
Write Priority Count Register **8–53**
Write Priority Count Register **8–54**

Dynamic Memory Controller (DMC)

DMC Features 9–1
Feature Exclusions 9–2
DMC Functional Description 9–2
ADSP-SC59x DMC Register List 9–3
ADSP-SC59x DMC Register List 9–4
Protocol Controller 9–5
Efficiency Controller 9–5
Page-Based Scheduling 9–5
Same Controller Transaction Scheduling 9–5
DMC Read Data Buffer 9–5
Closed Page Per Bank 9–6
SCB ID-Based Priority 9–6
Delaying up to Eight Auto-Refresh Commands 9–7
Page and Bank Interleaving 9–7
System Crossbar Completer Interface 9–7
Read/Write Command and Data Buffers 9–8
Peripheral Bus Completer Interface 9–8
Architectural Concepts 9–8
Controller On Die Termination (ODT) 9–8
Mode Register Set and Extended Mode Register Set Command 9–8
DDR3 Reset Functionality 9–9
DDR3 SDRAM Organization 9–9
DMC Clocking 9–10
DMC DMA 9–10
DMC Operating Modes 9–10
Self-Refresh Mode 9–10
DMC Event Control 9–11
DMC Programming Model 9–11
PHY DLL Calibration 9–11
DDR3 ZQ Calibration Short CMD 9–11
DDR3 ZQ Calibration Long CMD 9–12

On Die Termination (DDR3) 9–12
Leveling Techniques 9–12
Initializing the DMC 9–13
Resetting the DMC Lane 9–14
Performing ZQ Calibration 9–14
Programming Duty Cycles 9–15
Programming the DMC Controller 9–15
Programming DQ Delay Trim 9–16
DMC Performance Monitor (DMCPM) 9–17
ADSP-SC59x DMC Register Descriptions 9–17
Configuration Register 9–20
Control Register 9–22
DLL Control Register 9–26
Data Calibration Address Register 9–27
Data Calibration Data 0 Register 9–28
Data Calibration Data 1 Register 9–29
Efficiency Control Register 9–30
Shadow EMR3 Register 9–34
Shadow MR0 Register (DDR3) 9–35
Shadow MR1 Register (DDR3) 9–37
Shadow MR2 Register (DDR3) 9–40
Mask (Mode Register Shadow) Register 9–42
DMC Performance Control Register 9–44
Performance Monitor Idle Cycle DQ Count Register 9–47
Performance Monitor Idle Page Count Register 9–48
Performance Monitor Page Hit Count 9–49
Performance Monitor Page Miss Count 9–50
Performance Monitor Read to Write Turn Around Count Register 9–51
Performance Monitor Refresh Count Register 9–52
Performance Monitor Write to Read Turn Around Count Register 9–53

Priority ID Register 1 9–54
Priority ID Register 2 9–55
Priority ID Mask Register 1 9–56
Priority ID Mask Register 2 9–57
DMC Read Data Buffer ID Register 1 9–58
DMC Read Data Buffer ID Register 2 9–59
DMC Read Data Buffer Mask Register 1 9–60
DMC Read Data Buffer Mask Register 2 9–61
Performance Monitor Buffer Hit Count Register 9–62
SCB Performance AXI Transfer Count Register 9–63
SCB Performance AXI Transfer Count Register 9–64
SCB Performance Monitor Idle Cycle Count Register 9–65
SCB Performance Monitor Idle Cycle Count Register 9–66
Status Register 9–67
Timing 0 Register 9–70
Timing 1 Register 9–72
Timing 2 Register 9–73
ADSP-SC59x DMC Register Descriptions 9–74
DDR CA Lane Control Register 9–75
Data Lane 0 Control Register 0 9–77
Data Lane 0 Control Register 1 9–79
Data Lane 1 Control Register 0 9–80
Data Lane 1 Control Register 1 9–82
DDR ROOT Module Control Register 9–83
Scratch Register 2 9–85
Scratch Register 3 9–86
Scratch Register 4 9–87
Scratch Register 5 9–88
Scratch Register 6 9–89
Scratch Register 7 9–90

DDR Calibration Control Register 0 9–91
DDR Calibration Control Register 1 9–92
DDR Calibration Control Register 2 9–93

DDR Pre-fetch Buffer (DDRPFB)

DDRPFB Features 10–2
Smart Pre-fetch 10–2
Data Invalidation 10–2
Direct Access 10–2
Return Zero 10–2
Controller ID Based Direct Access 10–2
Range Based Pre-fetch 10–4
ADSP-SC59x DDRPFB Register List 10–5
DDRPFB Programming Model 10–6
Enabling the Pre-fetch Buffer and PFB Idle State 10–6
General Programming Guidelines 10–6
Guidelines For Achieving Maximum Performance 10–6
ADSP-SC59x DDRPFB Register Descriptions 10–7
Control Register 0 10–8
Control Register 1 10–10
Range Select Register 10–11
Requester ID 0 Register 10–12
Requester ID 0 Mask Register 10–13
Requester ID 1 Register 10–14
Requester ID 1 Mask Register 10–15
Requester ID 2 Register 10–16
Requester ID 2 Mask Register 10–17
Requester ID 3 Register 10–18
Requester ID 3 Mask Register 10–19
Requester ID Enable Register 10–20

Prefetch Buffer First Status Register 10–21
Return Zero End Address Register 10–22
Return Zero Start Address Register 10–23

One-Time Programmable Memory Controller (OTPC)

OTPC Features 11–1
OTPC Functional Description 11–1
ADSP-SC59x OTPC Register List 11–1
ADSP-SC59x OTPC Interrupt List 11–2
OTPC Memory Layout 11–2
Error Correction 11–5
OTPC Event Control 11–5
OTPC Interrupt Signals 11–6
OTPC Status and Error Signals 11–6
OTPC Programming Model 11–6
ADSP-SC59x OTPC Register Descriptions 11–6
OTP Security State Register 11–7
OTP Status Register 11–8

System Memory Protection Unit (SMPU)

SMPU Features 12–1
SMPU Functional Description 12–2
ADSP-SC59x SMPU Register List 12–3
SMPU Interrupts 12–4
Memory Writes 12–4
Memory Reads 12–4
ID Comparison 12–4
Memory Region 12–8
SMPU Definitions 12–9
SMPU Block Diagram 12–10
SMPU Architectural Concepts 12–10

SMPU Operating Modes 12–11
SMPU Interrupt Signals 12–11
SMPU Status and Error Signals 12–12
ADSP-SC59x SMPU Register Descriptions 12–12
Bus Error Address Register 12–14
Bus Error Details Register 12–15
SMPU Control Register 12–16
Exclusive Access IDn Address 12–18
Exclusive Access Status 12–19
Interrupt Address Register 12–20
Interrupt Details Register 12–21
Region n Address Register 12–22
Region n Control Register 12–23
SMPU Revision ID Register 12–26
Region n ID A Register 12–27
Region n ID B Register 12–28
Region n ID Mask A Register 12–29
Region n ID Mask B Register 12–30
SMPU Control Secure Accesses Register 12–31
Region n Control Secure Accesses Register 12–33
SMPU Status Register 12–34

General-Purpose Ports (PORT)

PORT Features 13–2
PORT Functional Description 13–2
ADSP-SC59x PORT Register List 13–3
ADSP-SC59x PORT Trigger List 13–3
ADSP-SC59x PINT Register List 13–4
ADSP-SC59x PINT Interrupt List 13–4
ADSP-SC59x PINT Trigger List 13–5

ADSP-SC59x PADS Register List 13–5
PORT Architectural Concepts 13–7
Internal Interfaces 13–7
External Interfaces 13–7
GPIO Pin Function 13–7
Input Mode 13–7
Output Mode 13–7
Trigger Toggle Mode 13–8
Open-Drain Mode 13–8
Port Multiplexing Control 13–8
PORT Event Control 13–9
PORT Interrupt Signals 13–9
PORT Programming Model 13–11
Programmable Pull-up Resistors for PORT and DAI 13–14
ADSP-SC59x PORT Register Descriptions 13–15
Port x GPIO Data Register 13–17
Port x GPIO Data Clear Register 13–21
Port x GPIO Data Set Register 13–25
Port x GPIO Output Toggle Register 13–28
Port x GPIO Direction Register 13–31
Port x GPIO Direction Clear Register 13–35
Port x GPIO Direction Set Register 13–39
Port x Function Enable Register 13–42
Port x Function Enable Clear Register 13–45
Port x Function Enable Set Register 13–48
Port x GPIO Input Enable Register 13–51
Port x GPIO Input Enable Clear Register 13–54
Port x GPIO Input Enable Set Register 13–57
Port x GPIO Lock Register 13–60
Port x Multiplexer Control Register 13–62

Port x GPIO Polarity Invert Register 13–64
Port x GPIO Polarity Invert Clear Register 13–68
Port x GPIO Polarity Invert Set Register 13–71
Port x GPIO Trigger Toggle Register 13–74
ADSP-SC59x PINT Register Descriptions 13–75
PINT Assign Register 13–77
PINT Edge Clear Register 13–79
PINT Edge Set Register 13–82
PINT Invert Clear Register 13–85
PINT Invert Set Register 13–88
PINT Latch Register 13–91
PINT Mask Clear Register 13–95
PINT Mask Set Register 13–98
PINT Pin State Register 13–101
PINT Request Register 13–105
ADSP-SC59x PADS Register Descriptions 13–109
DAI0 1 to 10 pins DS control 13–111
DAI0 11 to 20 pins DS control 13–114
DAI0 Port Input Enable Control Register 13–117
DAI1 1 to 10 pins DS control 13–118
DAI1 11 to 20 pins DS control 13–121
DAI1 Port Input Enable Control Register 13–124
DAI[n] Pull-Down Enable 13–125
DAI[n] Pull-Up Enable 13–126
Non-GPIO Drive Strength Register 13–127
Peripheral PAD Configuration0 Register 13–130
Peripheral Configuration1 Register 13–133
PORTA 0 to 7 pins DS control 13–136
PORTA 8 - 15 pins DS control 13–139
PORTB 0 to 7 pins DS control 13–142

PORTB 8 - 15 pins DS control 13–145
PORTC 0 to 7 pins DS control 13–148
PORTC 8 - 15 pins DS control 13–151
PORTD 0 to 7 pins DS control 13–152
PORTD 8 to 15 pins DS control 13–153
PORTE 0 to 7 pins DS control 13–154
PORTE 8 to 15 pins DS control 13–155
PORTF 0 to 7 pins DS control 13–156
PORTF 8 to 15 pins DS control 13–157
PORTG 0 to 7 pins DS control 13–158
PORTG 8 to 15 pins DS control 13–159
PORTH 0 to 7 pins DS control 13–160
PORTH 8 to 15 pins DS control 13–161
PORTI 0 to 7 pins DS control 13–162
PORT[n] Pull-Down Enable 13–163
PORT[n] Pull-Up Enable 13–164

Thermal Monitoring Unit (TMU)

TMU Features 14–1
TMU Functional Description 14–1
ADSP-SC59x TMU Register List 14–1
ADSP-SC59x TMU Interrupt List 14–2
ADSP-SC59x TMU Trigger List 14–2
TMU Definitions 14–2
TMU Block Diagram 14–3
TMU Architectural Concepts 14–3
TMU and HADC 14–4
TMU Event Control 14–5
Status and Error Signals 14–5
TMU Programming Guidelines 14–5

TMU Calibration 14–6
ADSP-SC59x TMU Register Descriptions 14–6
Alert High Limit Register 14–8
Alert Low Limit Register 14–9
Averaging Register 14–10
Temperature Conversion Blank Register 14–11
TMU Control Register 14–12
Fault High Limit Register 14–13
Fault Low Limit Register 14–14
Gain Value Register 14–15
Interrupt Mask Register 14–16
Offset Register 14–17
Temperature Refresh Counter 14–18
Status Register 14–19
Temperature Value Register 14–20

Housekeeping ADC (HADC)

HADC Features 15–1
HADC Functional Description 15–1
ADSP-SC59x HADC Register List 15–2
ADSP-SC59x HADC Interrupt List 15–2
ADSP-SC59x HADC Trigger List 15–3
HADC Definitions 15–3
HADC Block Diagram 15–4
HADC Signal Descriptions 15–5
HADC Architectural Concepts 15–5
Converter Operation 15–5
Auto-Scan 15–6
Channel Sequence Programming 15–6
ADC Transfer Function 15–7
Results 15–7

HADC Operating Modes 15–7
HADC Event Control 15–8
HADC Programming Model 15–8
ADSP-SC59x HADC Register Descriptions 15–8
Channel Mask Register 15–10
Control Register 15–11
Channel Data Registers 15–13
Interrupt Mask Register 15–14
Status Register 15–15

Controller Area Network Flexible Data Rate (CANFD)

CANFD Features 16–1
CANFD Functional Description 16–2
ADSP-SC59x CANFD Register List 16–3
ADSP-SC59x CANFD Interrupt List 16–4
ADSP-SC59x CANFD Trigger List 16–5
CANFD Architectural Concepts 16–5
Block Diagram 16–5
Message Buffer (MB) 16–6
Message Buffer Memory Map 16–12
Memory Partition 16–17
Rx FIFO Structure 16–18
CANFD Processes 16–21
Transmit Process 16–21
Arbitration Process 16–22
Lowest Number Mailbox First 16–22
Highest-Priority Mailbox First 16–22
Arbitration Completion 16–23
Arbitration Start and Stop Conditions 16–24
Receive Process 16–25
Matching Process 16–26

Receive Process Pretending Network Mode 16–31
Move Process 16–34
Move-In 16–34
Move-Out 16–36
Data Coherence 16–36
Transmission Abort Mechanism 16–36
Mailbox Inactivation 16–37
Mailbox Lock Mechanism 16–38
Rx FIFO 16–39
Rx FIFO Under DMA Operation 16–40
Clear FIFO Operation 16–40
CAN Protocol Features 16–41
CAN FD Frames 16–41
Transceiver Delay Compensation 16–45
Remote Frames 16–46
Overload Frames 16–47
Time Stamp 16–47
Protocol Timing 16–48
Arbitration and Matching Timing 16–51
Tx Arbitration Start Delay 16–52
CANFD Clock Domains and Restrictions 16–55
CANFD Operating Modes 16–58
Normal Mode 16–58
Freeze Mode 16–59
Loop-Back Mode 16–59
Listen-Only Mode 16–60
CAN FD Active Mode 16–60
Module Disable Mode 16–60
Doze Mode 16–61
Stop Mode 16–62
Pretended Network Mode 16–63

CANFD Event Control 16–64
Interrupts 16–64
Bus Interface 16–65
Detection and Correction of Memory Errors 16–66
Sources of Memory Access 16–68
Error Indication 16–68
Error Reporting 16–69
Response to Errors 16–69
Error Injection 16–70
CANFD Programming Model 16–71
Initialization 16–72
Pretended Network - Doze Mode 16–75
Pretended Network - Stop Mode 16–75
ADSP-SC59x CANFD Register Descriptions 16–75
Can Bit Timing Register 16–78
CRC Register 16–81
Control 1 Register 16–82
Pretended Networking Control1 Register 16–88
Control 2 Register 16–91
Pretended Networking Control2 Register 16–96
Error Counter Register 16–97
Error Injection Address Register 16–100
Error Injection Data Pattern Register 16–101
Error Injection Parity Pattern Register 16–102
Error Status Register 16–103
Error and Status 1 Register 16–106
Error and Status 2 Register 16–117
CANFD Bit Timing Register 16–119
CANFD CRC Register 16–121
CANFD Control Register 16–123

Pretended Networking DLC Filter Register 16–127
Pretended Networking ID Filter1 Register 16–128
Pretended Networking ID Filter2 / IDMask Register 16–129
Mailbox Interrupt Flag 1 Register 16–131
Mailbox Interrupt Flag 2 Register 16–142
Mailbox Interrupt Mask 1 Register 16–150
Mailbox Interrupt Mask 2 Register 16–158
Module Configuration Register 16–166
Memory Error Control Register 16–175
Pretended Networking Payload Low Filter2 Register 16–178
Pretended Networking Payload Low Filter1 Register 16–179
Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register 16–180
Pretended Networking Payload Low Filter2 / Payload Low Mask Register 16–181
Error Report Address Register 16–182
Error Report Data Register 16–184
Error Report Syndrome Register 16–185
Receive Mailbox14 Mask Register 16–187
Receive Mailbox15 Mask Register 16–188
Receive FIFO Global Mask Register 16–189
Receive FIFO Information Register 16–190
Receive Individual Mask Register 16–191
Receive Mailbox Global Mask Register 16–192
Free Running Timer Register 16–193
Wakeup Message Buffer Data 4-7 Register 16–194
Wakeup Message Buffer Data 0-3 Register 16–195
Wakeup Message ID Buffer Register 16–196
Wakeup Message Buffer Control/Status Register 16–197
Pretended Networking Wakeup Match Register 16–198
ADSP-SC59x MISCREG Register Descriptions 16–199
CANFD Low Power Selection Mode 16–200

Error Register 16–202
PLL2 Control Register 16–205
SH0 Prefetch Range Selection Register 16–207
SH1 Prefetch Range Selection Register 16–208
SPORT Direct Interrupt Enable for SH0 and SH1 16–209

Watchdog Timer (WDOG)

WDOG Features 17–1
WDOG Functional Description 17–2
ADSP-SC59x WDOG Register List 17–3
ADSP-SC59x WDOG Interrupt List 17–3
ADSP-SC59x WDOG Trigger List 17–3
WDOG Block Diagram 17–3
Internal Interface 17–4
External Interface 17–4
ADSP-SC59x WDOG Register Descriptions 17–4
Count Register 17–5
Control Register 17–6
Watchdog Timer Status Register 17–7
Watchdog Timer Window Register 17–9

Enhanced Mobile Storage Interface (eMSI)

eMSI Features 18–1
eMSI Functional Description 18–2
ADSP-SC59x EMSI Register List 18–2
ADSP-SC59x EMSI Interrupt List 18–4
ADSP-SC59x EMSI Trigger List 18–5
eMSI Block Diagram 18–5
eMSI Architectural Concepts 18–5
Generating SD/eMMC Commands 18–7
Block Count 18–7

Preset Value Registers 18–8
Automatic Command Generation (Auto CMD) 18–8
Data Transfer Modes 18–9
Direct Memory Access (DMA) 18–9
Non Descriptor (Single Operation) DMA 18–9
Descriptor (Advanced) DMA 18–10
Advanced DMA 2 (ADMA2) 18–10
Advanced DMA 3 (ADMA3) 18–14
Command Queueing Engine 18–17
Task Processing 18–18
Task Selection and Execution 18–19
Task Completion: Interrupts and Interrupt Coalescing 18–19
Direct Command (DCMD) Submission 18–19
Queue-Barrier (QBR) Tasks 18–20
eMMC Command Queuing Data Structures 18–20
Task Descriptor Structure 18–21
Transfer Descriptor Structure 18–22
Task Descriptor for Direct-Command (DCMD) Tasks 18–23
eMMC Boot Operation 18–24
Mandatory Boot Operation 18–24
Alternate Boot Operation 18–25
eMSI Event Control and Interrupts 18–26
Error Detection 18–26
Response Error Check 18–26
eMSI Programming Model 18–27
Initial Programming Sequence 18–27
Card Detection 18–28
eMSI Controller Setup Sequence 18–29
eMMC Device Setup 18–29
SD Interface Setup 18–30
Clock Control 18–31

eMSI Controller Clock Setup Sequence 18–31
eMSI Bus Clock Supply and Stop Sequence 18–32
eMSI Bus Clock Frequency Change Sequence 18–32
Card Interface Setup Sequence 18–34
SD Card Setup Sequence 18–34
eMMC Card Interface Setup 18–35
Setting Timeout on the eMSI Bus 18–35
Abort Transaction 18–36
Abort Command Sequence 18–36
Asynchronous Abort 18–38
eMMC Transaction Mode 18–38
Initializing and Identifying an eMMC Device 18–38
Switching to Various Speed Modes in an eMMC Device 18–40
Changing the Data Bus Width for an eMMC Device 18–40
Error Recovery in eMMC Mode 18–41
Issuing CMD With or Without Data Transfers for an eMMC Device 18–45
Issuing CMD Without Data Transfer 18–45
Issuing CMD With Data Transfer 18–48
Command Queueing 18–56
Initializing the Command Queuing Engine 18–56
Submitting and Completing a Task in the Command Queuing Engine 18–58
CQE Error Recovery 18–60
eMMC Device Boot Programming Sequences 18–61
Preparing for a Boot 18–61
Initiating a Mandatory Boot in SDMA Mode 18–62
Initiating a Mandatory Boot in ADMA2 Mode 18–64
Initiating Alternate Boot 18–66
SD Transaction Mode 18–66
SD Card Initialization and Identification 18–66
Changing Bus Width for an SD Card 18–67
Changing Bus Speed Mode for an SD Card 18–68

Issuing CMD With or Without Data Transfers for an SD Card 18–68
ADSP-SC59x EMSI Register Descriptions 18–69
ADMA System Address Register - Low 18–72
ADMA3 Integrated Descriptor Address Register - Low 18–73
ADMA Error Status Register 18–74
Argument Register 18–75
Auto CMD Status Register 18–76
Base Address 0 Register 18–78
Base Address 1 Register 18–79
16-bit Block Count Register 18–80
Block Size Register 18–81
eMMC Boot Control Register 18–83
Capabilities 1 Register - 0 to 31 18–85
Capabilities Register - 32 to 63 18–91
Clock Control Register 18–94
Command Register 18–96
Command Queuing Version Register 18–98
Command Queuing Capabilities Register 18–99
Command Queuing Configuration Register 18–100
CQ Command Response Argument Register 18–102
Command Response for Direct Command Register 18–103
CQ Command Response Index 18–104
Command Queuing Control Register 18–105
Device Pending Tasks Register 18–106
Device Queue Status Register 18–107
Command Queuing Interrupt Coalescing Register 18–108
Command Queuing Interrupt Status Register 18–111
Command Queuing Interrupt Status Enable Register 18–113
Command Queuing Interrupt Signal Enable Register 18–115
Command Response Mode Error Mask Register 18–116

CQ Send Status Configuration 1 Register 18–117
CQ Send Status Configuration 2 Register 18–119
Command Queuing Task Clear Register 18–120
Command Queuing Task Completion Notification Register 18–121
Command Queuing DoorBell Register 18–122
Command Queuing Task Descriptor List Base Address Register 18–123
CQ Task Error Information Register 18–124
EMSI Control 1 Register 18–126
EMSI Control 2 Register 18–128
SD Host Controller Specification Version 18–130
eMMC Control Register 18–131
Error Interrupt Status Register 18–133
Error Interrupt Status Enable Register 18–137
Error Interrupt Signal Enable Register 18–140
Force Event Register for Auto CMD Error Status Register 18–143
Force Event Register for Error Interrupt Status 18–145
Interrupt Status Register 18–148
Interrupt Status Enable Register 18–150
Interrupt Signal Enable Register 18–152
Preset Value for Default Speed 18–154
Preset Value for HSDDR 18–155
Preset Value for HSSDR 18–156
Preset Value for Initialization 18–157
Preset Value for Legacy Mode 18–158
Present State Register 18–159
Response Register 0 18–163
Response Register 1 18–164
Response Register 2 18–165
Response Register 3 18–166
SDMA System Address Register 18–167

Software Reset Register 18–169
Timeout Control Register 18–171
Transfer Mode Register 18–172
EMSI Version 18–175
EMSI Version Type 18–176
Wakeup Control Register 18–177

Link Port (LP)

LP Features 19–1
LP Functional Description 19–1
ADSP-SC59x LP Register List 19–2
ADSP-SC59x LP Interrupt List 19–2
ADSP-SC59x LP Trigger List 19–2
ADSP-SC59x LP DMA Channel List 19–3
Block Diagram 19–3
External Connections 19–4
Internal Blocks 19–4
Architectural Concepts 19–4
Link Port Protocol (SC59x) 19–4
FIFO Buffers 19–7
Handshake for Link Port Enable Process 19–9
Clocking 19–10
Multi-Processor Connectivity 19–10
LP Operating Modes 19–12
LP Data Transfer Modes 19–12
Core Data Transfers 19–12
DMA Data Transfers 19–12
LP Event Control 19–13
Interrupt Signals 19–13
Enabling Link Port Interrupts 19–13
Status and Error Signals 19–14

LP Programming Model 19–14
Setting Up a DMA Transmit Operation 19–14
Setting Up a DMA Receive Operation 19–15
Setting Up a Core Transmit Operation 19–16
Setting Up a Core Receive Operation 19–16
ADSP-SC59x LP Register Descriptions 19–17
Control Register 19–18
Clock Divider Value Register 19–20
Receive Buffer Register 19–21
Status Register 19–22
Transmit Buffer Register 19–24
Shadow Input Transmit Buffer Register 19–25
Shadow Output Transmit Buffer Register 19–26

Pulse Density Modulation (PDM) Microphone Interface

PDM Features 20–1
PDM Functional Description 20–1
ADSP-SC59x PDM Register List 20–2
PDM Block Diagram 20–2
PDM Architectural Concepts 20–3
PDM Initialization and Clocking 20–3
SPORT Interface 20–4
SPORT Timing 20–5
High Pass Filter 20–7
PDM Programming Model 20–8
Configuring the PDM Interface 20–8
Software Reset 20–9
ADSP-SC59x PDM Register Descriptions 20–9
PDM Control Register 20–10
High Pass Filter Control Register 20–11
Software Reset Register 20–12

Serial Port Control0 Register 20–13
Serial Port Control1 Register 20–15

Universal Serial Bus Controller (USBC)

USBC Features 21–1
USBC Functional Description 21–3
ADSP-SC59x USBC Register List 21–3
ADSP-SC59x USBULPI Interrupt List 21–6
USBC Block Diagram 21–6
USBC Definitions 21–6
Transmit and Receive FIFOs 21–8
DDR Mode 21–8
Restrictions 21–9
USBC Architectural Concepts 21–10
System Level Architecture 21–10
Bus Interface Unit (BIU) 21–10
Requester Bus Interface Unit (BIUM) 21–11
Completer Bus Interface Unit (BIUS) 21–11
Internal DMA Controller 21–14
Host Architecture 21–15
Device Architecture 21–16
Control and Status Registers (CSR) 21–18
Application Interface Unit (AIU) 21–18
DMA Scheduler (DSCH) 21–19
Periodic FIFO Control (PFC) 21–20
Periodic Transmit Data FIFO 21–20
Non-Periodic Transmit Data FIFO 21–20
Endpoint Information Controller (EPINFO_CTL) 21–23
Media Access Controller (MAC) 21–23
MAC Components 21–24
USB Transaction Handling 21–25

Host Protocol Handling 21–25
Device Protocol Handling 21–25
OTP Protocol Handling 21–25
USBC Programming Model 21–26
Program the Host Core 21–26
Initialize the Host 21–26
Connect the Host 21–26
Disconnect the Host 21–27
Program the Device Core 21–27
Initialize the Device 21–27
Connect the Device 21–28
Disconnect the Device 21–29
Initialize the Endpoint 21–29
Initialize upon USB Reset 21–29
Initialize upon Enumeration Completion 21–30
OTG Programming 21–31
A Device Host Negotiation 21–31
B Device Host Negotiation 21–32
ADSP-SC59x USBC Register Descriptions 21–33
Device All Endpoint Interrupt Status Register 21–36
Device All Endpoint Interrupt Mask Register 21–39
Device Configuration Register 21–43
Device Control Register 21–47
Device Control IN Endpoint 0 Control Register 21–52
Device Control IN Endpoint n Control Register 21–54
Device Control IN Endpoint 0 DMA Address Register 21–59
Device IN Endpoint 0 Buffer Address Register 21–60
Device Control IN Endpoint n DMA Buffer Address Register 21–61
Device Control IN Endpoint n DMA Address Register 21–62
Device IN Endpoint FIFO Empty Interrupt Mask Register 21–63
Device Control IN Endpoint 0 Interrupt Control Register 21–64

Device Control IN Endpoint n Interrupt Control Register **21–67**
Device IN Endpoint Common Interrupt Mask Register **21–70**
Device Control IN Endpoint 0 Transfer Size Register **21–72**
Device Control IN Endpoint n Transfer Size Register **21–74**
Device IN Endpoint 1 Transmit FIFO Size Register **21–76**
Device IN Endpoint 2 Transmit FIFO Size Register **21–77**
Device IN Endpoint 3 Transmit FIFO Size Register **21–79**
Device OUT Endpoint 0 Control Register **21–81**
Device OUT Endpoint n Control Register **21–83**
Device OUT Endpoint 0 DMA Address Register **21–88**
Device OUT Endpoint 0 Buffer Address Register **21–89**
Device OUT Endpoint n Buffer Address Register **21–90**
Device OUT Endpoint n DMA Address Register **21–91**
Device OUT Endpoint 0 Interrupt Register **21–92**
Device OUT Endpoint n Interrupt Register **21–96**
Device OUT Endpoint Common Interrupt Mask Register **21–100**
Device OUT Endpoint 0 Transfer Size Register **21–102**
Device OUT Endpoint n Transfer Size Register **21–103**
Device Status Register **21–104**
Device Threshold Control Register **21–106**
Device Control IN Endpoint Transmit FIFO Status Register **21–108**
Device Control IN Endpoint Transmit FIFO Status Register **21–109**
Device VBUS Discharge Time Register **21–110**
Device VBUS Pulsing Time Register **21–111**
Bus Configuration Register **21–112**
DFIFO Configuration Register **21–116**
User Hardware Configuration 1 Register **21–117**
User Hardware Configuration 2 Register **21–118**
User Hardware Configuration 3 Register **21–121**
User Hardware Configuration 4 Register **21–123**

Interrupt Mask Register **21–126**
Interrupt Status Register **21–129**
Non-periodic Transmit FIFO Size Register **21–138**
Non-periodic Transmit FIFO/Queue Status Register **21–139**
OTG Control and Status Register **21–141**
OTG Interrupt Register **21–147**
PHY Interface Control Register **21–149**
Reset Register **21–151**
Receive FIFO Size Register **21–156**
Receive Status Read/Pop Register **21–157**
Receive Status Debug Read Register **21–160**
Module ID Register **21–163**
USB Configuration Register **21–164**
Host All Channels Interrupt Register **21–171**
Host All Channels Interrupt Mask Register **21–172**
Host Channel n Characteristics Register **21–173**
Host Channel n DMA Buffer Address Register **21–176**
Host Channel n DMA Address Register **21–177**
Host Configuration Register **21–178**
Host Channel n Interrupt Mask Register **21–181**
Host Channel n Interrupt Status Register **21–182**
Host Channel n Split Control Register **21–186**
Host Channel n Transfer Size Register **21–188**
Host Frame Interval Register **21–190**
Host Frame List Base Address Register **21–191**
Host Frame Number/Frame Time Remaining Register **21–192**
Host Port Control and Status Register **21–193**
Host Periodic Transmit FIFO Size Register **21–198**
Power and Clock Gating Control Register **21–199**

Serial Peripheral Interface (SPI)

SPI Features 22–1
SPI Functional Description 22–2
ADSP-SC59x SPI Register List 22–2
ADSP-SC59x SPI Interrupt List 22–3
ADSP-SC59x SPI Trigger List 22–4
ADSP-SC59x SPI DMA Channel List 22–5
SPI Block Diagram 22–5
Transfer Protocol 22–6
Clock Considerations 22–8
Controlling Delay Between Frames 22–8
Flow Control 22–10
Target Select Operation 22–11
Beginning and Ending a Non-DMA SPI Transfer 22–12
Transmit Operation in Non-DMA Mode 22–13
Receive Operation in Non-DMA Mode 22–13
Dual I/O Mode 22–13
Quad I/O Mode (SPI1/SPI2) 22–14
Fast Mode 22–15
Memory-Mapped Mode (SPI2) 22–16
Memory-Mapped Description of Operation 22–17
Memory-Mapped Architectural Concepts 22–18
Memory-Mapped Read Accesses 22–20
Memory-Mapped High-Performance Features 22–23
Merged Read Accesses 22–23
Wrap Around Accesses 22–23
Execute-In-Place (XIP, SPI2 only) 22–24
Memory-Mapped Mode Error Status Bits 22–25
Memory-Mapped Programming Guidelines 22–26
SPI Interrupt Signals 22–28

Data Interrupts 22–28
Status Interrupts 22–29
Error Conditions 22–29
SPI Programming Concepts 22–30
Controller Operation in Non-DMA Modes 22–31
Target Operation in Non-DMA Modes 22–31
Configuring DMA Controller Mode 22–32
Configuring DMA Target Mode Operation 22–33
ADSP-SC59x SPI Register Descriptions 22–34
Clock Rate Register 22–36
Control Register 22–37
Delay Register 22–43
Masked Interrupt Condition Register 22–44
Masked Interrupt Clear Register 22–46
Interrupt Mask Register 22–49
Interrupt Mask Clear Register 22–51
Interrupt Mask Set Register 22–54
Memory Mapped Read Header 22–57
SPI Memory Top Address 22–61
Receive FIFO Data Register 22–62
Received Word Count Register 22–63
Received Word Count Reload Register 22–64
Receive Control Register 22–65
Slave Select Register 22–68
Status Register 22–71
Transmit FIFO Data Register 22–76
Transmitted Word Count Register 22–77
Transmitted Word Count Reload Register 22–78
Transmit Control Register 22–79

Octal Serial Peripheral Interface (OSPI)

OSPI Features 23–1
OSPI Functional Description 23–2
ADSP-SC59x OSPI Register List 23–2
ADSP-SC59x OSPI Interrupt List 23–3
OSPI Block Diagram 23–3
Architectural Concepts 23–4
Direct Access Controller (DAC) 23–5
Software Triggered Instruction Generator (STIG) 23–6
Servicing STIG Request 23–7
Arbitration Between DAC and STIG Access 23–7
Auto Polling for DAC Write Access 23–8
SPI Command Translation 23–8
Hold and Reset Control 23–9
Flash Instruction Type Support 23–10
Dual Data Rate (DDR) Operations 23–10
Data Sampling 23–14
PHY Module Architecture 23–14
PHY Pipeline Mode 23–18
Programming Concepts 23–18
Configuring OSPI after Reset 23–18
Programming Dummy Cycles 23–19
Configuring OSPI for Optimal Use 23–19
Configuring OSPI for DAC Read Operation 23–20
Configuring OSPI for DAC Write Operation 23–20
Issuing STIG Command 23–21
Entering XIP mode 23–23
Using PHY Mode 23–24
ADSP-SC59x OSPI Register Descriptions 23–26
Octal SPI Control Register 23–28

Device Delay Register **23–33**
Device Read Instruction Control Register **23–34**
Device Write Instruction Control Register **23–36**
Device Size Control Register **23–38**
DLL Observable Register (Lower) **23–39**
DLL Observable Register (Upper) **23–41**
Flash Command Address Register **23–42**
Flash Command Control Register **23–43**
Flash Command Control Memory Register **23–45**
Flash Command Read Data Register (Lower) **23–46**
Flash Command Read Data Register (Upper) **23–47**
Flash Command Write Data Register (Lower) **23–48**
Flash Command Write Data Register (Upper) **23–49**
Interrupt Mask Register **23–50**
Interrupt Status Register **23–51**
Lower Write Protection Register **23–53**
Mode Bit Control Register **23–54**
Module ID Register **23–55**
Polling Expiration Register **23–56**
Opcode Extension Register (Lower) **23–57**
Opcode Extension Register (Upper) **23–58**
PHY Control Register **23–59**
PHY DLL Master Control Register **23–60**
Polling Flash Status Register **23–61**
Read Data Capture Register **23–62**
Remap Address Register **23–63**
Upper Write Protection Register **23–64**
Write Protection Control Register **23–65**
Write Completion Control Register **23–66**

Universal Asynchronous Receiver/Transmitter (UART)

UART Features 24–1
UART Functional Description 24–2
ADSP-SC59x UART Register List 24–3
ADSP-SC59x UART Interrupt List 24–3
ADSP-SC59x UART Trigger List 24–4
ADSP-SC59x UART DMA Channel List 24–5
UART Block Diagram 24–5
UART Architectural Concepts 24–6
Internal Interface 24–6
External Interface 24–6
Hardware Flow Control 24–7
Bit Rate Generation 24–7
Autobaud Detection 24–8
UART Debug Features 24–9
UART Operating Modes 24–10
UART Mode 24–10
IrDA SIR Mode 24–11
Multi-Drop Bus Mode 24–11
UART Data Transfer Modes 24–12
UART Mode Transmit Operation (Core) 24–12
UART Mode LIN Break Command 24–13
UART Mode Receive Operation (Core) 24–14
IrDA Transmit Operation 24–15
IrDA Receive Operation 24–15
MDB Transmit Operation 24–16
MDB Receive Operation 24–16
DMA Mode 24–17
Mixing DMA and Core Modes 24–18
Setting Up Hardware Flow Control 24–18
UART Event Control 24–18

Interrupt Masks 24–19
Interrupt Servicing 24–19
Transmit Interrupts 24–19
Receive Interrupts 24–21
Status Interrupts 24–22
Multi-Drop Bus Events 24–23
UART Programming Model 24–23
Detecting Autobaud 24–23
Using Common Initialization Steps 24–24
Using Core Transfers 24–24
Using DMA Transfers 24–24
Using Interrupts 24–24
Setting Up Hardware Flow Control 24–24
ADSP-SC59x UART Register Descriptions 24–24
Clock Rate Register 24–26
Control Register 24–27
Interrupt Mask Register 24–33
Interrupt Mask Clear Register 24–37
Interrupt Mask Set Register 24–39
Receive Buffer Register 24–41
Receive Shift Register 24–42
Receive Counter Register 24–43
Scratch Register 24–44
Status Register 24–45
Transmit Address/Insert Pulse Register 24–50
Transmit Hold Register 24–51
Transmit Shift Register 24–52
Transmit Counter Register 24–53

Enhanced Parallel Peripheral Interface (EPPI)

EPPI Features 25–1
EPPI Functional Description 25–2
ADSP-SC59x EPPI Register List 25–3
ADSP-SC59x EPPI Interrupt List 25–3
ADSP-SC59x EPPI Trigger List 25–4
RGB Data Formats 25–4
Data Clipping 25–4
Data Mirroring 25–5
Windowing 25–6
Preamble, Blanking and Stripping Support 25–6
EPPI Definitions 25–7
EPPI Block Diagram 25–8
EPPI Architectural Concepts 25–8
Reset Operation 25–8
Frame Sync Polarity and Sampling Edge 25–9
Direct Memory Access (DMA) 25–9
EPPI Clock 25–10
EPPI Operating Modes 25–11
ITU-R 656 Modes 25–11
ITU-R 656 Background 25–11
ITU-R 656 Input Modes 25–15
Entire Field 25–15
Active Video 25–15
Vertical Blanking Interval (VBI) 25–15
ITU-R 656 Output in General-Purpose Transmit Modes 25–16
Frame Synchronization in ITU-R 656 Modes 25–17
General-Purpose EPPI Modes 25–18
General-Purpose 0 Frame Sync Mode 25–18
General-Purpose 1 Frame Sync Mode 25–18
General-Purpose 2 Frame Sync Mode 25–19
Data Enable in General-Purpose 2 Frame Sync Transmit Mode 25–19

General-Purpose 3 Frame Sync Mode 25–19
Supported Data Formats 25–20
Receive Data Formats 25–20
Transmit Data Formats 25–22
Data Transfer Modes 25–23
Data Packing for Receive Modes 25–23
Data Packing for Transmit Modes 25–24
Sign-Extended and Zero-Filled Data 25–24
Split Receive Modes 25–24
Split Transmit Modes 25–25
Clock Gating 25–25
Support for Delayed Start of EPPI Frame Syncs 25–25
Ignoring Premature External Frame Syncs for Data Consistency 25–26
EPPI Event Control 25–26
EPPI Status, Error, and Interrupt Signals 25–27
Frame and Line Track Errors 25–27
Line Track Errors 25–27
Frame Track Errors 25–27
Preamble Error Not Corrected Error 25–28
EPPI Programming Model 25–28
Receiving ITU-R 656 Frames 25–28
Transmitting ITU-R 656 Frames in GP Transmit Modes 25–28
Configuring Transfers in GP 0 FS Mode 25–29
Configuring Transfers in GP 1 FS Mode 25–29
Configuring Transfers in GP 2 FS Mode 25–30
Configuring Transfers in GP 3 FS Mode 25–31
Configuring the EPPI to Use the Windowing Feature 25–31
EPPI Mode Configuration 25–32
Configuring 8-Bit Receive Mode 25–32
Configuring 10/12/14-Bit Receive Modes 25–33
Configuring 16-Bit Receive Mode 25–35

Configuring 18-Bit Receive Mode 25–36
Configuring 8-Bit Split Receive Mode 25–37
Configuring 10/12/14/16-Bit Split Receive Mode with SPLTWRD=0 25–40
Configuring 16-Bit Split Receive Mode with SPLTWRD=1 25–41
Configuring 8-Bit Transmit Mode 25–42
Configuring 10/12/14-Bit Transmit Modes 25–43
Configuring 16-Bit Transmit Mode 25–43
Configuring 18-Bit Transmit Mode 25–44
Configuring 8-Bit Split Transmit Mode 25–44
Configuring 10/12/14/16-Bit Transmit Mode with SPLTWRD=0 25–47
Configuring 16-Bit Split Transmit Mode with SPLTWRD=1 25–49
EPPI Programming Concepts 25–50
ADSP-SC59x EPPI Register Descriptions 25–50
Clock Divide Register 25–52
Control Register 25–53
Control Register 2 Register 25–61
Clipping Register for EVEN (Luma) Data Register 25–62
Lines Per Frame Register 25–63
Frame Sync 1 Delay Value Register 25–64
FS1 Period Register / EPPI Active Samples Per Line Register 25–65
FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register 25–66
Frame Sync 2 Delay Value Register 25–67
FS2 Period Register / EPPI Active Lines Per Field Register 25–68
FS2 Width Register / EPPI Lines Of Vertical Blanking Register 25–69
Horizontal Transfer Count Register 25–70
Horizontal Delay Count Register 25–71
Interrupt Mask Register 25–72
Samples Per Line Register 25–74
Clipping Register for ODD (Chroma) Data Register 25–75
Status Register 25–76

Vertical Transfer Count Register 25–79
Vertical Delay Count Register 25–80

General-Purpose Timer (TIMER)

GP Timer Features 26–1
ADSP-SC59x TIMER Register List 26–2
ADSP-SC59x TIMER Interrupt List 26–2
ADSP-SC59x TIMER Trigger List 26–3
Timer Block Diagram 26–5
Internal Interface 26–5
Internal Timer Connections 26–6
External Interface 26–6
GP Timer Operating Modes 26–6
General Operation 26–7
Period, Width and Delay Register Interaction 26–7
Single-Pulse PWMOUT Mode 26–8
Continuous PWMOUT Mode 26–9
Width Capture (WIDCAP) Mode 26–10
Width Capture Mode Overflow 26–13
Windowed Watchdog (WATCHDOG) Modes 26–15
Windowed Watchdog Width Mode 26–15
Windowed Watchdog Period Mode 26–17
Pin Interrupt (PININT) Mode 26–19
External Clock (EXTCLK) Mode 26–19
GP Timer Programming Concepts 26–20
Setting Up Constantly Changing Timer Conditions 26–20
Configuring, Enabling, and Disabling One or More Timers 26–21
Configuring Timer Data and Status Interrupts 26–21
Configuring the Timer as a Trigger Receiver 26–21
Ordered Trigger Toggle Mode 26–22
Using the Timer Broadcast Feature 26–22

Timer Illegal States 26–23
Continuous PWMOUT Mode 26–23
Single Pulse PWMOUT Mode 26–25
WIDCAP Mode 26–25
EXTCLK Mode 26–25
WATCHDOG Events 26–26
ADSP-SC59x TIMER Register Descriptions 26–27
Broadcast Delay Register 26–29
Broadcast Period Register 26–30
Broadcast Width Register 26–31
Data Interrupt Latch Register 26–32
Data Interrupt Mask Register 26–33
Error Type Status Register 26–34
Run Register 26–38
Run Clear Register 26–39
Run Set Register 26–40
Status Interrupt Latch Register 26–41
Status Interrupt Mask Register 26–42
Stop Configuration Register 26–43
Stop Configuration Clear Register 26–44
Stop Configuration Set Register 26–45
Timer n Configuration Register 26–46
Timer n Counter Register 26–51
Timer n Delay Register 26–52
Timer n Period Register 26–53
Timer n Width Register 26–54
Trigger Receiver Enable Register 26–55
Trigger Generator Mask Register 26–56

General-Purpose Counter (CNT)

GP Counter Features 27–1
GP Counter Functional Description 27–1
ADSP-SC59x CNT Register List 27–2
ADSP-SC59x CNT Interrupt List 27–2
ADSP-SC59x CNT Trigger List 27–3
GP Counter Operating Modes 27–3
Quadrature Encoder Mode 27–3
Binary Encoder Mode 27–4
Up/Down Counter Mode 27–4
Direction Counter Mode 27–5
Timed Direction Mode 27–5
GP Counter Programming Model 27–5
GP Counter General Programming Flow 27–5
GP Counter Mode Configuration 27–5
Configuring GP Counter Push-Button Operation 27–6
Configuring Zero-Marker-Zeros-Counter Mode 27–6
Configuring Zero-Marker-Error Mode 27–6
Configuring Zero-Once Mode 27–6
Configuring Boundary Auto-Extend Mode 27–7
Configuring Boundary Capture Mode 27–7
Configuring Boundary Compare and Boundary Zero Modes 27–7
Configuring GP Counter Push-Button Operation 27–8
GP Counter Programming Concepts 27–8
CNT Input Noise Filtering 27–8
Capturing Counter Interval and CNT_CNTR Read Timing 27–9
Capturing Time Interval Between Successive Counter Events 27–10
GP Counter Event Control 27–11
Illegal Gray and Binary Code Events 27–11
Up/Down Count Events 27–11
Zero-Count Events 27–12
Overflow Events 27–12

Boundary Match Events 27–12
Zero Marker Events 27–12
ADSP-SC59x CNT Register Descriptions 27–12
Configuration Register 27–14
Command Register 27–17
Counter Register 27–20
Debounce Register 27–21
Interrupt Mask Register 27–23
Maximum Count Register 27–26
Minimum Count Register 27–27
Status Register 27–28

Media Local Bus (MLB)

Features 28–1
MLB Definitions 28–2
Clocking 28–3
Functional Description 28–3
ADSP-SC59x MLB Register List 28–4
ADSP-SC59x MLB Interrupt List 28–5
MediaLB Protocol 28–5
MLB Architectural Concepts 28–6
MediaLB Block Diagram 28–6
MediaLB Interface 28–7
Routing Fabric 28–8
Data Buffer RAM 28–8
Channel Table RAM 28–8
Address Mapping 28–8
Channel Allocation Table 28–9
Channel Set Up 28–10
Channel Descriptor Tables 28–11
AHB Descriptor Table (ADT) 28–15

Interrupt Interface Block 28–19
Operating Modes 28–20
Isochronous Data Exchange 28–20
Asynchronous and Control Data Exchange 28–21
Synchronous Data Exchange 28–22
Data Transfer 28–22
DMA 28–23
Programming Model 28–23
Channel Initialization 28–23
Configure the Hardware 28–23
Program the CAT and the CDT 28–23
Program the ADT 28–24
Service 28–25
Servicing the DMA Channel Interrupts 28–26
Servicing the MediaLB Status Interrupts 28–26
Polling for MediaLB System Commands 28–27
ADSP-SC59x MLB Register Descriptions 28–27
Peripheral Channel Mask 0 Register 28–29
Peripheral Channel Mask 1 Register 28–30
Peripheral Channel Status 0 Register 28–31
Peripheral Channel Status 1 Register 28–32
Bus Control Register 28–33
MediaLB Control 0 Register 28–35
Control 1 Register 28–37
MLB Global Control Register 28–38
HBI Channel Busy 0 Register 28–39
HBI Channel Busy 1 Register 28–40
HBI Channel Error 0 Register 28–41
HBI Channel Error 1 Register 28–42
HBI Channel Mask 0 Register 28–43

HBI Channel Mask 1 Register 28–44
HBI Control Register 28–45
Memory Interface Address Register 28–46
Memory Interface Control Register 28–47
Memory Interface Control Data 0 Register 28–48
Memory Interface Control Data 1 Register 28–49
Memory Interface Control Data 2 Register 28–50
Memory Interface Control Data 3 Register 28–51
Memory Interface Control Data Write Enable 0 Register 28–52
Memory Interface Control Data Write Enable 1 Register 28–53
Memory Interface Control Data Write Enable 2 Register 28–54
Memory Interface Control Data Write Enable 3 Register 28–55
Interrupt Enable Register 28–56
Channel Status 0 Register 28–59
Channel Status 1 Register 28–60
System Data Register 28–61
System Status Register 28–62
MediaLB 6-pin Control 0 Register 28–64

Two-Wire Interface (TWI)

TWI Features 29–1
TWI Functional Description 29–2
ADSP-SC59x TWI Register List 29–2
ADSP-SC59x TWI Interrupt List 29–3
TWI Block Diagram 29–3
External Interface 29–3
Serial Clock Signal (SCL) 29–4
Serial Data Signal (SDA) 29–4
Internal Interface 29–5
TWI Architectural Concepts 29–5
TWI Protocol 29–5

Clock Generation and Synchronization 29–6
Bus Arbitration 29–6
Start and Stop Conditions 29–7
General Call Support 29–7
Fast Mode 29–8
TWI Operating Modes 29–8
Repeated Start 29–8
Transmit Receive Repeated Start 29–8
Receive Transmit Repeated Start 29–9
Clock Stretching 29–10
Clock Stretching During FIFO Underflow 29–10
Clock Stretching During FIFO Overflow 29–11
Clock Stretching During Repeated Start 29–11
TWI Programming Model 29–12
General Setup 29–12
Completer Mode 29–13
Requester Mode Program Flow 29–14
Requester Mode Clock Setup 29–15
Requester Mode Transmit 29–15
Requester Mode Receive 29–16
ADSP-SC59x TWI Register Descriptions 29–17
SCL Clock Divider Register 29–18
Control Register 29–19
FIFO Control Register 29–21
FIFO Status Register 29–23
Interrupt Mask Register 29–24
Interrupt Status Register 29–26
Controller Mode Address Register 29–29
Controller Mode Registers 29–30
Controller Mode Status Register 29–33

Rx Data Double-Byte Register 29–36
Rx Data Single-Byte Register 29–37
Target Mode Address Register 29–38
Target Mode Control Register 29–39
Target Mode Status Register 29–41
Tx Data Double-Byte Register 29–42
Tx Data Single-Byte Register 29–43

Ethernet Media Access Controller (EMAC)

EMAC Features 30–1
EMAC Functional Description 30–7
ADSP-SC59x EMAC Register List 30–8
ADSP-SC59x EMAC Interrupt List 30–17
ADSP-SC59x EMAC Trigger List 30–18
EMAC Definitions 30–19
EMAC Block Diagram and Interfaces 30–22
EMAC CORE Sub Blocks 30–24
EMAC PHY Interface 30–26
RGMII Board Design Recommendations 30–27
Clock Sources 30–28
EMAC Architectural Concepts 30–29
EMAC System Crossbar Interface (EMAC SCB) 30–29
Priority of SCB Requests 30–31
SCB Interface Programming Options 30–31
Burst Reordering and Data Interleaving 30–32
Posted Writes 30–32
SCB Bus Transaction Status 30–33
DMA Controller (EMAC DMA) 30–33
DMA Bursts Using the SCB Interface 30–35
Application Data Buffer Alignment 30–35
Buffer Size Calculations 30–36

DMA Transmit Operation 30–36
DMA Receive Operation 30–41
DMA Error Response 30–42
DMA Descriptors 30–43
MAC Transaction Layer (MTL) 30–71
Transmit Path and Operation 30–71
Receive Path and Operation 30–73
EMAC CORE 30–74
EMAC CORE Transmission Engine 30–75
EMAC Core Reception 30–78
Source Address, VLAN, CRC Insertion/Replacement 30–81
Double VLAN Processing 30–81
Source Address and VLAN Insert, Replace, or Delete 30–83
Queue/Channel Based VLAN Tag Insertion on Transmit 30–84
CRC Insert and Replace 30–84
Packet Filtering 30–84
Source Address or Destination Address Filtering 30–85
VLAN Filtering 30–88
Layer 3 and Layer 4 Frame Filtering 30–94
EMAC Station Management Interface (SMI) 30–96
MDC Clock Frequency 30–97
MDIO Frame Structure 30–98
SMI Write Operation 30–99
SMI Read Operation 30–100
Preamble Suppression 30–100
Trailing Clocks and Back-to-Back Transactions 30–100
Interrupt for MDIO Transaction Completion 30–101
EMAC Management Counters (MMC) 30–101
MMC Receive Interrupt Register 30–103
MMC Transmit Interrupt Register 30–103
MMC Receive Checksum Offload Interrupt Register 30–103
EMAC Precision Time Protocol (PTP) Engine 30–103

IEEE1588 and the PTP Engine 30–103
Block Diagram 30–107
PTP Module Clock 30–108
Time Stamp Module 30–109
System Time 30–118
Target Time Trigger (Alarm) 30–121
Pulse-Per-Second (PPS) 30–121
PTP Interrupts 30–124
Media Clock Generation and Recovery (MCGR) 30–124
PTP Time Stamp Offload Function 30–128
One Step Time Stamp 30–133
Multiple Channels and Queues Support 30–140
Multiple Queues Support in the Transmit Path 30–140
Fixed Priority Scheme 30–140
Multiple Queues in the Receive Path 30–141
Assigning Tag Priorities to Transmit and Receive Queues 30–144
Distribution of Receive Packets from MAC to Receive Queues 30–145
Traffic Shaper 30–147
TCP/IP Offloading Features 30–157
Transmit Checksum Offload Engine 30–158
Receive Checksum Offload Engine 30–160
TCP/IP Segmentation Offload (TSO) Engine 30–165
UDP/IPv4 Fragment Offload (UFO) Engine 30–171
IPv4 ARP Offload Engine 30–174
Energy Efficient Ethernet (EEE) Mode 30–175
Transmit Path Functions 30–176
Receive Path Functions 30–177
LPI Timers 30–179
Low Power Interrupt (LPI) 30–179
Time-Sensitive Networking 30–180
Enhancements to Scheduled Traffic (EST) 30–180
Gate Control List (GCL) Implementation 30–182

Frame Preemption (FPE) 30–191
Time-Based Scheduling 30–203
Flow Control 30–209
Transmit Flow Control 30–209
Receive Flow Control 30–212
Loopback Mode 30–214
Guidelines for Using Loopback Mode 30–214
Enabling Loopback Mode 30–214
Endian Support 30–215
EMAC Event Control 30–215
EMAC Interrupt Signals 30–215
Interrupts from the MAC 30–216
Interrupts from the MTL 30–216
Interrupts from DMA 30–217
Periodic Scheduling of Transmit and Receive Interrupt 30–217
Per Channel Transfer Complete Interrupt 30–218
EMAC Triggers 30–219
EMAC Programming Model 30–219
Initializing DMA 30–220
Initializing the MTL Registers 30–221
Initializing the MAC 30–221
Performing Normal Transmit and Receive Operations 30–222
Stopping and Starting Transmission 30–223
Switching to New Descriptor List in Receive DMA 30–223
Multi-Channel Multi-Queuing 30–224
Recovering from DMA Channel Failure 30–225
Disabling the Receive Queue 30–226
IEEE 1588 Time Stamping (PTP) 30–226
Initialization Guidelines for System Time Generation 30–226
System Time Correction 30–227
Coarse Correction Method 30–227

Fine Correction Method 30–227
PTP Offloading 30–227
Automatic Periodic Generation of PTP Sync Messages 30–227
Periodic Generation of PTP Pdelay_Req Messages 30–228
Response Generation - Ordinary or Boundary Requester Mode 30–229
Response Generation - Ordinary or Boundary Completer Mode 30–229
Response Generation - Transparent Completer Mode 30–229
Response Generation - Transparent Requester Mode 30–230
Response Generation - Peer-to-Peer Transparent Mode 30–231
Using the Audio Video Feature 30–231
Initializing the DMA in Audio Video Feature 30–231
Enabling Slot Number Checking 30–232
Disabling Flow Control for AV Enabled Queues 30–233
Programming Energy Efficient Ethernet 30–233
Entering and Exiting the Transmit LPI Mode 30–233
Programming Flexible Pulse-Per-Second Output 30–234
Generating Single Pulse on PPS 30–234
Generating Next Pulse on PPS 30–234
Generating a Pulse Train on PPS 30–235
Programming TSO 30–235
Programming UFO 30–236
Programming Header Payload Split Receive 30–237
Programming VLAN Filtering on Receive 30–237
Programming Extended VLAN Filtering and Routing on Receive 30–238
Programming Queue/Channel Based VLAN Inclusion Registers 30–238
Writing to the Indirect Registers 30–238
Reading the Indirect Registers 30–239
Programming EST 30–239
GCL and GCL Linked Registers 30–239
EST Registers 30–240
Programming Launch Time in Time-Based Scheduling 30–241

Programming Media Clock Generation and Recovery 30–241
Media Clock Generation 30–241
Media Clock Recovery 30–241
EMAC Programming Concepts 30–242
IEEE 802.3 Ethernet Packet Structure 30–242
Frame Size Statistics for Application Software 30–243
Software Visualization of Programmable Packet Size 30–243
Ethernet Packet Structure in C 30–243
PTP Header Structure in C 30–244
ADSP-SC59x EMAC Register Descriptions 30–244
DMA Channel n Control Register 30–254
DMA Channel n Current Application Receive Buffer Address Register 30–256
DMA Channel n Current Application Receive Descriptor Register 30–257
DMA Channel n Current Application Transmit Buffer Address Register 30–258
DMA Channel n Current Application Transmit Descriptor Register 30–259
DMA Channel n Interrupt Enable Register 30–260
DMA Channel n Missed Frame Counter Register 30–264
DMA Channel n Receive Descriptor List Address Register 30–265
DMA Channel n Receive Descriptor Tail Pointer Register 30–266
DMA Channel n Receive Control Register 30–267
DMA Channel n Extended Receive Control Register 30–270
DMA Channel n Receive Interrupt Watchdog Timer Register 30–271
DMA Channel n Slot Function Control and Status register 30–273
DMA Channel n Status Register 30–275
DMA Channel n Transmit Descriptor List Address Register 30–282
DMA Channel n Transmit Descriptor Ring Length Register 30–283
DMA Channel n Transmit Descriptor Tail Pointer Register 30–284
DMA Channel n Transmit Control Register 30–285
DMA Debug Status 0 Register 30–289
DMA Debug Status 1 Register 30–293

DMA Debug Status 2 Register 30–297
DMA Interrupt Status Register 30–299
DMA Mode Register 30–302
DMA System Bus Mode Register 30–305
DMA TBS Control 0 Register 30–308
DMA TBS Control 1 Register 30–309
DMA TBS Control 2 Register 30–310
DMA TBS Control 3 Register 30–311
LPI One Microsecond Tic Counter Register 30–312
Address Resolution Protocol (ARP) Register 30–313
MAC Address 0 High Register 30–314
MAC Address 0 Low Register 30–315
MAC Address n High Register 30–316
MAC Address n Low Register 30–318
Auxiliary Time Stamp Control Register 30–319
Auxiliary Time Stamp Nanoseconds Register 30–321
Auxiliary Time Stamp Seconds Register 30–322
CSR Software Control Register 30–323
MAC Configuration Register 30–324
Debug Register 30–333
MAC Split Mode and Offset Configuration Register 30–335
MAC Extended Configuration Register 30–337
Frame Preemption Control and Status Register 30–341
Hardware Feature Presence Register 0 30–343
Hardware Feature Presence Register 1 30–347
Hardware Feature Presence Register 2 30–352
Hardware Feature Presence Register 3 30–355
Hash Table Register 0 30–359
Hash Table Register 1 30–360
Hash Table Register 2 30–361

Hash Table Register 3 30–362
Hash Table Register 4 30–363
Hash Table Register 5 30–364
Hash Table Register 6 30–365
Hash Table Register 7 30–366
Indirect Access Control Register 30–367
Indirect Access Data Register 30–369
Inner VLAN Tag Inclusion or Replacement Register 30–370
MAC Interrupt Enable Register 30–372
MAC Interrupt Status Register 30–374
Layer 3 and Layer 4 Control Register 30–379
Layer 3 and Layer 4 Control Register 30–384
LPI Control and Status Register 30–389
Transmit LPI Entry Timer Register 30–393
LPI Timers Control Register 30–394
Layer 3 Address 0 Register 7 30–395
Layer 3 Address 0 Register [i] 30–396
Layer 3 Address 1 Register 7 30–397
Layer 3 Address 1 Register [i] 30–398
Layer 3 Address 2 Register 7 30–399
Layer 3 Address 2 Register [i] 30–400
Layer 3 Address 3 Register 7 30–401
Layer 3 Address 3 Register [i] 30–402
Layer 4 Address Register [i] 30–403
Layer 4 Address Register [i] 30–404
PTP Log Message Interval Register 30–405
MDIO Address Register 30–407
MDIO Data Register 30–412
PHY Interface Control and Status Register 30–413
PPS n Interval Register 30–415

PPS n Target Time Nanoseconds Register 30–416
PPS n Target Time Seconds Register 30–417
PPS n Width Register 30–418
PPS Control Register 30–419
PTP Time Stamp Offload Control Register 30–425
MAC Packet Filter Register 30–428
Presentation Time Update Register 30–433
Presentation Time Nanosecond Register 30–434
Queue 0 Transmit Flow Control Register 30–435
Queue [i] Transmit Flow Control Register 30–438
Receive Queue Control 0 Register 30–441
Receive Queue Control 1 Register 30–444
Receive Queue Control 2 Register 30–448
Receive Queue Control 3 Register 30–450
Receive Queue Control 4 Register 30–452
Receive Flow Control Register 30–454
Receive Transmit Status Register 30–456
Source Port Identity 0 Register 30–459
Source Port Identity 1 Register 30–460
Source Port Identity 2 Register 30–461
Subsecond Increment Register 30–462
System Time Higher Word Seconds Register 30–463
System Time Nanoseconds Register 30–464
System Time Nanoseconds Update Register 30–465
System Time Seconds Register 30–467
System Time Seconds Update Register 30–468
Time Stamp Addend Register 30–469
Time Stamp Control Register 30–470
Time Stamp Egress Asymmetry Correction Register 30–477
Time Stamp Egress Correction Nanosecond Register 30–478

Time Stamp Egress Correction Sub Nanosecond Register 30–479
Time Stamp Egress Latency Register 30–480
Time Stamp Ingress Asymmetry Correction Register 30–481
Time Stamp Ingress Correction Nanosecond Register 30–482
Time Stamp Ingress Correction Sub Nanosecond Register 30–483
Time Stamp Ingress Latency Register 30–484
Time Stamp Status Register 30–485
Transmit Queue Priority Mapping 0 Register 30–491
Transmit Queue Priority Mapping 1 Register 30–492
Transmit Time Stamp Status Nanoseconds Register 30–493
Transmit Time Stamp Status Seconds Register 30–494
VLAN Hash Table Register 30–495
VLAN Tag Inclusion or Replacement Register 30–496
VLAN Tag Control Register 30–499
VLAN Tag Indirect Access Data Register 30–504
Watchdog Timeout Register 30–507
MMC Control Register 30–509
MMC Receive FPE Interrupt Status Register 30–512
MMC Receive FPE Interrupt Mask Register 30–514
MMC Transmit FPE Interrupt Status Register 30–516
MMC Transmit FPE Interrupt Mask Register 30–518
MMC Receive IPC Interrupt Status Register 30–519
MMC Receive IPC Interrupt Mask Register 30–528
MMC Receive FPE Fragment Counter Register 30–537
MMC Receive Interrupt Status Register 30–538
MMC Receive Interrupt Mask Register 30–547
MMC Receive FPE Packet Assembly Error Counter Register 30–556
MMC Receive FPE Packet Assembly Okay Counter Register 30–557
MMC Receive FPE Packet SMD Error Counter Register 30–558
MMC Transmit FPE Packet Fragment Counter Register 30–559

MMC Transmit FPE Hold Request Counter Register 30–560
MMC Transmit Interrupt Status Register 30–561
MMC Transmit Interrupt Mask Register 30–570
EST Control Register 30–579
EST Extended Control Register 30–582
EST Frame Size Capture Register 30–583
EST Frame Size Error Register 30–584
EST Gate Control List Control Register 30–585
EST Gate Control List Data Register 30–588
EST Interrupt Enable Register 30–589
EST Scheduling Error Register 30–591
EST Status Register 30–592
Frame Preemption Advance Register 30–595
MTL Frame Preemption Control and Status Register 30–596
MTL Interrupt Status Register 30–597
MTL Operation Mode Register 30–600
Queue 0 Interrupt Control and Status Register 30–602
Queue n Interrupt Control and Status Register 30–604
Receive Queue Control Register 30–606
Receive Queue 0 Debug Register 30–608
Receive Queue 0 Missed Packet and Overflow Counter Register 30–610
Receive Queue 0 Operation Mode Register 30–612
Receive Queue n Control Register 30–616
Receive Queue n Debug Register 30–618
Receive Queue n Missed Packet and Overflow Counter Register 30–620
Receive Queue n Operation Mode Register 30–622
Receive Queue to DMA Channel Mapping Register 0 30–626
Receive Queue to DMA Channel Mapping Register 1 30–630
MTL Time Based Scheduling Control Register 30–634
Transmit Queue 0 Debug Register 30–636

Transmit Queue 0 ETS Status Register 30–638
Transmit Queue 0 Operation Mode Register 30–639
Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register 30–641
Transmit Queue 0 Underflow Packet Counter Register 30–642
Transmit Queue n Debug Register 30–643
Transmit Queue n ETS Control Register 30–645
Transmit Queue n ETS Status Register 30–647
Transmit Queue n CBS HiCredit Register 30–648
Transmit Queue n CBS LoCredit Register 30–649
Transmit Queue n Operation Mode Register 30–650
Transmit Queue n IdleSlopeCredit, Quantum or Weights Register 30–652
Transmit Queue n SendSlopeCredit Register 30–654
Transmit Queue n Underflow Packet Counter Register 30–655
Receive ICMP Error Octets Counter Register 30–656
Receive ICMP Error Packets Counter Register 30–657
Receive ICMP Good Octets Counter Register 30–658
Receive ICMP Good Packets Counter Register 30–659
Receive IPv4 Fragmented Octets Counter Register 30–660
Receive IPv4 Fragmented Packets Counter Register 30–661
Receive IPv4 Good Octets Counter Register 30–662
Receive IPv4 Good Packets Counter Register 30–663
Receive IPv4 Header Error Octets Counter Register 30–664
Receive IPv4 Header Error Packets Counter Register 30–665
Receive IPv4 Payload Octets Counter Register 30–666
Receive IPv4 Payload Packets Counter Register 30–667
Receive IPv4 UDP Checksum Disable Octets Counter Register 30–668
Receive IPv4 UDP Checksum Disable Packets Counter Register 30–669
Receive IPv6 Good Octets Counter Register 30–670
Receive IPv6 Good Packets Counter Register 30–671
Receive IPv6 Header Error Octets Counter Register 30–672

Receive IPv6 Header Error Packets Counter Register 30–673
Receive IPv6 Payload Octets Counter Register 30–674
Receive IPv6 Payload Packets Counter Register 30–675
Receive TCP Error Octets Counter Register 30–676
Receive TCP Error Packets Counter Register 30–677
Receive TCP Good Octets Counter Register 30–678
Receive TCP Good Packets Counter Register 30–679
Receive UDP Error Octets Counter Register 30–680
Receive UDP Error Packets Counter Register 30–681
Receive UDP Good Octets Counter Register 30–682
Receive UDP Good Packets Counter Register 30–683
Receive 1024 - Maximum Octets Packets Good Bad Counter Register 30–684
Receive 128 - 255 Octets Packets Good Bad Counter Register 30–685
Receive 256 - 511 Octets Packets Good Bad Counter Register 30–686
Receive 512 - 1023 Octets Packets Good Bad Counter Register 30–687
Receive 64 Octets Packets Good Bad Counter Register 30–688
Receive 65 - 127 Octets Packets Good Bad Counter Register 30–689
Receive Alignment Error Packets Counter Register 30–690
Receive Broadcast Packets Good Counter Register 30–691
Receive CRC Error Packets Counter Register 30–692
Receive Control Packets Good Counter Register 30–693
Receive FIFO Overflow Packets Counter Register 30–694
Receive Jabber Error Packets Counter Register 30–695
Receive LPI Transition Counter Register 30–696
Receive LPI Microseconds Counter Register 30–697
Receive Length Error Packets Counter Register 30–698
Receive Multicast Packets Good Counter Register 30–699
Receive Octet Good Counter Register 30–700
Receive Octet Good Bad Counter Register 30–701
Receive Out of Range Type Packets Counter Register 30–702

Receive Oversize Packets Good Counter Register 30–703
Receive Packets Good Bad Counter Register 30–704
Receive Pause Packets Counter Register 30–705
Receive Error Packets Counter Register 30–706
Receive Runt Error Packets Counter Register 30–707
Receive Undersize Packets Good Counter Register 30–708
Receive Unicast Packets Good Counter Register 30–709
Receive VLAN Packets Good Bad Counter Register 30–710
Receive Watchdog Error Packets Counter Register 30–711
Transmit 1024-Max Octets Packets Good Bad Counter Register 30–712
Transmit 128-255 Octets Packets Good Bad Counter Register 30–713
Transmit 256-511 Octets Packets Good Bad Counter Register 30–714
Transmit 512-1023 Octets Packets Good Bad Counter Register 30–715
Transmit 64 Octets Packets Good Bad Counter Register 30–716
Transmit 65-127 Octets Packets Good Bad Counter Register 30–717
Transmit Broadcast Packets Good Counter Register 30–718
Transmit Broadcast Packets Good Bad Counter Register 30–719
Transmit Carrier Error Packets Counter Register 30–720
Transmit Deferred Packets Counter Register 30–721
Transmit Excessive Collision Packets Counter Register 30–722
Transmit Excessive Deferral Error Counter Register 30–723
Transmit LPI Transition Counter Register 30–724
Transmit LPI Microseconds Counter Register 30–725
Transmit Late Collision Packets Counter Register 30–726
Transmit Multicast Packets Good Counter Register 30–727
Transmit Multicast Packets Good Bad Counter Register 30–728
Transmit Multiple Collision Packets Good Counter Register 30–729
Transmit Osize Packets Good Counter Register 30–730
Transmit Octet Good Counter Register 30–731
Transmit Octet Good Bad Counter Register 30–732

Transmit Packet Good Counter Register 30–733
Transmit Packet Good Bad Counter Register 30–734
Transmit Pause Packets Counter Register 30–735
Transmit Single Collision Packets Good Counter Register 30–736
Transmit Underflow Error Packets Counter Register 30–737
Transmit Unicast Packets Good Bad Counter Register 30–738
Transmit VLAN Packets Good Counter Register 30–739

Digital Audio Interface (DAI)

SRU Features 31–1
Functional Description 31–1
ADSP-SC59x DAI Register List 31–2
ADSP-SC59x DAI Interrupt List 31–5
ADSP-SC59x DAI Trigger List 31–5
DAI Block Diagram 31–6
DAI Signal Naming Conventions 31–6
I/O Pin Buffers 31–7
Pin Buffer Signals 31–7
Pin Buffer Input Signal 31–7
Pin Buffer Enable Signal 31–7
Loopback Pin Buffers 31–8
Pin Buffer Functions 31–8
Pin Buffers as Signal Input 31–8
Pin Buffers As Signal Output 31–9
DAI Pin Buffer Status 31–9
DAIn Peripherals 31–10
Output Signals With Pin Buffer Enable Control 31–10
Output Signals Without Pin Buffer Enable Control 31–10
Signal Routing Units (SRUs) 31–10
Signal Routing Matrix by Groups 31–10
DAI Group Routing 31–11

Rules for SRU Connections 31–12
Miscellaneous Buffers and Functions 31–13
DAI Routing Unit (DRU) 31–13
DAI Routing Capabilities 31–14
DAI Default Routing 31–17
Unused DAI Connections 31–18
DAI Operating Modes 31–18
DAI Pin Buffer Polarity 31–18
DAI Miscellaneous Buffer Polarity 31–19
DAI System Interrupt Controller (SIC) 31–19
Signal Routing Unit Effect Latency 31–21
DAI Programming Model 31–21
Debug Features 31–22
DAI Sources Overview 31–22
Group A – Clock Routing Signals 31–23
Group B – Serial Data Source Signals 31–27
Group C – Frame Sync Source Signals 31–30
Group D – Pin Signal Assignment Source Signals 31–33
Group E – Miscellaneous Source Signals 31–41
Group F – Pin Output Enable Source Signals 31–44
DAI Destination Registers Overview 31–48
SPORT Grouping with DAI 31–56
Global PCG Enable 31–56
ADSP-SC59x DAI Register Descriptions 31–56
Clock Routing Control Register 0 31–59
Clock Routing Control Register 1 31–61
Clock Routing Control Register 2 31–63
Clock Routing Control Register 3 31–64
Clock Routing Control Register 4 31–65
Clock Routing Control Register 5 31–66

Serial Data Routing Control Register 0 ..... **31–68**
Serial Data Routing Control Register 1 ..... **31–69**
Serial Data Routing Control Register 2 ..... **31–70**
Serial Data Routing Control Register 3 ..... **31–71**
Serial Data Routing Control Register 4 ..... **31–72**
Serial Data Routing Control Register 5 ..... **31–73**
Serial Data Routing Control Register 6 ..... **31–74**
Extended Clock Routing Control Register 0 ..... **31–75**
Extended Clock Routing Control Register 1 ..... **31–77**
Extended Clock Routing Control Register 2 ..... **31–79**
Extended Clock Routing Control Register 3 ..... **31–80**
Extended Clock Routing Control Register 4 ..... **31–81**
Extended Clock Routing Control Register 5 ..... **31–82**
Extended Serial Data Routing Control Register 0 ..... **31–84**
Extended Serial Data Routing Control Register 1 ..... **31–85**
Extended Serial Data Routing Control Register 2 ..... **31–86**
Extended Serial Data Routing Control Register 3 ..... **31–87**
Extended Serial Data Routing Control Register 4 ..... **31–88**
Extended Serial Data Routing Control Register 5 ..... **31–89**
Extended Serial Data Routing Control Register 6 ..... **31–90**
Extended Frame Sync Routing Control Register 0 ..... **31–91**
Extended Frame Sync Routing Control Register 1 ..... **31–93**
Extended Frame Sync Routing Control Register 2 ..... **31–95**
Extended Frame Sync Routing Control Register 4 ..... **31–96**
Extended Miscellaneous Control Register 0 ..... **31–97**
Extended Miscellaneous Control Register 1 ..... **31–99**
Extended Miscellaneous Control Register 2 ..... **31–100**
Extended Pin Buffer Enable Register 0 ..... **31–101**
Extended Pin Buffer Enable Register 1 ..... **31–102**
Extended Pin Buffer Enable Register 2 ..... **31–103**

Extended Pin Buffer Enable Register 3 31–104
Extended Pin Buffer Assignment Register 0 31–105
Extended Pin Buffer Assignment Register 1 31–106
Extended Pin Buffer Assignment Register 2 31–107
Extended Pin Buffer Assignment Register 3 31–108
Extended Pin Buffer Assignment Register 4 31–109
Frame Sync Routing Control Register 0 31–110
Frame Sync Routing Control Register 1 31–112
Frame Sync Routing Control Register 2 31–114
Frame Sync Routing Control Register 4 31–115
Global SPORT Interrupt Grouping Register 31–116
Global PCG Enable Control Register 31–119
Global SPORT Enable Register 31–122
Falling-Edge Interrupt Mask Register 31–125
Core Interrupt Priority Assignment Register 31–128
Rising-Edge Interrupt Mask Register 31–131
High Priority Interrupt Latch Register 31–134
Shadow High Priority Interrupt Latch Register 31–137
Low Priority Interrupt Latch Register 31–141
Shadow Low Priority Interrupt Latch Register 31–144
Miscellaneous Control Register 0 31–148
Miscellaneous Control Register 1 31–150
Miscellaneous Control Register 1 31–152
Pin Buffer Enable Register 0 31–153
Pin Buffer Enable Register 1 31–154
Pin Buffer Enable Register 2 31–155
Pin Buffer Enable Register 3 31–156
Pin Buffer Assignment Register 0 31–157
Pin Buffer Assignment Register 1 31–158
Pin Buffer Assignment Register 2 31–159

Pin Buffer Assignment Register 3 31–160
Pin Buffer Assignment Register 4 31–161
Pin Status Register 31–163

Serial Port (SPORT)

Features 32–1
Signal Descriptions 32–3
SRU Programming 32–6
Functional Description 32–6
ADSP-SC59x SPORT Register List 32–6
ADSP-SC59x SPORT Interrupt List 32–7
ADSP-SC59x SPORT Trigger List 32–9
ADSP-SC59x SPORT DMA Channel List 32–10
Block Diagram 32–11
Architectural Concepts 32–12
Multiplexer Logic 32–13
Data Types and Companding 32–16
Companding as a Function 32–17
Transmit Path 32–17
Receive Path 32–19
Operating Modes and Options 32–19
Serial Word Length 32–21
Clock Sample and Drive Edges 32–21
Frame Sync Options 32–23
Data-Dependent versus Data-Independent Frame Syncs 32–23
Support for Edge-Detected and Level-Sensitive Frame Syncs 32–23
Early versus Late Frame Syncs 32–24
Framed versus Unframed Frame Syncs 32–25
Frame Sync Polarity 32–26
Premature Frame Sync Error Detection 32–26
Mode Selection 32–27

Standard DSP Serial Mode 32–27
Stereo Modes 32–28
I²S Mode 32–29
Left-Justified Mode 32–30
Right-Justified Mode 32–31
Multichannel (TDM) Mode 32–33
Packed I²S Mode 32–36
Gated Clock Mode 32–37
Data Transfers and Interrupts 32–38
Data Buffers 32–39
Data Buffer Status 32–40
Single-Word (Core) Transfers 32–41
DMA Transfers 32–41
Data Transfer Interrupt 32–42
Error Detection (Status) Interrupt 32–43
Grouping of SPORTs 32–45
SPORT Programming Model 32–46
Initializing Core-Driven (Non-MCM) Transfers 32–47
Initializing Multichannel Transfers 32–48
Using DMA for SPORT Transfers 32–49
Using Companding as a Function 32–49
Programming Global SPORT Groups 32–50
Disabling Global SPORT Groups 32–51
ADSP-SC59x SPORT Register Descriptions 32–51
Half SPORT 'A' Multichannel 0-31 Select Register 32–53
Half SPORT 'B' Multichannel 0-31 Select Register 32–54
Half SPORT 'A' Multichannel 32-63 Select Register 32–55
Half SPORT 'B' Multichannel 32-63 Select Register 32–56
Half SPORT 'A' Multichannel 64-95 Select Register 32–57
Half SPORT 'B' Multichannel 64-95 Select Register 32–58

Half SPORT 'A' Multichannel 96-127 Select Register 32–59
Half SPORT 'B' Multichannel 96-127 Select Register 32–60
Half SPORT 'A' Control 2 Register 32–61
Half SPORT 'B' Control 2 Register 32–62
Half SPORT 'A' Control Register 32–63
Half SPORT 'B' Control Register 32–71
Half SPORT 'A' Divisor Register 32–80
Half SPORT 'B' Divisor Register 32–81
Half SPORT 'A' Error Register 32–82
Half SPORT 'B' Error Register 32–84
Half SPORT 'A' Multichannel Control Register 32–86
Half SPORT 'B' Multichannel Control Register 32–88
Half SPORT 'A' Multichannel Status Register 32–90
Half SPORT 'B' Multichannel Status Register 32–91
Half SPORT 'A' Rx Buffer (Primary) Register 32–92
Half SPORT 'B' Rx Buffer (Primary) Register 32–93
Half SPORT 'A' Rx Buffer (Secondary) Register 32–94
Half SPORT 'B' Rx Buffer (Secondary) Register 32–95
Half SPORT 'A' Tx Buffer (Primary) Register 32–96
Half SPORT 'B' Tx Buffer (Primary) Register 32–97
Half SPORT 'A' Tx Buffer (Secondary) Register 32–98
Half SPORT 'B' Tx Buffer (Secondary) Register 32–99

Precision Clock Generators (PCG)

Features 33–1
Functional Description 33–2
ADSP-SC59x PCG Register List 33–2
ADSP-SC59x PCG Trigger List 33–3
Internal Interface 33–4
Serial Clock 33–4

Frame Sync 33–4
Frame Sync Output 33–4
Divider Mode Selection 33–5
Phase Shift 33–5
Pulse Width 33–6
Default Pulse Width 33–6
Input Clock Source Considerations 33–7
Timing Example for I²S Mode 33–7
Cross Mode Connections 33–7
Operating Modes 33–7
Normal Mode 33–7
Bypass Mode 33–8
One-Shot Mode 33–8
Audio System Example 33–9
Hardware Trigger Control 33–11
Clock Configuration Examples 33–11
Global PCG Enable 33–12
PCG Event Control 33–12
External Event Trigger 33–12
External Event Trigger Delay 33–13
Programming Model 33–13
Frame Sync Phase Setting 33–14
External Event Trigger 33–14
Debug Features 33–14
ADSP-SC59x PCG Register Descriptions 33–14
Precision Clock A Control 0 Register 33–16
Precision Clock A Control 1 Register 33–17
Precision Clock B Control 0 Register 33–18
Precision Clock B Control 1 Register 33–19
Precision Clock C Control 0 Register 33–20

Precision Clock C Control 1 Register 33–21
Precision Clock D Control 0 Register 33–22
Precision Clock D Control 1 Register 33–23
Precision Clock E Control 0 Register 33–24
Precision Clock E Control 1 Register 33–25
Precision Clock F Control 0 Register 33–26
Precision Clock F Control 1 Register 33–27
Precision Clock G Control 0 Register 33–28
Precision Clock G Control 1 Register 33–29
Precision Clock H Control 0 Register 33–30
Precision Clock H Control 1 Register 33–31
Precision Clock Pulse Width Control 1 Register 33–32
Precision Clock Pulse Width Control 2 Register 33–34
Precision Clock Pulse Width Control 3 Register 33–36
Precision Clock Pulse Width Control 4 Register 33–38
Precision Clock Frame Sync Synchronization 1 Register 33–40
Precision Clock Frame Sync Synchronization 2 Register 33–43
Precision Clock Frame Sync Synchronization 3 Register 33–46
Precision Clock Frame Sync Synchronization 4 Register 33–49

Asynchronous Sample Rate Converter (ASRC)

Features 34–1
Functional Description 34–2
ADSP-SC59x ASRC Register List 34–2
ASRC Interrupt List 34–2
ASRC Block Diagram 34–2
SRU Programming 34–3
Clocking 34–3
I/O Ports 34–4
De-Emphasis Filter 34–4

Mute Control 34–4
SRC Core 34–4
RAM FIFO 34–4
Digital Servo Loop 34–5
FIR Filter 34–5
Sample Rate Sensing 34–5
Digital Filter Group Delay 34–5
Data Format 34–6
Operating Modes 34–6
TDM Input Mode 34–6
TDM Output Mode 34–7
Matched-Phase Mode 34–7
Bypass Mode 34–8
De-Emphasis Mode 34–9
Dithering Mode 34–9
Muting Modes 34–9
Soft Mute 34–9
Hard Mute 34–10
Auto Mute 34–10
Interrupts 34–10
Sources 34–10
SRC Mute Out 34–10
Masking 34–10
Service 34–10
Programming Model 34–11
Debug Features 34–11
ADSP-SC59x ASRC Register Descriptions 34–11
Control Register for ASRC 0 and 1 34–12
Control Register for ASRC 2 and 3 34–17
Mute Register 34–22

Ratio Register for ASRC 0 and 1 34–23
Ratio Register for ASRC 2 and 3 34–25

Sony/Philips Digital Interface (S/PDIF)

Features 35–1
ADSP-SC59x SPDIF Register List 35–1
SRU Programming 35–3
S/PDIF Interrupt List 35–3
Clocking 35–3
S/PDIF Transmitter 35–3
Functional Description 35–3
Input Data Formats 35–5
Operating Modes 35–6
Full Serial Mode 35–6
Standalone Mode 35–6
Data Output Mode 35–7
S/PDIF Receiver 35–7
Functional Description 35–8
Clock Recovery 35–8
Output Data Format 35–9
Channel Status 35–9
Operating Modes 35–9
Compressed or Non-linear Audio Data 35–9
Emphasized Audio Data 35–10
Single-Channel Double-Frequency Mode 35–10
Clock Recovery Modes 35–10
Interrupts 35–11
Sources 35–11
Transmit Block Start 35–11
Receiver Status 35–11
Receiver Error 35–11

Masking 35–11
Service 35–11
Programming Model 35–11
Programming the Transmitter 35–11
Programming the Receiver 35–12
Interrupted Data Streams on the Receiver 35–12
Debug Features 35–13
Loopback Routing 35–13
ADSP-SC59x SPDIF Register Descriptions 35–13
Receive Control 35–15
Receive Status Register 35–17
Receive Status A0 Register 35–20
Receive Status B0 Register 35–21
Receive Status A1 Register 35–22
Receive Status B1 Register 35–23
Transmit Control Register 35–24
Transmit Status A0 Register 35–27
Transmit Status A1 Register 35–28
Transmit Status A2 Register 35–29
Transmit Status A3 Register 35–30
Transmit Status A4 Register 35–31
Transmit Status A5 Register 35–32
Transmit Status B0 Register 35–33
Transmit Status B1 Register 35–34
Transmit Status B2 Register 35–35
Transmit Status B3 Register 35–36
Transmit Status B4 Register 35–37
Transmit Status B5 Register 35–38
Transmit User Buffer A0 Register 35–39
Transmit User Buffer A1 Register 35–40

Transmit User Buffer A2 Register 35–41
Transmit User Buffer A3 Register 35–42
Transmit User Buffer A4 Register 35–43
Transmit User Buffer A5 Register 35–44
Transmit User Buffer B0 Register 35–45
Transmit User Buffer B1 Register 35–46
Transmit User Buffer B2 Register 35–47
Transmit User Buffer B3 Register 35–48
Transmit User Buffer B4 Register 35–49
Transmit User Buffer B5 Register 35–50
User Bit Update Register 35–51

Direct Memory Access (DMA)

DMA Channel Features 36–1
DMA Channel Functional Description 36–3
ADSP-SC59x DMA Register List 36–3
ADSP-SC59x DMA Channel List 36–3
DMA Definitions 36–5
Block Diagram 36–7
Architectural Concepts 36–9
DMA Channel SCB Interface 36–9
SCB Interface Signals 36–9
SCB Burst Transfers 36–9
Data Address Alignment 36–10
Descriptor Set Address Alignment 36–11
Peripheral Control Commands 36–11
Idle Command 36–11
Request-Data Command 36–12
Request-Data Urgent Command 36–12
Peripheral-Control Command Restrictions 36–12
DMA Channel Peripheral DMA Bus 36–12

Memory DMA and Triggering 36–12
Medium Band Width DMA Channel MMR Access Bus 36–15
DMA Channel Operation Flow 36–15
Startup Flow 36–15
Refresh Flow 36–17
Work Unit Transition Flow 36–18
Transfer Termination and Shutdown Flow 36–20
DMA Channel Errors 36–22
Status and Debug Errors 36–22
DMA Configuration Register Errors 36–22
Illegal Register Write During Run 36–23
Address Alignment Error 36–23
Memory Access Error 36–23
Trigger Overrun Error 36–24
Bandwidth-Monitor Error 36–24
Control Interface Error 36–24
DMA Operating Modes 36–24
Register-Based Flow Modes 36–24
Stop Mode 36–25
Autobuffer Mode 36–25
Descriptor-Based Flow Modes 36–25
Descriptor-Array Mode 36–26
Descriptor-List Mode 36–26
Descriptor-On-Demand Modes 36–27
Data Transfer Modes 36–28
Two-Dimensional DMA 36–28
DMA Channel Event Control 36–29
Event Signals 36–30
Work Unit State Events 36–31
Peripheral Interrupt Request Events 36–31
Peripheral Data Request Events 36–31

DMA Channel Triggers 36–32
Issuing Triggers 36–32
Waiting For Triggers 36–32
DMA Channel Programming Model 36–33
Mode Configuration 36–34
Register-Based Linear-Buffer Stop Flow Mode 36–34
Register-Based Autobuffer Flow Mode 36–35
Descriptor-Array Flow Mode 36–36
Descriptor-List Flow Mode 36–37
Register-Based Memory-to-Memory Transfer in Stop Flow Mode 36–38
Programming Concepts 36–39
Synchronization of Software and DMA 36–40
Interrupt and Trigger Event-Based Synchronization 36–40
Register Polling Based Synchronization 36–40
Descriptor Queues 36–41
Queues Using Event Generation for Every Descriptor Set 36–41
Queues Using Minimal Events 36–42
ADSP-SC59x DMA Register Descriptions 36–43
Start Address of Current Buffer Register 36–45
Current Address Register 36–46
Bandwidth Limit Count Register 36–47
Bandwidth Limit Count Current Register 36–48
Bandwidth Monitor Count Register 36–49
Bandwidth Monitor Count Current Register 36–50
Configuration Register 36–51
Current Descriptor Pointer Register 36–59
Pointer to Next Initial Descriptor Register 36–60
Previous Initial Descriptor Pointer Register 36–61
Status Register 36–62
Inner Loop Count Start Value Register 36–65
Current Count (1D) or Intra-row XCNT (2D) Register 36–66

Inner Loop Address Increment Register 36–67
Outer Loop Count Start Value (2D only) Register 36–68
Current Row Count (2D only) Register 36–69
Outer Loop Address Increment (2D only) Register 36–70

Extended Memory DMA (EMDMA)

EMDMA Features 37–1
EMDMA Functional Description 37–1
ADSP-SC59x EMDMA Register List 37–2
ADSP-SC59x EMDMA Interrupt List 37–2
ADSP-SC59x EMDMA Trigger List 37–3
DMA Addressing 37–3
DMA Burst Transfers 37–3
Transfer Control Block (TCB) Memory Storage 37–3
Chain Assignment 37–4
Starting Chain Loading 37–4
Buffered Chain Loading Register 37–4
TCB Storage 37–4
EMDMA Operating Modes 37–7
Standard DMA 37–8
Circular Buffered DMA 37–8
Chained DMA Mode 37–9
Data Direction On-the-Fly 37–10
Write Back Circular Index Pointer 37–10
Scatter/Gather DMA 37–10
Pre Modified Read/Write Index 37–13
Delay Line DMA 37–13
ADSP-SC59x EMDMA Register Descriptions 37–16
External Base Address Register 37–17
Circular Buffer Length Register 37–18

Chain Pointer Register 37–19
Internal Count Register 37–20
External Count Register 37–21
External Memory DMA Control Register 37–22
Internal Index Register 37–26
External Index Register 37–27
Internal Modifier Register 37–28
External Modifier Register 37–29
Delay Line Tap Count Register 37–30
Tap List Pointer Register 37–31

Cyclic Redundancy Check (CRC)

CRC Features 38–1
CRC Functional Description 38–2
ADSP-SC59x CRC Register List 38–2
ADSP-SC59x CRC Interrupt List 38–3
CRC Definitions 38–3
CRC Block Diagram 38–4
Peripheral DMA Bus 38–5
MMR Access Bus 38–5
Mirror Block 38–6
Data FIFO 38–6
DMA Request Generator 38–6
CRC Engine 38–6
Compare Logic 38–6
CRC Architectural Concepts 38–6
Look-up Table 38–7
Data Mirroring 38–7
FIFO Status and Data Requests 38–8
CRC Operating Modes 38–9

Data Transfer Modes 38–9
Memory Scan Compute-and-Compare Mode 38–10
Memory Scan Data Verify 38–10
Memory Transfer Compute-and-Compare Mode 38–10
Memory Transfer Data Fill Mode 38–11
CRC Event Control 38–11
Interrupt Signals 38–11
CRC Programming Model 38–12
CRC Mode Configuration 38–12
Look-up Table Generation 38–12
Core Driven Memory Scan Compute-and-Compare Mode 38–13
DMA Driven Memory Scan Compute-and-Compare Mode 38–14
Core Driven Memory Scan Data Verify Mode 38–16
DMA Driven Memory Scan Data Verify Mode 38–17
Core Driven Memory Transfer Compute-and-Compare Mode 38–19
DMA Driven Memory Transfer Compute-and-Compare Mode 38–20
DMA Driven Memory Transfer Data Fill Mode 38–22
ADSP-SC59x CRC Register Descriptions 38–23
Data Compare Register 38–25
Control Register 38–26
Data Word Count Register 38–29
Data Count Capture Register 38–30
Data Word Count Reload Register 38–31
Data FIFO Register 38–32
Fill Value Register 38–33
Interrupt Enable Register 38–34
Interrupt Enable Clear Register 38–35
Interrupt Enable Set Register 38–36
Polynomial Register 38–37
CRC Current Result Register 38–38
CRC Final Result Register 38–39

Status Register 38–40

System Security

Security Features 39–1
Security Functional Description 39–2
Security Mode Configuration 39–4
Status and Error Signals 39–4

System Protection Unit (SPU)

SPU Features 40–1
SPU Functional Description 40–1
ADSP-SC59x SPU Register List 40–1
ADSP-SC59x SPU Interrupt List 40–2
Peripheral Register Write Protection 40–2
SPU Block Diagram 40–5
SPU Architectural Concepts 40–5
SPU Event Control 40–5
SPU Programming Model 40–6
SPU Mode Configuration 40–7
Locking Write-Protect Registers 40–7
Protecting a Peripheral 40–7
Configuring Security Privileges of a Peripheral 40–7
ADSP-SC59x SPU Register Descriptions 40–8
Control Register 40–9
Secure Check Register 40–10
Secure Control Register 40–11
Secure Core Registers 40–13
Secure Peripheral Register 40–14
Status Register 40–15
Write Protect Register n 40–16
Write-Protect, Secure Peripheral, and Secure Core Registers 40–16

ADSP-SC59x Processor Specific Information 40–24

Security Packet Engine (PKTE)

PKTE Features 41–1
PKTE Functional Description 41–1
ADSP-SC59x PKTE Register List 41–1
ADSP-SC59x PKTE Interrupt List 41–3
PKTE Definitions 41–4
Cipher Module 41–5
Hash Module 41–5
Pseudo-Random Number Generator 41–6
Packet Engine Processing Details 41–8
Crypto Padding 41–8
Pad Generation and Insertion 41–9
Pad Types 41–9
Pad Length 41–10
Pad Verification and Consumption 41–12
Crypto and Hash Algorithms 41–14
IV Processing 41–17
ARC4 Processing 41–18
Hash State Loading 41–19
Sequence Number Processing 41–19
Sequence Number Processing in Extended SSL/TLS 41–19
Sequence Number Processing in DTLS 41–20
PKTE Block Diagram 41–23
PKTE Architectural Concepts 41–24
Packet Engine 41–24
Input/Output FIFO Buffers 41–24
Parallel Operations 41–24
DMA Controller 41–24
Interrupt Controller 41–25
Clock Controller 41–25

PKTE Operating Modes 41–25
Autonomous Ring Mode (ARM) 41–26
Target Command Mode (TCM) 41–27
Direct Host Mode (DHM) 41–27
PKTE Event Control 41–27
PKTE Interrupt Signals 41–27
PKTE Programming Model 41–30
PKTE Mode Configuration 41–33
PKTE Programming Concepts 41–33
Packet Engine Descriptor 41–33
Descriptor Processing 41–35
Descriptor Ownership 41–36
SA Record and State Record Structure 41–37
SA Record Structure 41–37
SA State Structure 41–40
ARC4 State Structure 41–40
Configuring Operations in the PKTE 41–41
Basic Operations and Decoding 41–41
Error Code Description 41–42
Extended Error Codes 41–43
Number Format 41–45
PKTE Programming Examples 41–46
Calculating SHA in Direct Host Mode 41–46
Performing AES Decryption in Direct Host Mode 41–47
ADSP-SC59x PKTE Register Descriptions 41–48
Packet Engine ARC4 State Record Address 41–51
Starting Entry of 256-byte ARC4 State Buffer 41–52
Packet Engine Buffer Pointer Register 41–53
Packet Engine Buffer Threshold Register 41–54
Packet Engine Command Descriptor Ring Base Address 41–56
Packet Engine Command Descriptor Count Register 41–57

Packet Engine Command Descriptor Count Increment Register 41–58
Packet Engine Configuration Register 41–59
PE Clock Control Register 41–62
PKTE Continue Register 41–64
Packet Engine Control Register 41–65
Starting Entry of 256-byte Data Input/Output Buffer 41–70
Packet Engine Destination Address 41–71
Packet Engine DMA Configuration Register 41–72
Packet Engine Endian Configuration Register 41–74
Packet Engine Halt Control Register 41–76
Packet Engine Halt Status Register 41–78
Interrupt Mask Disable Register 41–81
Interrupt Mask Enable Register 41–83
Interrupt Masked Status Register 41–85
Packet Engine Input Buffer Count Register 41–87
Packet Engine Input Buffer Count Increment Register 41–88
Interrupt Configuration Register 41–89
Interrupt Clear Register 41–90
Interrupt Enable Register 41–92
Interrupt Unmasked Status Register 41–94
Packet Engine Length Register 41–96
Packet Engine Output Buffer Count Register 41–98
Packet Engine Output Buffer Count Decrement Register 41–99
Packet Engine Result Descriptor Ring Base Address 41–100
Packet Engine Result Descriptor Count Registers 41–101
Packet Engine Result Descriptor Count Decrement Registers 41–102
Packet Engine Ring Configuration 41–103
Packet Engine Ring Pointer Status 41–104
Packet Engine Ring Status 41–105
Packet Engine Ring Threshold Registers 41–106

Packet Engine SA Address 41–108
ARC4 i and j Pointer Register 41–109
SA Command 0 41–110
SA Command 1 41–115
SA Inner Hash Digest Registers 41–119
SA Key Registers 41–120
SA Initialization Vector Register 41–121
SA Outer Hash Digest Registers 41–122
SA Ready Indicator 41–123
SA Sequence Number Register 41–124
SA Sequence Number Mask Registers 41–125
SA SPI Register 41–126
Packet Engine Source Address 41–127
Packet Engine Status Register 41–128
Packet Engine State Record Address 41–132
State Hash Byte Count Registers 41–133
State Inner Digest Registers 41–134
State Initialization Vector Registers 41–135
Packet Engine User ID 41–136

Public Key Accelerator (PKA)

PKA Features 42–1
PKA Functional Description 42–1
ADSP-SC59x PKA Register List 42–2
PKA Definitions 42–2
PKA Architectural Concepts 42–3
PKA Block Diagram 42–3
PKCP Vector Operations 42–4
Modular Exponentiation Operations 42–6
Modular Inversion 42–13

Modular Inversion with an Even Modulus 42–14
Modular Inversion with a Prime Modulus 42–14
ECC Operations 42–14
ADSP-SC59x PKA Register Descriptions 42–20
PKA Vector_A Length 42–21
PKA Vector_A Address 42–22
PKA Vector_B Length 42–23
PKA Vector_B Address 42–24
PKA Compare Result 42–25
PKA Vector_C Address 42–26
PKA Most-Significant-Word of Divide Remainder 42–27
PKA Vector_D Address 42–28
PKA Function 42–29
Start of PKA RAM space 42–32
PKA Most-Significant-Word of Result Vector 42–33
PKA Bit Shift Value 42–34

Public Key Interrupt Controller (PKIC)

PKIC Functional Description 43–1
ADSP-SC59x PKIC Register List 43–1
ADSP-SC59x PKIC Interrupt List 43–2
PKIC Programming Model 43–2
PKIC Programming Concepts 43–2
ADSP-SC59x PKIC Register Descriptions 43–3
Acknowledge Register 43–4
Enable Clear Register 43–5
Enable Control Register 43–6
Enable Set Register 43–7
Enabled Status Register 43–8
Polarity Control Register 43–9

Raw Status Register 43–10
Type Control Register 43–11

True Random Number Generator (TRNG)

TRNG Features 44–1
TRNG Functional Description 44–1
ADSP-SC59x TRNG Register List 44–1
Random Number Generation 44–2
Locking Detection and Prevention 44–3
Run Testing 44–4
Monobit Testing 44–4
Poker Testing 44–5
Data for Tests 44–6
X9.31 Postprocessing 44–6
TRNG Block Diagram 44–6
TRNG Architectural Concepts 44–7
TRNG Operating Modes 44–8
TRNG Data Transfer Modes 44–9
TRNG Event Control 44–9
TRNG Interrupt Signals 44–9
ADSP-SC59x TRNG Register Descriptions 44–10
TRNG Alarm Counter Register 44–12
TRNG Alarm Mask Register 44–14
TRNG Alarm Stop Register 44–15
TRNG Block Count Register 44–16
TRNG Configuration Register 44–17
Counter Register 44–19
TRNG Control Register 44–20
TRNG FRO De-tune Register 44–23
TRNG FRO Enable Register 44–24

TRNG Input Registers 44–25
TRNG Interrupt Acknowledge Register 44–26
Post-Process Key Registers 44–28
TRNG LFSR Access Register 44–29
TRNG LFSR Access Register 44–30
TRNG LFSR Access Register 44–31
TRNG Monobit Test Result Register 44–32
TRNG Output Registers 44–33
TRNG Poker Test Result Registers 44–34
TRNG Run Count Registers 44–35
TRNG Run Test State and Result Registers 44–36
TRNG Status Register 44–38
TRNG Test Register 44–40
TRNG Post-Process "V" Value Registers 44–43

FIR Accelerator (FIR)

Features 45–1
Clocking 45–1
Functional Description 45–2
ADSP-SC59x FIR Register List 45–3
ADSP-SC59x FIR Interrupt List 45–3
ADSP-SC59x FIR Trigger List 45–4
Compute Block 45–4
Partial Sum Register 45–5
Delay Line Memory 45–5
Coefficient Memory 45–5
Prefetch Data Buffer 45–5
Processing Output 45–6
System Memory Storage 45–7
Coefficients and Input Buffer Storage 45–7

Single Rate Input Filtering 45–7
Decimation 45–7
Interpolation 45–8
Operating Modes 45–8
Single Rate Processing 45–8
Single Iteration 45–8
Floating-point Multi-Iteration 45–8
Window Processing 45–9
Multi-Rate Processing 45–9
Decimation 45–9
Interpolation 45–9
Floating-Point Data Format 45–10
Fixed-Point Data Format 45–10
Auto Configuration Mode (ACM) 45–10
Data Transfer 45–11
Chain Assignment 45–11
DMA Access 45–13
Burst Access Support 45–13
Data and Coefficient Load in Parallel 45–13
Prefetch Buffer for DMA Channels 45–13
Accelerator TCB 45–14
Chain Pointer DMA 45–14
Programming Model 45–15
Legacy Mode 45–15
Auto Configuration Mode (ACM) 45–16
Debug Mode 45–17
Write to Local Memory 45–17
Read from Local Memory 45–18
Single-Step Mode 45–18
Computing FIR Output, Tap Length Greater than 4096 45–18

Debug Features 45–19
Local Memory Access 45–20
Single-Step Mode 45–20
Emulation Considerations 45–20
Interrupts 45–20
Sources 45–21
Window Complete 45–21
All Channels Complete 45–21
MAC Status 45–21
Service 45–21
ADSP-SC59x FIR Register Descriptions 45–22
FIR Chain Pointer Register 45–23
FIR Coefficient Count Register 45–24
FIR Coefficient Index Register 45–25
FIR Coefficient Modifier Register 45–26
FIR Global Control Register 45–27
FIR Channel Control Register 45–31
Debug Address Register 45–33
FIR Debug Control Register 45–34
FIR Debug Data Read Register 45–35
FIR Debug Data Write Register 45–36
FIR DMA Status Register 45–37
FIR Input Data Base Register 45–39
FIR Input Data Count Register 45–40
FIR Input Data Index Register 45–41
FIR Input Data Modifier Register 45–42
FIR MAC Status Register 45–43
FIR Output Data Base Register 45–46
FIR Output Data Count Register 45–47
FIR Output Data Index Register 45–48

FIR Output Data Modifier Register 45–49
Software Control Register 1 45–50
Software Control Register 2 45–51
Secondary Global Control Register 45–52

IIR Accelerator (IIR)

Features 46–1
IIR Definitions 46–1
Clocking 46–2
Functional Description 46–3
ADSP-SC59x IIR Register List 46–4
ADSP-SC59x IIR Interrupt List 46–5
ADSP-SC59x IIR Trigger List 46–6
Multiply and Accumulate (MAC) Unit 46–7
Input Data and Biquad State 46–7
Coefficient Memory 46–7
Internal Memory Storage 46–7
Coefficient Memory Storage 46–7
Operating Modes 46–8
Window Processing Mode 46–8
40-Bit Floating-Point Mode 46–8
Save Biquad State Mode 46–9
Auto Configuration Mode (ACM) 46–9
Data Transfers 46–10
IIR Accelerator TCB 46–10
DMA Access 46–12
Burst Mode Access 46–12
Chain Pointer DMA 46–12
Interrupts 46–13
Sources 46–14
Window Complete 46–14

All Channels Complete 46–14
MAC Status 46–15
Service 46–15
Programming Model 46–15
Programming Legacy Mode and ACM 46–15
Legacy Mode Programming 46–15
Auto Configuration Mode Programming 46–17
ADSP-SC59x IIR Register Descriptions 46–18
Chain Pointer Register 46–20
Coefficient Buffer Index Register 46–21
Coefficient Buffer Length Register 46–22
Coefficient Index Modifier Register 46–23
Global Control Register 46–24
Channel Control Register 46–28
IIR Debug Address Register 46–30
IIR Debug Control Register 46–31
IIR Debug Read Data High Register 46–32
IIR Debug Read Data Low Register 46–33
IIR Debug Write Data High Register 46–34
IIR Debug Write Data Low Register 46–35
DMA Status Register 46–36
Input Buffer Base Register 46–38
Input Data Index Register 46–39
Input Data Buffer Length Register 46–40
Input Data Index Modifier Register 46–41
MAC Status Register 46–42
Output Buffer Base Register 46–43
Output Data Buffer Index Register 46–44
IIR Output Data Buffer Length Register 46–45
IIR Output Data Index Modifier Register 46–46

Software Control Register1 46–47
Software Control Register2 46–48
Secondary Global Control Register 46–49

Boot ROM and Booting the Processor

SRAM Requirements 47–1
Preboot Operations 47–2
Start-up Sequence 47–3
Core Reset Sequencing 47–3
Core 0 Start-up 47–4
Core 1 Start-up 47–5
Core 2 Start-up 47–6
Fault Configuration 47–7
L2 Controller Configuration 47–7
L2 Memory Initialization 47–7
Idle On Entry 47–8
SPU Configuration 47–8
SMPU Configuration 47–8
Secure Debug Key Processing 47–9
CGU Configuration 47–10
Releasing All Cores from Reset 47–12
L1 Memory Initialization 47–12
Default Entry Point 47–13
NO-BOOT Processing 47–13
SYS_RESOUT Signal 47–14
DMC Configuration 47–14
Bypassing the Boot Process 47–15
Boot Mode Disable 47–15
Boot Command Customization 47–16
Boot Mode Specific SPU Configuration 47–16
Executing the Boot Mode 47–17

Boot Modes 47–18
No-Boot Mode 47–18
SPI Master Boot Mode 47–19
SPI Slave Boot Mode 47–24
Link Port Target Boot Mode 47–27
UART Controller Boot Mode 47–29
OSPI Controller Boot Mode 47–32
eMSI Controller Boot Mode 47–39
Default Booting Peripheral Pin-Mux 47–45
Boot Loader Stream 47–46
Block Types 47–49
Normal Block 47–50
First Block 47–50
Final Block 47–50
Indirect Block 47–51
Ignore Block 47–51
Fill Block 47–52
Init Block 47–52
Callback Block 47–54
Save Block 47–55
Single-Block Boot Streams 47–55
Direct Code Execution 47–56
Multi-Application Boot Streams 47–57
CRC32 Protection 47–59
Secure Boot 47–59
Terminology 47–61
Signing for Secure Boot Images 47–62
Secure Boot Image Types 47–62
Secure Boot Image Format 47–63
Secure Boot Image Attributes 47–66

Secure Boot Streams 47–67
Secure Debug Access 47–69
Boot ROM Errors and Failures 47–71
Boot ROM Programming Model 47–74
Boot Mode Driver API 47–74
Error Handler 47–76
Page Mode 47–77
Boot Hook Function 47–77
enum ROM_HOOK_CALL_CAUSE 47–77
Boot Return Feature 47–78
Boot Termination and Application Execution 47–78
Boot ROM OTP Customizations 47–79
API Reference 47–79
adi_rom_Boot() 47–79
adi_rom_BootKernel() 47–82
adi_rom_Crc32Init() 47–83
adi_rom_Crc32Poly() 47–83
adi_rom_GetAddress() 47–84
adi_rom_MemCompare() 47–84
adi_rom_MemCopy() 47–85
adi_rom_MemCrc() 47–86
adi_rom_MemDma() 47–86
adi_rom_MemFill() 47–89
adi_rom_PeriphDma() 47–90
callback() 47–91
initcode() 47–93
adi_rom_idle_loop() 47–94
adi_rom_CguInit() 47–94
adi_rom_DmcPhyCalibration() 47–94
adi_rom_DmcInit() 47–95

adi_rom_ShaInit() 47–95
adi_rom_Sha() 47–95
Data Structures 47–98
struct ADI_ROM_BOOT_BUFFER 47–98
struct ADI_ROM_BOOT_CONFIG 47–99
struct ADI_ROM_BOOT_HEADER 47–108
struct ADI_ROM_BOOT_INTER_BUFFER 47–109
struct ADI_ROM_BOOT_INTER_BUFFERS 47–109
struct ADI_ROM_BOOT_LINKPORT 47–110
struct ADI_ROM_BOOT_MODES 47–111
struct ADI_ROM_BOOT_REGISTRY 47–111
struct ADI_ROM_BOOT_SPI 47–112
struct ADI_ROM_BOOT_OSPI 47–115
struct ADI_ROM_BOOT_UART 47–116
struct ADI_ROM_OTP_BOOT_CFG 47–117
struct ADI_ROM_OTP_BOOT_CGU_INFO 47–119
struct ADI_ROM_OTP_BOOT_CMD_INFO 47–123
struct ADI_ROM_BOOT_EMSI 47–124
struct ADI_ROM_OTP_BOOT_INFO 47–125
struct ADI_ROM_OTP_DMC_CONFIG 47–126
struct ROM_BOOT_DMA_INSTANCE 47–126
struct ROM_BOOT_MDMA 47–127
struct ROM_BOOT_MDMA_REGS 47–127
struct ROM_DMA_MDMA_CONFIG 47–128
struct ROM_DMA_PDMA_CONFIG 47–130
struct OTP_DATA 47–131
Enumerations 47–134
enum ADI_ROM_BOOT_KEY_TYPE 47–134
enum ADI_ROM_BOOT_TYPE 47–134
enum ROM_BOOT_MDMA_CRC_SUPPORT 47–135

enum ROM_BOOT_RESULT 47–136
enum ROM_CORE_ID 47–140
enum ROM_DMA_DONE_DETECT_METHOD 47–141
enum ROM_DMA_MDMA_ID 47–141
enum ROM_DMA_MDMA_OPERATION 47–142
enum ROM_DMA_PDMA_OPERATION 47–143
enum ROM_GETADDR_VALUE 47–143
enum ROM_HOOK_CALL_CAUSE 47–144
enum ROM_SB_IMAGE_TYPE 47–145
enum ROM_SPI_PROTOCOL 47–146

System Crossbars (SCB)

SCB Features 48–1
SCB Functional Description 48–1
SCB Architectural Concepts 48–2
SCB Block Diagram 48–2
SCB Block Diagram 48–3
System Crossbars 48–13
SCB Bus Requester IDs 48–14
SCB Programming Model 48–17
SCB Programming Concepts 48–19
QoS Programming 48–23
SCB6 Programming 48–23
ADSP-SC59x SCB0 Register Descriptions 48–24
CRC0 Channel 0 Read Quality of Service Register 48–30
CRC0 Channel 0 Write Quality of Service Register 48–31
CRC0 Channel 1 Read Quality of Service Register 48–32
CRC0 Channel 1 Write Quality of Service Register 48–33
CRC1 Channel 0 Read Quality of Service Register 48–34
CRC1 Channel 0 Write Quality of Service Register 48–35
CRC1 Channel 1 Read Quality of Service Register 48–36

CRC1 Channel 1 Write Quality of Service Register 48–37
CRC2 Channel 0 Read Quality of Service Register 48–38
CRC2 Channel 0 Write Quality of Service Register 48–39
CRC2 Channel 1 Read Quality of Service Register 48–40
CRC2 Channel 1 Write Quality of Service Register 48–41
CRC3 Channel 0 Read Quality of Service Register 48–42
CRC3 Channel 0 Write Quality of Service Register 48–43
CRC3 Channel 1 Read Quality of Service Register 48–44
CRC3 Channel 1 Write Quality of Service Register 48–45
CRYPTO Read Quality of Service Register 48–46
CRYPTO Write Quality of Service Register 48–47
DBG Read Quality of Service Register 48–48
DBG Write Quality of Service Register 48–49
DLDMA0 Channel 0 Read Quality of Service Register 48–50
DLDMA0 Channel 0 Write Quality of Service Register 48–51
DLDMA0 Channel 1 Read Quality of Service Register 48–52
DLDMA0 Channel 1 Write Quality of Service Register 48–53
DLDMA1 Channel 0 Read Quality of Service Register 48–54
DLDMA1 Channel 0 Write Quality of Service Register 48–55
DLDMA1 Channel 1 Read Quality of Service Register 48–56
DLDMA1 Channel 1 Write Quality of Service Register 48–57
EMAC Read Quality of Service Register 48–58
EMAC Write Quality of Service Register 48–59
ETR Read Quality of Service Register 48–60
ETR Write Quality of Service Register 48–61
GIGE Read Quality of Service Register 48–62
GIGE Write Quality of Service Register 48–63
HSMDMA1 Channel 0 Read Quality of Service Register 48–64
HSMDMA1 Channel 0 Write Quality of Service Register 48–65
HSMDMA1 Channel 1 Read Quality of Service Register 48–66

HSMDMA1 Channel 1 Write Quality of Service Register 48–67
HSMDMA Channel 0 Read Quality of Service Register 48–68
HSMDMA Channel 0 Write Quality of Service Register 48–69
HSMDMA Channel 1 Read Quality of Service Register 48–70
HSMDMA Channel 1 Write Quality of Service Register 48–71
LP0 Read Quality of Service Register 48–72
LP0 Write Quality of Service Register 48–73
LP1 Read Quality of Service Register 48–74
LP1 Write Quality of Service Register 48–75
MLB Read Quality of Service Register 48–76
MLB Write Quality of Service Register 48–77
MSMDMA1 Channel 0 Read Quality of Service Register 48–78
MSMDMA1 Channel 0 Write Quality of Service Register 48–79
MSMDMA1 Channel 1 Read Quality of Service Register 48–80
MSMDMA1 Channel 1 Write Quality of Service Register 48–81
MSMDMA Channel 0 Read Quality of Service Register 48–82
MSMDMA Channel 0 Write Quality of Service Register 48–83
MSMDMA Channel 1 Read Quality of Service Register 48–84
MSMDMA Channel 1 Write Quality of Service Register 48–85
PL310 M0 Read Quality of Service Register 48–86
PL310 M0 Write Quality of Service Register 48–87
PL310 M1 Read Quality of Service Register 48–88
PL310 M1 Write Quality of Service Register 48–89
PL310 MMR Read Quality of Service Register 48–90
PL310 MMR Write Quality of Service Register 48–91
PPI F0 Read Quality of Service Register 48–92
PPI F0 Write Quality of Service Register 48–93
PPI F1 Read Quality of Service Register 48–94
PPI F1 Write Quality of Service Register 48–95
SCB6 Synchronization Mode Register 48–96

SDIO0 Read Quality of Service Register 48–97
SDIO0 Write Quality of Service Register 48–98
SH0 DPORT Read Quality of Service Register 48–99
SH0 DPORT Write Quality of Service Register 48–100
SH0 FIR Channel 0Read Quality of Service Register 48–101
SH0 FIR Channel 0Write Quality of Service Register 48–102
SH0 FIR Channel 1Read Quality of Service Register 48–103
SH0 FIR Channel 1Write Quality of Service Register 48–104
SH0 IIR Channel 0 Read Quality of Service Register 48–105
SH0 IIR Channel 0 Write Quality of Service Register 48–106
SH0 IIR Channel 1 Read Quality of Service Register 48–107
SH0 IIR Channel 1 Write Quality of Service Register 48–108
SH0 IPORT Read Quality of Service Register 48–109
SH0 IPORT Write Quality of Service Register 48–110
SH0 MMR Read Quality of Service Register 48–111
SH0 MMR Write Quality of Service Register 48–112
SH1 DPORT Read Quality of Service Register 48–113
SH1 DPORT Write Quality of Service Register 48–114
SH1 FIR Channel 0 Read Quality of Service Register 48–115
SH1 FIR Channel 0 Write Quality of Service Register 48–116
SH1 FIR Channel 1 Read Quality of Service Register 48–117
SH1 FIR Channel 1 Write Quality of Service Register 48–118
SH1 IIR Channel 0 Read Quality of Service Register 48–119
SH1 IIR Channel 0 Write Quality of Service Register 48–120
SH1 IIR Channel 1 Read Quality of Service Register 48–121
SH1 IIR Channel 1 Write Quality of Service Register 48–122
SH1 IPORT Read Quality of Service Register 48–123
SH1 IPORT Write Quality of Service Register 48–124
SH1 MMR Read Quality of Service Register 48–125
SH1 MMR Write Quality of Service Register 48–126

SP0A Read Quality of Service Register **48–127**
SP0A Write Quality of Service Register **48–128**
SP0B Read Quality of Service Register **48–129**
SP0B Write Quality of Service Register **48–130**
SP1A Read Quality of Service Register **48–131**
SP1A Write Quality of Service Register **48–132**
SP1B Read Quality of Service Register **48–133**
SP1B Write Quality of Service Register **48–134**
SP2A Read Quality of Service Register **48–135**
SP2A Write Quality of Service Register **48–136**
SP1B Read Quality of Service Register **48–137**
SP1B Write Quality of Service Register **48–138**
SP3A Read Quality of Service Register **48–139**
SP3A Write Quality of Service Register **48–140**
SP3B Read Quality of Service Register **48–141**
SP3B Write Quality of Service Register **48–142**
SP4A Read Quality of Service Register **48–143**
SP4A Write Quality of Service Register **48–144**
SP4B Read Quality of Service Register **48–145**
SP4B Write Quality of Service Register **48–146**
SP5A Read Quality of Service Register **48–147**
SP5A Write Quality of Service Register **48–148**
SP5B Read Quality of Service Register **48–149**
SP5B Write Quality of Service Register **48–150**
SP6A Read Quality of Service Register **48–151**
SP6A Write Quality of Service Register **48–152**
SP6B Read Quality of Service Register **48–153**
SP6B Write Quality of Service Register **48–154**
SP7A Read Quality of Service Register **48–155**
SP7A Write Quality of Service Register **48–156**

SP7B Read Quality of Service Register 48–157
SP7B Write Quality of Service Register 48–158
SPI0 RX Read Quality of Service Register 48–159
SPI0 RX Write Quality of Service Register 48–160
SPI0 TX Read Quality of Service Register 48–161
SPI0 TX Write Quality of Service Register 48–162
SPI1 RX Read Quality of Service Register 48–163
SPI1 RX Write Quality of Service Register 48–164
SPI1 TX Read Quality of Service Register 48–165
SPI1 TX Write Quality of Service Register 48–166
SPI2 RX Read Quality of Service Register 48–167
SPI2 RX Write Quality of Service Register 48–168
SPI2 TX Read Quality of Service Register 48–169
SPI2 TX Write Quality of Service Register 48–170
SPI3 RX Read Quality of Service Register 48–171
SPI3 RX Write Quality of Service Register 48–172
SPI3 TX Read Quality of Service Register 48–173
SPI3 TX Write Quality of Service Register 48–174
UART0 RX Read Quality of Service Register 48–175
UART0 RX Write Quality of Service Register 48–176
UART0 TX Read Quality of Service Register 48–177
UART0 TX Write Quality of Service Register 48–178
UART1 RX Read Quality of Service Register 48–179
UART1 RX Write Quality of Service Register 48–180
UART1 TX Read Quality of Service Register 48–181
UART1 TX Write Quality of Service Register 48–182
UART2 RX Read Quality of Service Register 48–183
UART2 RX Write Quality of Service Register 48–184
UART2 TX Read Quality of Service Register 48–185
UART2 TX Write Quality of Service Register 48–186

UART3 RX Read Quality of Service Register 48–187
UART3 RX Write Quality of Service Register 48–188
UART3 TX Read Quality of Service Register 48–189
UART3 TX Write Quality of Service Register 48–190
USB0 Read Quality of Service Register 48–191
USB0 Write Quality of Service Register 48–192
ADSP-SC59x SCB1 Register Descriptions 48–192
DMC Fabric (CLK03) Synchronization Mode Register 48–193
ADSP-SC59x SCB3 Register Descriptions 48–193
DDR MMRG Fabric (CLK03) Synchronization Mode Register 48–194
DDR MMRG Fabric (CLK04) Synchronization Mode Register 48–195
DDR MMRG Fabric (CLK08) Synchronization Mode Register 48–196
ADSP-SC59x SCB4 Register Descriptions 48–196
Fabric Acc MMR Read Quality of Service Register 48–198
Fabric Acc MMR Write Quality of Service Register 48–199
Fabric S1PORT Read Quality of Service Register 48–200
Fabric S1PORT Write Quality of Service Register 48–201
Fabric S2PORT Read Quality of Service Register 48–202
Fabric S2PORT Write Quality of Service Register 48–203
FIR Channel 0 Read Quality of Service Register 48–204
FIR Channel 0 Write Quality of Service Register 48–205
FIR Channel 1 Read Quality of Service Register 48–206
FIR Channel 1 Write Quality of Service Register 48–207
IIR0 Channel 0 Read Quality of Service Register 48–208
IIR0 Channel 0 Write Quality of Service Register 48–209
IIR0 Channel 1 Read Quality of Service Register 48–210
IIR0 Channel 1 Write Quality of Service Register 48–211
IIR1 Channel 0 Read Quality of Service Register 48–212
IIR1 Channel 0 Write Quality of Service Register 48–213
IIR1 Channel 1 Read Quality of Service Register 48–214

IIR1 Channel 1 Write Quality of Service Register 48–215
IIR2 Channel 0 Read Quality of Service Register 48–216
IIR2 Channel 0 Write Quality of Service Register 48–217
IIR2 Channel 1 Read Quality of Service Register 48–218
IIR2 Channel 1 Write Quality of Service Register 48–219
IIR3 Channel 0 Read Quality of Service Register 48–220
IIR3 Channel 0 Write Quality of Service Register 48–221
IIR3 Channel 1 Read Quality of Service Register 48–222
IIR3 Channel 1 Write Quality of Service Register 48–223
DPORT Read Quality of Service Register 48–224
DPORT Write Quality of Service Register 48–225
ADSP-SC59x SCB5 Register Descriptions 48–225
SPI2/OSPI Memory Map Address Remap Register 48–226
ADSP-SC59x SCB6 Register Descriptions 48–226
A55 M0 Ib.fn Mod 48–227

System Watchpoint Unit (SWU)

SWU Features 49–1
SWU Functional Description 49–1
ADSP-SC59x SWU Register List 49–1
ADSP-SC59x SWU Interrupt List 49–2
ADSP-SC59x SWU Trigger List 49–3
SWU Definitions 49–4
SWU Architectural Concepts 49–4
SWU-to-SCB Interface 49–4
SWU Block Diagram 49–4
SCB Interface Block 49–5
MMR Interface Block 49–5
SWU Operating Modes 49–5
Bandwidth Mode 49–5
Watchpoint Mode 49–5

Match Block 49–5
Scaling 49–6
SWU Event Control 49–6
SWU Interrupts 49–6
SWU Status and Errors 49–6
Triggers 49–7
SWU Programming Model 49–7
SWU Mode Configuration 49–8
Configuring the SWU for Bandwidth Mode 49–8
Configuring the SWU for Watchpoint Mode 49–9
ADSP-SC59x SWU Register Descriptions 49–10
Count Register n 49–11
Control Register n 49–12
Current Register n 49–17
Global Control Register 49–18
Global Status Register 49–19
Bandwidth History Register n 49–23
ID Register n 49–24
Lower Address Register n 49–25
Target Register n 49–26
Upper Address Register n 49–27

Memory Error Protection Unit (MEPU)

Memory Error Controller (MEC) 50–1
MEC Features 50–1
Parity Controller (PCTL) 50–2
PCTL Features 50–2
MEC Functional Description 50–2
ADSP-SC59x MEC Register List 50–2
MEC Trigger List 0

MEPU Block Diagram 50–3
MEC Block Diagram 50–4
PCTL Block Diagram 50–4
MECx Input and Output Block Diagram 50–5
MEC Architectural Concepts 50–6
MEC Configuration 50–6
PCTL Integration 50–8
ADSP-SC59x MEC Register Descriptions 50–8
A55 Core Memory Errors Interrupt Request Global Control Register 50–10
A55 Error Interrupt Request Global Status Register 50–11
A55 Error Status Register 50–12
A55 Error Control Register 50–13
A55 Error Interrupt Mask Register 50–14
Component ID0 Register 50–15
Component ID1 Register 50–16
Component ID2 Register 50–17
Component ID3 Register 50–18
Clear Register 50–19
ECC Error Control Register 50–20
ECC Error Interrupt Mask Register 50–21
ECC Error Status Register 50–22
ECC Error Interrupt Request Global Control Register 50–23
ECC Error Interrupt Request Global Status Register 50–24
Parity Error Interrupt Request Global Control Register 50–25
Parity Error Interrupt Request Global Status Register 50–26
Parity Error Control Register 50–27
Parity Error Control Register 50–29
Parity Error Interrupt Mask Register 50–30
Parity Error Interrupt Mask Register 50–32
Parity Error Status Register 50–33

Parity Error Status Register 50–35
Peripheral ID0 Register 50–36
Peripheral ID1 Register 50–37
Peripheral ID2 Register 50–38
Peripheral ID3 Register 50–39
Peripheral ID4 Register 50–40

System Debug and Trace Unit (DBG)

DBG Features 51–1
Debug Functional Description 51–2
ADSP-SC59x CSPFT Register List 51–2
ADSP-SC59x TAPC Register List 51–3
DBG Block Diagram 51–3
DBG Definitions 51–4
Test Access Port Controller (TAPC) 51–5
Embedded Trace Macrocell (ETM) 51–5
Debug Access Ports 51–5
Trace Unit 51–5
Programmable Flow Trace (CSPFT) 51–6
System Trace Module (STM) 51–6
Embedded Cross Trigger (ECT) 51–7
CTI Debug Trigger Tables 51–8
ADSP-SC59x CSPFT Register Descriptions 51–12
Address Comparator Access Type Register 51–14
Address Comparator Value Register 51–15
Authentication Status Register 51–16
Configuration Code Extension Register 51–17
Component ID0 Register 51–18
Component ID1 Register 51–19
Component ID2 Register 51–20

Component ID3 Register 51–21
Context ID Comparator Mask Register 51–22
Context ID Comparator Value 51–23
Claim Tag Clear Register 51–24
Claim Tag Set Register 51–25
Counter Enable Event Register 51–26
Counter Reload Event Register 51–29
Counter Reload Value Register 51–32
Counter Value Register 51–33
Main Control Register 51–34
Device Type Identifier Register 51–36
External Output Event Register 51–37
Hardware Feature Register 51–40
Lock Access Register 51–42
Lock Status Register 51–43
Peripheral ID0 Register 51–44
Peripheral ID1 Register 51–45
Peripheral ID2 Register 51–46
Peripheral ID3 Register 51–47
Peripheral ID4 Register 51–48
Status Register 51–49
Synchronization Frequency Register 51–50
CoreSight Trace ID Register 51–51
ADSP-SC59x TAPC Register Descriptions 51–51
Debug Control Register 51–52
IDCODE Register 51–54
Secure Debug Key 0 Register 51–55
Secure Debug Key 1 Register 51–56
Secure Debug Key 2 Register 51–57
Secure Debug Key 3 Register 51–58

Secure Debug Key Control Register 51–59
Secure Debug Key Status Register 51–60
USERCODE Register 51–61
ADSP-SC59x CSCTI Register Descriptions 51–61
External Multiplexor Control Register 51–64
Authentication Status 51–65
Claim Tag Clear Register 51–66
Claim Tag Set Register 51–67
Component ID0 51–68
Component ID1 51–69
Component ID2 51–70
Component ID3 51–71
CTI Application Trigger Clear Register 51–72
CTI Application Pulse Register 51–73
CTI Application Trigger Set Register 51–74
CTI Channel In Status Register 51–75
CTI Channel Out Status Register 51–76
CTI Control Register 51–77
Enable CTI Channel Gate Register 51–78
CTI Trigger 0 to Channel Enable Register 51–79
CTI Trigger 1 to Channel Enable Register 51–80
CTI Trigger 2 to Channel Enable Register 51–81
CTI Trigger 3 to Channel Enable Register 51–82
CTI Trigger 4 to Channel Enable Register 51–83
CTI Trigger 5 to Channel Enable Register 51–84
CTI Trigger 6 to Channel Enable Register 51–85
CTI Trigger 7 to Channel Enable Register 51–86
CTI Interrupt Acknowledge Register 51–87
CTI Channel to Trigger 0 Enable Register 51–88
CTI Channel to Trigger 1 Enable Register 51–89

CTI Channel to Trigger 2 Enable Register 51–90
CTI Channel to Trigger 3 Enable Register 51–91
CTI Channel to Trigger 4 Enable Register 51–92
CTI Channel to Trigger 5 Enable Register 51–93
CTI Channel to Trigger 6 Enable Register 51–94
CTI Channel to Trigger 7 Enable Register 51–95
CTI Trigger In Status Register 51–96
CTI Trigger Out Status Register 51–97
Device ID 51–98
Device Type 51–99
Integration Test Channel Input 51–100
ITCHINACK 51–101
Integration Test Channel Output 51–102
ITCHOUTACK 51–103
Integration Mode Control Register 51–104
Integration Test Trigger Input 51–105
ITTRIGINACK 51–106
Integration Test Trigger Output 51–107
ITTRIGOUTACK 51–108
Lock Access Register 51–109
Lock Status Register 51–110
Peripheral ID0 51–111
Peripheral ID1 51–112
Peripheral ID2 51–113
Peripheral ID3 51–114
Peripheral ID4 51–115
Peripheral ID5 51–116
Peripheral ID6 51–117
Peripheral ID7 51–118

ADSP-SC59x Register List

Preface

Thank you for purchasing and developing systems using a ADSP-SC59x SHARC+® processor from Analog Devices, Inc.

Purpose of This Manual

The ***ADSP-SC596/SC598 SHARC+ Processor Hardware Reference*** provides architectural information about the ADSP-SC59x processors. This hardware reference provides the main architectural information about these processors. The architectural descriptions cover functional blocks, buses, and ports, including all features and processes that they support.

For timing, electrical, and package specifications, refer to the ***SHARC+ Dual-Core DSP with Arm Cortex-A55 ADSP-SC596/SC598 Data Sheet***.

Intended Audience

The primary audience for this manual is a programmer who is familiar with Analog Devices processors. The manual assumes the audience has a working knowledge of the appropriate processor architecture and instruction set. Programmers who are unfamiliar with Analog Devices processors can use this manual, but should supplement it with other texts, such as programming reference books and data sheets, that describe their target architecture.

What's New in This Manual

This manual is the third preliminary revision (0.3) of the *ADSP-SC596/SC598 SHARC+ Processor Hardware Reference*. The following changes are included in this edition:

- Added the programming model topics in the USB chapter
- Updated the SCB Controlled DMA Channel Peripherals and Bus Requester ID tables.
- Fixed typo in Secure Boot Header table, Keyless Format BLx value.
- Added Loopback Pin Buffers topic in the DAI chapter
- Added Default Booting Peripheral Pin-Mux section in Boot Chapter
- Updated eMSI chapter
- Updated the title of the book to reflect correct product offering.

NOTE: Analog Devices is in the process of updating documentation to provide terminology and language that is culturally appropriate. This is a process with a wide scope and will be phased in as quickly as possible. Thank you for your patience.

Technical or Customer Support

You can reach customer and technical support for processors and peripherals from Analog Devices using the following resources:

- Post your questions in the processors and DSP support community at ***EngineerZone***®:

  http://ez.analog.com/community/dsp

- Submit your questions to technical support at ***Connect with ADI Specialists***:

  http://www.analog.com/support

- Contact your Analog Devices sales office or authorized distributor. Locate one at:

  http://www.analog.com/adi-sales

Product Information

Product information can be obtained from the Analog Devices Internet site and the CrossCore Embedded Studio® (CCES) online Help system.

Analog Devices Web Site

The Analog Devices Web site, http://www.analog.com, provides information about a broad range of products—analog integrated circuits, amplifiers, converters, and digital signal processors.

To access a complete technical library for each processor family, go to: http://www.analog.com/processors/technical_library The manuals selection opens a list of current manuals related to the product as well as a link to the previous revisions of the manuals. When locating your manual title, note a possible errata check mark next to the title that leads to the current correction report against the manual.

Also note, MyAnalog.com is a free feature of the Analog Devices Web site that allows customization of a Web page to display only the latest information about products you are interested in. You can choose to receive weekly e-mail notifications containing updates to the Web pages that meet your interests, including documentation errata against all manuals. MyAnalog.com provides access to books, application notes, data sheets, code examples, and more.

Visit MyAnalog.com to sign up. If you are a registered user, just log on. Your user name is your e-mail address.

EngineerZone

EngineerZone is a technical support forum from Analog Devices. It allows you direct access to ADI technical support engineers. You can search FAQs and technical information to get quick answers to your embedded processing and DSP design questions.

Use EngineerZone to connect with other DSP developers who face similar design challenges. You can also use this open forum to share knowledge and collaborate with the ADI support team and your peers. Visit http://ez.analog.com to sign up.

Supported Processors

The following is the list of Analog Devices, Inc. processors.

Blackfin+® (ADSP-BF7xx) Processors

The name *Blackfin+* refers to the enhanced fixed-point Blackfin core architecture featured by the ADSP-BF70x processor product line, which is a family of 16-bit embedded processors.

Blackfin® (ADSP-BF6xx/BF5xx) Processors

The name *Blackfin* refers to the fixed-point core architecture featured on the following processors: ADSP-BF5xx and ADSP-BF6xx.

SHARC® (ADSP-21xxx) Processors

The name *SHARC* refers to the high-performance, 32-bit, floating-point core architecture featured on the following processors: ADSP-2116x, ADSP-2126x, ADSP-213xx, and ADSP-214xx. These processors can be used in speech, sound, graphics, and imaging applications.

SHARC+® (ADSP-SC5xx, ADSP-215xx) Processors

The name *SHARC+* refers to the enhanced high-performance, 32-bit, floating-point core architecture featured on the following processors: ADSP-215xx/ADSP-SC5xx. The connected SHARC+ ADSP-SC5xx processors also contain an Arm® Cortex®-A5 core. These products can be used in speech, sound, graphics, and imaging applications.

The following is the list of Analog Devices, Inc. processors supported by the IAR Embedded WorkBench® development tools. For information about the IAR Embedded WorkBench product and software download, go to http://www.iar.com/en/Products/IAR-Embedded-Workbench .

Notation Conventions

Text conventions used in this manual are identified and described as follows. Additional conventions, which apply only to specific chapters, may appear throughout this document.

Example	Description
File > ***Close***	Titles in reference sections indicate the location of an item within the CrossCore Embedded Studio IDE's menu system (for example, the ***Close*** command appears on the ***File*** menu).
`{this | that}`	Alternative required items in syntax descriptions appear within curly brackets and separated by vertical bars; read the example as `this` or `that`. One or the other is required.

Example	Description
`[this | that]`	Optional items in syntax descriptions appear within brackets and separated by vertical bars; read the example as an optional `this` or `that`.
`[this, …]`	Optional item lists in syntax descriptions appear within brackets delimited by commas and terminated with an ellipse; read the example as an optional comma-separated list of `this`.
`.SECTION`	Commands, directives, keywords, and feature names are in text with `Letter Gothic` font.
`filename`	Non-keyword placeholders appear in text with italic style format.
NOTE:	***NOTE:*** For correct operation, ... A note provides supplementary information on a related topic. In the online version of this book, the word ***NOTE:*** appears instead of this symbol.
CAUTION:	***CAUTION:*** Incorrect device operation may result if ... ***CAUTION:*** Device damage may result if ... A caution identifies conditions or inappropriate usage of the product that could lead to undesirable results or product damage. In the online version of this book, the word ***CAUTION:*** appears instead of this symbol.
ATTENTION:	***ATTENTION:*** Injury to device users may result if ... A warning identifies conditions or inappropriate usage of the product that could lead to conditions that are potentially hazardous for devices users. In the online version of this book, the word ***ATTENTION:*** appears instead of this symbol.
Registers/Bits	All registers and bits in this manual are linked (clickable) to their respective descriptions in the "Register Descriptions" of each chapter.
Miscellaneous Conventions	Interrupt and internal signals are shown in all caps with no other formatting. For example the SPDIFn_RX or SCLK signal or the PKTE0_IRQ interrupt. An overbar denotes an active-low signal as in $\overline{\text{SYS_FAULT}}$.

Register Documentation Conventions

Register diagrams use the following conventions:

- The descriptive name of the register appears at the top with the short form of the name.
- If a bit has a short name, the short name appears first in the bit description, followed by the long name.
- The reset value appears in binary in the individual bits and in hexadecimal to the left of the register.
- Bits marked ***X*** have an unknown reset value. Consequently, the reset value of registers that contain such bits is undefined or dependent on pin values at reset.
- Shaded bits are reserved

NOTE: To ensure upward compatibility with future implementations, write back the value that is read for reserved bits in a register, unless otherwise specified.

Register description tables use the following conventions:

- Each bit's or bit field's access type appears beneath the bit number in the table in the form (read-access/write-access). The access types include:
  - R = read, RC = read clear, RS = read set, R0 = read zero, R1 = read one, Rx = read undefined
  - W = write, NW = no write, W1C = write one to clear, W1S = write one to set, W0C = write zero to clear, W0S = write zero to set, WS = write to set, WC = write to clear, W1A = write one action
- Many bit and bit field descriptions include enumerations, identifying bit values and related functionality. Unless otherwise indicated (with a prefix), these enumerations are decimal values.

1 Arm Cortex-A55 Subsystem

The ADSP-596/598 processor includes an Arm® Cortex-A55® core. The Cortex-A55 core is a mid-range, low-power core that implements the Armv8-A architecture with support for the Armv8.1-A extension, the Armv8.2-A extension and the reliability, availability, and serviceability (RAS) extension. The core has a Level 1 (L1) memory system, and private Level 2 (L2) cache. The core is implemented inside the DynamIQ Shared Unit (DSU) as a Little core.

The Cortex-A55 sub-system in the ADSP-596/598 processor includes a Data Engine unit that implements the Advanced SIMD and floating-point architecture support, Cryptographic Extension, Generic Interrupt Controller, Performance Monitoring Unit, and a Generic Timer. The Cortex-A55 also includes support for a L1-Cache sub-system and a full-fledged Memory Management Unit. The Cortex-A55 implements the Armv8 architecture and runs 64-bit Arm instructions in AArch64 execution state and 32-bit Arm and 32-bit Thumb instructions in AArch32 execution state.

This document describes the Arm Cortex-A55 core and memory architecture used on the ADSP-596/598 processor but does not provide detailed programming information for the Arm processor. For more information about programming the Arm processor, visit the Arm Information Center:

- ARM Documentation

The applicable documentation for programming the Arm Cortex-A5 processor include:

- Arm Cortex-A55 Core Technical Reference Manual, Revision: r2p0
- Arm Cortex-A Series Programmer's Guide for ARMv8-A, Version: 1.0
- ARM CoreLink NIC-400 Network Interconnect Technical Reference Manual
- Arm Architecture Reference Manual Armv8, for Armv8-A architecture profile
- Arm DynamIQ™ Shared Unit Technical Reference Manual, Revision: r4p1
- ARM NEON™ Programmer’s Guide
- Arm Cortex®-A55 Core Advanced SIMD and Floating-point Support Technical Reference Manual
- Arm Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and 4.0

Cortex-A55 Features

The Arm Cortex-A55 subsystem has the following features.

Core Features

- Full implementation of the Arm8.2-A A64, A32, and T32 instruction sets
- Both the AArch32 and AArch64 execution states at all Exception levels (EL0 to EL3)
- In-order pipeline with direct and indirect branch prediction
- Separate L1 data and instruction side memory systems with a Memory Management Unit (MMU)
- Support for Arm TrustZone® technology
- Data Engine unit that implements the Advanced SIMD and floating-point architecture support
- Cryptographic Extension. This architectural extension is only available if the Data Engine is present
- Generic Interrupt Controller (GIC) CPU interface to connect to an external distributor
- Generic Timers interface that support a 64-bit count input from an external system counter

Cache Features

- Unified private L2 cache
- L1 and L2 cache protection in the form of Error Correction Code (ECC) or parity on all RAM instances

Debug Features

- Reliability, Availability, and Serviceability (RAS) extension
- Arm8.2-A debug logic
- Performance Monitoring Unit (PMU)
- Embedded Trace Macrocell (ETM) that supports instruction trace only

Functional Description

The following sections provide information on the function of the subsystem.

Cortex-A55 Block Diagram

The ***A55 Subsystem Block Diagram*** shows the primary blocks of the Cortex-A55 subsystem. The cluster consists of the core and the DynamIQ Shared Unit (DSU), which connects the cores to an external memory system.

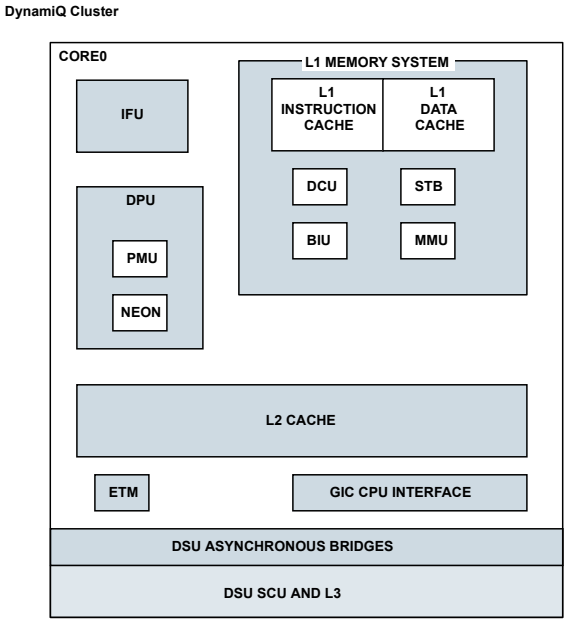


Figure 1-1: A55 Subsystem Block Diagram

Instruction Fetch Unit (IFU)

The IFU fetches instructions from the instruction cache or from external memory and predicts the outcome of branches in the instruction stream. It passes the instructions to the Data Processing Unit (DPU) for processing.

Data Processing Unit (DPU)

The DPU decodes and executes instructions. It executes instructions that require data transfer to or from the memory system by interfacing to the Data Cache Unit (DCU). The DPU includes the PMU, the Advanced SIMD and floating-point support, and the Cryptographic Extension.

Performance Monitor Unit (PMU)

The Cortex-A55 core includes performance monitors that can be configured to gather various statistics on the operation of the core and its memory system during runtime. These provide useful information about the behavior of the code that can be used for debug and code profiling. The PMU provides six counters. Each counter can count any of the events available in the core.

See the Performance Monitor Unit (PMU) section below for details of the PMU register set and events.

Advanced SIMD and Floating-point support

Advanced SIMD is a media and signal processing architecture that adds instructions primarily for audio, video, 3D graphics, image, and speech processing. The floating-point architecture provides support for single-precision and double-precision floating-point operations.

The Cortex-A55 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set and the Advanced SIMD and floating-point instructions in the A32 and T32 instruction sets. The A64 instruction set offers additional Advanced SIMD instructions, including double-precision floating-point vector operations.

The Cortex-A55 floating-point implementation:

- Does not generate floating-point exceptions
- Implements all scalar operations in hardware with support for all combinations of:
    - Rounding modes
    - Flush-to-zero
    - Default Not a Number (NaN) modes

The Advanced SIMD architecture, its associated implementation, and supporting software, are also referred to as NEON™ technology.

See the *Arm® Cortex®-A55 Core Advanced SIMD and Floating-point Support Technical Reference Manual* for details of the instruction and register set.

Cryptographic Extension

The Cortex-A55 core Cryptographic Extension supports the Armv8-A cryptographic extension. The cryptographic extension adds new A64, A32, and T32 instructions to advanced SIMD that accelerate:

- Advanced Encryption Standard (AES) encryption and decryption
- The Secure Hash Algorithm (SHA) functions SHA-1, SHA-224, and SHA-256
- Finite field arithmetic used in algorithms such as Galois/Counter mode and elliptic curve cryptography

See the *Arm® Architecture Reference Manual Armv8*, for Armv8-A architecture profile for details of the instruction and register set.

Memory Management Unit (MMU)

The MMU is responsible for translating addresses of code and data from the virtual view of memory to the physical addresses in the real system. The translation is carried out by the MMU hardware and is transparent to the application. In addition, the MMU controls such things as memory access permissions, memory ordering and cache policies for each region of memory.

The MMU enables tasks or applications to be written in a way that requires them to have no knowledge of the physical memory-map of the system, or about other programs that might be running simultaneously. This enables you to use the same virtual memory address space for each program. It also lets you work with a contiguous virtual memory map, even if the physical memory is fragmented. This virtual address space is separate from the actual physical map of memory in the system. Applications are written, compiled, and linked to run in the virtual memory space. Virtual addresses are those used by you, and the compiler and linker, when placing code in memory. Physical addresses are those used by the actual hardware system.

The three main functions of the MMU are:

Page table entries support:

- Control the translation table walk hardware that accesses translation tables in main memory
- Translate virtual addresses (VAs) to physical addresses (PAs)
- Provide fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables

Each stage of address translation uses a set of address translations and associated memory properties that are held in memory-mapped tables called translation tables. Translation table entries can be cached into a Translation Lookaside Buffer (TLB). The following table describes the components included in the MMU.

Table 1-1: TLBs and TLB Caches in the MMU

Component	Description
Instruction L1 TLB	15 entries, fully associative
Data L1 TLB	16 entries, fully associative
L2 TLB	1024 entries, 4-way set associative
Walk cache RAM	64 entries, 4-way set associative
IPA cache RAM	64 entries, 4-way set associative

The Cortex-A55 core supports a 40-bit physical address range, which allows 1TB of physical memory to be addressed.

The Cortex-A55 core implements a two-level TLB structure.

L1 TLB

The first level of caching for the translation table information is an L1 TLB. It is implemented on both the instruction and data sides. The Cortex-A55 L1 instruction TLB supports 4 KB, 16 KB, 64 KB, and 2 MB pages. The Cortex-A55 L1 data TLB supports 4 KB pages only. Any other page sizes are fractured after the L2 TLB and the appropriate page size are sent to the L1 TLB. All TLB-related maintenance operations result in flushing both the instruction and data L1 TLBs.

L2 TLB

A unified L2 TLB handles the misses from the L1 instruction and data TLBs using a 4-way, set-associative, 1024-entry cache. In implementations with core cache protection, parity bits protect the TLB RAMs by detecting any single-bit error. If an error is detected, the entry is invalidated and fetched again.

IPA Cache RAM

The Intermediate Physical Address (IPA) cache RAM holds mappings between the IPAs and Physical Addresses (PAs). Only non-secure EL1 and EL0 stage 2 translations use the IPA cache. When a stage 2 translation completes,

the cache is updated. The IPA cache is checked whenever a stage 2 translation is required. Like the L2 TLB, the IPA cache RAM can hold entries for different sizes.

Walk Cache RAM

The walk cache RAM holds the result of a stage 1 translation up to, but not including the last level.

NOTE: Virtual memory translation tables are typically created by operating systems and are often dynamically managed by the memory management layer. However, even a bare metal system can enable the MMU. For this, a flat mapping technique is used, where all virtual memory addresses are programmed exactly as the physical memory addresses in the system.

L1 Memory System

The Cortex-A55 processor L1 memory system consists of separate instruction and data caches that run at Arm core clock speed. The L1 instruction cache size is 32 KB and data cache size is 32 KB.

L1 Instruction-side Memory System

The L1 instruction-side memory system provides an instruction stream to the DPU. Its key features are:

- 64-byte instruction side cache line length
- 4-way set associative L1 instruction cache
- 128-bit read interface to the L2 memory system

The Cortex-A55 core uses extensive branch prediction to improve Instructions Per Clock (IPC) and power efficiency.

L1 Data-side Memory System

The L1 data-side memory system responds to load and store requests from the DPU. It also responds to SCU snoop requests from other cores, or external controllers. Its key features are:

- 64-byte data side cache line length
- 4-way set associative L1 data cache
- A read buffer that services both the Data Cache Unit (DCU), and the Instruction Fetch Unit (IFU)
- A 64-bit read path from the data L1 memory system to the data path
- A 128-bit write path from the data path to the L1 memory system
- Merging store buffer capability which writes to all types of memory (device, normal cacheable and normal non-cacheable)
- A data side prefetch engine that detects patterns of strides with multiple streams are allowed in parallel, capable of detecting both constant and patterns of strides

L2 Memory System

The Cortex-A55 L2 memory system is required to interface the Cortex-A55 cores to the L3 memory system. The main features of the L2 memory system are:

- Strictly exclusive with L1 data cache
- Pseudo-inclusive with L1 instruction cache
- Private per-core unified L2 cache
- 40-bit physical address space
- Physically indexed, physically tagged

The L2 memory subsystem consists of:

- A 4-way, set-associative L2 cache with a size of 256 KB. Cache lines have a fixed length of 64 bytes.
- Optional ECC protection for tag, data, and L2 data buffer RAM structures

Reliability, Availability, and Serviceability (RAS)

The Cortex-A55 core implements the RAS extension to the Armv8-A architecture which provides mechanisms for the standardized reporting of errors generated by cache protection mechanisms. When configured with core cache protection, the Cortex-A55 core can detect and correct a 1-bit error in any RAM and detect 2-bit errors in some RAMs.

See the *Arm Cortex-A55 Core Technical Reference Manual* for more information on RAS.

Generic Interrupt Controller (GIC)

The GIC-600 is an Arm architecture compliant System-on-Chip (SoC) peripheral. It is a high-performance, area-optimized interrupt controller. The GIC implements the Arm generic interrupt controller architecture. The GIC takes interrupts asserted at the system level and signals them to each connected processor as appropriate. The GIC has the following features.

- Registers for managing interrupt sources, interrupt behavior, and interrupt routing to one or more processors
- Support for the Arm architecture security extensions
- The ability to enable, disable, and generate processor interrupts from hardware (peripheral) interrupt sources
- The ability to generate software interrupts
- Support for interrupt masking and prioritization

Refer to the GIC Overview section for more information on the processor-specific configuration of the GIC.

The GIC-600 contains a performance monitoring unit (PMU) for counting key GIC events from the distributor.

Refer to the GIC Performance Monitoring Unit (GIC PMU) section for more information.

A55 Core in DynamIQ Shared Unit (DSU)

The ADSP-SC596/SC598 processor implements a single A55 Core (LITTLE core) with L1 and L2 cache inside the DynamIQ. The LITTLE cluster does not contain shared/L3 cache, only the Snoop Control Unit (SCU) with snoop filter. The A55 core is configured to run synchronously with the DSU.

See the *Arm DynamIQ™ Shared Unit Technical Reference Manual* for further details on DSU.

System Control

The system registers control and provide status information for the functions that the core implements. The main functions of the system registers are:

- Overall system control and configuration
- MMU configuration and management
- Cache configuration and management
- System performance monitoring
- GIC configuration and management

The system registers are accessible in the AArch64 and AArch32 execution states.

Generic Timer

The generic timer can schedule events and trigger interrupts that are based on an incrementing counter value. It generates timer events as active-low interrupt outputs and event streams. The timer contains:

- An EL1 non-secure physical timer
- An EL2 hypervisor physical timer
- An EL3 secure physical timer
- A virtual timer
- A hypervisor virtual timer

The Cortex-A55 core does not include the system counter. This resides in the SoC. The system counter value is distributed to the core with a 64-bit bus.

See the *Arm DynamIQ Shared Unit Technical Reference Manual* and the *Arm Architecture Reference Manual Armv8*, for Armv8-A architecture profile for more information on the generic timer.

Generic Timer Interrupts

Timers can be configured to generate an interrupt. The active-low, level sensitive timer interrupt signals are routed to the GIC as a private peripheral interrupt (PPI). The specific interrupts are shown in the *GIC Interrupt* table.

Table 1-2: Generic Timer Interrupts

Interrupt ID	Name	Description	Sensitivity
26	CNTHP_IRQ	Hypervisor physical timer event	Level
27	CNTV_IRQ	Virtual timer event	Level
28	CNTHV_IRQ	Hypervisor virtual timer event	Level
29	CNTPS_IRQ	Secure physical timer event	Level
30	CNTPNS_IRQ	Non-Secure physical timer event	Level

ADSP-SC59x GIC Interrupt List

Table 1-3: ADSP-SC59x GIC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
339	GIC0_SPI_SOFT0_INT	GIC0 GIC Software Interrupt 0		
340	GIC0_SPI_SOFT1_INT	GIC0 GIC Software Interrupt 1		
341	GIC0_SPI_SOFT2_INT	GIC0 GIC Software Interrupt 2		
342	GIC0_SPI_SOFT3_INT	GIC0 GIC Software Interrupt 3		
343	GIC0_SPI_SOFT4_INT	GIC0 GIC Software Interrupt 4		
344	GIC0_SPI_SOFT5_INT	GIC0 GIC Software Interrupt 5		
345	GIC0_SPI_SOFT6_INT	GIC0 GIC Software Interrupt 6		
346	GIC0_SPI_SOFT7_INT	GIC0 GIC Software Interrupt 7		
347	GIC0_SPI_SOFT8_INT	GIC0 GIC Software Interrupt 8		
348	GIC0_SPI_SOFT9_INT	GIC0 GIC Software Interrupt 9		
349	GIC0_SPI_SOFT10_INT	GIC0 GIC Software Interrupt 10		
350	GIC0_SPI_SOFT11_INT	GIC0 GIC Software Interrupt 11		
351	GIC0_SPI_SOFT12_INT	GIC0 GIC Software Interrupt 12		
352	GIC0_SPI_SOFT13_INT	GIC0 GIC Software Interrupt 13		
353	GIC0_SPI_SOFT14_INT	GIC0 GIC Software Interrupt 14		
354	GIC0_SPI_SOFT15_INT	GIC0 GIC Software Interrupt 15		

A55 Configurations

The ***Core Configuration*** and ***DSU Configuration*** tables describe the Cortex A55 and processor configurations.

Table 1-4: Core Configuration

Core Feature	Comment
L1 instruction cache size	32 KB
L1 data cache size	32 KB
L2 cache	Included
L2 cache size	256 KB
ECC or parity core cache protection	Included
Advanced SIMD and floating-point support (including Dot Product instruction support)	Included
Cryptographic Extension	Included
CoreSight Embedded Logic Analyzer (ELA)	False
CoreSight ELA RAM address size	N/A
Page Based Hardware Attributes (PBHA) support	Not included

Table 1-5: DSU Configuration

Core Feature	Comment
NUM_BIG_CORES	0
NUM_LITTLE_CORES	1
NUM_OTHER_CORES	0
BIG_CORE_TYPE	N/A
LITTLE_CORE_TYPE	Ananke
OTHER_CORE_TYPE	N/A
MODULE	N/A
MODULE_DEBUG_BLOCK	N/A
ACE	True
MASTER_DATA_WIDTH	128
ACP	False
PERIPH_PORT	True
SCU_CACHE_PROTECTION	True
L3_CACHE	False

DynamIQ Shared Unit Interfaces

This section describes the DSU interface signals.

Clock and Clock Enable Signals

The ***Clock Signals*** and ***Clock Enable Signals*** tables describe the clock and clock enable signals.

Table 1-6: Clock Signals

Signal Name	Description	Connection
SCLK	Clock for the SCU/L3 and the AMBA interface	CLKO2
PCLK	Clock for the debug peripheral interface and the timers, power management, and other miscellaneous logic	SYSCLK
ATCLK	Clock for the ATB trace interface	SYSCLK
GICCLK	Clock for the GIC interface	SYSCLK

The *Default Configuration* table describe the default configuration of the Cortex-A55.

Table 1-7: Default Configuration

Configuration	Status
Default Execution Mode in EL3	AArch64
Endianness configuration, controls the reset value of the SCTLR_EL3/SCTLR EE bit	Little Endian
Enable Thumb exceptions, controls the reset value of the SCTLR.TE bit	Enabled
Value read in ClusterID Affinity Level-2 field, MPIDR bits [23:16]	8'h0
Value read in ClusterID Affinity Level-3 field, MPIDR bits [39:32]	8'h0
Disables the Cryptographic Extensions	False
Globally disables the CPU interface logic and routes the external interrupt signals directly to the cores	False

GIC Signals

This section describes the GIC default configuration.

Table 1-8: GIC Default Configuration (Interrupts)

Signal Name	Description	Connection
$\overline{\text{FIQ}}$	Active-low, level-sensitive fast interrupt request	Disabled
$\overline{\text{IRQ}}$	Active-low, level-sensitive interrupt request	Disabled
$\overline{\text{VFIQ}}$	Active-low, level-sensitive virtual fast interrupt request	Disabled
$\overline{\text{VIRQ}}$	Active-low, level-sensitive virtual interrupt request	Disabled
IRIT I/F	Distributor to GIC CPU Interface messages	From GIC
ICC I/F	GIC CPU Interface to distributor messages	To GIC

WFE Wakeup Event Signals

This section describes the wakeup event signals

Table 1-9: WFE Wakeup Event Signals

Signal Name	Description	TRU Requester/ Completer ID
EVENTIREQ	Event input request for core wake-up from WFE state. It must remain asserted until EVENTIACK is asserted and must not be re-asserted until EVENTIACK is low.	TRU Completer 2
EVENTIACK	Event input request acknowledge. It is not asserted until EVENTIREQ is high, and then remains asserted until after EVENTIREQ goes low.	TRU Requester 8
EVENTOREQ	Event output request for core wake-up, triggered by SEV instruction. It is only asserted when EVENTOACK is low, and then remains high until after EVENTOACK goes high.	TRU Requester 9
EVENTOACK	Event output request acknowledge. It must not be asserted until EVENTOREQ is high, and then must remain asserted until after EVENTOREQ goes low.	TRU Completer 3

Error Signals

This section describes the error signals

Table 1-10: Error Signals

Signal Name	Description
$\overline{\text{ERRIRQ[0]}}$ A55_SCU_ERR (A55 SCU Error)	Active-low, level-sensitive error indicator for an ECC error that causes potential data corruption or loss of coherency in snoop filter RAMs or ACE write transactions with a write response condition. It is output from the SCLK domain.
$\overline{\text{ERRIRQ[1]}}$ A55_L1L2_ERR (A55 L1,L2 Error)	Active-low, level-sensitive error indicator for an ECC error that causes potential data corruption or loss of coherency for the L1 and L2 caches in the core. The bit is output from the PERIPHCLK domain.

Debug Block APB Interface Signals (Debug)

The interface between the debug block and the DSU consists of a pair of APB interfaces, one in each direction. The following table shows the signals between the cluster APB controller and the debug block APB target.

Table 1-11: Debug Block APB Interface Signals (Debug)

Signal Name	Description
$\overline{\text{COMMIRQ}}$	Active-low, level-sensitive comms channel receive or transmit interrupt request
DBGEN	Invasive debug enable
NIDEN	Non-invasive debug enable
SPIDEN	Secure privilege invasive debug enable
SPNIDEN	Secure privilege non-invasive debug enable

Performance Monitor Unit (PMU)

The Cortex-A55 core includes performance monitors that can be configured to gather various statistics on the operation of the core and its memory system during runtime. These provide useful information about the behavior of the code that can used for debug and code profiling. The PMU provides six counters. Each counter can count any of the events available in the core.

For complete register information refer to the *Arm Cortex-A55 Core Technical Reference Manual* and the *Arm Cortex-A Series Programmer's Guide for ARMv8-A*.

Functional Description

This section describes the functionality of the PMU. The PMU includes the following interfaces and counters:

- Events from all other units from across the design are provided to the PMU
- The PMU registers can be programmed using the system registers or the external peripheral interface
- The PMU has 32-bit counters that increment when they are enabled, based on events, and a 64-bit cycle counter
- The Cortex-A55 core supports access to the performance monitor registers from the internal system register interface and a memory-mapped interface

PMU Events

The PMU event counters can gather various statistics on the operation of the core and its memory system, like cycle count, cache miss/hit, memory read/write, load and prefetch instructions. For the all the PMU events, please refer to *Arm Cortex-A55 Core Technical Reference Manual*.

PMU Interrupts

The Cortex-A55 core asserts the nPMUIRQ signal when the PMU generates an interrupt. The active-low, level sensitive PMU interrupt signal is routed to the GIC as a private peripheral interrupt.

Interrupt ID	Name	Description	Sensitivity
23	PMU_IRQ	PMU interrupt request	Level

2 Clock Generation Unit (CGU)

The Clock Generation Unit (CGU) includes the phase locked loop (PLL) and the PLL control unit (PCU). The PLL generates a requester clock that runs at a frequency that is a multiple of the CLKIN input clock frequency. The PCU divides down the requester clock to generate various system clocks and synchronization signals.

CGU Features

The CGU module supports the following features:

- Provides smooth transitions from the current clock condition to a new condition with PLL logic and executes the changes to clocks due to register programming.
- Provides PLL and clock domain status reporting for event management.
- Supports the capability to bypass the PLL for power savings.
- Software controlled dynamic power management allows control of the core clock frequency (f_{CCLK}).
- Controls clock gating of the core and system clocks.

NOTE: For more information about processor-specific CGU features, see the processor data sheet.

CGU Functional Description

The CGU generates all on-chip clocks and synchronization signals based on the programmed PLL multiplication factor and dividers. The CGU provides the following functionality.

Change the PLL Clock Frequency

The CGU allows programs to change the PLL clock frequency by writing new values to bits in the control register. Any time the PLL relocks, the CGU aligns all core and system clocks.

Change Other Clock Frequencies

The CGU allows programs to change the CCLKn, SYSCLK, SCLKn, DCLK, and OCLK frequencies by writing values to the CGU_DIV register. Any time the clock frequency is changed, the OCLK, CCLKn, SYSCLK, DCLK, and SCLKn clocks exit the frequency change sequence aligned.

Perform Clock Alignment

The CGU can align all clocks by writing to the CGU_DIV register. This function aligns all PLL-based clocks.

For more information on these functions, see the CGU Programming Model section.

ADSP-SC59x CGU Register List

The clock generation unit (CGU) includes the phase locked loop (PLL) and the PLL control unit (PCU). The PLL generates a clock, running at a frequency that is a multiple of the CLKIN input clock's frequency. The CGU also generates all on-chip clocks and synchronization signals. The PCU permits application software control of the PLL's operation. A set of registers govern CGU operations. For more information on CGU functionality, see the CGU register descriptions.

Table 2-1: ADSP-SC59x CGU Register List

Name	Description
CGU_CCBF_DIS	Core Clock Buffer Disable Register
CGU_CCBF_STAT	Core Clock Buffer Status Register
CGU_CLKOUTSEL	CLKOUT Select Register
CGU_CTL	Control Register
CGU_DIV	Clocks Divisor Register
CGU_DIVEX	DIV Register Extension
CGU_OSCWDCTL	Oscillator Watchdog Register
CGU_PLLCTL	PLL Control Register
CGU_REVID	Revision ID Register
CGU_SCBF_DIS	System Clock Buffer Disable Register
CGU_SCBF_STAT	System Clock Buffer Status Register
CGU_STAT	Status Register
CGU_TSCOUNT0	Time Stamp Counter 32 LSB Register
CGU_TSCOUNT1	Time Stamp Counter 32 MSB Register
CGU_TSCTL	Time Stamp Control Register
CGU_TSVALUE0	Time Stamp Counter Initial 32 LSB Value Register
CGU_TSVALUE1	Time Stamp Counter Initial MSB Value Register

ADSP-SC59x CGU Interrupt List

Table 2-2: ADSP-SC59x CGU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
1	CGU0_EVT	CGU0 Event	Edge	
2	CGU1_EVT	CGU1 Event	Edge	

ADSP-SC59x CGU Trigger List

Table 2-3: ADSP-SC59x CGU Trigger List Generators

Trigger ID	Name	Description	Sensitivity
1	`CGU0_EVT`	CGU0 Event	Edge
2	`CGU1_EVT`	CGU1 Event	Edge

Table 2-4: ADSP-SC59x CGU Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

CGU Definitions

DPM

The Dynamic Power Management module (DPM) works with the CGU to provide flexible power dissipation modes for the processor.

PCU

The PLL control unit (PCU) in the CGU controls PLL operations. The MMR registers of the CGU are implemented in the PCU.

PLL

The phase-locked loop (PLL) operates within the CGU.

RCU

The reset control unit (RCU) provides input to the CGU to manage clocks during processor reset.

CDU

The clock distribution unit distributes the clocks from the CGU to different clock domains

CGU

The clock generation unit (CGU) is comprised of the PLL and PCU. The CGU generates the clocks listed in the ***Clock Descriptions*** table.

Table 2-5: Clock Descriptions

Clock	Description
CCLK0_0	CCLK0 derived from CGU0
CCLK1_0	CCLK1 derived from CGU0
CCLK2_0	CCLK2 derived from CGU0 *1
SYSCLK_0	SYSCLK derived from CGU0
SCLK0_0	SCLK0 derived from CGU0
SCLK1_0	SCLK1 derived from CGU0
DCLK_0	DCLK derived from CGU0
OCLK_0	OCLK derived from CGU0
CCLK0_1	CCLK0 derived from CGU1
CCLK2_1	CCLK2 derived from CGU1*1
SYSCLK_1	SYSCLK derived from CGU1
SCLK0_1	SCLK0 derived from CGU1
SCLK1_1	SCLK1 derived from CGU1
DCLK_1	DCLK derived from CGU1
OCLK_1	OCLK derived from CGU1
DCLK_2	DCLK derived from CGU2

*1. CCLK2_x is directly derived from the VCO clock; it is divided by 3. Hence, the frequency of CCLK2 = (PLL clock x 2/3).

CGU PLL Block Diagram

The ***CGU PLL Block Diagram*** provides a top-level block diagram of the phase locked loop (PLL). The main blocks of the PLL are the phase/frequency detector (PFD), the charge pump, the loop filter and the voltage controlled oscillator (VCO). The VCO multiplies the `SYS_CLKIN0` or `SYS_CLKIN1`, input to a higher frequency.

Additional configuration options are configured using the Clock Distribution Unit.

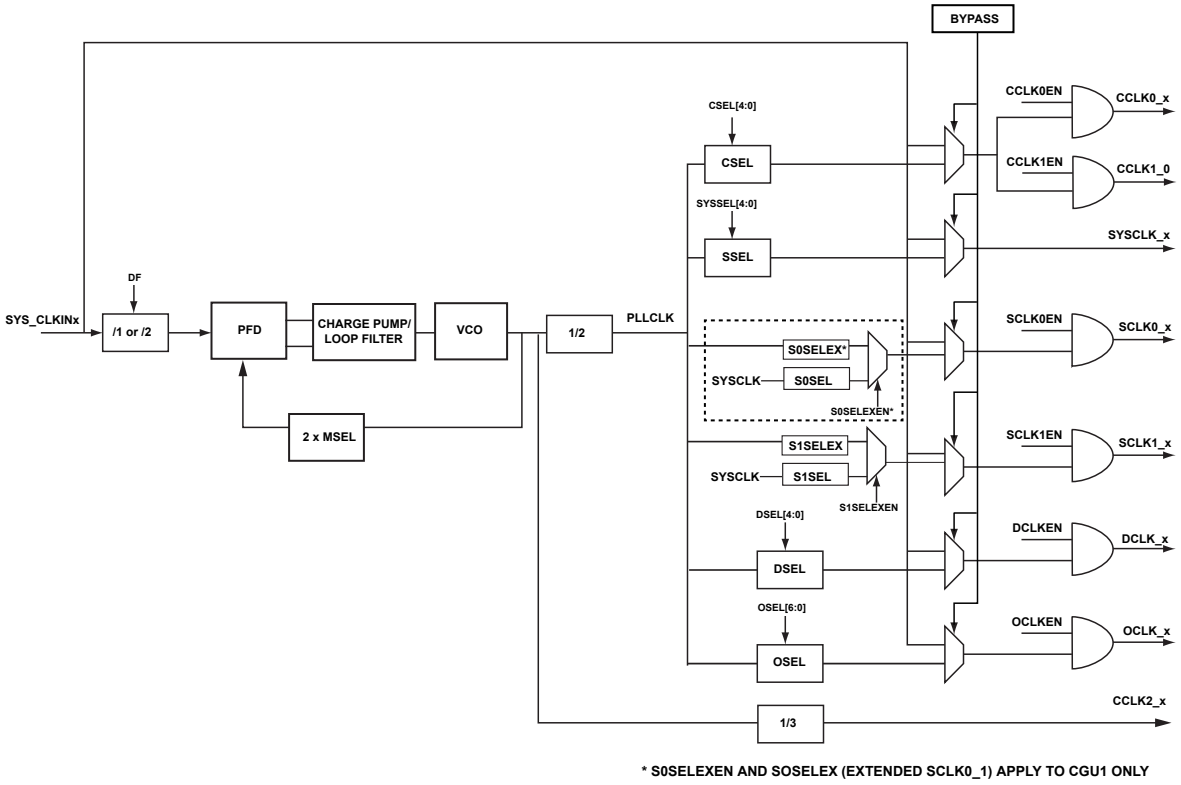


Figure 2-1: CGU1 PLL Block Diagram

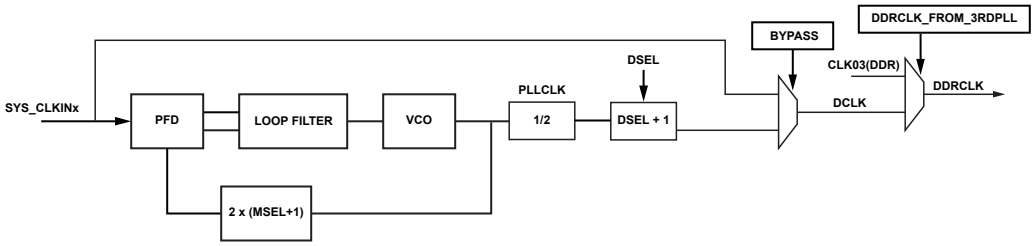


Figure 2-2: CGU2 PLL Block Diagram

CLKOUT Selections

The ***SYS_CLKOUT Generation*** figure is a conceptual representation of the CLKOUT module. Different clocks that originate from the CGU blocks are available on the SYS_CLKOUT output pin. The selection of the clock output on the SYS_CLKOUT pin is controlled by the `CGU_CLKOUTSEL.CLKOUTSEL` bit field. Most of the clock frequencies are divided with a fixed divider value, for example CGU0_SYSCLK0 has a fixed divider of 8 for the CLKOUT pin.

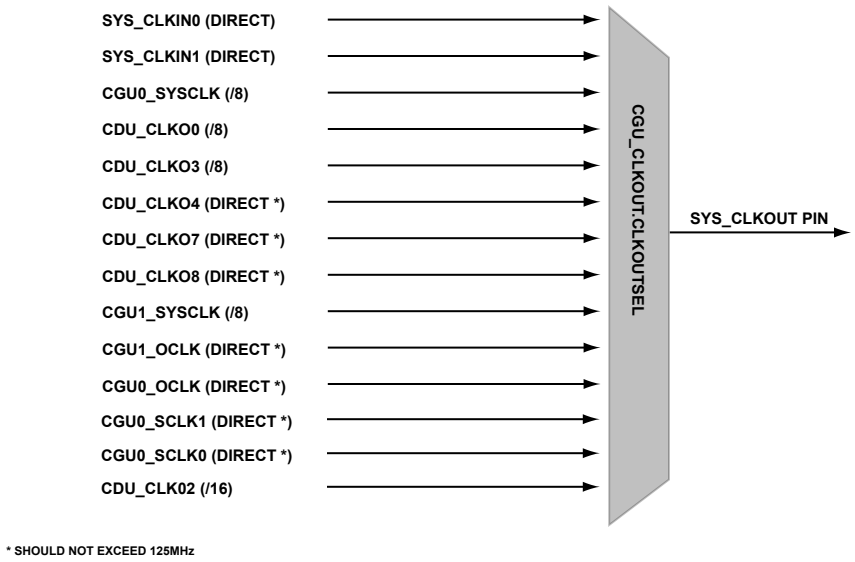


Figure 2-3: SYS_CLKOUT Generation

The BMODE pin affects the CLKOUT pins after a hardware reset is deasserted.

BMODE= 0 — When a hardware reset is deasserted, CGU0_OCLK/PLLCLK0 is selected to come out of the CLKOUT pin. The default value of OSEL and MSEL is the same. This functionality enables the user to verify PLL functionality by checking that the CLKOUT period is identical to CLKIN period. While programming all values of the `CGU_CLKOUTSEL.CLKOUTSEL` bit field is possible, changing it to 0 (SYS_CLKIN) is not possible until and unless a non-zero value is first programmed to the `CGU_CLKOUTSEL.CLKOUTSEL` bit field.

BMODE = (non-zero) — When a hardware reset is deasserted, SYS_CLKIN is selected by default.

CGU Operating Modes

The CGU does not have configurable operating modes, but CGU operations affect the operating modes of other modules. Some CGU operation issues that affect operation of other modules include the following:

- The PLL of the CGU operates in either normal mode (CGU clock divisors applied) or bypass mode (CGU PLL is bypassed and clock divisors ignored).
- The SCB uses the CGU for clock synchronization across clock domains. For more information, see System Crossbars (SCB).
- The DPM uses the CGU for clock management as power state transitions occur. For more information, see the Dynamic Power Management (DPM) chapter.
- The CGU uses clock gating control to obtain flexible low-power modes.

CGU Power-up Sequence

See the product data sheet for exact power-up requirements. The processor is configured to come up in clock bypass mode. The programs is required to configure full speed clocks and safety monitors. SYS_CLKIN0 or SYS_CLKIN1, and all supplies should be stable before the SYS_HWRST signal is deasserted.

CGU Event Control

The CGU generates an event or error for several different reasons. Events and errors are described in the following sections.

Events

After a frequency change, a CGU event indicates that the PLL has locked, and clocks are synchronized. If a core was idled while changing frequencies, the CGU can use an event interrupt to break the core idle. While in active mode, a CGU event indicates that the PLL has locked.

CGU Errors

A CGU error occurs under following conditions:

- A write access to the CGU_DIV/CGU_DIVEX registers trigger an alignment sequence while the PLL is locked and is still aligning the clocks.

  The CGU_STAT.WDIVERR bit state indicates this error. If this error occurs, clear the CGU_STAT.WDIVERR bit and rewrite the desired values to the CGU_DIV/CGU_DIVEX registers.

- A change to the CGU_DIV/CGU_DIVEX registers occur while the PLL is locked and is still aligning the clocks.

  The CGU_STAT.WDIVERR bit state indicates this error. If this error occurs, clear the CGU_STAT.WDIVERR bit and rewrite the desired values to the CGU_DIV/CGU_DIVEX registers.

- A write access to the CGU_CTL.DF bit field occurs or a write access to the CGU_CTL.MSEL bit field occurs while the PLL is locking.

  The CGU_STAT.WDFMSERR bit state indicates this error. If this error occurs, wait until the PLL has finished locking, cleared the error, and rewritten the desired value change.

- A write access to CGU_DIV.S1SEL when CGU_CTL.S1SELEXEN is enabled or a write access to CGU_DIVEX.S1SELEX when CGU_CTL.S1SELEXEN is disabled.

  A write access to CGU_DIV.S0SEL when CGU_CTL.S0SELEXEN is enabled or a write access to CGU_DIVEX.S0SELEX when CGU_CTL.S0SELEXEN is disabled.

  The CGU_STAT.WDIVERR bit state indicates this error. If this error occurs, clear the CGU_STAT.WDIVERR bit and rewrite the desired values to the CGU_DIV and CGU_DIVEX registers.

The CGU monitors changes to the following fields:

- SYSCLK Divisor – CGU_DIV.SYSSEL

CGU Generated Bus Errors

The CGU generates a bus error when a read or write transaction to an unused address within the CGU address range is attempted. It also generates a bus error when a misaligned access is made to a CGU register. In addition to the bus error, the CGU_STAT.ADDRERR bit is set. If a write to a write-protected CGU register is attempted, the CGU generates a bus error. In addition, the CGU_STAT.LWERR bit is set.

Oscillator Watchdog

The oscillator watchdog detects the absence of input clock transitions and provides a fault warning through the SYS_FAULT pin. To detect harmonic or subharmonic crystal oscillator behavior, the watchdog (under programmable control) also detects and reports input oscillator frequencies above and below the specified limits. Use an internal asynchronous, local 1 MHz oscillator combined with a series of programmable counters for this detection. Set the CGU_OSCWDCTL.MONDIS bit and clear the CGU_OSCWDCTL.FAULTEN bit to optionally disable all the input clock monitor and fault detection functions.

Set the CGU_OSCWDCTL.HODEN bit to enable harmonic oscillation detection. The CGU uses the CGU_OSCWDCTL.HODF bit field to indicate the desired lower fail limit (in MHz) for the harmonic oscillation detection. The upper limit is always twice the lower limit.

The ***HODF Settings for Different Input Clock Frequencies*** table shows the recommended values of the CGU_OSCWDCTL.HODF bit settings for different input clock frequencies.

Table 2-6: HODF Settings for Different Input Clock Frequencies

CGU_OSCWDCTL.HODF[5:0]	**Subharmonic Frequency (MHz)**	**Nom. Lower Fail Limit (MHz)**	**Input Clock Frequency (MHz)**	**Nom. Upper Fail Limit (MHz)**	**Second Harmonic Frequency (MHz)**
16	10	16	20	32	40
16	12.5	16	25	32	50
20	15	20	30	40	60

NOTE: Use a HODF value of 16 for frequency ranges greater than 20 MHz and less than 25 MHz. For frequency ranges greater than 25 MHz and less than 30 MHz, use a HODF value of 20.

The CGU uses the CGU_OSCWDCTL.BOUF (Bad Oscillator Upper Frequency limit) asynchronous control bit field to indicate the desired upper fail limit for the bad oscillation detection. Set the CGU_OSCWDCTL.BOUEN bit to enable upper-limit bad oscillation detection. A bad oscillation detection condition signals a fault before any processor operations occur. This detection occurs (even in bypass mode) whenever a clock frequency exceeds the specifications.

The CGU_OSCWDCTL.BOUF =0 operation starts with a target of 32 MHz and each additional LSB increases the frequency test limit by 2 MHz. For example:

Target Upper Frequency Limit = CGU_OSCWDCTL.BOUF × 2 MHz + 32 MHz

The CGU_STAT.OSCWDSTATFC status bits indicate the nature of the fault.

The ***Fault Map*** table shows the fault values.

Table 2-7: Fault Map

CGU_STAT.OSCWDSTATFC Bitfield Values	**Fault Type**
0	No Fault
1	No Input Clock
2	Subharmonic CLKIN
3	Harmonic CLKIN
4	No AUX_CLK
5	CLKIN > Upper Freq Limit (BOUF)
6	Reserved
7	Multiple Limit Faults

There is a priority to the faults given in the case of multiple fault errors. The highest priority is given to No Input clock followed by No AUX_CLK. The other three fault cases share the lowest priority. Multiple limit faults are asserted if more than one type of subharmonic CLKIN, harmonic CLKIN, or BOUF faults are observed.

NOTE: All the CGU_STAT.OSCWDSTATFC faults other than the absence of AUX_CLK (for example, CGU_STAT.OSCWDSTATFC =4) are not reliable and used for debug only.

Program and enable the OSCWDOG to match the actual crystal, before bringing the PLL out of bypass.

CGU Programming Model

The programming model for the CGU involves the various mode configuration techniques.

Configuring CGU Modes

Use the following procedures to configure the clocks and PLL.

NOTE: The program needs to clear the CGU_STAT.CLKSALGN bit before changing clocks. The following sequence is executed once, inside the application, after coming out of reset.

```
*pREG_CGU0_PLLCTL |= BITM_CGU_PLLCTL_PLLBPCL; // come out of bypass and enter
Full ON
while( (pADI_CGU0 ->STAT & 0xF) != 0x5 ) { } // poll
// now clocks are running with hardware default divisors.
// now program can change frequencies If desired the program can put the PLL
again into bypass.
```

Changing the Clock Frequencies

Applications change clock frequencies in two ways. The first way is modifying the PLL multiplication value by writing to the CGU_CTL register and the second is modifying the clock dividers by writing to the CGU_DIV register. Both actions have different implications even if the frequencies of the final clock are the same. Write accesses to change the CGU_CTL.DF or CGU_CTL.MSEL bit fields while the PLL is locking set the CGU_STAT.WDFMSERR error bit. The CGU_STAT.WDIVERR error bit is set when one of following accesses is attempted while the PLL is locked, but still aligning the clocks:

- A write access to the CGU_DIV/CGU_DIVEX to trigger an alignment sequence.
- A write access to the CGU_DIV/CGU_DIVEX to change the CGU_DIV.CSEL/, CGU_DIV.SYSSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, or CGU_DIV.DSEL/CGU_DIVEX.S1SELEX, CGU_DIVEX.S0SELEX bits.

Read-after-write accesses to these registers return the new value, even if the frequency of the clock change is still in-progress.

Modifying the PLL multiplier requires the PLL to relock. Once the PLL locks, the CGU synchronizes the clocks. Changes to the CGU_CTL.DF or CGU_CTL.MSEL bit field result in bypassing the PLL. If necessary, clear the CGU_DIV.UPDT bit to avoid multiple clock alignment sequences. If the CGU_DIV register is not updated, the CGU uses the current values to determine the frequencies of the clock. It is the programs responsibility to guarantee that the new CGU_CTL.DF or CGU_CTL.MSEL and CGU_DIV combinations are legal.

Changing the PLL Clock Frequency

To change the phase-locked loop clock (*PLLCLK*) frequency, write new values to the CGU_CTL.MSEL field or CGU_CTL.DF field. Any time the PLL relocks, all core and system clocks are aligned.

1. Read CGU_STAT register and verify that:
   a. The CGU_STAT.PLLEN bit =1 (PLL enabled)
   b. The CGU_STAT.PLOCK bit =1 (PLL is not locking)
   c. The CGU_STAT.CLKSALGN bit =0 (clocks aligned)
2. Write the desired values to the clock divisor select fields of the CGU_DIV/CGU_DIVEX register with the CGU_DIV.UPDT bit =0.
3. Write the desired values to the CGU_CTL.DF and CGU_CTL.MSEL fields.

This sequence performs these actions:

1. Updates the corresponding CGU registers.
2. Bypasses the PLL.
3. Makes the PLL lock to the new values in the CGU_CTL.MSEL or CGU_CTL.DF fields.
4. Changes the clock frequencies.

5. Exits the PLL bypass with all clocks aligned.

When exiting the PLL bypass state, a CGU event occurs.

The CGU_STAT register exits this sequence with the CGU_STAT.PLLEN bit =1, the CGU_STAT.PLOCK bit =1, the CGU_STAT.PLLBP bit =0, and the CGU_STAT.CLKSALGN bit =0. Poll the CGU_STAT.PLOCK bit, CGU_STAT.PLLBP bit, and CGU_STAT.CLKSALGN bit to discover when the PLL is locked and the clocks are aligned.

Changing the frequency of the PLL is allowed while the PLL is bypassed. In this case the new PLLCLK frequency is not used until the PLL is no longer bypassed.

Changing the CCLKn or SYSCLK Frequency Without Modifying the PLLCLK Frequency

To change the clock frequencies, write new values to CGU_DIV.CSEL or CGU_DIV.SYSSEL bits. The frequency change occurs only when the PLL is not bypassed. Any time the CCLKn or SYSCLK clock frequencies are changed, they exit the frequency change sequence aligned.

1. Read the CGU_STAT register to verify that the CGU_STAT.CLKSALGN bit =0 (clocks aligned).
2. Write the desired CGU_DIV.CSEL, CGU_DIV.SYSSEL, and CGU_DIV.OSEL bitfield values with the CGU_DIV.UPDT bit = 1 and/or, write to the CGU_DIVEX.S1SELEX field. This write updates the CGU_DIV register, changes the SCLKn and SYSCLK frequencies, and aligns the clocks. When the clocks are aligned, a CGU event occurs.

The CGU_STAT register exits this sequence with the CGU_STAT.CLKSALGN bit =0. Poll the CGU_STAT.CLKSALGN bit to discover when the clocks are aligned. Any write attempt to change the CGU_DIV.S0SEL or CGU_DIV.S1SEL bit fields while CGU_STAT.CLKSALGN bit =1 (clocks alignment in progress) triggers an MMR access bus error and the CGU_DIV register is not modified.

Also, any write attempt to change the CGU_DIVEX.S1SELEX field while the CGU_STAT.CLKSALGN bit =1 (clocks alignment in progress) triggers an MMR access bus error and the CGU_DIVEX register is not modified.

Programming the SYSCLK frequency to a higher value than CCLKn also triggers an MMR access bus error and the CGU_DIV/CGU_DIVEX register is not modified.

Writing to the CGU_DIV/CGU_DIVEX register is allowed while the processor is in active (PLL bypassed) mode. In this case the effect of the write is visible only after the transition to full-on (PLL not bypassed) mode.

Accessing the DDR memory while changing the SYSCLK frequency is not supported and can have unpredictable results.

Changing the OCLK Frequency

To change the OCLK clock frequency, write a new CGU_DIV.OSEL bit value. Any time the OCLK clock frequency is changed, the OCLK, CCLKn, SYSCLK, and SCLKn clocks exit the frequency change sequence aligned.

1. Read the CGU_STAT register to verify that the CGU_STAT.CLKSALGN bit =0 (clocks aligned).
2. Write the desired CGU_DIV.OSEL value with the CGU_DIV.UPDT bit =1. This write updates the CGU_DIV register, changes the OCLK frequency, and aligns all clocks except OCLK.

The CGU_STAT register exits this sequence with the CGU_STAT.CLKSALGN bit =0. Poll the CGU_STAT.CLKSALGN bit to discover when the clocks are aligned. Any write attempt to change the CGU_DIV.DSEL field while the CGU_STAT.CLKSALGN bit =1 (clock alignment in progress) triggers an MMR access bus error and the CGU_DIV/CGU_DIVEX is not modified. When the clocks are aligned, a CGU event occurs.

Writing to the CGU_DIV.OSEL bit field is allowed while the processor is in active (PLL bypassed) mode. In this case the effect of the write is visible only after the transition to full-on (PLL not bypassed) mode.

Selecting SCLK1_X and SCLK0_X Source

Programming the CGU_DIVEX enables the PLL clock as a source (bypassing SYSCLK0_0 that continues to source other peripherals). When CGU_DIVEX is enabled, the PLL clock is divided from CGU_DIVEX register value.

Changing SCLK1_X or SCLK0_X Frequencies

Complete the following steps to change the SCLK0 or SCLK1 frequency.

1. Read the CGU_STAT register. Verify that CGU_STAT.CLKSALGN = 0 (Clocks aligned).
2. Read the CGU_CTL register. Verify that the CGU_DIVEX.S1SELEX or CGU_DIVEX.S0SELEX bit is set as needed. (These bits can only be modified when the PLL is bypassed).
3. Read the CGU_STAT register. Verify that the PLL is not bypassed. (If it is bypassed, see PLL Bypass and PLL Disable topic.)
4. Read the CGU_DIV register. Verify that the CGU_DIV.UPDT bit is set.
5. Write the desired CGU_DIVEX.S1SELEX or CGU_DIVEX.S0SELEX value to the CGU_DIVEX registers .

Aligning All Clocks

To align the clocks, write 1 to the CGU_DIV.ALGN bit. The frequency can also be changed, if necessary. The clocks aligned include:

- CCLKn
- SYSCLK
- SCLKn
- DCLK
- OCLK

1. Read the CGU_STAT register to verify that CGU_STAT.CLKSALGN bit =0 (clocks aligned).
2. Write 1 to the CGU_DIV.ALGN bit. All other fields can change.

   ADDITIONAL INFORMATION: This write does not alter the CGU_DIV/CGU_DIVEX register unless one of the clock-select fields is modified. When the clocks are aligned, a CGU event occurs.

The CGU_STAT register exits this sequence with the CGU_STAT.CLKSALGN bit =0. Poll the CGU_STAT.CLKSALGN bit to discover when the clocks are aligned. Any write to the CGU_DIV/CGU_DIVEX register intended to align clocks or to change a clock select field while the CGU_STAT.CLKSALGN bit =1 (clocks alignment in progress) triggers an MMR access bus error. In this case, the CGU_DIV/CGU_DIVEX register is not modified.

Writing 1 to the CGU_DIV.ALGN bit has no effect while the processor is in active (PLL bypassed) mode.

Shutting Off CCLKn from Another Controller

CCLKn can be shut off to save power when it is not in use.

1. Disable interrupts to core n.
2. Set the RCU_SIDIS.SI[n] bit to disable the interfaces of core n to:
   a. Stop DMA accesses to its L1
   b. Stop memory accesses to core 0
   c. Stop accesses to MMRs
3. Test the RCU_SISTAT.SI[n] bit to detect when accesses to core n have been disabled and all the pending transactions have completed.
4. Set the CGU_CCBF_DIS.CCBF0 bit to disable the CCLKn buffer.
5. Check the CGU_CCBF_STAT.CCBF0 bit.

If the CGU_CCBF_STAT.CCBF0 bit is set, continue.

Reenable CCLKn From Another Controller

1. Clear the CGU_CCBF_DIS.CCBF0 to enable CCLKn.
2. Check the CGU_CCBF_STAT.CCBF0 bit.
   a. If the CGU_CCBF_DIS.CCBF0 bit is cleared, continue
3. Clear the RCU_SIDIS.SI[n] bit. The core deasserts its acknowledge signal in response to the RCU_SYSRST0signal. This operation clears the RCU_SISTAT.SI[n] bit.

Valid Clock Multiplier Settings

Processor operations depend on valid settings in the CGU_CTL and CGU_DIV registers. These registers control the clock multiplier and divisor values. Set these registers such that the minimum and maximum clocks specified in the data sheet are not violated. All other clock specifications in the data sheet must also be adhered to for correct operation of the processor.

PLL Bypass and PLL Disable

Writing 1 to the CGU_PLLCTL.PLLBPST bit tells the PLL to apply OSC_CLKIN clock to CCLK, SYSCLK, SCLK0, SCLK1, DCLK (PLL Bypass), and OCLK outputs. Writing 1 to the CGU_PLLCTL.PLLBPCL bit tells the PLL to exit its PLL Bypass state and make all output clocks align and transition to their programmed frequencies.

The PLL can be disabled by clearing the CGU_PLLCTL.PLLEN bit while in the bypass state. If necessary, clock buffers can be disabled.

CCLK0 clock can be disabled or enabled by writing 1 or 0 to the corresponding bit in the CGU_CCBF_DIS register.

To determine which core clock buffers were disabled since the last read, software reads the CGU_CCBF_STAT register. The SCLK0, SCLK1, DCLK and OCLK clocks can be disabled or enabled by writing 1 or 0 to the corresponding bit in the CGU_SCBF_DIS register. Software cannot disable SYSCLK.

DCLK from CGU2

PLL relocking with MSEL update:

1. Clear the MISCREG_PLL2_CONTROL.BYPASSB, MISCREG_PLL2_CONTROL.DSEL_DIV_CHG and MISCREG_PLL2_CONTROL.MSELDF_CHG bits.
2. Wait 20 core clock cycles for the PLL of the third SYSCLK to reflect CLKIN.
3. Update the MISCREG_PLL2_CONTROL.MSEL value and set the MISCREG_PLL2_CONTROL.MSELDF_CHG bit.
4. Wait 768 core clock cycles for the MSELDF_CHG pulse.
5. Wait 2048 CLKIN cycles for the PLL to lock.
6. Set the MISCREG_PLL2_CONTROL.BYPASSB bit.
7. Set the MISCREG_PLL2_CONTROL.BYPASSB bit

DCLK divider update:

1. Clear the MISCREG_PLL2_CONTROL.BYPASSB, MISCREG_PLL2_CONTROL.DSEL_DIV_CHG, and MISCREG_PLL2_CONTROL.MSELDF_CHG bits.
2. Update the MISCREG_PLL2_CONTROL.DSEL value and set the MISCREG_PLL2_CONTROL.DSEL_DIV_CHG bit.

3. Wait 768 core clock cycles for the MSELDF_CHG pulse.
4. Set the MISCREG_PLL2_CONTROL.BYPASSB control bit.
5. Set the MISCREG_PLL2_CONTROL.DDRCLK_FROM_3RDPLL bit to connect the third PLL DCLK to DMC.

   This bit determines whether the DCLK is sourced from the third PLL.

ADSP-SC59x CGU Register Descriptions

Clock Generation Unit (CGU) contains the following registers.

Table 2-8: ADSP-SC59x CGU Register List

Name	Description
CGU_CCBF_DIS	Core Clock Buffer Disable Register
CGU_CCBF_STAT	Core Clock Buffer Status Register
CGU_CLKOUTSEL	CLKOUT Select Register
CGU_CTL	Control Register
CGU_DIV	Clocks Divisor Register
CGU_DIVEX	DIV Register Extension
CGU_OSCWDCTL	Oscillator Watchdog Register
CGU_PLLCTL	PLL Control Register
CGU_REVID	Revision ID Register
CGU_SCBF_DIS	System Clock Buffer Disable Register
CGU_SCBF_STAT	System Clock Buffer Status Register
CGU_STAT	Status Register
CGU_TSCOUNT0	Time Stamp Counter 32 LSB Register
CGU_TSCOUNT1	Time Stamp Counter 32 MSB Register
CGU_TSCTL	Time Stamp Control Register
CGU_TSVALUE0	Time Stamp Counter Initial 32 LSB Value Register
CGU_TSVALUE1	Time Stamp Counter Initial MSB Value Register

Core Clock Buffer Disable Register

The CGU_CCBF_DIS register controls each core's clock buffer to determine if the CCLK is enabled.

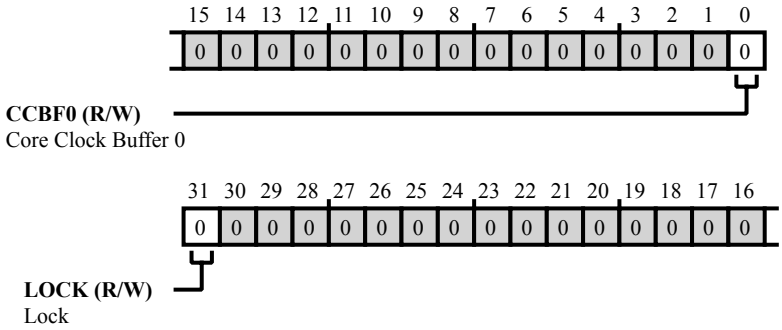


Figure 2-4: CGU_CCBF_DIS Register Diagram

Table 2-9: CGU_CCBF_DIS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If set (=1) the CGU_CCBF_DIS.LOCK bit locks the CGU_CCBF_DIS register.	
		0	Unlock register
		1	Lock register
0 (R/W)	CCBF0	Core Clock Buffer 0. The CGU_CCBF_DIS.CCBF0 bit enables (=0) or disables (=1) CCLK0s buffer.	
		0	Enable buffer
		1	Disable buffer

Core Clock Buffer Status Register

The CGU_CCBF_STAT register shows which core clock buffer(s) are disabled. For example clearing the CGU_CCBF_DIS.CCBF0 bit clears the CGU_CCBF_STAT.CCBF0 bit after a number of cycles. To guarantee that the correct value is read, this register should be read twice and the second result used.

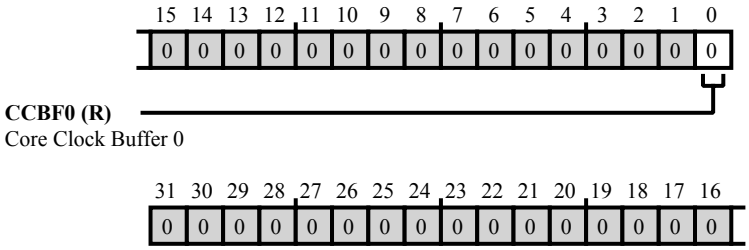


Figure 2-5: CGU_CCBF_STAT Register Diagram

Table 2-10: CGU_CCBF_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	CCBF0	Core Clock Buffer 0. The CGU_CCBF_STAT.CCBF0 bit reports the status of the CGU_CCBF_DIS.CCBF0 bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled

CLKOUT Select Register

The CGU_CLKOUTSEL selects the signal that the CGU drives through the CLKOUT multiplexer.

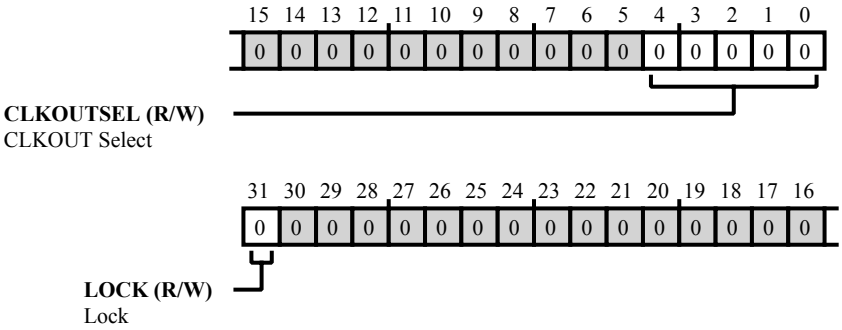


Figure 2-6: CGU_CLKOUTSEL Register Diagram

Table 2-11: CGU_CLKOUTSEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the CGU_CLKOUTSEL.LOCK bit is set, the CGU_CLKOUTSEL register is read only (locked).	
		0	Unlock
		1	Lock
4:0 (R/W)	CLKOUTSEL	CLKOUT Select. The CGU_CLKOUTSEL.CLKOUTSEL selects the signal that the CGU drives through the CLKOUT pin multiplexer.	
		0	CLKIN0
		1	CLKIN1
		2	PLLTOP0.SYSCLKO
		3	PLLTOP0.CCLK0
		4	CDU.CLKO3 for DDR
		5	CDU.CLKO4 for CAN
		6	CDU.CLKO7 for EMAC
		7	CDU.CLKO8 for LP
		8	PLLTOP1.SYSCLKO
		9	PLLTOP1.OCLK
		10	PLLTOP0.OCLK

Table 2-11: CGU_CLKOUTSEL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		11	Reserved
		12	Reserved
		13	Reserved
		14	PLLTOP0.SCLK1
		15	Reserved
		16	PLLTOP0.SCLK0
		17	Reserved
		18	Reserved
		19	CDU.CLKO2 for ARM Core
		20	GND

Control Register

The CGU_CTL controls the clock generation divisors for SYS_CLKIN and the PLL. Read after write accesses to the CGU_CTL register returns the new value even if the clock's frequency change is still in progress.

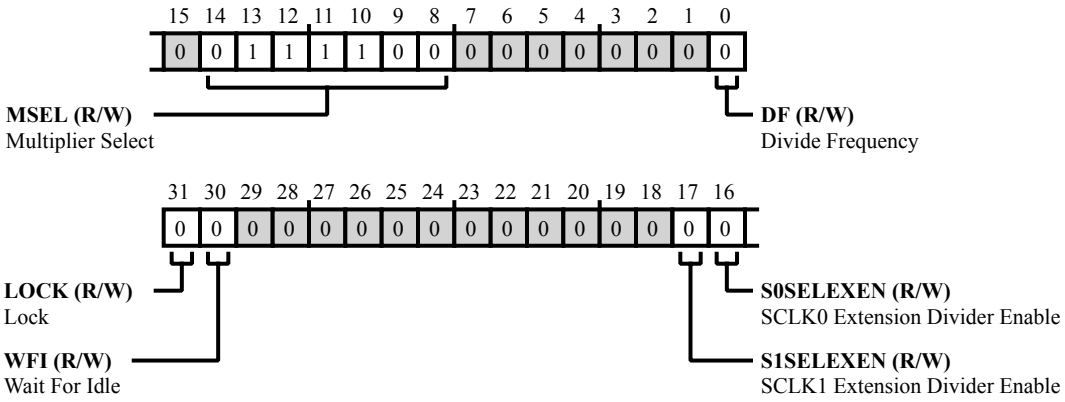


Figure 2-7: CGU_CTL Register Diagram

Table 2-12: CGU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the CGU_CTL.LOCK bit is set, the CGU_CTL register is read only (locked).	
		0	Unlock
		1	Lock
30 (R/W)	WFI	Wait For Idle. Modifying the PLL multiplier requires the PLL to re-lock and once the PLL locks, clocks have to be synchronized. Changes to the CGU_CTL.MSEL and the CGU_CTL.DF bit values results in bypassing the PLL. The CGU_CTL.WFI bit forces the PLL to wait for all processor cores to be in an idle or reset state before changing frequencies as a result of changes to the CGU_CTL.MSEL or CGU_CTL.DF bits. Write accesses to the CGU_CTL to change the CGU_CTL.DF or CGU_CTL.MSEL bit values while the PLL is locking sets the CGU_STAT.WDFMSERR bit.	
		0	Update Immediately
		1	Wait for Idle

Table 2-12: CGU_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	S1SELEXEN	SCLK1 Extension Divider Enable. The `CGU_CTL.S1SELEXEN` directs the CGU to select S1SELEX instead of S1SEL. This bit can only be updated when the PLL is bypassed.	
		0	Selects SYSCLK to SCLK1 Divider
		1	Selects PLLCLK to SCLK1 Divider
16 (R/W)	S0SELEXEN	SCLK0 Extension Divider Enable. The `CGU_CTL.S0SELEXEN` directs the CGU to select S0SELEX instead of S0SEL. This bit can only be updated when the PLL is bypassed.	
		0	Selects SYSCLK to SCLK0 Divider
		1	Selects PLLCLK to SCLK0 Divider
14:8 (R/W)	MSEL	Multiplier Select. The `CGU_CTL.MSEL` bit field selects the multiplier in the PLLCLK equation: PLLCLK frequency = (SYS_CLKIN frequency / (DF+1)) * MSEL Where the value of MSEL is between 1 and 127.	
		0	MSEL = 128
		1-127	MSEL = 1 to 127
0 (R/W)	DF	Divide Frequency. The `CGU_CTL.DF` bit selects whether or not the CLKIN input is divided by two before being passed to the PLL.	
		0	Pass OSC_CLKIN to PLL
		1	Pass OSC_CLKIN/2 to PLL

Clocks Divisor Register

The CGU_DIV register controls clock divisors for core clocks, system clocks, external (off core) memory clocks, and output clock. Read after write accesses to the CGU_DIV register returns the new value even if the clock's frequency change is still in progress.

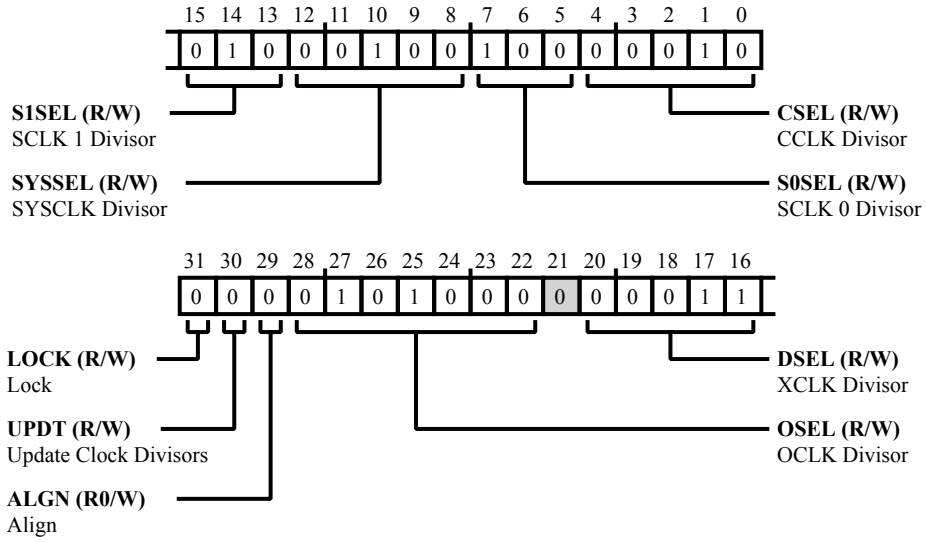


Figure 2-8: CGU_DIV Register Diagram

Table 2-13: CGU_DIV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the CGU_DIV.LOCK bit is set, the CGU0_DIV and CGU0_DIVEX registers are read only (locked).	
		0	Unlock
		1	Lock
30 (R/W)	UPDT	Update Clock Divisors. The CGU_DIV.UPDT controls whether the CGU drives new CGU_DIV.CSEL, CGU_DIV.SYSSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, CGU_DIV.DSEL, and CGU_DIV.OSEL values to PLL after CGU_DIV register update.	
		0	No PLL Update
		1	Drive Updated SEL Values to PLL

Table 2-13: CGU_DIV Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R0/W)	ALGN	Align. The CGU_DIV.ALGN directs the CGU to align the PLL-based clocks. The divisor selections (CGU_DIV.CSEL, CGU_DIV.SYSSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, CGU_DIV.DSEL, and/or CGU_DIV.OSEL) do not have to change.	
		0	No Action
		1	Align PLL Clocks
28:22 (R/W)	OSEL	OCLK Divisor. The CGU_DIV.OSEL selects the divisor in the OCLK equation: OCLK frequency = (SYS_CLKIN frequency / (DF+1)) * MSEL / CGU_DIV.OSEL Where the value of CGU_DIV.OSEL is between 1 and 127.	
		0	OSEL = 128
		1-127	OSEL = 1 to 127
20:16 (R/W)	DSEL	XCLK Divisor. The CGU_DIV.DSEL selects the divisor in the XCLK equation: XCLK frequency = (SYS_CLKIN frequency/(DF+1)) MSEL/CGU_DIV.DSEL Where the value of CGU_DIV.DSEL is between 1 and 31.	
		0	DSEL = 32
		1-31	DSEL = 1 to 31
15:13 (R/W)	S1SEL	SCLK 1 Divisor. The CGU_DIV.S1SEL selects the divisor in the SCLK1 equation: SCLK1 frequency = (SYSCLK frequency) / CGU_DIV.S1SEL Where the value of CGU_DIV.S1SEL is between 1 and 7.	
		0	S1SEL = 8
		1-7	S1SEL = 1 to 7
12:8 (R/W)	SYSSEL	SYSCLK Divisor. The CGU_DIV.SYSSEL selects the divisor in the SYSCLK equation: SYSCLK frequency = (SYS_CLKIN frequency/(DF+1)) MSEL/CGU_DIV.SYSSEL Where the value of CGU_DIV.SYSSEL is between 1 and 31.	
		0	SYSSEL = 32
		1-31	SYSSEL = 1 to 31

Table 2-13: CGU_DIV Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:5 (R/W)	S0SEL	SCLK 0 Divisor. The CGU_DIV.S0SEL selects the divisor in the SCLK0 equation: SCLK0 frequency = (SYSCLK frequency) / CGU_DIV.S0SEL Where the value of CGU_DIV.S0SEL is between 1 and 7.	
		0	S0SEL = 8
		1-7	S0SEL = 1 to 7
4:0 (R/W)	CSEL	CCLK Divisor. The CGU_DIV.CSEL bit field selects the divisor in the CCLK equation: CCLK frequency = (SYS_CLKIN frequency / (DF+1)) * MSEL / CGU_DIV.CSEL Where the value of CGU_DIV.CSEL is between 1 and 31.	
		0	CSEL = 32
		1-31	CSEL= 1 to 31

DIV Register Extension

The CGU_DIVEX register contains bitfields for programming the divisor values of the system clock [SCLK0 and SCLK1] in extended divisor mode.

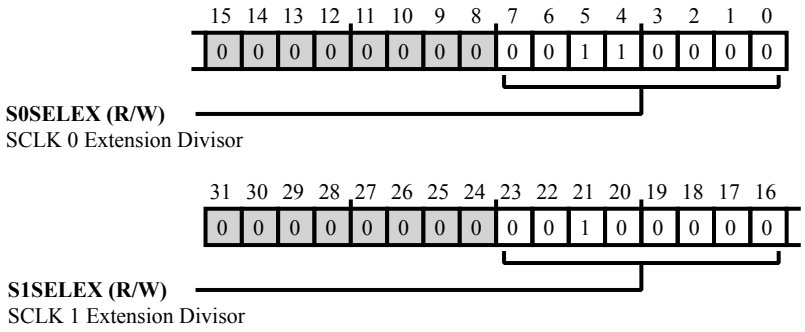


Figure 2-9: CGU_DIVEX Register Diagram

Table 2-14: CGU_DIVEX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:16 (R/W)	S1SELEX	SCLK 1 Extension Divisor. The CGU_DIVEX.S1SELEX selects the divisor in the SCLK1EX equation: SCLK1EX frequency = (PLLCLK frequency) / CGU_DIVEX.S1SELEX Where the value of CGU_DIVEX.S1SELEX is between 1 and 255.	
		0	S1SELEX = 256
		1-255	S1SELEX = 1 to 255
7:0 (R/W)	S0SELEX	SCLK 0 Extension Divisor. The CGU_DIVEX.S0SELEX selects the divisor in the SCLK0EX equation: SCLK0EX frequency = (PLLCLK frequency) / CGU_DIVEX.S0SELEX Where the value of CGU_DIVEX.S0SELEX is between 1 and 255.	
		0	S1SELEX = 256
		1-255	S1SELEX = 1 to 255

Oscillator Watchdog Register

The CGU_OSCWDCTL register configures the CGU to allow the detection of the absence of input clock transitions and provides a fault warning via the SYS_FAULT pin. The CGU_OSCWDCTL register also detects and reports input oscillator frequencies above and below specified limits, in order to specifically detect harmonic or sub-harmonic crystal oscillator behavior. This detection is achieved by using an internal asynchronous, local 1 MHz oscillator combined with a series of programmable counters.

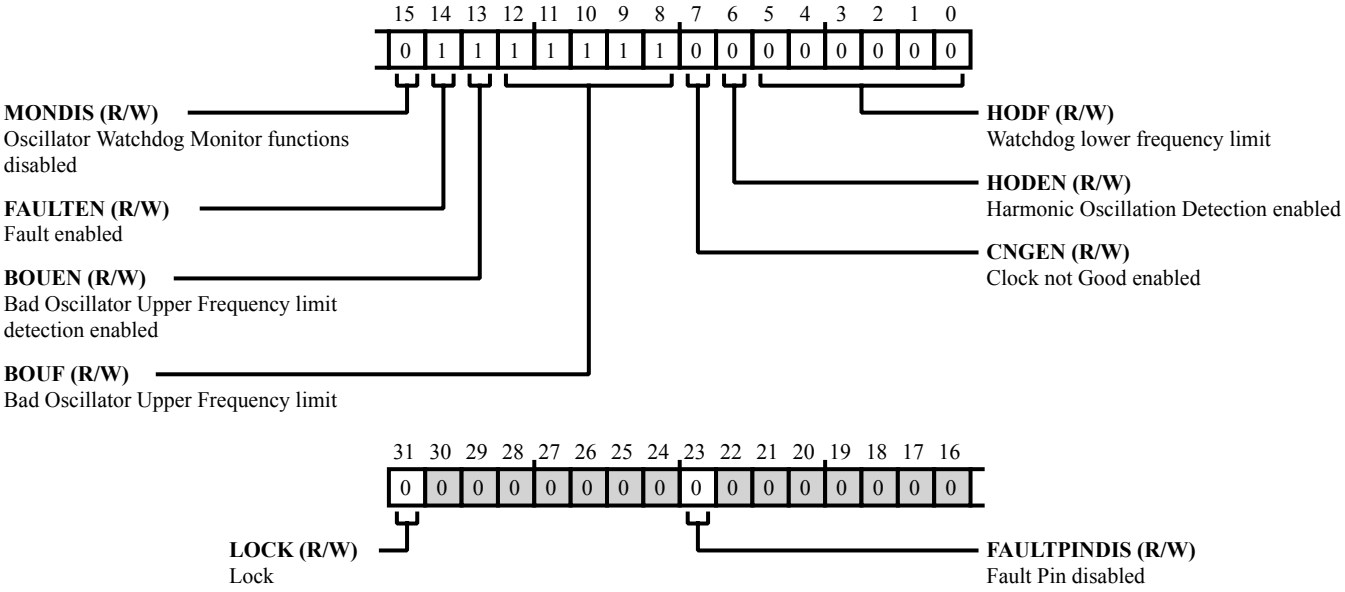


Figure 2-10: CGU_OSCWDCTL Register Diagram

Table 2-15: CGU_OSCWDCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	LOCK	Lock.
23 (R/W)	FAULTPINDIS	Fault Pin disabled. The CGU_OSCWDCTL.FAULTPINDIS bit disables pin fault detection.
15 (R/W)	MONDIS	Oscillator Watchdog Monitor functions disabled. The CGU_OSCWDCTL.MONDIS bit disables all the input clock monitor and fault detection functions.
14 (R/W)	FAULTEN	Fault enabled. The CGU_OSCWDCTL.FAULTEN bit enables fault detection.
13 (R/W)	BOUEN	Bad Oscillator Upper Frequency limit detection enabled. The CGU_OSCWDCTL.BOUEN bit enables upper limit bad oscillation detection.

Table 2-15: CGU_OSCWDCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
12:8 (R/W)	BOUF	Bad Oscillator Upper Frequency limit. The `CGU_OSCWDCTL.BOUF` bits indicate the desired upper fail limit for the bad oscillation detection.
7 (R/W)	CNGEN	Clock not Good enabled. The `CGU_OSCWDCTL.CNGEN` bit enables the detection of an oscillator watchdog clock fault.
6 (R/W)	HODEN	Harmonic Oscillation Detection enabled. The `CGU_OSCWDCTL.HODEN` bit enables harmonic oscillation detection.
5:0 (R/W)	HODF	Watchdog lower frequency limit. The `CGU_OSCWDCTL.HODF` bit field is used to indicate the desired lower fail limit for the harmonic oscillation detection in MHz.

PLL Control Register

The CGU_PLLCTL register contains bits that enable and disable the PLL as well as control its function.

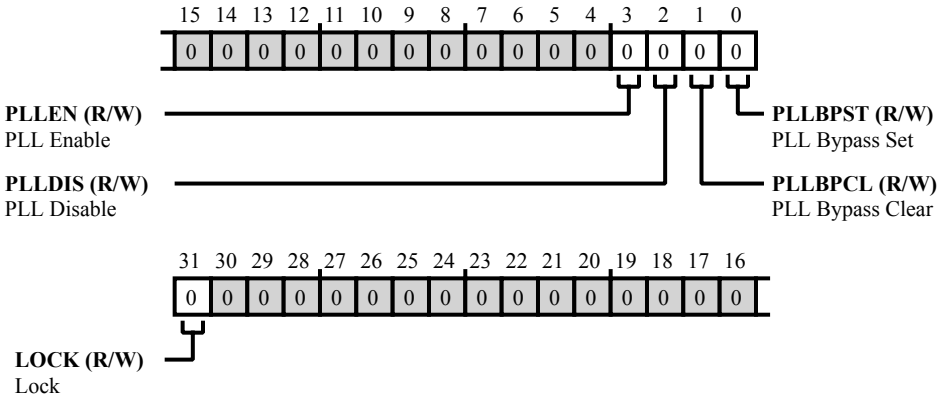


Figure 2-11: CGU_PLLCTL Register Diagram

Table 2-16: CGU_PLLCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. Setting (=1) the CGU_PLLCTL.LOCK bit locks access to the CGU_PLLCTL register.	
		0	Unlock register
		1	Lock register
3 (R/W)	PLLEN	PLL Enable. Setting (=1) the CGU_PLLCTL.PLLEN bit enables the PLL. Check the CGU_STAT.PLLEN bit to verify that the action is complete.	
		0	No action
		1	Enable PLL
2 (R/W)	PLLDIS	PLL Disable. Setting (=1) the CGU_PLLCTL.PLLDIS bit disables the PLL. Check the CGU_STAT.PLLEN bit to verify that the action is complete.	
		0	No action
		1	Disable PLL
1 (R/W)	PLLBPCL	PLL Bypass Clear. Setting (=1) the CGU_PLLCTL.PLLBPCL bit takes the PLL out of bypass mode. Check the CGU_STAT.PLLBP bit to verify that the action is complete.	
		0	No action
		1	Exit bypass mode

Table 2-16: CGU_PLLCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	PLLBPST	PLL Bypass Set. Setting (=1) the `CGU_PLLCTL.PLLBPST` bit bypasses the PLL and all the clocks run on CLKIN. Check the `CGU_STAT.PLLBP` bit to verify that the action is complete.	
		0	Use PLL
		1	Bypass PLL

Revision ID Register

The CGU_REVID register reports the version of the CGU.

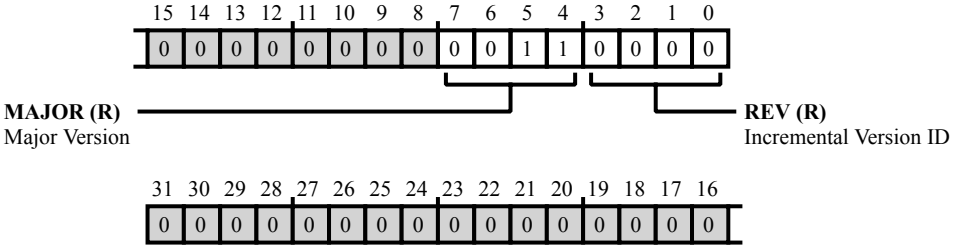


Figure 2-12: CGU_REVID Register Diagram

Table 2-17: CGU_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major Version.
3:0 (R/NW)	REV	Incremental Version ID.

System Clock Buffer Disable Register

The CGU_SCBF_DIS register controls each system's clock buffer to determine if the SCLKn buffer is enabled.

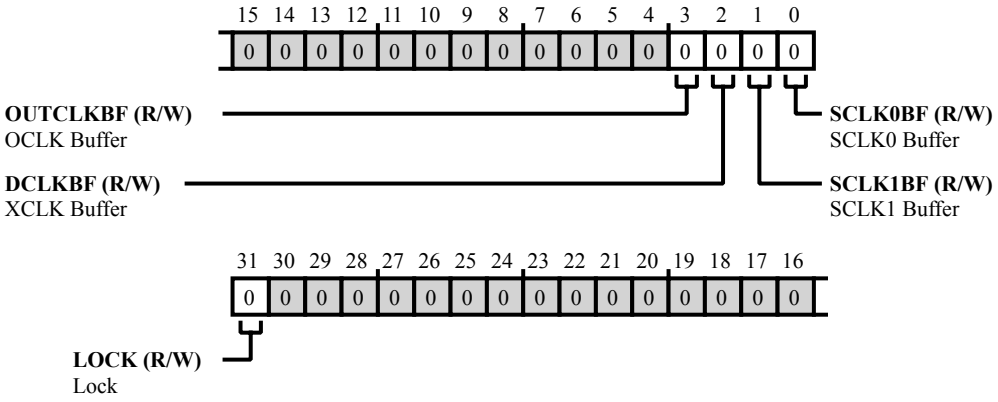


Figure 2-13: CGU_SCBF_DIS Register Diagram

Table 2-18: CGU_SCBF_DIS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. The CGU_SCBF_DIS.LOCK bit allows writes to the CGU_SCBF_DIS register when cleared (=0) or blocks writes if set (=1) and the SPU_CTL.GLCK bit is set.	
		0	Unlock register
		1	Lock register
3 (R/W)	OUTCLKBF	OCLK Buffer. The CGU_SCBF_DIS.OUTCLKBF bit enables (=0, default) or disables (=1) OCLKs buffer.	
		0	Enable buffer
		1	Disable buffer
2 (R/W)	DCLKBF	XCLK Buffer. The CGU_SCBF_DIS.DCLKBF bit enables (=0, default) or disables (=1) XCLKs buffer.	
		0	Enable buffer
		1	Disable buffer

Table 2-18: CGU_SCBF_DIS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	SCLK1BF	SCLK1 Buffer. The `CGU_SCBF_DIS.SCLK1BF` bit enables (=0, default) or disables (=1) SCLK1s buffer.	
		0	Enable buffer
		1	Disable buffer
0 (R/W)	SCLK0BF	SCLK0 Buffer. The `CGU_SCBF_DIS.SCLK0BF` bit enables (=0, default) or disables (=1) SCLK0s buffer.	
		0	Enable buffer
		1	Disable buffer

System Clock Buffer Status Register

The CGU_SCBF_STAT register shows which system clock buffer(s) are disabled. For example clearing the CGU_CCBF_DIS.CCBF0 bit clears the CGU_SCBF_STAT.SCLK0BF bit after a number of cycles. To guarantee that the correct value is read, this register should be read twice and the second result used.

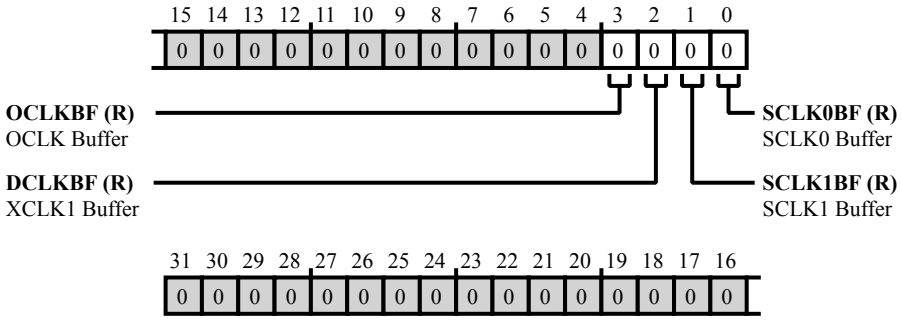


Figure 2-14: CGU_SCBF_STAT Register Diagram

Table 2-19: CGU_SCBF_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	OCLKBF	OCLK Buffer. The CGU_SCBF_STAT.OCLKBF bit reports the status of the CGU_SCBF_DIS.OUTCLKBF bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled
2 (R/NW)	DCLKBF	XCLK1 Buffer. The CGU_SCBF_STAT.DCLKBF bit reports the status of the CGU_SCBF_DIS.DCLKBF bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled
1 (R/NW)	SCLK1BF	SCLK1 Buffer. The CGU_SCBF_STAT.SCLK1BF bit reports the status of the CGU_SCBF_DIS.SCLK1BF bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled

Table 2-19: CGU_SCBF_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SCLK0BF	SCLK0 Buffer. The `CGU_SCBF_STAT.SCLK0BF` bit reports the status of the `CGU_SCBF_DIS.SCLK0BF` bit where 0 = enabled and 1 = disabled.	
		0	Enabled
		1	Disabled

Status Register

The CGU_STAT register reflects the PLL status and errors detected during the PLL configuration. This register may be cleared asynchronously by a reset signal from the RCU module. All bits---except those defined as W1C (write-1-to-clear)---are read only.

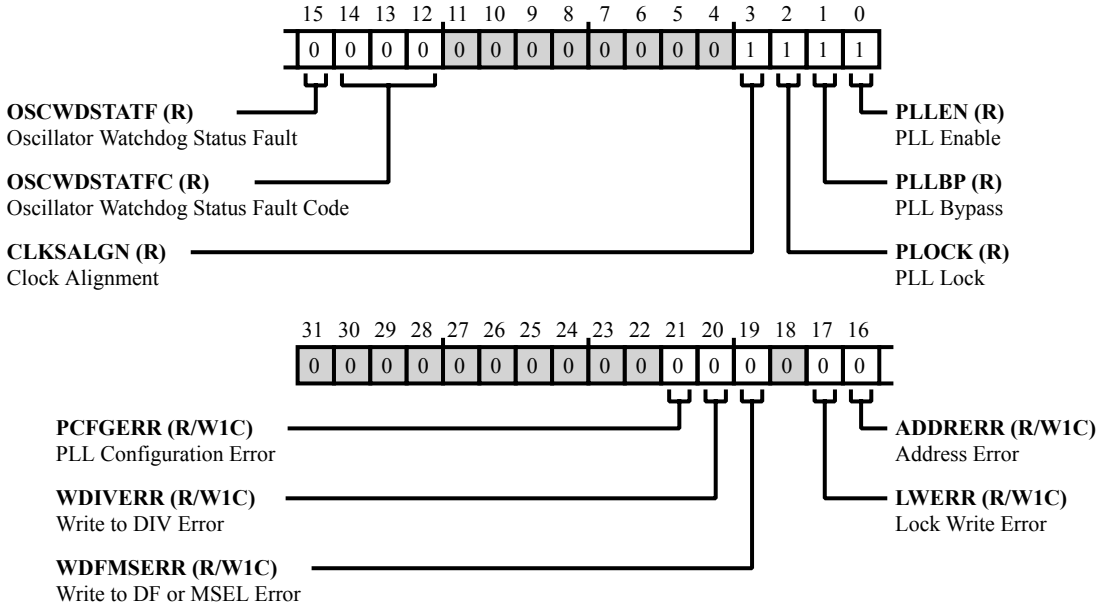


Figure 2-15: CGU_STAT Register Diagram

Table 2-20: CGU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W1C)	PCFGERR	PLL Configuration Error. If the CGU_PLLCTL.PLLBPST and the CGU_PLLCTL.PLLBPCL bits are set (=1) simultaneously or the CGU_PLLCTL.PLLDIS bit was set (=1) in full-on mode or while trying to enter full-on mode (CGU_PLLCTL.PLLBPCL =1), the CGU_STAT.PCFGERR bit triggers the bus error.	
		0	No Error
		1	Configuration Error

Table 2-20: CGU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W1C)	WDIVERR	Write to DIV Error. The CGU_STAT.WDIVERR bit indicates a write access to the CGU_DIV register (to trigger an alignment sequence or to change the CGU_DIV.CSEL, CGU_DIV.SYSSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, or CGU_DIV.DSEL bit values) while the PLL is locked, but still aligning the clocks. Read after write accesses to the CGU_STAT and CGU_DIV registers return the new value even if the clock frequency change is still in progress.	
		0	No Error
		1	Write DIV Error
19 (R/W1C)	WDFMSERR	Write to DF or MSEL Error. The CGU_STAT.WDFMSERR bit indicates a write access to the CGU_CTL register to change the CGU_CTL.DF or CGU_CTL.MSEL bit values while the PLL is locking.	
		0	No Error
		1	Write DF/MSEL Error
17 (R/W1C)	LWERR	Lock Write Error. The CGU_STAT.LWERR bit indicates an attempt to write to write-protected (locked) CGU registers. The CGU issues a bus error for this condition.	
		0	No Error
		1	Lock Write Error
16 (R/W1C)	ADDRERR	Address Error. The CGU_STAT.ADDRERR bit indicates an attempt to make a read or write access to unimplemented addresses or accesses are non-aligned. The CGU issues a bus error for this condition.	
		0	No Error
		1	Address Error
15 (R/NW)	OSCWDSTATF	Oscillator Watchdog Status Fault. The CGU_STAT.OSCWDSTATF bit indicates a fault in the oscillator watchdog (CGU's OSC_WDSTAT[1:0]) input pins.	
		0	No Fault
		1	Fault
14:12 (R/NW)	OSCWDSTATFC	Oscillator Watchdog Status Fault Code. The CGU_STAT.OSCWDSTATFC bit field indicates the nature of the fault in the oscillator watchdog (CGU's OSC_WDSTAT[1:0]) input pins.	
		0	No Fault
		1	No Input Clock

Table 2-20: CGU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	Subharmonic CLKIN
		3	Harmonic CLKIN
		4	No AUX_CLK
		5	CLKIN > Upper Frequency Limit (BOUF)
		6	Reserved
		7	Multiple Limit Faults
3 (R/NW)	CLKSALGN	Clock Alignment. The CGU_STAT.CLKSALGN bit indicates whether a clock alignment sequence is in progress. This bit is set when clocks alignment is required by changes to CGU_DIV.CSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, CGU_DIV.DSEL, or CGU_DIV.OSEL. The CGU_STAT.CLKSALGN bit is cleared when clocks are aligned. Note that (after a PLL frequency change in active state) the CGU_STAT.CLKSALGN bit may indicate that clocks are not aligned even though the clocks are aligned (all clocks are aligned and running at CLKIN frequency).	
		0	Clocks are Aligned
		1	Clocks not Aligned (alignment in progress)
2 (R/NW)	PLOCK	PLL Lock. The CGU_STAT.PLOCK bit indicates whether the PLL is locked. This bit is set when the PLL locks (PLL lock counter end-of-count). The CGU_STAT.PLOCK bit is cleared when requested PLL frequency change (for PLL reset, PLL disable-to-enable transition, or a change to the CGU_CTL.MSEL or CGU_CTL.DF values) is in progress.	
		0	PLL not Locked (PLL frequency change in progress)
		1	PLL Locked
1 (R/NW)	PLLBP	PLL Bypass. The CGU_STAT.PLLBP bit indicates whether the PLL is bypassed. The default value for the CGU_STAT.PLLBP bit is determined by the bypass strap pin.	
		0	PLL not Bypassed
		1	PLL Bypassed
0 (R/NW)	PLLEN	PLL Enable. The CGU_STAT.PLLEN bit indicates whether the PLL is enabled.	
		0	Disabled
		1	Enabled

Time Stamp Counter 32 LSB Register

The CGU_TSCOUNT0 register address is used to read the CoreSight time stamp counter LSB 32-bit (bits [31:0]) value.

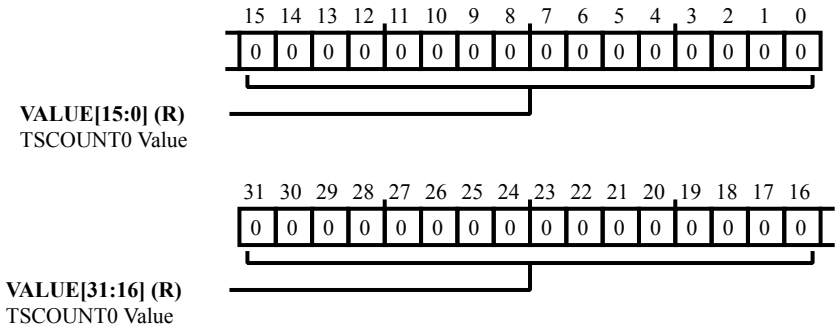


Figure 2-16: CGU_TSCOUNT0 Register Diagram

Table 2-21: CGU_TSCOUNT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	TSCOUNT0 Value. The CGU_TSCOUNT0.VALUE bit field holds the time stamp counter 32 LSBs.

Time Stamp Counter 32 MSB Register

The CGU_TSCOUNT1 register address is used to read the CoreSight time stamp counter MSB 32-bit (bits [63:32]) value.

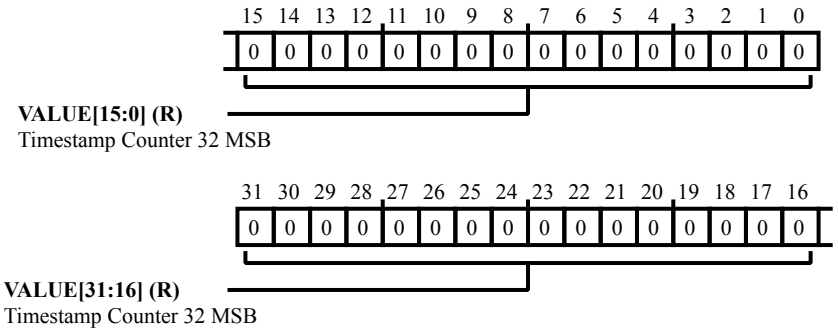


Figure 2-17: CGU_TSCOUNT1 Register Diagram

Table 2-22: CGU_TSCOUNT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Timestamp Counter 32 MSB. The CGU_TSCOUNT1.VALUE bit field holds the time stamp counter 32 MSBs.

Time Stamp Control Register

The CGU_TSCTL register controls the operation of the CoreSight time stamp counter.

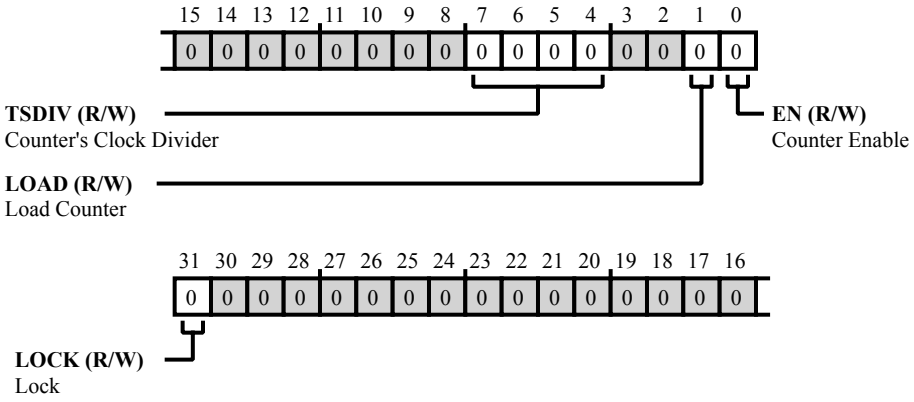


Figure 2-18: CGU_TSCTL Register Diagram

Table 2-23: CGU_TSCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. Setting the CGU_TSCTL.LOCK bit locks this register.	
		0	Unlock
		1	Lock
7:4 (R/W)	TSDIV	Counter's Clock Divider. The CGU_TSCTL.TSDIV bit field divides SYSCLK by 2^{TSDIV}.	
		0-15	Divides SYSCLK by 2^{TSDIV}
1 (R/W)	LOAD	Load Counter. Writing one to the CGU_TSCTL.LOAD bit causes CoreSight time stamp counter to be loaded from the CGU_TSVALUE0 and CGU_TSVALUE1 registers.	
		0	Always read as "0"
0 (R/W)	EN	Counter Enable. The CGU_TSCTL.EN bit enables or disables the CoreSight time stamp counter.	
		0	Counter Disabled
		1	Counter Enabled

Time Stamp Counter Initial 32 LSB Value Register

The CGU_TSVALUE0 register holds the least significant bits (bits [31:0]) value that is initially loaded to the Core-Sight time stamp counter.

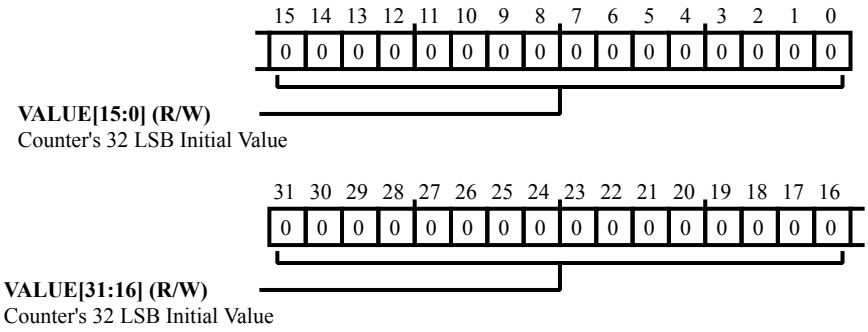


Figure 2-19: CGU_TSVALUE0 Register Diagram

Table 2-24: CGU_TSVALUE0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Counter's 32 LSB Initial Value. The CGU_TSVALUE0.VALUE bit field holds the LSBs value that is initially loaded to the CoreSight time stamp counter.

Time Stamp Counter Initial MSB Value Register

The CGU_TSVALUE1 register holds the most significant bits (bits [63:32]) value that is initially loaded to the CoreSight time stamp counter.

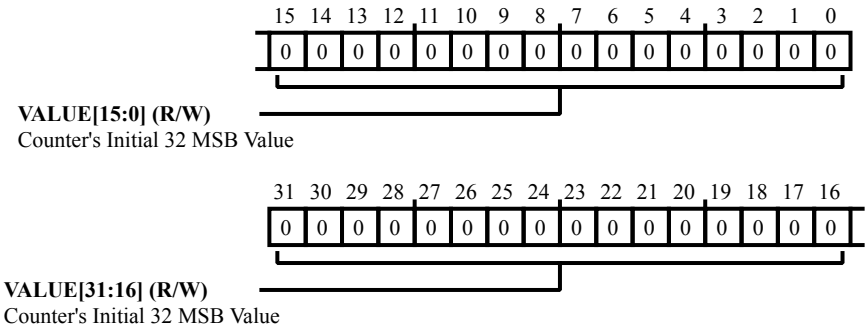


Figure 2-20: CGU_TSVALUE1 Register Diagram

Table 2-25: CGU_TSVALUE1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Counter's Initial 32 MSB Value. The CGU_TSVALUE1.VALUE bit field holds the MSBs value that is initially loaded to the CoreSight time stamp counter.

ADSP-SC59x MISCREG Register Descriptions

Misc registers for module are for integration purpose (MISCREG) contains the following registers.

Table 2-26: ADSP-SC59x MISCREG Register List

Name	Description
MISCREG_CAN_SYSCTL	CANFD Low Power Selection Mode
MISCREG_ECO_REG9	Error Register
MISCREG_PLL2_CONTROL	PLL2 Control Register
MISCREG_SH0_PFB_RANGE_SELECT	SH0 Prefetch Range Selection Register
MISCREG_SH1_PFB_RANGE_SELECT	SH1 Prefetch Range Selection Register
MISCREG_SHARC_BRIDGE_REMAP	SPORT Direct Interrupt Enable for SH0 and SH1

CANFD Low Power Selection Mode

The MISCREG_CAN_SYSCTL register contains the control bits for the different low power modes supported in the CANFD module.

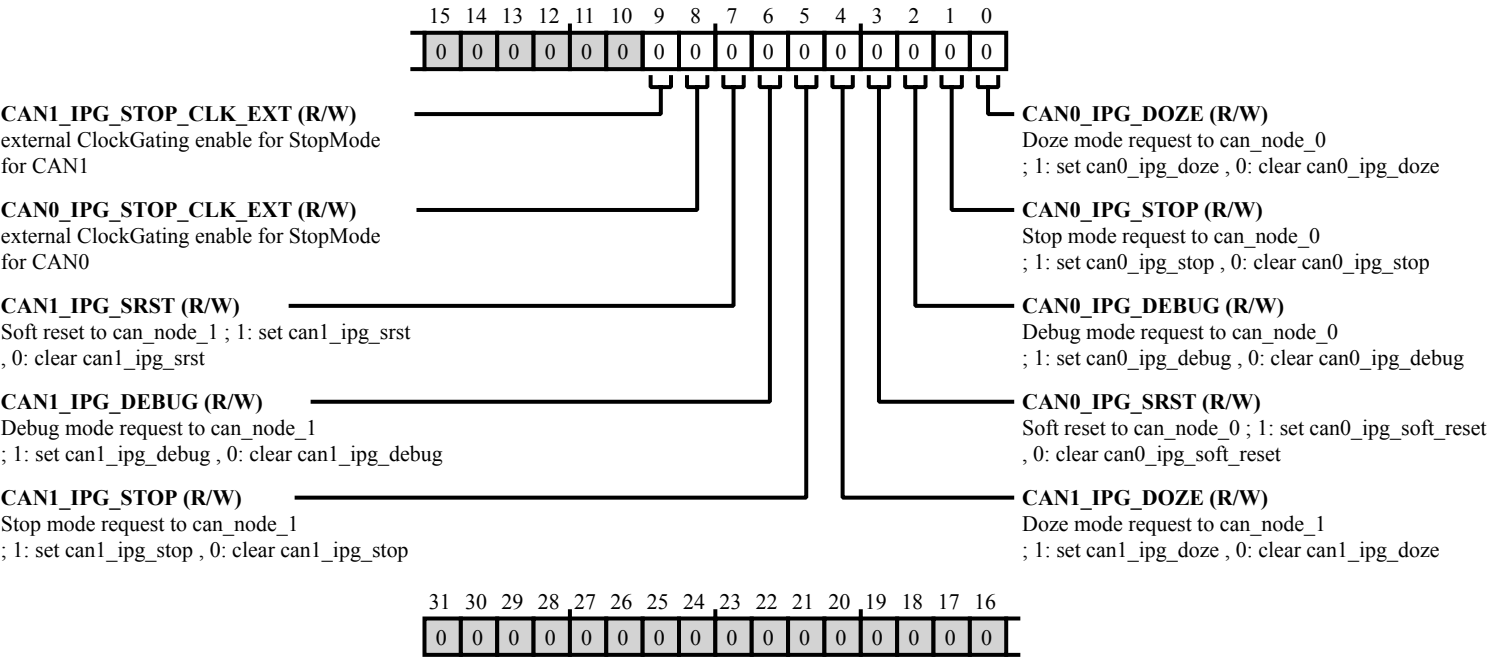


Figure 2-21: MISCREG_CAN_SYSCTL Register Diagram

Table 2-27: MISCREG_CAN_SYSCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	CAN1_IPG_STOP_CLK_EXT	external ClockGating enable for StopMode for CAN1.
8 (R/W)	CAN0_IPG_STOP_CLK_EXT	external ClockGating enable for StopMode for CAN0.
7 (R/W)	CAN1_IPG_SRST	Soft reset to can_node_1 ; 1: set can1_ipg_srst , 0: clear can1_ipg_srst.
6 (R/W)	CAN1_IPG_DEBUG	Debug mode request to can_node_1 ; 1: set can1_ipg_debug , 0: clear can1_ipg_debug.
5 (R/W)	CAN1_IPG_STOP	Stop mode request to can_node_1 ; 1: set can1_ipg_stop , 0: clear can1_ipg_stop.
4 (R/W)	CAN1_IPG_DOZE	Doze mode request to can_node_1 ; 1: set can1_ipg_doze , 0: clear can1_ipg_doze.

Table 2-27: MISCREG_CAN_SYSCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W)	CAN0_IPG_SRST	Soft reset to can_node_0 ; 1: set can0_ipg_soft_reset , 0: clear can0_ipg_soft_reset.
2 (R/W)	CAN0_IPG_DEBUG	Debug mode request to can_node_0 ; 1: set can0_ipg_debug , 0: clear can0_ipg_debug.
1 (R/W)	CAN0_IPG_STOP	Stop mode request to can_node_0 ; 1: set can0_ipg_stop , 0: clear can0_ipg_stop.
0 (R/W)	CAN0_IPG_DOZE	Doze mode request to can_node_0 ; 1: set can0_ipg_doze , 0: clear can0_ipg_doze.

Error Register

The MISCREG_ECO_REG9 register contains eight Tx channels, eight Rx channels, and DMA errors for both EMAC0 and EMAC1.

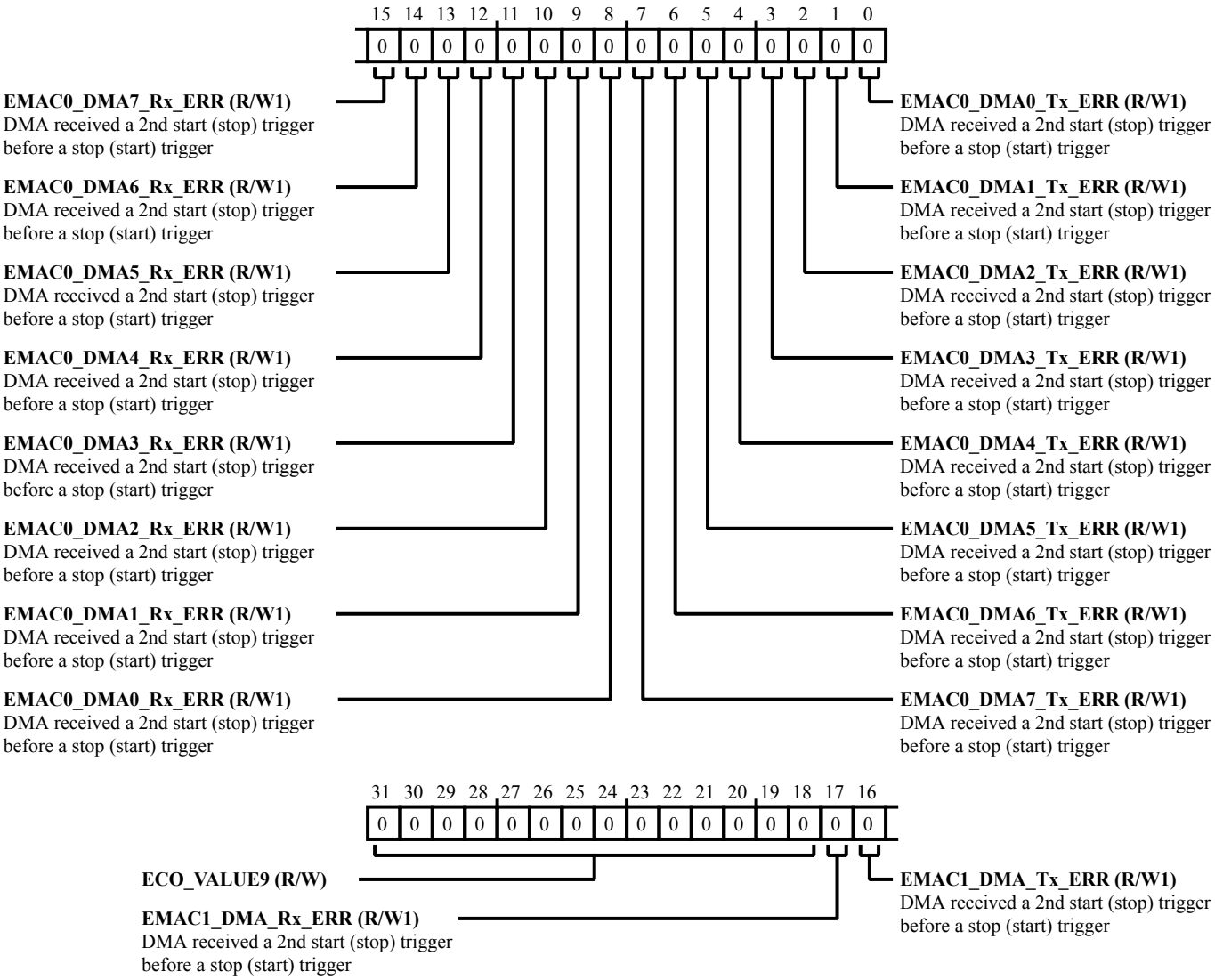


Figure 2-22: MISCREG_ECO_REG9 Register Diagram

Table 2-28: MISCREG_ECO_REG9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:18 (R/W)	ECO_VALUE9	.

Table 2-28: MISCREG_ECO_REG9 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W1)	EMAC1_DMA_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
16 (R/W1)	EMAC1_DMA_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
15 (R/W1)	EMAC0_DMA7_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
14 (R/W1)	EMAC0_DMA6_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
13 (R/W1)	EMAC0_DMA5_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
12 (R/W1)	EMAC0_DMA4_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
11 (R/W1)	EMAC0_DMA3_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
10 (R/W1)	EMAC0_DMA2_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
9 (R/W1)	EMAC0_DMA1_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
8 (R/W1)	EMAC0_DMA0_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
7 (R/W1)	EMAC0_DMA7_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
6 (R/W1)	EMAC0_DMA6_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
5 (R/W1)	EMAC0_DMA5_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
4 (R/W1)	EMAC0_DMA4_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
3 (R/W1)	EMAC0_DMA3_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
2 (R/W1)	EMAC0_DMA2_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.

Table 2-28: MISCREG_ECO_REG9 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W1)	EMAC0_DMA1_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
0 (R/W1)	EMAC0_DMA0_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.

PLL2 Control Register

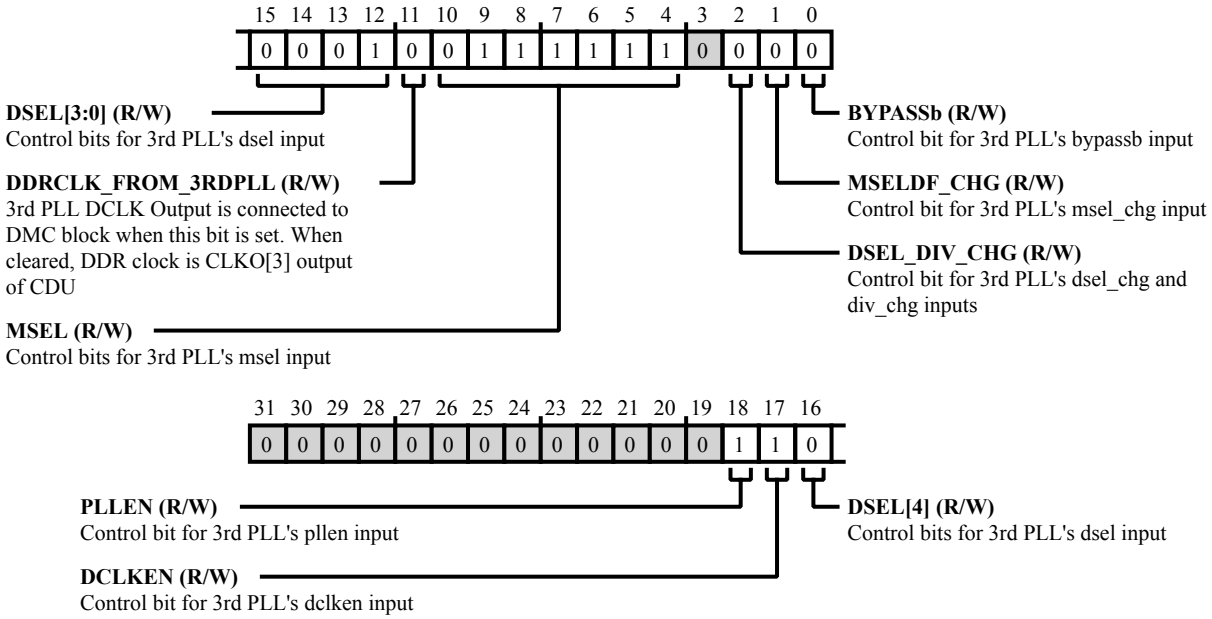


Figure 2-23: MISCREG_PLL2_CONTROL Register Diagram

Table 2-29: MISCREG_PLL2_CONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/W)	PLLEN	Control bit for 3rd PLL's pllen input. Internal signal from Dynamic Power Management Unit (DPM) which enables/disables the PLL system (e.g. in Active mode, during BYPASS.)
17 (R/W)	DCLKEN	Control bit for 3rd PLL's dclken input. Enables DCLK clock buffer
16:12 (R/W)	DSEL	Control bits for 3rd PLL's dsel input. Selects DCLK divide ratio
11 (R/W)	DDRCLK_FROM_3RDPLL	3rd PLL DCLK Output is connected to DMC block when this bit is set. When cleared, DDR clock is CLKO[3] output of CDU. 3rd PLL DCLK Output is connected to DMC block when this bit is set. When cleared, DDR clock is CLKO[3] output of CDU
10:4 (R/W)	MSEL	Control bits for 3rd PLL's msel input. The `MISCREG_PLL2_CONTROL.MSEL` selects the multiplier in the PLLCLK equation: PLLCLK frequency = (SYS_CLKIN frequency / (DF+1)) * MSEL * 2 Where the value of MSEL is between 1 and 127.

Table 2-29: MISCREG_PLL2_CONTROL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/W)	DSEL_DIV_CHG	Control bit for 3rd PLL's dsel_chg and div_chg inputs. Signal from PCU indicating a DDR clock frequency change
1 (R/W)	MSELDF_CHG	Control bit for 3rd PLL's msel_chg input. Signal from PLL Control Unit indicating a frequency change resulting from a change in either the DF or MSEL bits.
0 (R/W)	BYPASSB	Control bit for 3rd PLL's bypassb input. Active-low control bit to bypass the PLL which forces CLKIN on all output clocks

SH0 Prefetch Range Selection Register

In the MISCREG_SH0_PFB_RANGE_SELECT register, each bit field controls if the corresponding address range will be prefetched. When an address range is set it is prefetched and when it is zero there is no prefetch.

The MISCREG_SH0_PFB_RANGE_SELECT[31:16] bits control the range selection for the instruction cache, and MISCREG_SH0_PFB_RANGE_SELECT[15:0] bits control the range selection for the data cache.

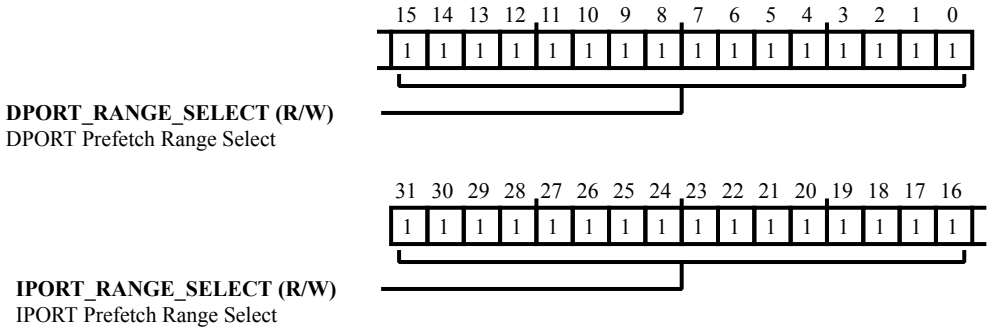


Figure 2-24: MISCREG_SH0_PFB_RANGE_SELECT Register Diagram

Table 2-30: MISCREG_SH0_PFB_RANGE_SELECT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	IPORT_RANGE_SELECT	IPORT Prefetch Range Select. Prefetch Range Selection for IPORT
15:0 (R/W)	DPORT_RANGE_SELECT	DPORT Prefetch Range Select. Prefetch Range Selection for DPORT

SH1 Prefetch Range Selection Register

Each bit field controls if the corresponding address range will be prefetched. When an address range is set it is prefetched an when it is zero there is no prefetch.

The MISCREG_SH1_PFB_RANGE_SELECT[31:16] bits control the range selection for the instruction cache, and MISCREG_SH1_PFB_RANGE_SELECT[15:0] bits control the range selection for the data cache.

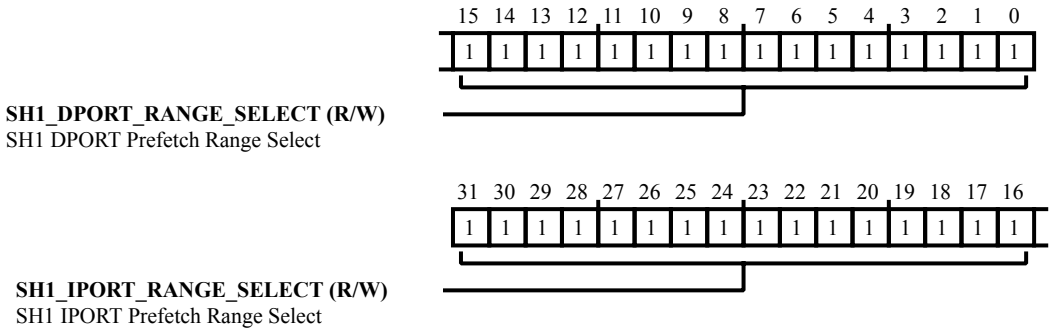


Figure 2-25: MISCREG_SH1_PFB_RANGE_SELECT Register Diagram

Table 2-31: MISCREG_SH1_PFB_RANGE_SELECT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	SH1_IPORT_RANGE_SELECT	SH1 IPORT Prefetch Range Select. Prefetch Range Selection for IPORT
15:0 (R/W)	SH1_DPORT_RANGE_SELECT	SH1 DPORT Prefetch Range Select. Prefetch Range Selection for DPORT

SPORT Direct Interrupt Enable for SH0 and SH1

The bits in the MISCREG_SHARC_BRIDGE_REMAP register control the direct connection of SPORT interrupts to the SHARC core.

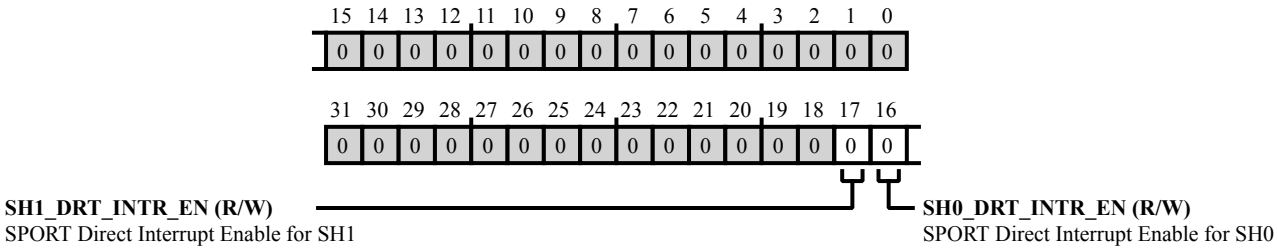


Figure 2-26: MISCREG_SHARC_BRIDGE_REMAP Register Diagram

Table 2-32: MISCREG_SHARC_BRIDGE_REMAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	SH1_DRT_INTR_EN	SPORT Direct Interrupt Enable for SH1. If the MISCREG_SHARC_BRIDGE_REMAP.SH1_DRT_INTR_EN bit is set a direct connection is enabled for SH1. Do not set the MISCREG_SHARC_BRIDGE_REMAP.SH1_DRT_INTR_EN bit when IIR1 of the SH1 is in use. If this bit is zero the direct connection is disabled.
16 (R/W)	SH0_DRT_INTR_EN	SPORT Direct Interrupt Enable for SH0. If the MISCREG_SHARC_BRIDGE_REMAP.SH0_DRT_INTR_EN bit is set a direct connection is enabled for SH0. Do not set the MISCREG_SHARC_BRIDGE_REMAP.SH0_DRT_INTR_EN bit when IIR1 of the SH0 is in use. If this bit is zero the direct connection is disabled.

3 Clock Distribution Unit (CDU)

The Clock Distribution Unit (CDU) consists of an array of multiplexors that select clocks originated from up to four different clock sources. These sources are different clocks that are generated from the Clock Generation Units (CGUs). The multiplexors are configured by software. All clocks multiplexors have up to four different sources. Unused input clocks are grounded internally and never selected. The output clock signal for each multiplexor is assigned to one or more destinations within the processor.

CDU Features

The CDU modules supports the following features:

- Generation of up to 14 output clocks
- Output clock buffers that can be disabled by software
- Multiplexors that are configured by writing to configuration registers (`CDU_CFG[n]`)
- Clocks originated in CGU0 that are selected by default
- A CDU status register (`CDU_STAT`) that indicates a configuration change is in-progress

CDU Functional Description

The CDU functions as a set of software-configurable multiplexors that select clocks from different sources.

CDU Block Diagram

The ***CGU/CDU Block Diagram*** figure shows the functional blocks within the CDU. As shown in the figure, the CDU takes different clocks generated by the CGU blocks and provides flexibility to route any clock from possible options to the output CDU clocks. The output clocks from the CDU are connected to specific targets such as the DDR module. All of the targets are clocked through the CDU output clocks rather than being directly clocked by CGU clocks.

This configuration provides the flexibility to meet the specific clock requirements of the different modules in the system (such as the core, DDR, or SPI module) without compromising the clocking of other modules. Such a flexibility is not possible with a single CGU in the system. With two CGUs:

- Each CGU has its own set of clock multipliers and dividers, providing a greater number of orthogonal clocks than possible with a single CGU.
- The `CDU_CLKINSEL.CGU1` bit determines the input for CGU1.

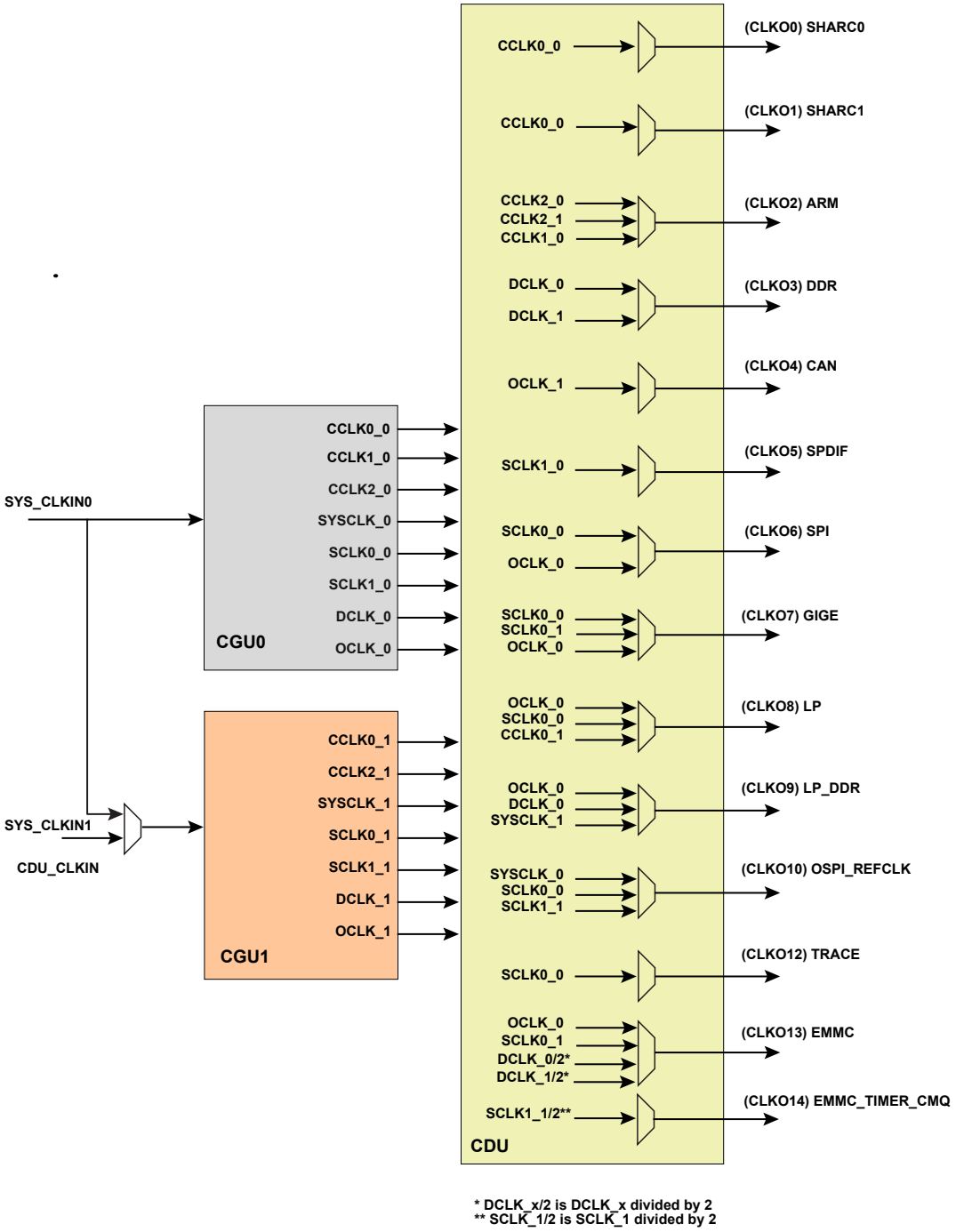


Figure 3-1: CGU/CDU Block Diagram

CDU Definitions

The ***Clock Descriptions*** table provides a brief description of the clocks supported by the processor.

Table 3-1: Clock Descriptions

Clock	Description
CCLK0_0	CCLK0 derived from CGU0
CCLK1_0	CCLK1 derived from CGU0
CCLK2_0	CCLK2 derived from CGU0*1
SYSCLK_0	SYSCLK derived from CGU0
SCLK0_0	SCLK0 derived from CGU0
SCLK1_0	SCLK1 derived from CGU0
DCLK_0	DCLK derived from CGU0
OCLK_0	OCLK derived from CGU0
CCLK0_1	CCLK0 derived from CGU1
CCLK2_1	CCLK2 derived from CGU1*1
SYSCLK_1	SYSCLK derived from CGU1
SCLK0_1	SCLK0 derived from CGU1
SCLK1_1	SCLK1 derived from CGU1
DCLK_1	DCLK derived from CGU1
OCLK_1	OCLK derived from CGU1
CDU_CLKOn	Clocks that come out of the CDU module and go to different blocks
DCLK_2	DCLK derived from CGU2

*1 CCLK2_x is directly derived from the VCO clock; it is divided by 3. Hence, the frequency of CCLK2 = (PLL clock x 2/3).

ADSP-SC59x CDU Register List

Clock Distribution Unit

Table 3-2: ADSP-SC59x CDU Register List

Name	Description
CDU_CFG[n]	CDU Configuration
CDU_CLKINSEL	CLKIN Select
CDU_REVID	CDU Revision ID
CDU_STAT	CDU Status

CDU Clock Configuration Options

The *CDU Targets* table provides information about clock source and destination options.

Table 3-3: CDU Targets

CDU0 Input				CDU0 Output	Target
IN0_CLKOn	IN1_CLKOn	IN2_CLKOn	IN3_CLKOn		
CCLK0_0	N/A	N/A	N/A	CLKO0	SHARC0 and corresponding accelerators
CCLK0_0	N/A	N/A	N/A	CLKO1	SHARC1 and corresponding accelerators
CCLK1_0	N/A	CCLK2_0	CCLK2_1	CLKO2	ARM
DCLK_0	DCLK_1	N/A	N/A	CLKO3	DDR
N/A	OCLK_1	N/A	N/A	CLKO4	CAN
SCLK1_0	N/A	N/A	N/A	CLKO5	SPDIF
SCLK0_0	OCLK_0	N/A	N/A	CLKO6	SPI
SCLK0_0	SCLK0_1	OCLK_0	N/A	CLKO7	GIGE
OCLK_0	SCLK0_0	CCLK0_1	N/A	CLKO8	LP
OCLK_0	DCLK_0	SYSCLK_1	N/A	CLKO9	LP_DDR
SYSCLK_0	SCLK0_0	SCLK1_1	N/A	CLKO10	OSPI_REFCLK
SCLK0_0	N/A	N/A	N/A	CLKO12	TRACE
OCLK_0	SCLK0_1	DCLK_0/2	DCLK_1/2	CLKO13	EMMC
N/A	SCLK1_1/2	N/A	N/A	CLKO14	EMMC_TIMER_CMQ

NOTE: CLKO11 is used internally by the processor and is not user configurable.

CDU Programming Model

The `CDU_CFG[n]` registers are a system configuration resource. These registers are accessed by a system configuration routine that also handles the configuration of other system modules. Writes to a `CDU_CFG[n]` register must occur when there is not a CDU_CLKOn configuration change in progress. If a CGUn inputs are selected, that CGUn configuration must be completed first.

Changing the PLL and Clock Frequency

1. Read the `CGU_STAT` register. Verify that:
   - `CGU_STAT.PLLEN` =1 (enabled)
   - `CGU_STAT.PLOCK` =1 (PLL is not locking)
   - `CGU_STAT.CLKSALGN` =0 (clocks aligned)

2. Write the desired values into the CGU_DIV.CSEL, CGU_DIV.S0SEL, CGU_DIV.SYSSEL, CGU_DIV.S1SEL, CGU_DIV.DSEL and CGU_DIV.OSEL bits with the CGU_DIV.UPDT bit cleared (= 0).
3. Write the desired values to the CGU_CTL.DF and CGU_CTL.MSEL bits.
4. Read the CGU_STAT register. Verify that:
   - CGU_STAT.PLLEN = 1 (enabled)
   - CGU_STAT.PLOCK = 1 (not locking)
   - CGU_STAT.CLKSALGN = 0 (clocks aligned)
5. Read the CDU_STAT register. Verify that the CDU_STAT.CLKO0 = 0 (no CLKOn configuration change in progress).
6. Write to the CDU_CLKINSEL register to select the CGU's CLKIN input clock.
7. Write to the CDU_CFG[n].SEL bit to select the clock source (the CDU_CFG[n].EN bit should =1.)
8. Read the CDU_CFG[n] register. Verify that the CDU_CLKINSEL has the programmed value.

CLKOn is reconfigured.

Changing the Clock Frequency

Use the following procedure to change a clock frequency.

1. Read the CGU_STAT register to verify that the CGU_STAT.CLKSALGN bit =0 (clocks aligned).
2. Write the desired values into the CGU_DIV.CSEL, CGU_DIV.S0SEL, CGU_DIV.SYSSEL, CGU_DIV.S1SEL, CGU_DIV.DSEL and CGU_DIV.OSEL bits with the CGU_DIV.UPDT bit = 1.
3. Read the CGU_DIV registers to verify that the CGU_DIV.CSEL, CGU_DIV.S0SEL, CGU_DIV.SYSSEL, CGU_DIV.S1SEL, CGU_DIV.DSEL and CGU_DIV.OSEL bit values are correct.
4. Read the CGU_STAT register to verify that the CGU_STAT.CLKSALGN = 0 (clocks aligned).
5. If clocks switch from CGUm clocks to CGUn input clocks, read the CGU_SCBF_STAT and CGU_CCBF_STAT registers. The CGU_CCBF_STAT[1:0] bit field corresponds to the CCLK1 and CCLK0 clock buffers, respectively. The CGU_SCBF_STAT[3:0] bit field corresponds to the OUTCLK, DCLK, SCLK1, and SCLK0 clock buffers, respectively.
6. Read the CDU_STAT register.
7. Write to the CDU_CFG[n].SEL bit to select the clock source (CDU_CFG[n].EN = 1).
8. Read the CDU_CFG[n] register. Verify that the CDU_CLKINSEL has the programmed value.
9. Verify that the CDU_STAT.CLKO0 = 0.

ADSP-SC59x CDU Register Descriptions

Clock Distribution Unit (CDU) contains the following registers.

Table 3-4: ADSP-SC59x CDU Register List

Name	Description
CDU_CFG[n]	CDU Configuration
CDU_CLKINSEL	CLKIN Select
CDU_REVID	CDU Revision ID
CDU_STAT	CDU Status

CDU Configuration

The CDU_CFG[n] registers control the configuration of the clock multiplexors. CDU0_CFG[n] corresponds to output clock CDU_CLKO[n].

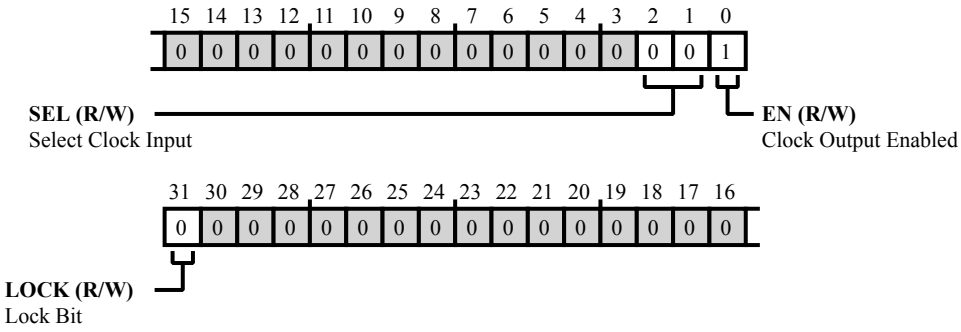


Figure 3-2: CDU_CFG[n] Register Diagram

Table 3-5: CDU_CFG[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit.	
2:1 (R/W)	SEL	Select Clock Input.	
		0	IN0_CLKOn Selected
		1	IN1_CLKOn Selected
		2	IN2_CLKOn Selected
		3	IN3_CLKOn Selected
0 (R/W)	EN	Clock Output Enabled. The CDU_CFG[n].EN bit enables clock output.	

CLKIN Select

The CDU_CLKINSEL register controls the configuration of the CLKIN multiplexors. One bit is assigned to each CGU in the system. This bit selects either CLKIN0 or CLKINn CGUn inputs.

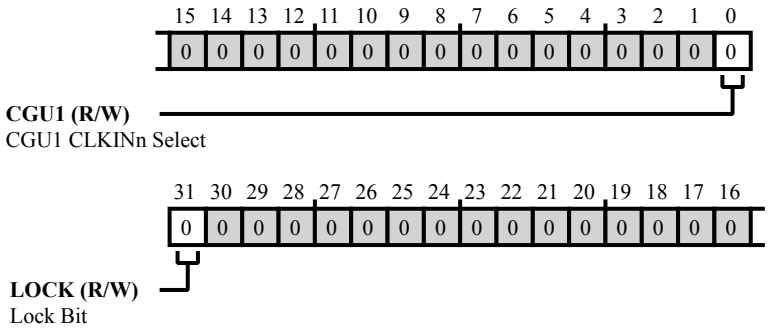


Figure 3-3: CDU_CLKINSEL Register Diagram

Table 3-6: CDU_CLKINSEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit.	
0 (R/W)	CGU1	CGU1 CLKINn Select. The CDU_CLKINSEL.CGU1 bit drives CDU_CLKIN_SEL[0] to CGU1.	
		0	Selects CLKIN0
		1	Selects CLKIN1

CDU Revision ID

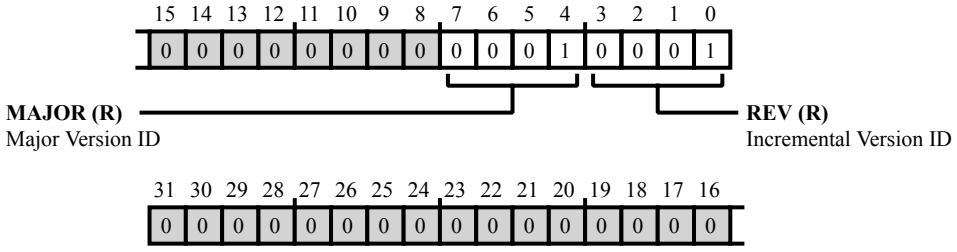


Figure 3-4: CDU_REVID Register Diagram

Table 3-7: CDU_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major Version ID.
3:0 (R/NW)	REV	Incremental Version ID.

CDU Status

The CDU_STAT register reflects the status a change in the configuration of the clock muxes.

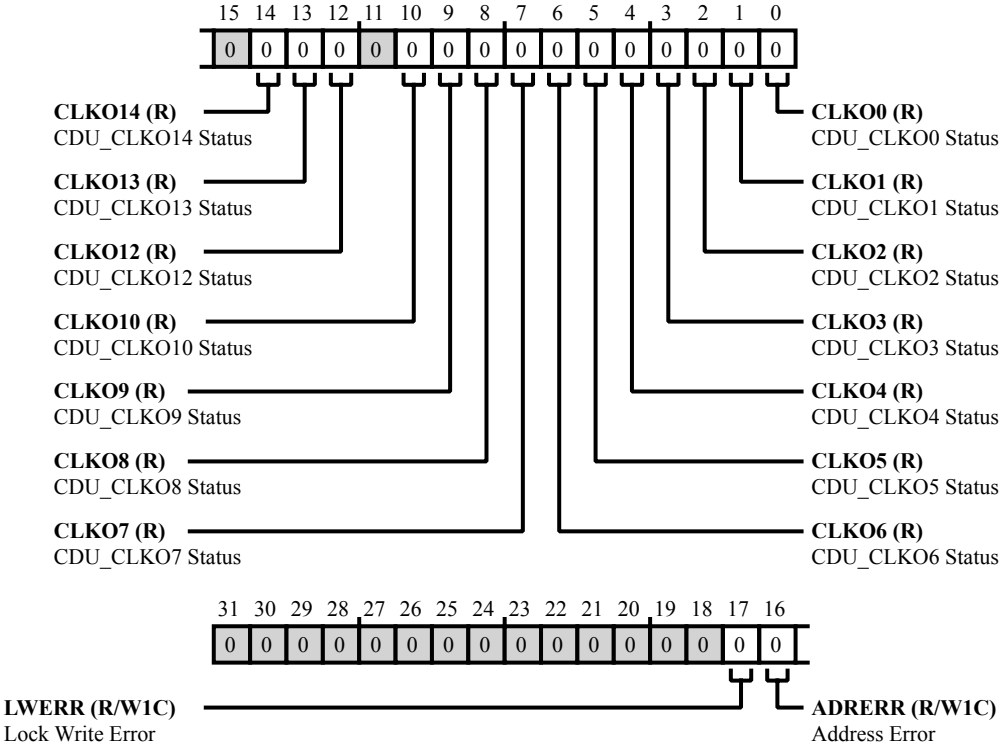


Figure 3-5: CDU_STAT Register Diagram

Table 3-8: CDU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	LWERR	Lock Write Error. The CDU_STAT.LWERR bit indicates a lock error where write transactions try to access write protected registers.	
		0	No Lock Write Error
		1	Lock Write Error
16 (R/W1C)	ADRERR	Address Error. The CDU_STAT.ADRERR bit indicates an address error where read or write transactions try to access unimplemented addresses or accesses are non-aligned.	
		0	No Address Error
		1	Address Error

Table 3-8: CDU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/NW)	CLKO14	CDU_CLKO14 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
13 (R/NW)	CLKO13	CDU_CLKO13 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
12 (R/NW)	CLKO12	CDU_CLKO12 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
10 (R/NW)	CLKO10	CDU_CLKO10 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
9 (R/NW)	CLKO9	CDU_CLKO9 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
8 (R/NW)	CLKO8	CDU_CLKO8 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
7 (R/NW)	CLKO7	CDU_CLKO7 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
6 (R/NW)	CLKO6	CDU_CLKO6 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
5 (R/NW)	CLKO5	CDU_CLKO5 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
4 (R/NW)	CLKO4	CDU_CLKO4 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress

Table 3-8: CDU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	CLKO3	CDU_CLKO3 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
2 (R/NW)	CLKO2	CDU_CLKO2 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
1 (R/NW)	CLKO1	CDU_CLKO1 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress
0 (R/NW)	CLKO0	CDU_CLKO0 Status.	
		0	No Configuration Change in Progress
		1	Configuration Change in Progress

4 Dynamic Power Management (DPM)

The dynamic power management (DPM) unit of the processor controls transitions between different power-saving modes.

DPM provides facility for shutting down clocks for various peripherals. Shutting down peripheral's clocks is considered a power saving feature. To support peripheral clock gating functionality, DPM_PER_DIS0 and DPM_PER_DIS1 are available in the DPM module. The bits in these registers need to be programmed to shut off the clocks to peripherals that are not used in an application to save clock switching power.

DPM Features

The DPM allows programs to control the power mode of the processor as follows.

- Permits operation of multiple, external wake-up sources

DPM Functional Description

The DPM can be programmed to transition between power modes.

ADSP-SC59x DPM Register List

A set of registers govern DPM operations. For more information on DPM functionality, see the DPM register descriptions.

Table 4-1: ADSP-SC59x DPM Register List

Name	Description
DPM_CTL	Control Register
DPM_PER_DIS0	Peripherals Disable Register 0
DPM_PER_DIS1	Peripherals Disable Register 1
DPM_REVID	Revision ID
DPM_STAT	Status Register

DPM Definitions

To make the best use of the DPM, it is useful to understand the following terms.

CGU

Acronym for the clock generation unit (CGU), which is comprised of the PLL and PCU

DPM

Acronym for the dynamic power management (DPM) controller.

Full-on mode

The normal operating mode in which all clock domains are derived from the PLL.

PCU

Acronym for the PLL control unit (PCU).

PLL

Acronym for the phase-locked loop (PLL).

RCU

Acronym for the reset control unit (RCU).

DPM Operating Modes

The DPM includes several operating modes. The modes are:

- Reset
- Full-on

Reset State

Reset is the initial state of the processor and is the result of a hardware or software triggered event. The DPM itself does not trigger entering reset. The external SYS_HWRST pin or the RCU triggers entering reset. The DPM responds to reset by transitioning to its default state.

From Reset, the DPM always transitions to PLL Bypassed state.

Full-on Mode

Full-on mode is the default state of the DPM after Reset.

In full-on mode, the processor can reach its maximum clock rate and power dissipation can be at its highest.

DPM Event Control

The DPM event is triggered when an enabled wake-up is asserted. The DPM generates bus errors when a misaligned access to a register occurs. It also generates errors when an attempt is made to access unused DPM address space or a write-protected register.

DPM Events

The DPM event interrupt is triggered when any bit in the DPM_STAT register is set, indicating that an enabled wake-up was asserted. The DPM event interrupt stays active until the user clears any bits that are set in the DPM_STAT register.

DPM Errors

The DPM generates a bus error when a read or write transaction is attempted to an unused address within the DPM address range. It also generates a bus error when a misaligned access is made to a DPM register. In addition to the bus error, the DPM sets the DPM_STAT.ADDRERR bit.

If a write to a write-protected DPM register is attempted, the DPM generates a bus error. In addition, the DPM sets the DPM_STAT.LWERR bit.

DPM Programming Model

The ***DPM_PER_DIS0 Register Mapping*** and ***DPM_PER_DIS1Register Mapping*** tables show the module clocks and the corresponding peripheral. The DPM_PER_DIS0 and DPM_PER_DIS1 registers are used to shut off the clock to each peripheral if it is not required by the application.

Table 4-2: DPM0_PER_DIS0 Register Mapping

Peripheral Name	Gated Module Clocks	Type of sync on PER-DIS bit	Effect latency (In cycles of module clocks)	DPM0_PER_DISn bit
FIR0/FFT	cclk	Not Required	1	0
IIR0	cclk	Not Required	1	1
DAI0	sclk0 sclk1 sysclk	Single Flop Single Flop Not Required	2 2 1	5
DAI1	sclk0 sclk1 sysclk	Single Flop Single Flop Not Required	2 2 1	6

Table 4-2: DPM0_PER_DIS0 Register Mapping (Continued)

Peripheral Name	Gated Module Clocks	Type of sync on PER-DIS bit	Effect latency (In cycles of module clocks)	DPM0_PER_DISn bit
MLB0	sysclk	Not Required	1	9
EMDMA0	sysclk	Not Required	1	13
EMDMA1	sysclk	Not Required	1	14
CRYPTO ACCELERA-TOR-0 (EIP-150/PKP)	sysclk	Not Required	1	16
CRYPTO ACCELERA-TOR-1 (EIP-93/SPE)	sysclk	Not Required	1	17
SMPU-SPIF	sysclk	Not Required	1	19
SMPU-L2CTL-CL2-0	sysclk	Not Required	1	20
SMPU-L2CTL-DL2-0	sysclk	Not Required	1	21

Table 4-3: DPM0_PER_DIS1 Register Mapping

Peripheral Name	Gated Module Clocks	Type of sync on PER-DIS bit	Effect latency (In cycles of module clocks)	DPM0_PER_DISn bit
TWI0	sclk0	Single Flop	2	0
TWI1	sclk0	Single Flop	2	1
TWI2	sclk0	Single Flop	2	2
TWI3	sclk0	Single Flop	2	3
TWI4	sclk0	Single Flop	2	4
TWI5	sclk0	Single Flop	2	5
OSPI	sysclk	Not Required	1	27

ADSP-SC59x DPM Register Descriptions

Dynamic Power Management (DPM) contains the following registers.

Table 4-4: ADSP-SC59x DPM Register List

Name	Description
DPM_CTL	Control Register
DPM_PER_DIS0	Peripherals Disable Register 0
DPM_PER_DIS1	Peripherals Disable Register 1

Table 4-4: ADSP-SC59x DPM Register List (Continued)

Name	Description
DPM_REVID	Revision ID
DPM_STAT	Status Register

Control Register

The DPM_CTL register controls sleep modes selections and PLL operations of the DPM. A write protect feature permits locking out changes to this register.

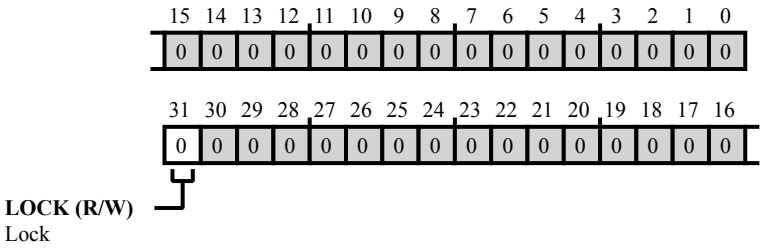


Figure 4-1: DPM_CTL Register Diagram

Table 4-5: DPM_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the DPM_CTL.LOCK bit is set, the DPM_CTL register is read only (locked).	
		0	Unlock
		1	Lock

Peripherals Disable Register 0

The DPM_PER_DIS0 register is used to shut off the clocks to peripherals that are not used in an application in order to save clock switching power.

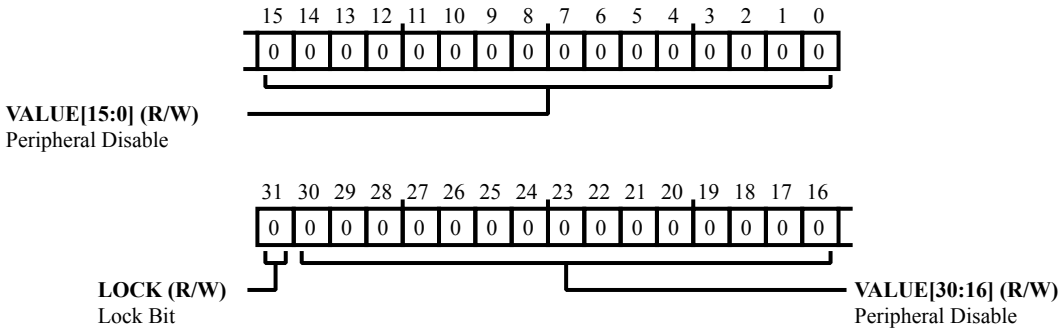


Figure 4-2: DPM_PER_DIS0 Register Diagram

Table 4-6: DPM_PER_DIS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit.	
30:0 (R/W)	VALUE	Peripheral Disable. The DPM_PER_DIS0.VALUE bits are used to shut of clocks in individual peripherals.	
		0	SH0_FIR0
		1	SH0_IIR0
		2	SH0_IIR1
		3	SH0_IIR2
		4	SH0_IIR3
		5	DAI0
		6	DAI1
		7	Reserved
		8	Reserved
		9	MLB0
		10	EMAC0 (10/100/1000)
		11	EMAC1 (10/100)
		12	MSHC0
		13	EMDMA0/CH1

Table 4-6: DPM_PER_DIS0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		14	EMDMA0/CH0
		15	Reserved
		16	SMPU_SPIF (SPI2)
		17	CRYPTO ACCELERATOR-1 (EIP-150/PKP)
		18	CRYPTO ACCELERATOR-0 (EIP-93/SPE)
		19	SMPU-SMOY
		20	SMPU-L2CTL-CL2-0
		21	SMPU-L2CTL-DL2-0
		22	SMPU-L2CTL-CL2-1
		23	SMPU-L2CTL-DL2-1
		24	SMPU-L2CTL-CL2-2
		25	Reserved
		26	Reserved
		27	SMPU-DMC0
		28	Reserved
		29	CAN0
		30	CAN1

Peripherals Disable Register 1

The DPM_PER_DIS1 register is used to shut off the clocks to peripherals that are not used in an application to save clock switching power.

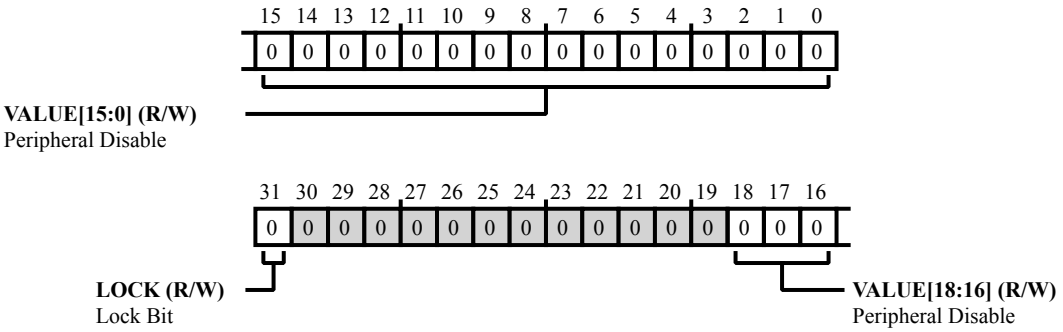


Figure 4-3: DPM_PER_DIS1 Register Diagram

Table 4-7: DPM_PER_DIS1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit.	
18:0 (R/W)	VALUE	Peripheral Disable. The DPM_PER_DIS1.VALUE bits are used to shut of clocks in individual peripherals.	
		0	TWI0
		1	TWI1
		2	TWI2
		3	TWI3
		4	TWI4
		5	TWI5
		6	xSPI_SYSCLK
		7	USBULPI
		8	Reserved
		9	Reserved
		10	Reserved
		11	Reserved
		12	Reserved
		13	Reserved

Table 4-7: DPM_PER_DIS1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		14	XSPI_CLK010
		15	PDM
		16	Reserved
		17	Reserved
		18	Reserved

Revision ID

The DPM_REVID register provides the revision of the DPM module.

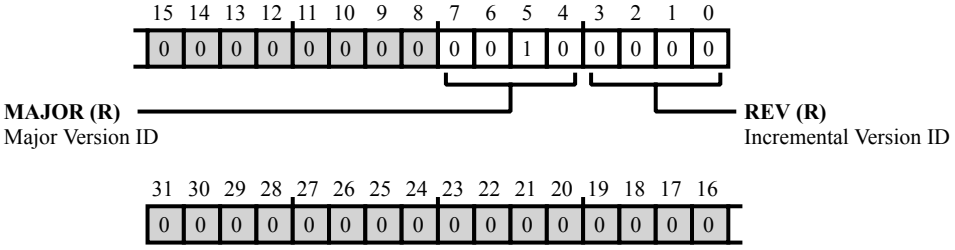


Figure 4-4: DPM_REVID Register Diagram

Table 4-8: DPM_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major Version ID.
3:0 (R/NW)	REV	Incremental Version ID.

Status Register

The DPM_STAT register contains bits that report the state of the module and various errors.

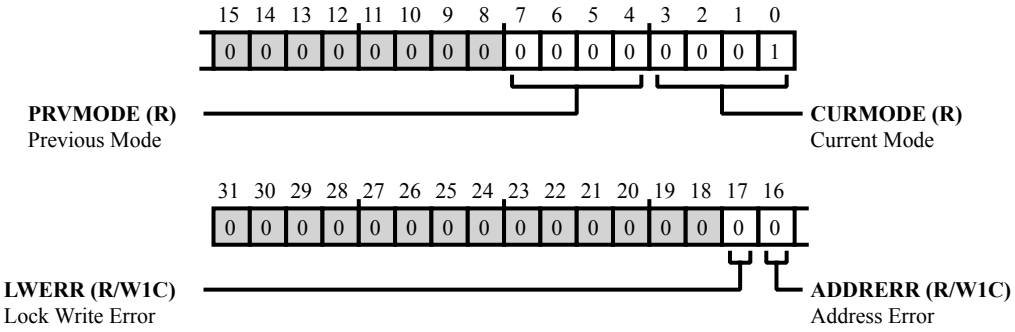


Figure 4-5: DPM_STAT Register Diagram

Table 4-9: DPM_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	LWERR	Lock Write Error. The DPM_STAT.LWERR bit indicates that a write transaction attempted an access to a write protected register. Triggers the DPMLV_PSLVERR interrupt.	
		0	Inactive
		1	Active
16 (R/W1C)	ADDRERR	Address Error. The DPM_STAT.ADDRERR bit indicates that a read or write transaction attempted an access to an unimplemented address or a write transaction attempted an access to a read only register or accesses are non aligned. Triggers the DPMLV_PSLVERR interrupt.	
		0	Inactive
		1	Active
7:4 (R/NW)	PRVMODE	Previous Mode. The DPM_STAT.PRVMODE bit field indicates the previous mode of the module.	
		0	Reset
		1	Full-On
		2	Reserved
		3	Reserved
		4	Reserved
		5-15	Reserved

Table 4-9: DPM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/NW)	CURMODE	Current Mode. The `DPM_STAT.CURMODE` bit field indicates the current mode of the module.	
		0	Reserved
		1	Full-On
		2	Reserved
		3	Reserved
		4-15	Reserved

5 Reset Control Unit (RCU)

Reset is the initial state of the processor at power-on or the run-time state of any core, as controlled by another core in the device using the RCU or as a result of a hardware or software triggered event. In this state, all control registers are set to their default values and functional units are idle. Exiting a full system hardware reset results with only the host core being released from reset and ready to boot (Core 0 on the devices ADSP-SC5x, Core 1 on the ADSP-215xx devices).

The following table shows the cores that are released from reset. Exiting a Core n only reset starts with this Core n being ready to execute the code from the software vector registers (RCU_SVECT0 , RCU_SVECT2).

Additional information on reset and booting can be found in the *Boot ROM and Booting the Processor* chapter.

Product	Core Released from Reset by RCU
ADSP-SC59x	Core 0
ADSP-2159x	Core 1

The Reset Control Unit (RCU) controls how all the functional units enter and exit reset. Differences in functional requirements and clocking constraints define how reset signals are generated. Programs must guarantee that none of the reset functions puts the system into an undefined state or causes resources to stall. This functionality is important when only one of the cores is reset (programs must ensure that there is no pending system activity involving the core that is being reset). While core resets and software system resets are controlled directly in the RCU, hardware resets can come from the TRU, SEC, or CGU Oscillator Watchdog.

RCU Features

The RCU module supports the following features:

- Hardware reset through the $\overline{\text{SYS_HWRST}}$ pin
- Software system reset through the RCU control (RCU_CTL) register
- Hardware system reset through:
  - TRU module
  - SEC module (System Fault Unit)

- A clock not good reset state (safe state of chip under reset) from the Oscillator Watchdog.
- Core reset through RCU Core Reset Output (RCU_CRCTL) register

RCU Functional Description

This section provides information on the function of RCU module.

ADSP-SC59x RCU Register List

The Reset Control Unit (RCU) controls how all the functional units in the processor enter and exit Reset. Differences in functional requirements and clocking constraints exist (units in different clock domains have to enter reset asynchronously, but units exit reset in a deterministic way), and these differences define how reset signals are generated. Reset signals propagate through all functional units asynchronously. For more information on RCU functionality, see the RCU register descriptions.

Table 5-1: ADSP-SC59x RCU Register List

Name	Description
RCU_BCODE	Boot Code Register
RCU_CRCTL	Core Reset Outputs Control Register
RCU_CRSTAT	Core Reset Outputs Status Register
RCU_CTL	Control Register
RCU_MSG	Message Register
RCU_MSG_CLR	Message Clear Bits Register
RCU_MSG_SET	Message Set Bits Register
RCU_SIDIS	System Interface Disable Register
RCU_SISTAT	System Interface Status Register
RCU_SRRQSTAT	System Reset Request Status Register
RCU_STAT	Status Register
RCU_SVECT0	Software Vector Register 0
RCU_SVECT1	Software Vector Register 1
RCU_SVECT2	Software Vector Register 2
RCU_SVECT_LCK	SVECT Lock Register

ADSP-SC59x RCU Trigger List

Table 5-2: ADSP-SC59x RCU Trigger List Generators

Trigger ID	Name	Description	Sensitivity
None			

Table 5-3: ADSP-SC59x RCU Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
50	RCU0_SYSRST0	RCU0 System Reset Slave 0	Pulse
51	RCU0_SYSRST1	RCU0 System Reset Slave 1	Pulse

RCU Definitions

To make the best use of the RCU, it is useful to understand the terms in this section.

The target or source defines the following are types of resets.

Hardware Reset (by target)

All functional units except a small subsection of debug interfaces are set to their default states. State is lost in all non-volatile storage.

System Reset (by target)

All functional units except the RCU, flash interface, and debug are set to their default states.

Core n Only Reset (by target)

Affects Core n only. The system software must guarantee that a bus requester cannot access the core in reset state.

Hardware Reset (by source)

The $\overline{\text{SYS_HWRST}}$ input signal is asserted active (pulled low).

System Reset (by source)

Software can trigger the reset by writing to the RCU_CTL register or by another functional unit such as the TRU or any of the generic reset inputs.

RCU Architectural Concepts

To understand the architecture of the RCU, it is important to consider the reset sources and how differing resets affect the functional units of the processor.

The RCU provides the hardware that controls how all the functional units enter and exit reset. Differences in functional requirements and clocking constraints define how reset signals are generated. For example, units in different clock domains must enter reset asynchronously but exit reset in a deterministic way.

The ***RCU Reset Sources*** table defines how reset sources affect the different functional units.

Table 5-4: RCU Reset Sources

Reset Source	Reset Type	Affected Functional Units
SYSCLK clock domain system reset by the Fault Unit (FMU) or	System Reset	All functional units, except: • RTC (if present) • RCU_STAT • RCU_BCODE • the units on the VDDEXT power domain
RCU_CTL.SYSRST bit set (software triggered reset)	System Reset	All functional units, except: • RTC (if present) • RCU_STAT • RCU_BCODE • the units on the VDDEXT power domain
RCU_CRCTL.CR[n] bit set (software triggered reset)	Core Only Reset	Core n only

RCU Status and Error Signals

The RCU_STAT register reflects status and error information. There are three kinds of errors that can occur in the RCU. The reset out error is triggered when RSTOUT is both asserted and deasserted at the same time. The lock write error occurs if an attempt is made to write a lock RCU register. The address error occurs if a read-only register is written to or if an attempt is made to a reserved address within the RCU MMR address range.

Resetting the Arm Core through Another Core/System Requester

The RCU allows reset of a given core n using another core or system requester. Core 0 can be individually reset by software, either setting any of CR0 bit in the RCU_CRCTL register. Cores that reset themselves cannot guarantee that all the system transactions to or from it have completed. Although a core n reset can be triggered by core n itself, it is recommended that another core or system requester trigger it. Core n can be reset to restore its functionality when it cannot execute software.

The following steps show the suggested programming sequence to reset core n only.

1. Clear the RCU_CRSTAT.CR[n] bit.
2. Disable interrupts to core n.
3. Set the RCU_CRCTL.CR[n] bit to reset core n.
4. Poll the RCU_CRSTAT.CR[n] bit until core n is in reset.
5. Clear the RCU_CRCTL.CR[n] bit to take core n out of reset.

6. Poll the RCU_CRSTAT.CR[n] bit until core n is out of reset.

Resetting a SHARC+ Core Through Another Core

Resetting a SHARC+ core involves a software handshake between the requester core which issues a reset to the SHARC+ core. The handshake is done using a core interrupt along with message passing through a variable in shared L2 memory RAM. Each SHARC+ core needs two bits for handshaking.

- Core Reset Request bit (CRR). This bit is set by the requesting core to indicate to the SHARC+ core that a core reset needs to be done. When the CRR bit is set the SHARC+ core should disable all interrupts, stop all system and memory accesses, and enter the IDLE state.
- IDLE acknowledgement (IDLE). Once the SHARC+ core is ready for reset, it sets this bit before entering the IDLE state to inform the requesting core that it is ready for reset.

Two bits are needed for SHARC0 core, and two bits are needed for SHARC1 core. These bits are:

- CCR0–SHARC0 core reset request bit in shared variable
- IDLE0–SHARC0 IDLE acknowledgement bit in shared variable
- CCR1–SHARC1 core reset request bit in shared variable
- IDLE1–SHARC1 IDLE acknowledgement bit in shared variable

Use the following programming to reset the SHARC+ core.

1. The requesting core checks the IDLE status bits in shared variable for the corresponding SHARC+ core (RCU_MSG.C1IDLE and RCU_MSG.C0IDLE).
2. When one of the core idle bits is set, then the program jumps to step 10. Otherwise the process continues to step 3.
3. The requesting core sets the CCRx bit in message register and raises the SOFT0 software interrupt through the SEC to the SHARC+ core (see Programming Examples for more information).
4. The SHARC+ core goes to into the ISR and checks the “CCRx” status bit in RCU_MSG register to ensure that software interrupt is for a core reset.

   ADDITIONAL INFORMATION: The SOFT0 software interrupt handler can be used for other non-reset/ general purposes as well.

   ADDITIONAL INFORMATION: The ISR should be in L1 memory.
5. The core should disable all interrupts before entering IDLE for reset.
6. The core sets the appropriate idle status bit (RCU_MSG.C1IDLE and RCU_MSG.C0IDLE).
7. Core enters IDLE.

8. The requester polls for the IDLEx status bit
9. The requester keeps a timeout option where if within N number of cycles the SHARC core doesn't respond then a system reset is initiated.

   ADDITIONAL INFORMATION: N can be decided by the user depending on timing criticality of the application.
10. Once the IDLEx status bit is found set, the requesting core initiates a core reset though the RCU.
11. Clear the `RCU_CRSTAT.CR[n]` bit.
12. Set the `RCU_SIDIS.SI[n]` bit to disable the interfaces for core n, to stop DMA accesses to its L1, to stop accesses to memory for core n, and stop accesses to MMRs.
13. Test the `RCU_SISTAT.SI[n]` bit to detect when accesses to core n have been disabled and all the pending transactions have completed.
14. Set the `RCU_CRCTL.CR[n]` bit to reset core n.
15. Poll the `RCU_CRSTAT.CR[n]` bit until core n is in reset.
16. Once the core is in reset, clear the `RCU_SIDIS.SI[n]` bit to re-enable the core interfaces.
17. Clear the `RCU_CRCTL.CR[n]` bit to take core n out of reset.
18. Poll the `RCU_CRSTAT.CR[n]` bit until core n is out of reset.

When a core is servicing a higher priority interrupt and gets stuck, it may not respond to the SEC.

ADSP-SC59x RCU Register Descriptions

Reset Control Unit (RCU) contains the following registers.

Table 5-5: ADSP-SC59x RCU Register List

Name	Description
RCU_BCODE	Boot Code Register
RCU_CRCTL	Core Reset Outputs Control Register
RCU_CRSTAT	Core Reset Outputs Status Register
RCU_CTL	Control Register
RCU_MSG	Message Register
RCU_MSG_CLR	Message Clear Bits Register
RCU_MSG_SET	Message Set Bits Register
RCU_SIDIS	System Interface Disable Register
RCU_SISTAT	System Interface Status Register

Table 5-5: ADSP-SC59x RCU Register List (Continued)

Name	Description
RCU_SRRQSTAT	System Reset Request Status Register
RCU_STAT	Status Register
RCU_SVECT0	Software Vector Register 0
RCU_SVECT1	Software Vector Register 1
RCU_SVECT2	Software Vector Register 2
RCU_SVECT_LCK	SVECT Lock Register

Boot Code Register

The RCU_BCODE register can be used to determine if and how core boots. This register is set to its default values by RESET.

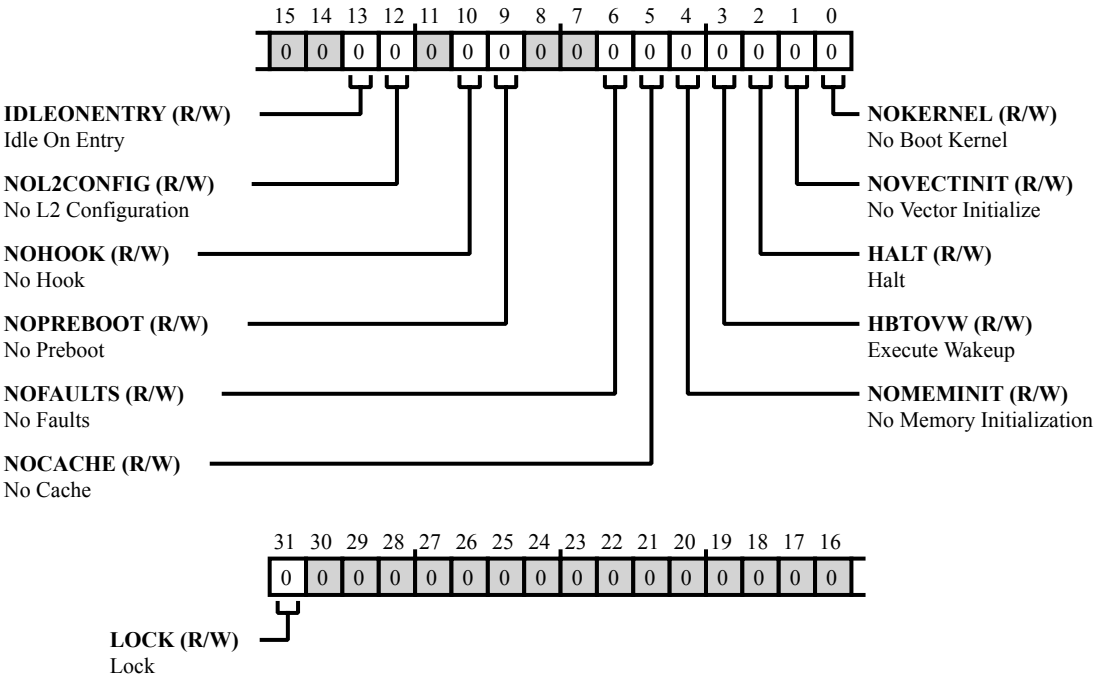


Figure 5-1: RCU_BCODE Register Diagram

Table 5-6: RCU_BCODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_BCODE.LOCK bit is set, the RCU_BCODE register is read only (locked).	
		0	Unlock
		1	Lock
13 (R/W)	IDLEONENTRY	Idle On Entry. The RCU_BCODE.IDLEONENTRY bit configures the RCU to enter the idle state at startup.	
		0	Do not enter idle state
		1	Enter idle state

Table 5-6: RCU_BCODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	NOL2CONFIG	No L2 Configuration. The `RCU_BCODE.NOL2CONFIG` bit configures the RCU to not perform the L2 memory configuration.	
		0	Configure L2 memory
		1	Do not configure L2 memory
10 (R/W)	NOHOOK	No Hook. The `RCU_BCODE.NOHOOK` bit configures the RCU to not perform the hook routine.	
		0	Perform hook routine
		1	Do not perform hook routine
9 (R/W)	NOPREBOOT	No Preboot. The `RCU_BCODE.NOPREBOOT` bit configures the RCU to not perform the customer preboot routine.	
		0	Perform preboot
		1	Do not perform preboot
6 (R/W)	NOFAULTS	No Faults. The `RCU_BCODE.NOFAULTS` bit configures the RCU to not perform fault initialization.	
		0	Perform fault initialization
		1	Do not perform fault initialization
5 (R/W)	NOCACHE	No Cache. The `RCU_BCODE.NOCACHE` bit configures the RCU to not perform a cache initialization and to not enable the cache.	
		0	Enable and initialize cache
		1	Do not initialize or enable cache
4 (R/W)	NOMEMINIT	No Memory Initialization. The `RCU_BCODE.NOMEMINIT` bit configures the RCU to not perform a memory initialization.	
		0	Perform memory initialization
		1	Do not perform memory initialization

Table 5-6: RCU_BCODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	HBTOVW	Execute Wakeup. The `RCU_BCODE.HBTOVW` bit configures the RCU to execute a wakeup.	
		0	Do not wakeup
		1	Execute wakeup
2 (R/W)	HALT	Halt. The `RCU_BCODE.HALT` bit configures the RCU to execute the no boot routine.	
		0	Do not execute routine
		1	Execute routine
1 (R/W)	NOVECTINIT	No Vector Initialize. The `RCU_BCODE.NOVECTINIT` bit configures the RCU to not vector to the application.	
		0	Vector
		1	Do not vector
0 (R/W)	NOKERNEL	No Boot Kernel. The `RCU_BCODE.NOKERNEL` bit configures the RCU to not execute the boot kernel.	
		0	Execute boot kernel
		1	Do not execute boot kernel

Core Reset Outputs Control Register

The RCU core reset control n registers (RCU_CRCTL) include a lock bit (RCU_CRCTL.LOCK) and a core reset bit (RCU_CRCTL.CR[n]) for each core reset signal on the product.

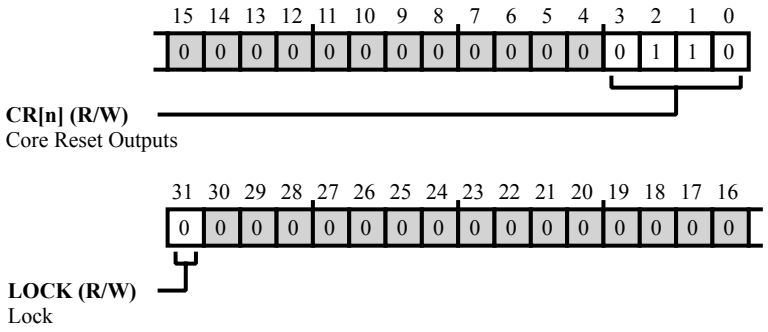


Figure 5-2: RCU_CRCTL Register Diagram

Table 5-7: RCU_CRCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_CRCTL.LOCK bit is set, the RCU_CRCTL register is read only (locked).	
		0	Unlock
		1	Lock
3:0 (R/W)	CR[n]	Core Reset Outputs. The RCU_CRCTL.CR[n] bits control CRES[1:0] core reset signals. The RCU_CRES[n] signals can be individually controlled. They are reset to their default value by a hard reset or a system reset. For each RCU_CRES[n], the selected RCU0_CRMSKi[n] bit is cleared.	
		0	RCU_CRES[3:0] Deasserted
		1	Reserved
		2	RCU_CRES[1] SHARC0 reset control bit asserted
		4	RCU_CRES[2] SHARC1 reset control bit asserted
		8	RCU_CRES[3] DEBUG reset control bit asserted
		15	RCU_CRES[3:0] Asserted

Core Reset Outputs Status Register

The RCU core reset status register (RCU_CRSTAT) contains status bits, indicating which core reset signals have been asserted.

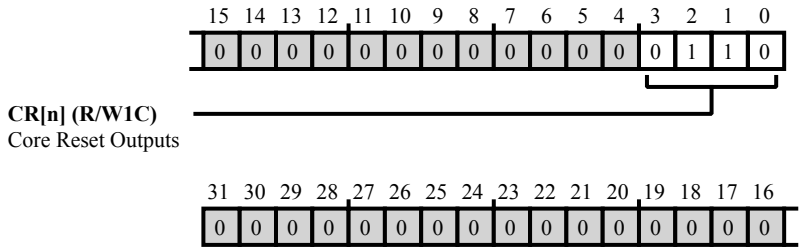


Figure 5-3: RCU_CRSTAT Register Diagram

Table 5-8: RCU_CRSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W1C)	CR[n]	Core Reset Outputs. The RCU_CRSTAT.CR[n] bits indicate which cores have been reset since the last time the bit was cleared. Bits masked by CORE_DISABLE_MASK[15:0] are permanently disabled and the corresponding CR bits set. CR bits are sticky, they need to be cleared by software.	
		0	RCU_CRES[1:0] deasserted. CR[n] corresponds to RCU_CRES[n].
		1	RCU_CRES[0] ARM A5 reset control bit asserted
		2	RCU_CRES[1] SHARC0 reset control bit asserted
		4	RCU_CRES[2] SHARC1 reset control bit asserted
		8	RCU_CRES[3] DEBUG reset control bit asserted
		15	RCU_CRES[3:0] were asserted since the last time bits were cleared. CR[n] corresponds to RCU_CRES[n].

Control Register

The RCU control register (RCU_CTL) provides a register lock, enables for the core and system reset requests inputs and control for the Reset Output pin.

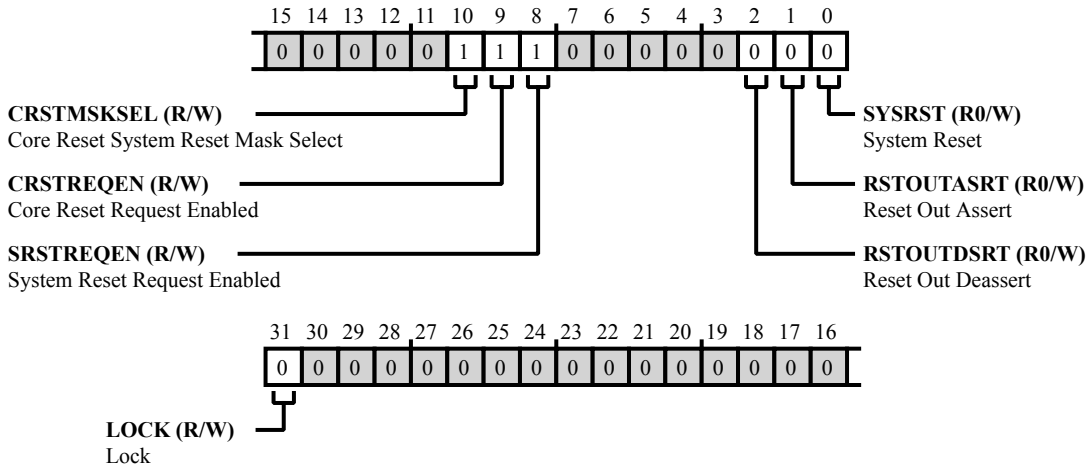


Figure 5-4: RCU_CTL Register Diagram

Table 5-9: RCU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_CTL.LOCK bit is set, the RCU_CTL register is read only (locked). This bit is cleared by a hard reset or any system reset event.	
		0	Unlock
		1	Lock
10 (R/W)	CRSTMSKSEL	Core Reset System Reset Mask Select. The RCU_CTL.CRSTMSKSEL bit selects the core reset system reset mask. This bit is cleared by a hard reset.	
9 (R/W)	CRSTREQEN	Core Reset Request Enabled. The RCU_CTL.CRSTREQEN bit controls whether the SYSCLK domain source(s) of reset is/are enabled to reset the core(s) when asserted. This bit is cleared by hard reset or any system reset event.	
		0	Disabled
		1	Enabled

Table 5-9: RCU_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	SRSTREQEN	System Reset Request Enabled. The RCU_CTL.SRSTREQEN bit controls whether the SYSCLK domain sources of reset are enabled to do a system reset when asserted. This bit is cleared by a hard reset.	
		0	Disabled
		1	Enabled
2 (R0/W)	RSTOUTDSRT	Reset Out Deassert. The RCU_CTL.RSTOUTDSRT bit controls the deassertion of the system reset pin.	
		0	No Action
		1	Deassert RSTOUT
1 (R0/W)	RSTOUTASRT	Reset Out Assert. The RCU_CTL.RSTOUTASRT bit controls assertion of the system reset pin.	
		0	No Action
		1	Assert RSTOUT
0 (R0/W)	SYSRST	System Reset. The RCU_CTL.SYSRST bit provides reset for all system units.	
		0	No Action
		1	System Reset

Message Register

The RCU_MSG is a general-purpose register. It is intended to provide flexibility for Boot ROM code and to pass predefined variables to the debugger. Please see the Booting chapter for product-specific details.

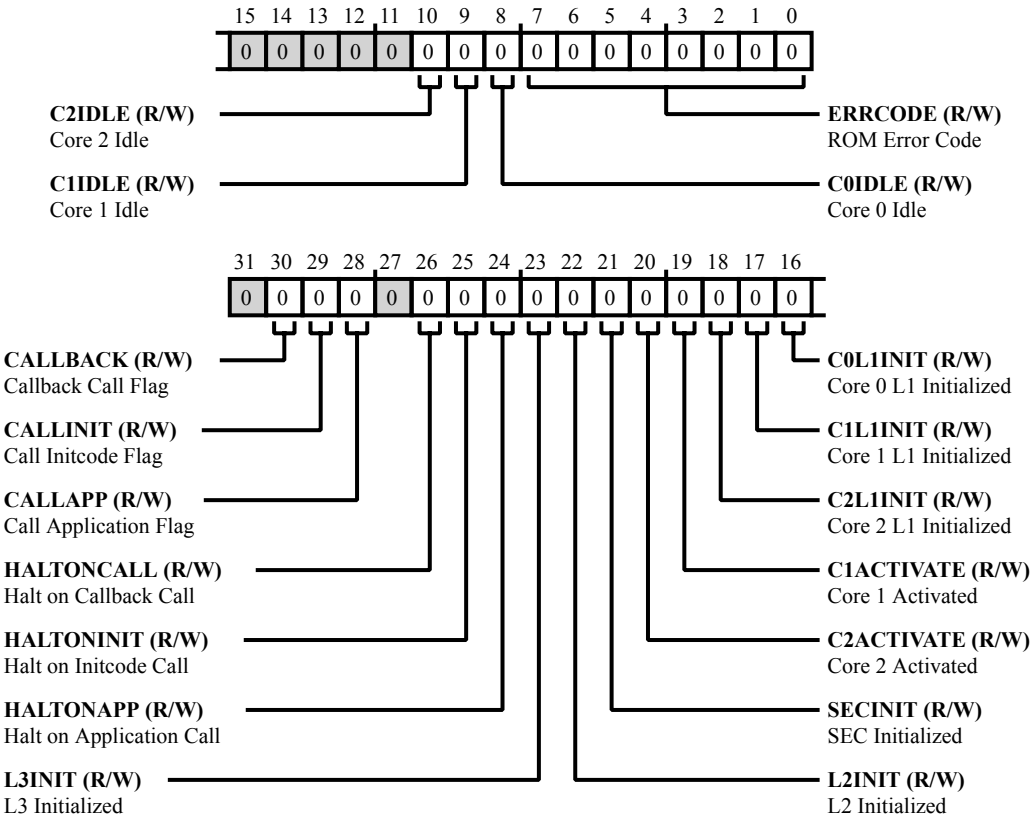


Figure 5-5: RCU_MSG Register Diagram

Table 5-10: RCU_MSG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	CALLBACK	Callback Call Flag. The RCU_MSG.CALLBACK bit indicates that a flag has been set by the boot code prior to a callback call.	
		0	Flag not set
		1	Flag set

Table 5-10: RCU_MSG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	CALLINIT	Call Initcode Flag. The RCU_MSG.CALLINIT bit indicates that a flag has been set by the boot code prior to an initcode call.	
		0	Flag not set
		1	Flag set
28 (R/W)	CALLAPP	Call Application Flag. The RCU_MSG.CALLAPP bit indicates that a flag has been set by the boot code prior to an application call.	
		0	Flag not set
		1	Flag set
26 (R/W)	HALTONCALL	Halt on Callback Call. The RCU_MSG.HALTONCALL bit generates an emulation exception prior to a callback call.	
		0	Do not generate exception
		1	Generate exception
25 (R/W)	HALTONINIT	Halt on Initcode Call. The RCU_MSG.HALTONINIT bit generates an emulation exception prior to an initcode call.	
		0	Do not generate exception
		1	Generate exception
24 (R/W)	HALTONAPP	Halt on Application Call. The RCU_MSG.HALTONAPP bit generates an emulation exception prior to an application call.	
		0	Do not generate exception
		1	Generate exception
23 (R/W)	L3INIT	L3 Initialized. The RCU_MSG.L3INIT bit indicates that the L3 resource is initialized.	
		0	Resource not initialized
		1	Resource initialized
22 (R/W)	L2INIT	L2 Initialized. The RCU_MSG.L2INIT bit indicates that the L2 resource is initialized.	
		0	Resource not initialized
		1	Resource initialized

Table 5-10: RCU_MSG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	SECINIT	SEC Initialized. The RCU_MSG.SECINIT bit is used by tools for initialization of the SEC.	
20 (R/W)	C2ACTIVATE	Core 2 Activated. The RCU_MSG.C2ACTIVATE bit is used by tools for activation of Core 2.	
19 (R/W)	C1ACTIVATE	Core 1 Activated. The RCU_MSG.C1ACTIVATE bit is used by tools for activation of Core 1.	
18 (R/W)	C2L1INIT	Core 2 L1 Initialized. The RCU_MSG.C2L1INIT bit indicates that the core 2 L1 resource is initialized.	
17 (R/W)	C1L1INIT	Core 1 L1 Initialized. The RCU_MSG.C1L1INIT bit indicates that the core 1 L1 resource is initialized.	
		0	Resource not initialized
		1	Resource initialized
16 (R/W)	C0L1INIT	Core 0 L1 Initialized. The RCU_MSG.C0L1INIT bit indicates that the core 0 L1 resource is initialized.	
		0	Resource not initialized
		1	Resource initialized
10 (R/W)	C2IDLE	Core 2 Idle. The RCU_MSG.C2IDLE bit indicates that core 2 is in a safe idle state in ROM.	
9 (R/W)	C1IDLE	Core 1 Idle. The RCU_MSG.C1IDLE bit indicates that core 1 is in a safe idle state in ROM.	
8 (R/W)	C0IDLE	Core 0 Idle. The RCU_MSG.C0IDLE bit indicates that core 0 is in a safe idle state in ROM.	
7:0 (R/W)	ERRCODE	ROM Error Code. The RCU_MSG.ERRCODE bit indicates the error code of the ROM. It is valid only when in the error handler.	

Message Clear Bits Register

The RCU_MSG_CLR register is used to clear bits in RCU_MSG register. Reading this register returns 0x00000000.

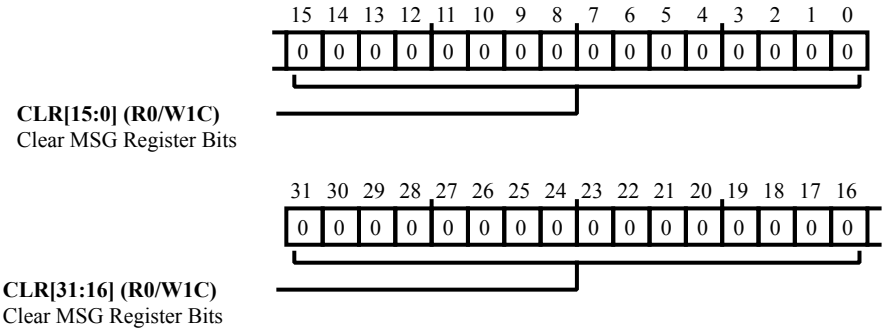


Figure 5-6: RCU_MSG_CLR Register Diagram

Table 5-11: RCU_MSG_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W1C)	CLR	Clear MSG Register Bits. The RCU_MSG_CLR.CLR bit resets MSG bit n.

Message Set Bits Register

The RCU_MSG_SET register is used to set bits in RCU_MSG register. Reading this register returns 0x00000000.

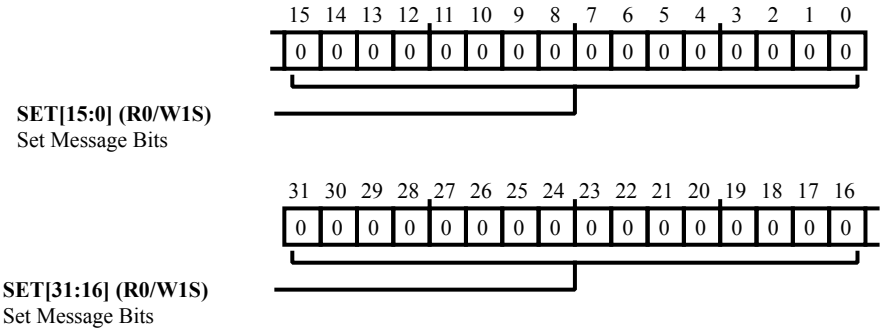


Figure 5-7: RCU_MSG_SET Register Diagram

Table 5-12: RCU_MSG_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W1S)	SET	Set Message Bits. The RCU_MSG_SET.SET bit sets MSG bit n.

System Interface Disable Register

The RCU system interface disable register (RCU_SIDIS) lets the RCU assert a system interface disable request to functional units in the processor. This register is set to its default values by a hard reset or any system reset event. For information on mapping between RCU_SIDIS bits and functional units, see the RCU functional description.

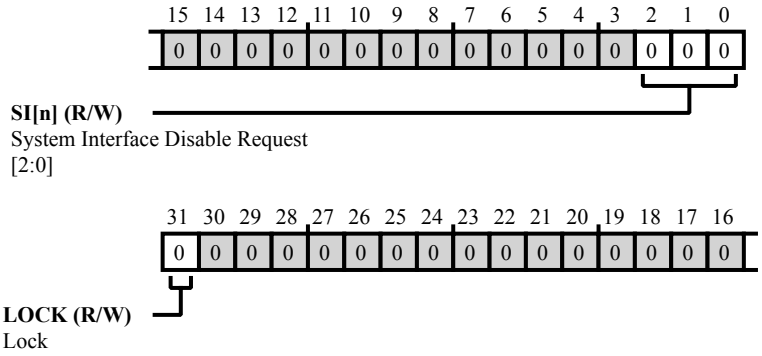


Figure 5-8: RCU_SIDIS Register Diagram

Table 5-13: RCU_SIDIS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_SIDIS.LOCK bit is set, the RCU_SIDIS register is read only (locked).	
		0	Unlock
		1	Lock
2:0 (R/W)	SI[n]	System Interface Disable Request [2:0]. Each RCU_SIDIS.SI[n] bit corresponds to a functional unit in the processor that supports the system interface disable request-acknowledge protocol.	
		0	RCU_SI_DISABLE_REQ[2:0] deasserted
		1	RCU_SI_DISABLE_REQ[0] system interface disable request to M33 asserted
		2	RCU_SI_DISABLE_REQ[1] system interface disable request to SHARC1 asserted
		4	Reserved
		7	RCU_SI_DISABLE_REQ[2:0] asserted

System Interface Status Register

The RCU system interface status register (RCU_SISTAT) indicates whether a functional unit has or has not acknowledged an RCU unit disable request.

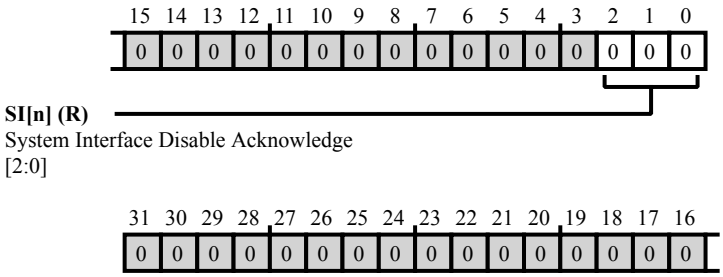


Figure 5-9: RCU_SISTAT Register Diagram

Table 5-14: RCU_SISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/NW)	SI[n]	System Interface Disable Acknowledge [2:0]. The RCU_SISTAT.SI[n] bit indicates whether a functional unit has or has not acknowledged an RCU unit disable request.	
		0	No Acknowledge
		1	RCU_SI_DISABLE_ACK[0] system interface disable acknowledge from SHARC0 asserted
		2	RCU_SI_DISABLE_ACK[1] system interface disable acknowledge from SHARC1 asserted
		4	Reserved
		7	SI_DISABLE_ACK[2:0] asserted

System Reset Request Status Register

The RCU system reset request status register (RCU_SRRQSTAT) contains status bits, indicating which system reset request input triggered a system reset. This register is set to its default values by a hard reset.

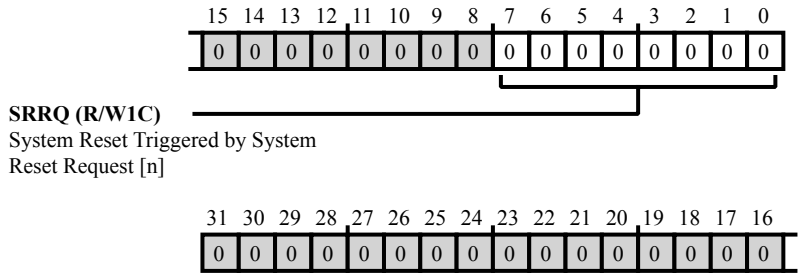


Figure 5-10: RCU_SRRQSTAT Register Diagram

Table 5-15: RCU_SRRQSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/W1C)	SRRQ	System Reset Triggered by System Reset Request [n]. The RCU_SRRQSTAT.SRRQ bits are set by the assertion of the corresponding system reset request input and deasserted by writing "1" to the bit. The RCU_SRRQSTAT register is cleared by a hard reset.	
		1	RCU_SRRQ[0], System Reset Request from SEC asserted
		2	RCU_SRRQ[1], System Reset Request from TRGS_RCU0_SYSRST0 asserted
		4	RCU_SRRQ[2], System Reset Request from TRGS_RCU0_SYSRST1 asserted
		8	RCU_SRRQ[3], System Reset Request from CTI3 asserted

Status Register

The RCU status register (RCU_STAT) contains status bits for all RCU reset sources, reset status, and boot mode inputs. Status bits for reset sources are sticky and can cleared by software. Error status bits are cleared by any reset event.

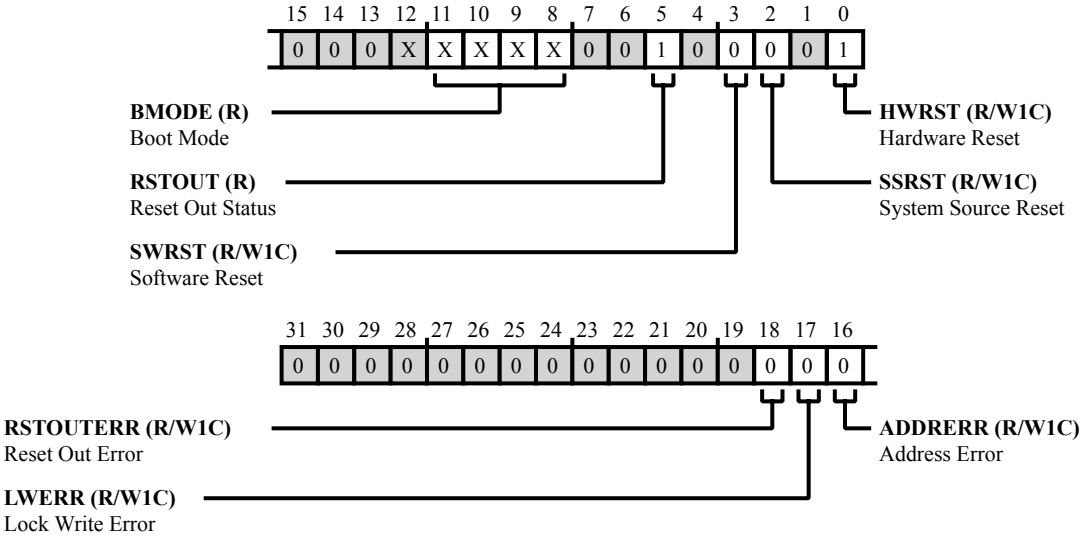


Figure 5-11: RCU_STAT Register Diagram

Table 5-16: RCU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W1C)	RSTOUTERR	Reset Out Error. The RCU_STAT.RSTOUTERR bit indicates (if set) that a write attempted to set the RCU_CTL.RSTOUTASRT and RCU_CTL.RSTOUTDSRT simultaneously. This condition triggers a bus error.	
		0	No Error
		1	Error Occurred
17 (R/W1C)	LWERR	Lock Write Error. The RCU_STAT.LWERR bit indicates (when set) there was an attempted write to an RCU register while the RCU_CTL.LOCK bit was set and the global lock bit is enabled (SPU_CTL.GLCK bit =1). This status bit is sticky; write-1-to-clear	
		0	No Error
		1	Error Occurred

Table 5-16: RCU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W1C)	ADDRERR	Address Error. The RCU_STAT.ADDRERR bit indicates that the RCU generated an address error. This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
11:8 (R/NW)	BMODE	Boot Mode. The RCU_STAT.BMODE bits indicate the input on the boot mode pins.	
5 (R/NW)	RSTOUT	Reset Out Status. The RCU_STAT.RSTOUT bit indicates the assertion status of the system reset pin.	
		0	RSTOUT Deasserted
		1	RSTOUT Asserted
3 (R/W1C)	SWRST	Software Reset. The RCU_STAT.SWRST bit indicates that a system reset (which was triggered by software) has occurred since the last time a hardware reset occurred or since the RCU_STAT.SWRST bit was cleared by software.	
		0	Inactive
		1	Reset Occurred
2 (R/W1C)	SSRST	System Source Reset. The RCU_STAT.SSRST bit indicates that a system reset triggered by hardware in the system clock domain, clock A domain, or clock B domain has occurred since the last time a hardware reset occurred or since the RCU_STAT.SSRST bit was cleared by software.	
		0	Inactive
		1	Reset Occurred
0 (R/W1C)	HWRST	Hardware Reset. The RCU_STAT.HWRST bit indicates that a hardware reset has occurred.	
		0	Inactive
		1	Reset Occurred

Software Vector Register 0

The RCU_SVECT0 register contains the default location of the first instruction to execute after a reset.

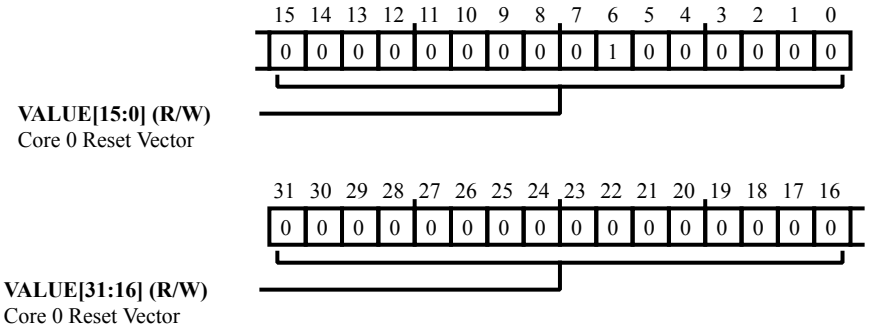


Figure 5-12: RCU_SVECT0 Register Diagram

Table 5-17: RCU_SVECT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Core 0 Reset Vector. Core0 jumps to the address in the RCU_SVECT0.VALUE bit field after reset.

Software Vector Register 1

The RCU_SVECT1 register contains the default location of the first instruction to execute after a reset.

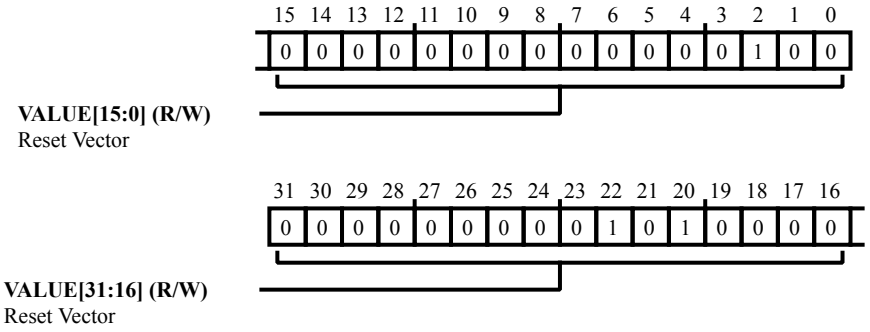


Figure 5-13: RCU_SVECT1 Register Diagram

Table 5-18: RCU_SVECT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Reset Vector. Core1 jumps to the address in the RCU_SVECT1.VALUE bit field after reset.

Software Vector Register 2

The RCU_SVECT2 register contains the default location of the first instruction to execute after a reset.

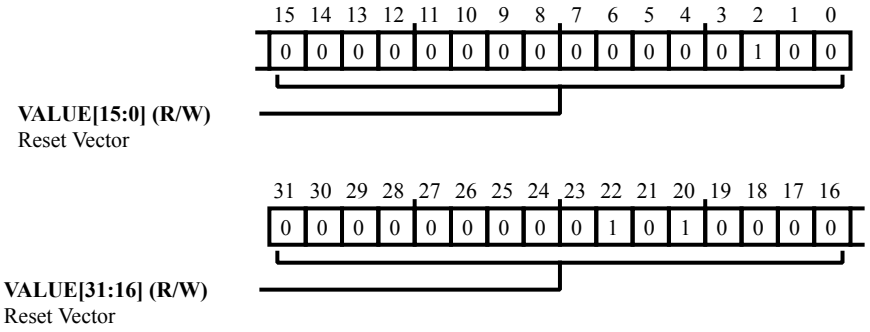


Figure 5-14: RCU_SVECT2 Register Diagram

Table 5-19: RCU_SVECT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Reset Vector. Core2 jumps to the address in the RCU_SVECT2.VALUE bit field after reset.

SVECT Lock Register

The RCU software vector lock register (RCU_SVECT_LCK) provides a register lock and software vector n enable bits for each processor core on the product. This register is set to its default values by a hard reset or any system reset event.

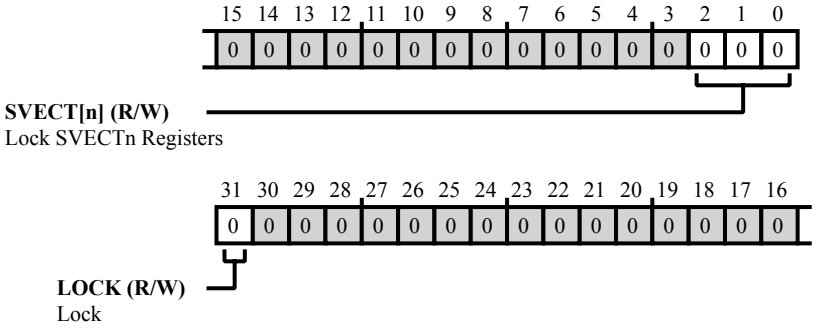


Figure 5-15: RCU_SVECT_LCK Register Diagram

Table 5-20: RCU_SVECT_LCK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_SVECT_LCK.LOCK bit is set, the RCU_SVECT_LCK register is read only (locked).	
		0	Unlock
		1	Lock
2:0 (R/W)	SVECT[n]	Lock SVECTn Registers. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the RCU_SVECT_LCK.SVECT[n] bit is set, the SVECT registers are read only (locked).	

6 System Event Controller (SEC) and Generic Interrupt Controller (GIC)

There are two interrupt controllers—a generic interrupt controller (GIC) for the Arm core and the system event controller (SEC) for the SHARC cores.

System event management is the responsibility of the system event controller (SEC). The SEC manages the configuration of all system event sources. The SEC also manages the propagation of system events to all connected cores and the system fault interface.

All the peripheral interrupts are routed using a single SEC interrupt to the desired core. The SEC allows programmability of the peripheral interrupt's priority, supporting up to 256 priority levels that are arbitrated within the SEC itself. The SEC also allows these interrupts to be grouped and masked by priority level and provides the flexibility to choose which core(s) the interrupt is routed to.

The SEC also supports self-nesting of interrupts, which is required when sharing a single interrupt request to an individual core, as this allows for a higher-priority peripheral interrupt to be passed to the core while it is currently servicing a lower-priority peripheral interrupt. For more information, refer to "Self-Nesting Mode for System Event Controller Interrupt (SECI)" in the *SHARC+ Core Programming Reference*.

For more information about the Arm GIC, visit the Arm Information Center.

SEC Features

The following list describes the system event controller features.

- Comprehensive system event source management including interrupt enable, fault enable, priority, core mapping, and source grouping.
- Fault management including fault action configuration, timeout, external indication, and system reset.
- Determinism where all system events have the same propagation delay and provide unique identification of a specific system event source.
- Distributed programming model where each system event source control and all status fields are independent of all others.

- Completer control port which provides access to all SEC registers for configuration, status, and interrupt or fault service model.
- Global locking supports a register level protection model to prevent writes to “locked” registers.

SEC Functional Description

The following sections provide a functional description of the SEC.

The *SEC/GIC Interrupt Signal Flow* figure shows an overview of the interrupt systems.

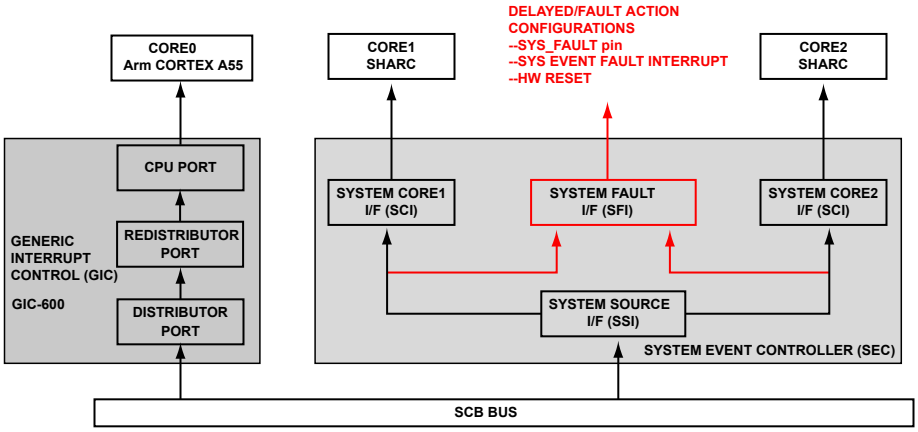


Figure 6-1: SEC/GIC Interrupt Signal Flow

ADSP-SC59x SEC Register List

The System Event Controller (SEC) manages the system fault sources, including control features such as enable/disable and active/pending source status. For more information on SEC functionality, see the SEC register descriptions.

Table 6-1: ADSP-SC59x SEC Register List

Name	Description
SEC_CACT[n]	SCI Active Register n
SEC_CCTL[n]	SCI Control Register n
SEC_CGMSK[n]	SCI Group Mask Register n
SEC_CPLVL[n]	SCI Priority Level Register n
SEC_CPMSK[n]	SCI Priority Mask Register n
SEC_CPND[n]	Core Pending Register n
SEC_CSID[n]	SCI Source ID Register n
SEC_CSTAT[n]	SCI Status Register n
SEC_END	Global End Register

Table 6-1: ADSP-SC59x SEC Register List (Continued)

Name	Description
SEC_FCOPP	Fault COP Period Register
SEC_FCOPP_CUR	Fault COP Period Current Register
SEC_FCTL	Fault Control Register
SEC_FDLY	Fault Delay Register
SEC_FDLY_CUR	Fault Delay Current Register
SEC_FEND	Fault End Register
SEC_FSID	Fault Source ID Register
SEC_FSRDLY	Fault System Reset Delay Register
SEC_FSRDLY_CUR	Fault System Reset Delay Current Register
SEC_FSTAT	Fault Status Register
SEC_GCTL	Global Control Register
SEC_GSTAT	Global Status Register
SEC_RAISE	Global Raise Register
SEC_SCTL[n]	Source Control Register n
SEC_SSTAT[n]	Source Status Register n

ADSP-SC59x SEC Interrupt List

Table 6-2: ADSP-SC59x SEC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
0	SEC0_ERR	SEC0 Error	Level	

ADSP-SC59x SEC Trigger List

Table 6-3: ADSP-SC59x SEC Trigger List Generators

Trigger ID	Name	Description	Sensitivity
76	SEC0_FAULT	SEC0 Fault	Edge

Table 6-4: ADSP-SC59x SEC Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

Combined SEC and GIC Interrupt List

The ***Combined SEC and GIC Interrupt List*** table provides a complete list of interrupts supported by the SHARC+ processor and Arm core. Note that the DAI has its own system interrupt controllers. For more information see the DAI System Interrupt Controller (SIC).

Table 6-5: Combined SEC and GIC Interrupt List

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
NA	NA	NA	GIC	GIC0_SOFT00	Arm Software Interrupt 0	0
NA	NA	NA	GIC	GIC0_SOFT01	Arm Software Interrupt 1	1
NA	NA	NA	GIC	GIC0_SOFT02	Arm Software Interrupt 2	2
NA	NA	NA	GIC	GIC0_SOFT03	Arm Software Interrupt 3	3
NA	NA	NA	GIC	GIC0_SOFT04	Arm Software Interrupt 4	4
NA	NA	NA	GIC	GIC0_SOFT05	Arm Software Interrupt 5	5
NA	NA	NA	GIC	GIC0_SOFT06	Arm Software Interrupt 6	6
NA	NA	NA	GIC	GIC0_SOFT07	Arm Software Interrupt 7	7
NA	NA	NA	GIC	GIC0_SOFT08	Arm Software Interrupt 8	9
NA	NA	NA	GIC	GIC0_SOFT09	Arm Software Interrupt 9	9
NA	NA	NA	GIC	GIC0_SOFT10	Arm Software Interrupt 10	10
NA	NA	NA	GIC	GIC0_SOFT11	Arm Software Interrupt 11	11
NA	NA	NA	GIC	GIC0_SOFT12	Arm Software Interrupt 12	12
NA	NA	NA	GIC	GIC0_SOFT13	Arm Software Interrupt 13	13
NA	NA	NA	GIC	GIC0_SOFT14	Arm Software Interrupt 14	14
NA	NA	NA	GIC	GIC0_SOFT15	Arm Software Interrupt 15	15
NA	NA	NA	NA	NA	Reserved	16-21
NA	NA	NA	PPI	COMM_IRQ	Comms channel receive or transmit interrupt request	22
NA	NA	NA	PPI	PMU_IRQ	PMU interrupt request	23
NA	NA	NA	PPI	CTI_IRQ	CTI interrupt	24
NA	NA	NA	PPI	VCPUMNT_IRQ	Virtual CPU interface maintenance interrupt PPI output	25
NA	NA	NA	PPI	CNTHP_IRQ	Hypervisor physical timer event	26
NA	NA	NA	PPI	CNTV_IRQ	Virtual timer event	27
NA	NA	NA	PPI	CNTHV_IRQ	Hypervisor virtual timer event	28

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
NA	NA	NA	PPI	CNTPS_IRQ	Secure physical timer event	29
NA	NA	NA	PPI	CNTPNS_IRQ	Non-secure physical timer event	30
NA	NA	NA	NA	NA	Reserved	31
0	0	NA	SEC	SEC0_ERR	SEC0 Error	32
1	1	NA	CGU	CGU0_EVT	CGU0 Event	33
2	2	NA	CGU	CGU1_EVT	CGU1 Event	34
3	3	NA	WDOG	WDOG0_EXP	WDOG0 Expiration	35
4	4	NA	WDOG	WDOG1_EXP	WDOG1 Expiration	36
5	5	NA	WDOG	WDOG2_EXP	WDOG2 Expiration	37
6	6	NA	OTPC	OTPC0_ERR	OTPC0 Dual bit Error Interrupt	38
7	7	NA	TMU	TMU0_FAULT	TMU0 Fault Event	39
8	8	NA	TMU	TMU0_ALERT	TMU0 Alert Event	40
9	9	NA	TAPC	TAPC0_KEYFAIL	Test/User Key Fail Interrupt	41
10	10	NA	L2CTL	L2CTL0_ECC_ERR	L2CTL0 ECC Error	42
11	11	NA	CORE-SIGHT/ L2CTL	ETR_BUFINTR	CoreSight Trace ETR (by default) or L2CTL0 ECC Warning	43
12	12	NA	L2CTL	L2CTL0_EVT	L2CTL0 Scrub/Initialization Done	44
13	13	NA	MEC	MEC1_EEIRQ0	MEC1 L2CTL0/CAN ECC Error to Core 1	45
		Reserved				Reserved
15	15	NA	MEC	MEC1_MEIRQ0	MEC1 Core 0 Memory Error Interrupt to Core 1	47
16	16	NA	MEC	MEC1_PEIRQ1	MEC1 Core 1 Parity Error Interrupt to Core 1	48
17	17	NA	MEC	MEC1_PEIRQ2	MEC1 Core 2 Parity Error Interrupt to Core 1	49
18	18	NA	MEC	MEC1_PEIRQ3	MEC1 System Peripheral Parity Error Interrupt to Core 1	50
19	19	NA	EMAC	EMAC0_DMA3_TX	EMAC0 DMA3 Tx interrupt	51
20	20	NA	EMAC	EMAC0_DMA4_TX	EMAC0 DMA4 Tx interrupt	52
21	21	NA	EMAC	EMAC0_DMA5_TX	EMAC0 DMA5 Tx interrupt	53

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
22	22	NA	EMAC	EMAC0_DMA6_TX	EMAC0 DMA6 Tx interrupt	54
23	23	NA	EMAC	EMAC0_DMA7_TX	EMAC0 DMA7 Tx interrupt	55
24	24	NA	EMAC	EMAC0_DMA0_RX	EMAC0 DMA0 Rx interrupt	56
25	25	NA	MEC	MEC2_EEIRQ0	MEC2 L2CTL0/CAN ECC Error to Core 2	57
		Reserved				Reserved
27	27	NA	MEC	MEC2_MEIRQ0	MEC2 Core 0 Memory Error Interrupt to Core 2	59
28	28	NA	MEC	MEC2_PEIRQ1	MEC2 Core 1 Parity Error Interrupt to Core 2	60
29	29	NA	MEC	MEC2_PEIRQ2	MEC2 Core 2 Parity Error Interrupt to Core 2	61
30	30	NA	MEC	MEC2_PEIRQ3	MEC2 Systerm Peripheral Parity Error Interrupt to Core 2	62
31	31	NA	EMAC	EMAC0_DMA1_RX	EMAC0 DMA1 Rx interrupt	63
32	32	NA	EMAC	EMAC0_DMA2_RX	EMAC0 DMA2 Rx interrupt	64
33	33	NA	EMAC	EMAC0_DMA3_RX	EMAC0 DMA3 Rx interrupt	65
34	34	NA	CORE	C1_IRQ0	CORE1 Data Read Interrupt	66
35	35	NA	CORE	C1_IRQ1	CORE1 Data Write Interrupt	67
36	36	NA	CORE	C1_IRQ2	CORE1 Instruction Read Interrupt	68
37	37	NA	SYSTEM	SOFT8	Software-driven Interrupt 8	69
38	38	NA	CORE	C2_IRQ0	CORE2 Data Read Interrupt	70
39	39	NA	CORE	C2_IRQ1	CORE2 Data Write Interrupt	71
40	40	NA	CORE	C2_IRQ2	CORE2 Instruction Read Interrupt	72
41	41	NA	SYSTEM	SOFT9	Software-driven Interrupt 9	73
42	42	NA	CORE	CORE1_DS_SLV_ACC_INTR	Core 1 Completer Access request, if core is in Deep Sleep	74
43	43	NA	CORE	CORE2_DS_SLV_ACC_INTR	Core2 Completer Access request, if core is in Deep Sleep	75
44	44	NA	DAI	DAI0_IRQH	DAI0 High Priority Interrupt	76
45	45	NA	DAI	DAI1_IRQH	DAI1 High Priority Interrupt	77
46	46	NA	DAI	DAI0_IRQL	DAI0 Low Priority Interrupt	78

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
47	47	NA	DAI	DAI1_IRQL	DAI1 Low Priority Interrupt	79
48	48	NA	TIMER	TIMER0_TMR0	TIMER0 Timer 0	80
49	49	NA	TIMER	TIMER0_TMR1	TIMER0 Timer 1	81
50	50	NA	TIMER	TIMER0_TMR2	TIMER0 Timer 2	82
51	51	NA	TIMER	TIMER0_TMR3	TIMER0 Timer 3	83
52	52	NA	TIMER	TIMER0_TMR4	TIMER0 Timer 4	84
53	53	NA	TIMER	TIMER0_TMR5	TIMER0 Timer 5	85
54	54	NA	TIMER	TIMER0_TMR6	TIMER0 Timer 6	86
55	55	NA	TIMER	TIMER0_TMR7	TIMER0 Timer 7	87
56	56	NA	TIMER	TIMER0_TMR8	TIMER0 Timer 8	88
57	57	NA	TIMER	TIMER0_TMR9	TIMER0 Timer 9	89
58	58	NA	TIMER	TIMER0_TMR10	TIMER0 Timer 10	90
59	59	NA	TIMER	TIMER0_TMR11	TIMER0 Timer 11	91
60	60	NA	TIMER	TIMER0_TMR12	TIMER0 Timer 12	92
61	61	NA	TIMER	TIMER0_TMR13	TIMER0 Timer 13	93
62	62	NA	TIMER	TIMER0_TMR14	TIMER0 Timer 14	94
63	63	NA	TIMER	TIMER0_TMR15	TIMER0 Timer 15	95
64	64	NA	TIMER	TIMER0_STAT	TIMER0 Status	96
65	65	NA	PINT	PINT0_BLOCK	PINT0 Pin Interrupt Block	97
66	66	NA	PINT	PINT1_BLOCK	PINT1 Pin Interrupt Block	98
67	67	NA	PINT	PINT2_BLOCK	PINT2 Pin Interrupt Block	99
68	68	NA	PINT	PINT3_BLOCK	PINT3 Pin Interrupt Block	100
69	69	NA	PINT	PINT4_BLOCK	PINT4 Pin Interrupt Block	101
70	70	NA	PINT	PINT5_BLOCK	PINT5 Pin Interrupt Block	102
71	71	NA	PINT	PINT6_BLOCK	PINT6 Pin Interrupt Block	103
72	72	NA	PINT	PINT7_BLOCK	PINT7 Pin Interrupt Block	104
73	73	NA	SYSTEM	SOFT0	Software-driven Interrupt 0	105
74	74	NA	SYSTEM	SOFT1	Software-driven Interrupt 1	106
75	75	NA	SYSTEM	SOFT2	Software-driven Interrupt 2	107
76	76	NA	SYSTEM	SOFT3	Software-driven Interrupt 3	108

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
77	77	NA	SYSTEM	SOFT4	Software-driven Interrupt 4	109
78	78	NA	SYSTEM	SOFT5	Software-driven Interrupt 5	110
79	79	NA	SYSTEM	SOFT6	Software-driven Interrupt 6	111
80	80	NA	SYSTEM	SOFT7	Software-driven Interrupt 7	112
81	81	NA	SPORT	SPORT0_A_DMA	SPORT0 Channel A DMA	113
82	82	NA	SPORT	SPORT0_A_STAT	SPORT0 Channel A Status	114
83	83	NA	SPORT	SPORT0_B_DMA	SPORT0 Channel B DMA	115
84	84	NA	SPORT	SPORT0_B_STAT	SPORT0 Channel B Status	116
85	85	NA	SPORT	SPORT1_A_DMA	SPORT1 Channel A DMA	117
86	86	NA	SPORT	SPORT1_A_STAT	SPORT1 Channel A Status	118
87	87	NA	SPORT	SPORT1_B_DMA	SPORT1 Channel B DMA	119
88	88	NA	SPORT	SPORT1_B_STAT	SPORT1 Channel B Status	120
89	89	NA	SPORT	SPORT2_A_DMA	SPORT2 Channel A DMA	121
90	90	NA	SPORT	SPORT2_A_STAT	SPORT2 Channel A Status	122
91	91	NA	SPORT	SPORT2_B_DMA	SPORT2 Channel B DMA	123
92	92	NA	SPORT	SPORT2_B_STAT	SPORT2 Channel B Status	124
93	93	NA	SPORT	SPORT3_A_DMA	SPORT3 Channel A DMA	125
94	94	NA	SPORT	SPORT3_A_STAT	SPORT3 Channel A Status	126
95	95	NA	SPORT	SPORT3_B_DMA	SPORT3 Channel B DMA	127
96	96	NA	SPORT	SPORT3_B_STAT	SPORT3 Channel B Status	128
97	97	NA	SPORT	SPORT4_A_DMA	SPORT4 Channel A DMA	129
98	98	NA	SPORT	SPORT4_A_STAT	SPORT4 Channel A Status	130
99	99	NA	SPORT	SPORT4_B_DMA	SPORT4 Channel B DMA	131
100	100	NA	SPORT	SPORT4_B_STAT	SPORT4 Channel B Status	132
101	101	NA	SPORT	SPORT5_A_DMA	SPORT5 Channel A DMA	133
102	102	NA	SPORT	SPORT5_A_STAT	SPORT5 Channel A Status	134
103	103	NA	SPORT	SPORT5_B_DMA	SPORT5 Channel B DMA	135
104	104	NA	SPORT	SPORT5_B_STAT	SPORT5 Channel B Status	136
105	105	NA	SPORT	SPORT6_A_DMA	SPORT6 Channel A DMA	137
106	106	NA	SPORT	SPORT6_A_STAT	SPORT6 Channel A Status	138

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
107	107	NA	SPORT	SPORT6_B_DMA	SPORT6 Channel B DMA	139
108	108	NA	SPORT	SPORT6_B_STAT	SPORT6 Channel B Status	140
109	109	NA	SPORT	SPORT7_A_DMA	SPORT7 Channel A DMA	141
110	110	NA	SPORT	SPORT7_A_STAT	SPORT7 Channel A Status	142
111	111	NA	SPORT	SPORT7_B_DMA	SPORT7 Channel B DMA	143
112	112	NA	SPORT	SPORT7_B_STAT	SPORT7 Channel B Status	144
113	113	NA	SPORT	GLOB-AL_SPORT_INT0_DMA	SPORT DMA GROUP0 interrupt	145
114	114	NA	SPORT	GLOB-AL_SPORT_INT1_DMA	SPORT DMA GROUP1 interrupt	146
115	115	NA	SPORT	GLOB-AL_SPORT_INT2_DMA	SPORT DMA GROUP2 interrupt	147
116	116	NA	SPORT	GLOB-AL_SPORT_INT3_DMA	SPORT DMA GROUP3 interrupt	148
117	117	NA	LP	LP0_DMA	LP0 DMA Data	149
118	118	NA	LP	LP0_STAT	LP0 Status	150
119	119	NA	LP	LP1_DMA	LP1 DMA Data	151
120	120	NA	LP	LP1_STAT	LP1 Status	152
121	121	NA	SPI	SPI0_TXDMA	SPI0 TX DMA Channel	153
122	122	NA	SPI	SPI0_RXDMA	SPI0 RX DMA Channel	154
123	123	NA	SPI	SPI0_STAT	SPI0 Status	155
124	124	NA	SPI	SPI0_ERR	SPI0 Error	156
125	125	NA	SPI	SPI1_TXDMA	SPI1 TX DMA Channel	157
126	126	NA	SPI	SPI1_RXDMA	SPI1 RX DMA Channel	158
127	127	NA	SPI	SPI1_STAT	SPI1 Status	159
128	128	NA	SPI	SPI1_ERR	SPI1 Error	160
129	129	NA	SPI	SPI2_TXDMA	SPI2 TX DMA Channel	161
130	130	NA	SPI	SPI2_RXDMA	SPI2 RX DMA Channel	162

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
131	131	NA	SPI	SPI2_STAT	SPI2 Status	163
132	132	NA	SPI	SPI2_ERR	SPI2 Error	164
133	133	NA	SPI	SPI3_TXDMA	SPI3TX DMA Channel	165
134	134	NA	SPI	SPI3_RXDMA	SPI3 RX DMA Channel	166
135	135	NA	SPI	SPI3_STAT	SPI3 Status	167
136	136	NA	SPI	SPI3_ERR	SPI3 Error	168
137	137	NA	OSPI	OSPI0_IRQ	OSPI Interrupt request	169
138	138	NA	UART	UART0_TXDMA	UART0 Transmit DMA	170
139	139	NA	UART	UART0_RXDMA	UART0 Receive DMA	171
140	140	NA	UART	UART0_STAT	UART0 Status	172
141	141	NA	UART	UART1_TXDMA	UART1 Transmit DMA	173
142	142	NA	UART	UART1_RXDMA	UART1 Receive DMA	174
143	143	NA	UART	UART1_STAT	UART1 Status	175
144	144	NA	UART	UART2_TXDMA	UART2 Transmit DMA	176
145	145	NA	UART	UART2_RXDMA	UART2 Receive DMA	177
146	146	NA	UART	UART2_STAT	UART2 Status	178
147	147	NA	UART	UART3_TXDMA	UART3 Transmit DMA	179
148	148	NA	UART	UART3_RXDMA	UART3 Receive DMA	180
149	149	NA	UART	UART3_STAT	UART3 Status	181
150	150	NA	TWI	TWI0_DATA	TWI0 Data Interrupt	182
151	151	NA	TWI	TWI1_DATA	TWI1 Data Interrupt	183
152	152	NA	TWI	TWI2_DATA	TWI2 Data Interrupt	184
153	153	NA	TWI	TWI3_DATA	TWI3 Data Interrupt	185
154	154	NA	TWI	TWI4_DATA	TWI4 Data Interrupt	186
155	155	NA	TWI	TWI5_DATA	TWI5 Data Interrupt	187
156	156	NA	CNT	CNT0_STAT	CNT0 Status	188
157	157	NA	CORE	C0_CPMUIRQ	Core 0 Cluster Performance Monitor Unit Interrupt	189
158	158	NA	CTI	ECT_C1_EVT	Core 1 CTI Event (CTI1)	190
159	159	NA	CTI	ECT_C2_EVT	Core 2 CTI Event (CTI2)	191

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
160	160	NA	PKIC	PKIC0_IRQ	Public Key Interrupt (PKA, TRNG, SL)	192
161	161	NA	PKTE	PKTE0_IRQ	Security Packet Engine Interrupt	193
162	162	NA	EMAC	EMAC0_DMA4_RX	EMAC0 DMA4 Rx Interrupt	194
163	163	NA	EMAC	EMAC0_DMA5_RX	EMAC0 DMA5 Rx Interrupt	195
164	164	NA	EMAC	EMAC0_DMA6_RX	EMAC0 DMA6 Rx Interrupt	196
165	165	NA	EMAC	EMAC0_DMA7_RX	EMAC0 DMA7 Rx Interrupt	197
166	166	NA	FIR	C1_FIR0_DMA	Core 1 FIR0 DMA	198
167	167	NA	FIR	C1_FIR0_STAT	Core 1 FIR0 Status	199
168	168	NA	IIR	C1_IIR0_DMA	Core 1 IIR0 DMA	200
169	169	NA	IIR	C1_IIR0_STAT	Core 1 IIR0 Status	201
170	170	NA	IIR	C1_IIR1_DMA	Core 1 IIR1 DMA	202
171	171	NA	IIR	C1_IIR1_STAT	Core 1 IIR1 Status	203
172	172	NA	IIR	C1_IIR2_DMA	Core 1 IIR2 DMA	204
173	173	NA	IIR	C1_IIR2_STAT	Core 1 IIR2 Status	205
174	174	NA	IIR	C1_IIR3_DMA	Core 1 IIR3 DMA	206
175	175	NA	IIR	C1_IIR3_STAT	Core 1 IIR3 Status	207
176	176	NA	FIR	C2_FIR0_DMA	Core 2 FIR0 DMA	208
177	177	NA	FIR	C2_FIR0_STAT	Core 2 FIR0 Status	209
178	178	NA	IIR	C2_IIR0_DMA	Core 2 IIR0 DMA	210
179	179	NA	IIR	C2_IIR0_STAT	Core 2 IIR0 Status	211
180	180	NA	IIR	C2_IIR1_DMA	Core 2 IIR1 DMA	212
181	181	NA	IIR	C2_IIR1_STAT	Core 2 IIR1 Status	213
182	182	NA	IIR	C2_IIR2_DMA	Core 2 IIR2 DMA	214
183	183	NA	IIR	C2_IIR2_STAT	Core 2 IIR2 Status	215
184	184	NA	IIR	C2_IIR3_DMA	Core 2 IIR3 DMA	216
185	185	NA	IIR	C2_IIR3_STAT	Core 2 IIR3 Status	217
186	186	NA	HADC	HADC0_EVT	HADC0 Interrupt	218
187	187	NA	MLB	MLB0_INT0	MLB0 AHB interrupt 0	219
188	188	NA	MLB	MLB0_INT1	MLB0 AHB Interrupt 1	220

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
189	189	NA	MLB	MLB0_STAT	MLB0 Status	221
190	190	NA	CRC	CRC0_ERR	CRC0 Error	222
191	191	NA	CRC	CRC1_ERR	CRC1 Error	223
192	192	NA	CRC	MDMA0_SRC	MDMA Source 0 (Enh BW DMA)/CRC0 In	224
193	193	NA	CRC	MDMA0_DST	MDMA Dest 0 (Enh BW DMA)/ CRC0 Out	225
194	194	NA	CRC	MDMA1_SRC	MDMA Source 1 (Enh BW DMA)/CRC1 In	226
195	195	NA	CRC	MDMA1_DST	MDMA Dest 1 (Enh BW DMA)/ CRC1 Out	227
196	196	NA	CRC	CRC0_DCNTEXP	CRC0 Datacount expiration	228
197	197	NA	CRC	CRC1_DCNTEXP	CRC1 Datacount expiration	229
198	198	NA	CRC	CRC2_ERR	CRC2 Error	230
199	199	NA	CRC	CRC3_ERR	CRC3 Error	231
200	200	NA	CRC	MDMA4_SRC	MDMA Source 4 (Enh BW DMA)/CRC2 In	232
201	201	NA	CRC	MDMA4_DST	MDMA Dest 4 (Enh BW DMA)/ CRC2 Out	233
202	202	NA	CRC	MDMA5_SRC	MDMA Source 5 (Enh BW DMA)/CRC3 In	234
203	203	NA	CRC	MDMA5_DST	MDMA Dest 5 (Enh BW DMA)/ CRC3 Out	235
204	204	NA	CRC	CRC2_DCNTEXP	CRC2 Datacount expiration	236
205	205	NA	CRC	CRC3_DCNTEXP	CRC3 Datacount expiration	237
206	206	NA	MDMA	MDMA2_SRC	Enh BW DMA Channel 0	238
207	207	NA	MDMA	MDMA2_DST	Enh BW DMA Channel 1	239
208	208	NA	MDMA	MDMA3_SRC	Max BW DMA Channel 0	240
209	209	NA	MDMA	MDMA3_DST	Max BW DMA Channel 1	241
210	210	NA	EMDMA	EMDMA0_DONE	EMDMA0 DMA Done	242
211	211	NA	EMDMA	EMDMA1_DONE	EMDMA1 DMA Done	243
212	212	NA	MDMA	MDMA7_SRC	Max BW DMA 1 Channel 0	244

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
213	213	NA	MDMA	MDMA7_DST	Max BW DMA 1 Channel 1	245
214	214	NA	MDMA	MDMA6_SRC	Enh BW DMA 1 Channel 0	246
215	215	NA	MDMA	MDMA6_DST	Enh BW DMA 1 Channel 1	247
216	216	NA	SPU	SPU0_INT	SPU0 Event	248
217	217	NA	SMPU	SMPU0_AGGR_INT	SMPU Aggregated Event	249
218	218	NA	EPPI	EPPI0_CH0_DMA	EPPI0 DMA Channel 0	250
219	219	NA	EPPI	EPPI0_CH1_DMA	EPPI0 DMA Channel 1	251
220	220	NA	EPPI	EPPI0_STAT	EPPI0 Status	252
221	221	NA	EMAC	EMAC0_STAT	EMAC0 Status	253
222	222	NA	EMAC	EMAC0_PWR	EMAC0 Power	254
223	223	NA	EMAC	EMAC0_DMA0_TX	EMAC0 DMA0 Transmit	255
224	224	NA	EMAC	EMAC0_DMA1_TX	EMAC0 DMA1 Transmit	256
225	225	NA	EMAC	EMAC0_DMA2_TX	EMAC0 DMA2 Transmit	257
226	226	NA	EMAC	EMAC0_MAC	EMAC0 MAC	258
227	227	NA	EMAC	EMAC1_STAT	EMAC1 Status	259
228	228	NA	EMAC	EMAC1_PWR	EMAC1 Power	260
229	229	NA	EMAC	EMAC1_DMA0_TX	EMAC1 DMA0 Transmit	261
230	230	NA	EMAC	EMAC1_MAC	EMAC1 MAC	262
231	231	NA	CAN	CAN0_WU_IRQ	CAN0 wakeup interrupts	263
232	232	NA	EMAC	EMAC1_DMA0_RX	EMAC1 DMA0 Receive	264
233	233	NA	CAN	CAN0_IRQ	CAN0 interrupts	265
234	234	NA	MEC	MEC_MFIRQC0	SHARC MECs (logical OR of MEC2, MEC1) Core0 Memory Fault Interrupt	266
235	235	NA	CAN	CAN0_MSG_IRQ	CAN0 message receive/transmit interrupt	267
236	236	NA	CAN	CAN1_WU_IRQ	CAN1 wakeup interrupts	268
237	237	NA	EMAC	eMMC_STAT	eMMC Status Interrupt	269
238	238	NA	CAN	CAN1_IRQ	CAN1 interrupts	270
239	239	NA	EMAC	eMMC_WAK	eMMC Wakeup interrupt	271

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
240	240	NA	CAN	CAN1_MSG_IRQ	CAN1 message receive/transmit interrupt	272
241	241	NA	USB	USB0_INT	USB0 Interrupt	273
242	242	NA	TRU	TRU0_SLV4	TRU0 Interrupt 4	274
243	243	NA	TRU	TRU0_SLV5	TRU0 Interrupt 5	275
244	244	NA	TRU	TRU0_SLV6	TRU0 Interrupt 6	276
245	245	NA	TRU	TRU0_SLV7	TRU0 Interrupt 7	277
246	246	NA	TRU	TRU0_SLV8	TRU0 Interrupt 8	278
247	247	NA	TRU	TRU0_SLV9	TRU0 Interrupt 9	279
248	248	NA	TRU	TRU0_SLV10	TRU0 Interrupt 10	280
249	249	NA	TRU	TRU0_SLV11	TRU0 Interrupt 11	281
		Reserved				Reserved
251	251	0	FIR	C1_FIR0_BUS_ERR	Core 1 FIR0 bus error	283
251	252	1	IIR	C1_IIR0_BUS_ERR	Core 1 IIR0 bus error	284
251	253	2	IIR	C1_IIR1_BUS_ERR	Core 1 IIR1 bus error	285
251	254	3	IIR	C1_IIR2_BUS_ERR	Core 1 IIR2 bus error	286
251	255	4	IIR	C1_IIR3_BUS_ERR	Core 1 IIR3 bus error	287
251	256	5	FIR	C2_FIR0_BUS_ERR	Core 2 FIR0 bus error	288
251	257	6	IIR	C2_IIR0_BUS_ERR	Core 2 IIR0 bus error	289
251	258	7	IIR	C2_IIR1_BUS_ERR	Core 2 IIR1 bus error	290
251	259	8	IIR	C2_IIR2_BUS_ERR	Core 2 IIR2 bus error	291
251	260	9	IIR	C2_IIR3_BUS_ERR	Core 2 IIR3 bus error	292
252	261	0	SPI	SPI0_TXDMA_ERR	SPI0 TX DMA Channel Error	293
252	262	1	SPI	SPI0_RXDMA_ERR	SPI0 RX DMA Channel Error	294
252	263	2	SPI	SPI1_TXDMA_ERR	SPI1 TX DMA Channel Error	295
252	264	3	SPI	SPI1_RXDMA_ERR	SPI1 RX DMA Channel Error	296
252	265	4	SPI	SPI2_TXDMA_ERR	SPI2 TX DMA Channel Error	297
252	266	5	SPI	SPI2_RXDMA_ERR	SPI2 RX DMA Channel Error	298
252	267	6	SPI	SPI3_TXDMA_ERR	SPI3 TX DMA Channel Error	299
252	268	7	SPI	SPI3_RXDMA_ERR	SPI3 RX DMA Channel Error	300

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
252	269	8	UART	UART0_TXDMA_ERR	UART0 Transmit DMA Error	301
252	270	9	UART	UART0_RXDMA_ERR	UART0 Receive DMA Error	302
252	271	10	UART	UART1_TXDMA_ERR	UART1 Transmit DMA Error	303
252	272	11	UART	UART1_RXDMA_ERR	UART1 Receive DMA Error	304
252	273	12	UART	UART2_TXDMA_ERR	UART2 Transmit DMA Error	305
252	274	13	UART	UART2_RXDMA_ERR	UART2 Receive DMA Error	306
252	275	14	UART	UART3_TXDMA_ERR	UART3 Transmit DMA Error	307
252	276	15	UART	UART3_RXDMA_ERR	UART3 Receive DMA Error	308
252	277	16	LP	LP0_DMA_ERR	LP0 DMA Data Error	309
252	278	17	LP	LP1_DMA_ERR	LP1 DMA Data Error	310
252	279	18	EPPI	EP-PI0_CH0_DMA_ERR	EPPI0 DMA Channel 0 Error	311
252	280	19	EPPI	EP-PI0_CH1_DMA_ERR	EPPI0 DMA Channel 1 Error	312
253	281	0	CRC	MDMA0_SRC_ERR	Enh BW Source 0 DMA Channel Error	313
253	282	1	CRC	MDMA0_DST_ERR	Enh BW Dest 0 DMA Channel Error	314
253	283	2	CRC	MDMA1_SRC_ERR	Enh BW Source 1 DMA Channel Error	315
253	284	3	CRC	MDMA1_DST_ERR	Enh BW Dest 1 DMA Channel Error	316
253	285	4	MDMA	MDMA2_SRC_ERR	Enh BW DMA Channel 0 Error	317
253	286	5	MDMA	MDMA2_DST_ERR	Enh BW DMA Channel 1 Error	318
253	287	6	MDMA	MDMA3_SRC_ERR	Max BW DMA Channel 0 Error	319
253	288	7	MDMA	MDMA3_DST_ERR	Max BW DMA Channel 1 Error	320
253	289	8	CRC	MDMA4_SRC_ERR	Enh BW Source 4 DMA Channel Error	321
253	290	9	CRC	MDMA4_DST_ERR	Enh BW Dest 4 DMA Channel Error	322
253	291	10	CRC	MDMA5_SRC_ERR	Enh BW Source 5 DMA Channel Error	323

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
253	292	11	CRC	MDMA5_DST_ERR	Enh BW Dest 5 DMA Channel Error	324
253	293	12	MDMA	MDMA6_SRC_ERR	Enh BW DMA Channel 4 Error	325
253	294	13	MDMA	MDMA6_DST_ERR	Enh BW DMA Channel 5 Error	326
253	295	14	MDMA	MDMA7_SRC_ERR	Max BW DMA Channel 4 Error	327
253	296	15	MDMA	MDMA7_DST_ERR	Max BW DMA Channel 5 Error	328
254	297	0	SPORT	SPORT0_A_DMA_ERR	SPORT0 Channel A DMA Error	329
254	298	1	SPORT	SPORT0_B_DMA_ERR	SPORT0 Channel B DMA Error	330
254	299	2	SPORT	SPORT1_A_DMA_ERR	SPORT1 Channel A DMA Error	331
254	300	3	SPORT	SPORT1_B_DMA_ERR	SPORT1 Channel B DMA Error	332
254	301	4	SPORT	SPORT2_A_DMA_ERR	SPORT2 Channel A DMA Error	333
254	302	5	SPORT	SPORT2_B_DMA_ERR	SPORT2 Channel B DMA Error	334
254	303	6	SPORT	SPORT3_A_DMA_ERR	SPORT3 Channel A DMA Error	335
254	304	7	SPORT	SPORT3_B_DMA_ERR	SPORT3 Channel B DMA Error	336
254	305	8	SPORT	SPORT4_A_DMA_ERR	SPORT4 Channel A DMA Error	337
254	306	9	SPORT	SPORT4_B_DMA_ERR	SPORT4 Channel B DMA Error	338
254	307	10	SPORT	SPORT5_A_DMA_ERR	SPORT5 Channel A DMA Error	339
254	308	11	SPORT	SPORT5_B_DMA_ERR	SPORT5 Channel B DMA Error	340
254	309	12	SPORT	SPORT6_A_DMA_ERR	SPORT6 Channel A DMA Error	341
254	310	13	SPORT	SPORT6_B_DMA_ERR	SPORT6 Channel B DMA Error	342
254	311	14	SPORT	SPORT7_A_DMA_ERR	SPORT7 Channel A DMA Error	343
254	312	15	SPORT	SPORT7_B_DMA_ERR	SPORT7 Channel B DMA Error	344
255	314	1	SWU	SWU1_EVT	SWU1 Event CL2_0	346
255	315	2	SWU	SWU2_EVT	SWU2 Event DL2_0	347
255	316	3	SWU	SWU7_EVT	SWU7 Event SHARC0 S1	348
255	317	4	SWU	SWU8_EVT	SWU8 Event SHARC0 S2	349
255	318	5	SWU	SWU9_EVT	SWU9 Event SHARC1 S1	350
255	319	6	SWU	SWU10_EVT	SWU10 Event SHARC2 S2	351
255	320	7	SWU	SWU11_EVT	SWU11 Event SMMR	352
255	321	8	SWU	SWU12_EVT	SWU12 Event SPI2/OSPI	353

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
255	322	9	SWU	SWU13_EVT	SWU13 Event DMC0	354
255	323	10	SWU	SWU3_EVT	SWU3 Event CL2_1	355
255	324	11	SWU	SWU4_EVT	SWU4 Event DL2_1	356
255	325	12	SWU	SWU5_EVT	SWU5 Event CL2_2	357
NA	326	NA	MEC	MEC0_EEIRQ0	MEC0 L2CTL0/CAN ECC Error to Core 0	358
NA	328	NA	MEC	MEC0_MEIRQ0	MEC0 L2CTL0/CAN ECC Error to Core 0	360
NA	329	NA	MEC	MEC0_PEIRQ1	MEC0 Core 1 Parity Error Interrupt to Core 0	361
NA	330	NA	MEC	MEC0_PEIRQ2	MEC0 Core 2 Parity Error Interrupt to Core 0	362
NA	331	NA	MEC	MEC0_PEIRQ3	MEC0 Systerm Peripheral Parity Error Interrupt to Core 0	363
NA	332	NA	Reserved	Reserved	Reserved	364
NA	333	NA	CTI	ECT_C0_EVT	Core 0 CTI Event (CTI 0)	365
NA	334	NA	TRU	TRU0_SLV0	TRU0 Completer Interrupt Request 0	366
NA	335	NA	TRU	TRU0_SLV1	TRU0 Completer Interrupt Request 1	367
NA	336	NA	TRU	TRU0_SLV2	TRU0 Completer Interrupt Request 2	368
NA	337	NA	TRU	TRU0_SLV3	TRU0 Completer Interrupt Request 3	369
NA	339	NA	GIC	SPI_SOFT00	Software-driven SPI Interrupt 0	371
NA	340	NA	GIC	SPI_SOFT01	Software-driven SPI Interrupt 1	372
NA	341	NA	GIC	SPI_SOFT02	Software-driven SPI Interrupt 2	373
NA	342	NA	GIC	SPI_SOFT03	Software-driven SPI Interrupt 3	374
NA	343	NA	GIC	SPI_SOFT04	Software-driven SPI Interrupt 4	375
NA	344	NA	GIC	SPI_SOFT05	Software-driven SPI Interrupt 5	376
NA	345	NA	GIC	SPI_SOFT06	Software-driven SPI Interrupt 6	377
NA	346	NA	GIC	SPI_SOFT07	Software-driven SPI Interrupt 7	378
NA	347	NA	GIC	SPI_SOFT08	Software-driven SPI Interrupt 8	379

Table 6-5: Combined SEC and GIC Interrupt List (Continued)

SEC ID	Interrupt Number SHARC+	SEA ID	Module	SEC/GIC Interrupt Name	SEC/GIC Interrupt Description	Interrupt Number Arm
NA	348	NA	GIC	SPI_SOFT09	Software-driven SPI Interrupt 9	380
NA	349	NA	GIC	SPI_SOFT10	Software-driven SPI Interrupt 10	381
NA	350	NA	GIC	SPI_SOFT11	Software-driven SPI Interrupt 11	382
NA	351	NA	GIC	SPI_SOFT12	Software-driven SPI Interrupt 12	383
NA	352	NA	GIC	SPI_SOFT13	Software-driven SPI Interrupt 13	384
NA	353	NA	GIC	SPI_SOFT14	Software-driven SPI Interrupt 14	385
NA	354	NA	GIC	SPI_SOFT15	Software-driven SPI Interrupt 15	386

SEC Definitions

The event controller uses the following definitions.

SCI

SEC core interface, core interface subblock of the SEC

SEA

SEC system event aggregator generates an aggregate request to the SEC based on the active events at a given time.

SID (Identification, unique)

Source numeric identifier for each system source connected to the SEC.

SFI

SEC Fault Interface, fault management subblock of the SEC.

SPR

SEC prioritizer determines the highest priority pending interrupt and the highest priority active interrupt. The SPR provides these interrupts in the appropriate registers of the SCI for the priority and nesting model of the SCI.

SSI

SEC source interface, system event source control, and status subblock of the SEC.

System Events

System source indications including interrupts and faults.

System Source

Point of origin of system event.

SEC System Event Aggregator (SEA)

System event sources request event service from the SEC. To reduce the number of required SSI channels in the SEC, the System Event Aggregator (SEA) acts as an intermediate aggregator between the source and the SEC. All unique system events have dedicated inputs to the SEA. The SEA generates an aggregate request to the SEC based on which events are active at a given time.

To determine the active event for a given SEC ID, read the `SEC_SSTAT[n].CHID` bit field. The *Combined SEC and GIC Interrupt List* table shows five SEAs connected to SEC ID 251-255. All the sources grouped over the SEA are error events for a set of peripherals like the FIR-IIR, SPI-UART-LP-EPPI, MDMA, SPORT, and SWU. An error event from a peripheral is only expected when that peripheral is enabled. To mask SEA events from sources, disable the source peripheral conditions. There are no registers to mask events from the SEA, but there is a provision to mask the complete group output at the SEC.

Events are handled with a fixed priority order, on a first-come, first-served basis. If multiple events corresponding to the same SEC ID are active at the same time, the priority level is order dependent with the lower SEA IDs having the higher priority. The SEA events are level-based, so they remain high until the events are cleared at the source ensuring no event is missed. There is no support for nesting in a group since they all have same the SEC ID. An SEA event must be handled completely before addressing the next event.

SEC Block Diagram

The ***SEC Block Diagram*** shows the event management architecture.

System sources connect to the SEC through the SSI. Each core has a dedicated SCI. The SFI provides fault action connections to the rest of the system.

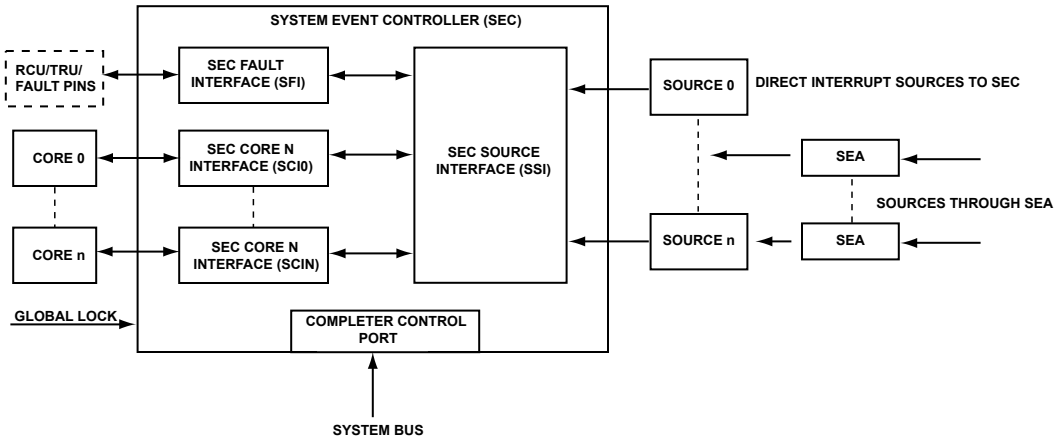


Figure 6-2: SEC Block Diagram

As shown in the figure, the SEC has three blocks for event management. The SFI monitors and manages any fault event triggered from various fault input sources. System interrupt sources are routed to the SFI through SEC source interface (SSI).

SEC Fault Interface (SFI)

The SFI manages fault events and associated actions. The fault management support provided in the SEC helps satisfy the safety requirements of various applications. The SSI provides the highest priority pending source that is enabled as a fault. The SFI captures this value and enables a countdown, and once the countdown expires, takes the prescribed action.

Fault actions which can be configured, as shown in ***SFI Block Diagram***, include

- Trigger Output
- System Reset
- Fault Output
    - Computer Operating Properly (COP) mode
    - Fault Mode

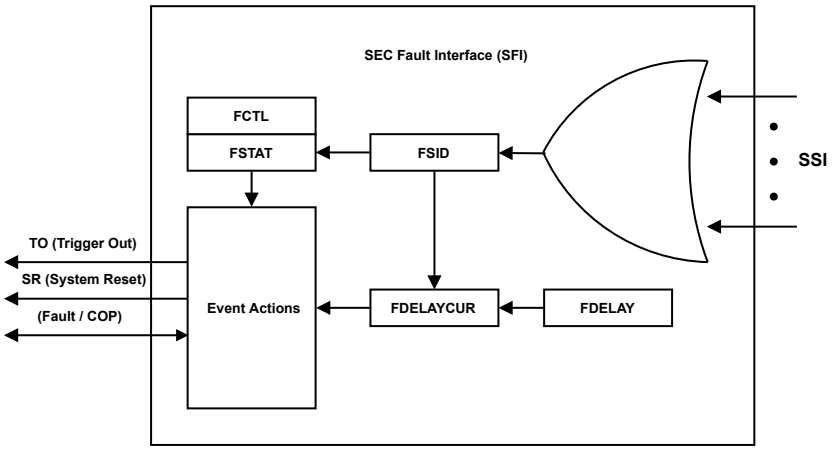


Figure 6-3: SFI Block Diagram

Fault Management

System sources can be enabled as fault sources in the SEC_SCTL[n] register. When a source enabled as a fault moves to pending, it is forwarded to the SFI as a fault indication. The pending bit (SEC_FSTAT.PND) indicates a source has signaled a fault assertion but it has not yet triggered the event actions (if delay is enabled). The SEC fault interface sets the SEC_FSTAT.PND bit when the fault source ID register (SEC_FSID) is updated on assertion of a fault source input. The system source pending triggers a fault pending and after a programmable delay the fault moves to active. Event actions then execute if appropriate action is not taken by the core. The SEC_FSTAT.ACT bit indicates that the SEC has received a fault source input, the delay has expired, and the fault actions are enabled.

The SEC_FSTAT.NPND bit indicates if one or more sources have signaled a fault assertion, but the input has not yet triggered the fault pending detection in the SEC fault interface. The SEC sets the SEC_FSTAT.NPND bit when the fault interface detects assertion of any enabled fault source input, while either the SEC_FSTAT.PND or SEC_FSTAT.ACT bits are set. The SEC clears the SEC_FSTAT.NPND bit when there are no fault sources waiting.

A fault indication from an external device can also be detected on sampling the fault signals. When a fault is detected the SEC_FSTAT.ACT and SEC_FSID.FEXT bits are set. The assertion of either signal results in a fault input detection.

The SEC_FEND register receives a fault end indication from the core. The core writes the SID of the fault to the SEC_FEND register. If the SID matches the value in the SEC_FSID register, the SEC_FSTAT.PND and SEC_FSTAT.ACT bits are cleared.

SEC Core Interface (SCI)

The SCI manages communication between the corresponding core and the SEC. The SEC prioritizer (SPR) of the SCI receives pending, active, and priority information from the SSI for each system event source assigned to this SCI. The SPR determines the highest priority pending system event and the SCI determines whether it propagates to the core. The SCI maintains the coherency for the system event service model implemented on the connected core.

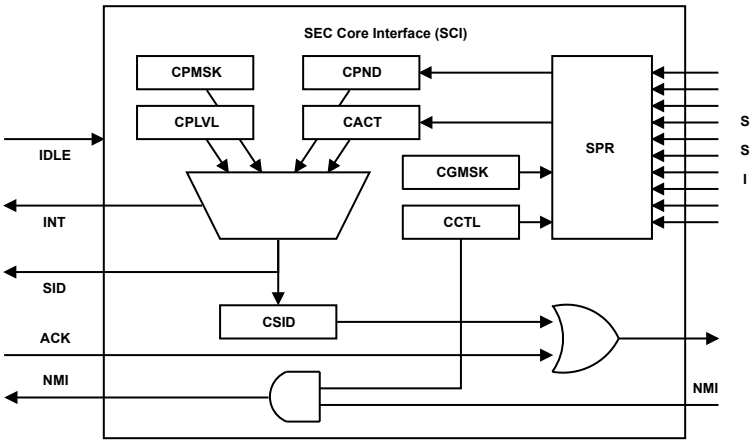


Figure 6-4: SCI Overview Block Diagram

SEC Source Interface (SSI)

The SSI manages all of the system event sources. It maintains the status of each source in the corresponding `SEC_SSTAT[n]` register. The corresponding `SEC_SCTL[n]` register manages the control of each source. A pending and enabled event passes its indication and priority to the SCI to which it is assigned for further processing.

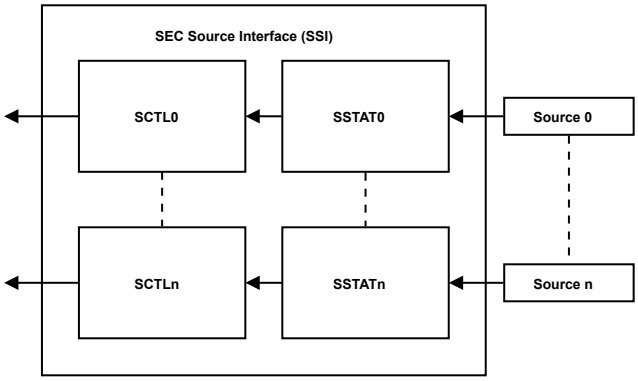


Figure 6-5: SSI Overview Block Diagram

SEC Architectural Concepts

The following sections describe SEC architectural features.

System Interrupt Acknowledge

A system interrupt acknowledge occurs when the core provides an indication that it has acquired the SID of the interrupt last issued by the SEC. The SEC core interface option allows generation by:

- A completer port write to the `SEC_CSID[n]` register.
- The assertion of an input acknowledge signal (the connected core generates the signal).

System Interrupt Groups

System sources can be assigned to groups using the SEC_SCTL[n].GRP bit field. Source groups allow fast context switching for system interrupts at each SCI. The SEC_CGMSK[n] register allows quick masking of interrupt groups of unlimited size with a single write operation.

System Interrupt Flow

An enabled and asserted system interrupt source is latched at the SSI and routed to the appropriate SCI based on the core target select (SEC_SCTL[n].CTG) bit field setting. The SEC priority ordering determines the highest priority pending system interrupt and the SCI updates the SEC_CPND[n].SID and SEC_CACT[n].PRIO bit field values. The SCI compares the SEC_CPND[n] register value against the highest priority active source in the SEC_CACT[n] register).

The priority level register (SEC_CPLVL[n]) determines how many of the MSBs the SEC uses in the comparison. The priority mask register (SEC_CPMSK[n]) and the group mask register (SEC_CGMSK[n]) determines which pending interrupt sources participate. If the SEC_CPND[n] register value is a higher priority (lower value) than the priority of the SEC_CACT[n] register from the comparison based on the SEC_CPLVL[n] register, the system interrupt output is asserted. The source ID register (SEC_CSID[n]) is updated with the SEC_CPND[n].SID bit field value and forwarded to the connected core.

After the core provides an interrupt acknowledgment, the interrupt source is active, until the SEC completes interrupt service with a write to the SEC_END.SID bit field with the same value. Note the following:

- Interrupt acknowledgement occurs with an MMR write of the SEC_CSID[n] register or the core version of the SEC_CSID[n] register.
- Interrupt active status indication is SEC_SSTAT[n].ACT==1.

The following sequence shows the example flow for a single interrupt.

1. The SEC compares the SEC_CPND[n] register value to the SEC_CACT[n] register value. If the interrupt in the SEC_CPND[n] register is higher priority, continue.
2. The SEC copies the SEC_CPND[n] register value to the SEC_CSID[n] register and asserts the interrupt signal.
3. The core reads the SEC_CSID[n] register (or core version).
4. The core writes to the SEC_CSID[n] register (or core version, asserts the acknowledge signal).
5. The SEC deasserts the interrupt signal and clears the SEC_SSTAT[n].PND bit and sets the SEC_SSTAT[n].ACT bit of the source going active.
6. The core writes the SEC_CSID[n] of the active interrupt to the SEC_END register.
7. The SEC clears the SEC_SSTAT[n].ACT bit of the source being ended.

The following sequence shows the example flow for interrupt nesting where interrupt A is a lower priority and occurs earlier than interrupt B.

1. The SEC compares the SEC_CPND[n] (A) register value to the SEC_CACT[n] register and if the interrupt in the SEC_CPND[n] register is a higher priority, continue.
2. The SEC copies SEC_CPND[n] (A) register to the SEC_CSID[n] register and asserts the interrupt signal.
3. The core reads the SEC_CSID[n] (A) register (or core version).
4. The core writes to the SEC_CSID[n] register (or core version, asserts the acknowledge signal).
5. The SEC deasserts the INT signal and clears the SEC_SSTAT[n].PND bit and sets the SEC_SSTAT[n].ACT bit of the source (A) going active.
6. The SEC compares the SEC_CPND[n] (B) register value to the SEC_CACT[n] (A) register value. If the SEC_CACT[n] (B) register value is a higher priority, continue.
7. The SEC copies the SEC_CPND[n] (B) register value to SEC_CSID[n] register and asserts the interrupt signal.
8. The core reads the SEC_CSID[n] (B) register (or core version).
9. The core writes to the SEC_CSID[n] register (or core version, asserts the acknowledge signal).
10. The SEC deasserts the INT signal and clears the SEC_SSTAT[n].PND bit and sets the SEC_SSTAT[n].ACT bit of the source (B) going active.
11. The core writes the SEC_CSID[n] of the active interrupt (B) to the SEC_END register.
12. The SEC clears the SEC_SSTAT[n].ACT bit of the source (B) being ended.
13. The core writes the SEC_CSID[n] of the active interrupt (A) to the SEC_END register.
14. The SEC clears the SEC_SSTAT[n].ACT bit of the source (A) being ended.

System Interrupt Priorities

Each system interrupt source has its own programmable priority level which is configured using the SEC_SCTL[n].PRIO bit field. The SCI evaluates the priority of all pending sources to determine the source of the highest priority pending system interrupt for forwarding to the attached core. If more than one source of the pending system interrupt has the same priority setting, the SCI chooses the one with the lowest SID. For example, if SID 0, SID 1, and SID 2 are all pending and have the same priority setting, the SCI chooses SID 0 as the highest-priority source.

SEC Error

The processor includes an SEC error (SEC_GSTAT.ERR) as a source input to the SEC to allow for handling the error as an interrupt or fault.

SEC Programming Model

Implementing a system interrupt service model using the SEC requires, at a minimum:

- Proper configuration of a system interrupt source (for example a peripheral or DMA)
- A core interrupt or event service model

The core must be configured for response to system interrupts from the SEC. The SEC must be configured to enable and map the system interrupt source to the correct SCI and to forward interrupts to the connected core.

The system interrupt source must be configured to generate interrupt assertions. Alternatively, the processor can use software triggering for interrupt assertion. Software driven interrupts are generated by writing the source ID of the interrupt to be triggered to the SEC_RAISE register.

Programming Concepts

The following list provides the basic programming concepts necessary for configuring the SEC.

- Configuring an SSI as a system interrupt for a specific core.
- Configuring an SCI to provide system interrupts to the connected core (See Configuring a System Source to Interrupt a Core).
- Configuring an SSI as a system fault (See Configuring a System Source as a Fault).
- Configuring the SFI to manage system faults.

Programming Examples

This section provides example programming tasks that are typical for SEC usage.

Fault Management Interface Programming Model

The SFI interface can be programmed to manage fault events from system sources and associated actions such as issuing a system reset when watchdog expiration event occurs.

1. Set the SEC_GCTL.EN bit to enable the SEC.
2. Write to the SEC_FCTL register to configure specific fault actions.
    - Trigger Output. Set the SEC_FCTL.TOEN bit for the SEC to produce trigger outputs when a fault becomes active. The SEC_FCTL.TES bit can be programmed to select the event that directs the SEC to assert trigger output when a fault is pending or active. Configure receivers for SEC fault trigger generator output.

      NOTE: If the SEC_FCTL.TOEN and or the SEC_FCTL.TES bits =1 (Trigger Output Enabled and Trigger on Fault Pending), an external fault (if enabled by the SEC_FCTL.FIEN bit) will not issue a trigger since Fault Pending is bypassed for external faults.
    - System Reset. The Reset Control Unit (RCU) controls how the functional units enter and exit reset. Configure the RCU_CTL.SRSTREQEN bit. This bit controls whether the sources of reset are enabled to perform a system reset. To issue a system reset request when a fault becomes active, set the SEC_FCTL.SREN bit. The SEC fault system reset delay register (SEC_FSRDLY) can be programmed for the delay, if required, from a fault becoming active to system reset request assertion.

- Fault Output. This configuration allows the SEC to indicate the fault status based on the SEC_FCTL.CMS bit configuration.
  - Computer Operating Normally (COP) mode. To configure fault output for COP mode, set the SEC_FCTL.FOEN bit to enable fault output. Set the SEC_FCTL.CMS bit to select COP mode to toggle the fault pin when no fault is active. Program the SEC_FCOPP period register with a desired width value for the COP toggled output pin.
  - Fault mode. Set the SEC_FCTL.FOEN bit to enable fault output. The SEC_FCTL.CMS bit should be set to Fault mode to toggle the fault pin when a fault is active.

3. If required, program the Fault Input to sample fault inputs from external devices on fault pins. Configure the SEC_FCTL.FIEN bit to enable the SEC to sample a fault input from an external device.

   ADDITIONAL INFORMATION: The SEC_FCTL.FIEN bit should be set only while the SEC_FCTL.EN bit is low. If the SEC_FCTL.EN bit is already high and the SEC_FCTL.FIEN bit needs to be set, the SEC_FCTL.EN bit should be cleared first. Fault input can only be enabled when Fault mode is selected by the SEC_FCTL.CMS bit.

4. Program the required fault delay to the SEC_FDLY.COUNT bit field if a delay between fault source assertion and the fault response is required.

5. Configure the SEC_FCTL register to enable the SEC.

   ADDITIONAL INFORMATION: The SEC_FCTL.EN bit should be set only while the SEC_FSTAT.ACT bit is low.

6. Write to the control register of a specific source register using the SEC_SCTL[n] register to enable the source as a fault.

Configuring a System Source to Interrupt a Core

To configure a system source to interrupt a core, the SEC itself must be enabled with the source interface (SSI) and core interface (SCI) properly initialized. Specifically, the SCI must be set up to accept interrupt signaling from the SEC and pass them to the specified core, and the SSI must properly enable each of the peripheral interrupt sources to generate interrupt signals and optionally define a priority scheme that overrides the default priority settings. In summary:

1. Write to the SEC_GCTL register to enable the SEC.
2. Write to the appropriate SCI SEC_CCTL[n] register to enable SEC interrupts to be sent to that core.
3. Write to the appropriate SSI SEC_SCTL[n] register to enable that peripheral as an interrupt source and to set the core target field to map the source to the desired SCI.
4. (Optional) By default, all the SEC interrupts are grouped as a single priority level, so passing of peripheral interrupt requests from the SEC is based solely on the default enumerated source ID. By programming the SEC_CPLVL[n].PLVL register, interrupt sources can be grouped into priority levels within the SEC such that arbitration is first performed by source ID within a grouped priority level before proceeding to the next

priority level, thus providing the flexibility to have lower-priority interrupt sources considered before higher-priority sources.

ADDITIONAL INFORMATION: The SEC_CPMSK[n] and SEC_CGMSK[n] registers must also can be programmed to mask the interrupts based on the customized levels and grouping.

Core/SEC Handshake Requirements to Ensure Proper Interrupt Handling

A specific handshake with the SEC is required to handle interrupts associated with an individual core. The handshake ensures that nested interrupts are properly tracked and that new peripheral interrupts being raised within the SEC are either passed immediately to the core or held off and queued within the SEC for later servicing.

Use the following procedure to write a custom dispatcher inside the Interrupt Service Routine. Note that the core needs to read and acknowledge the SEC_CSID[n] register by writing the same value. The core must also write to the SEC_END register after the ISR execution completes.

1. Read the SEC_CSID[n] register to obtain the source ID of the peripheral interrupt request.
2. Write the read value back to the SEC_CSID[n] register to send the acknowledge signal to the SEC that the core has accepted and begun processing the interrupt request.
3. Execute the actual ISR (typically a call to a specific handler function from a look-up table based on the peripheral source ID). Write to the SEC_GCTL register to enable the SEC.
4. Write the value of the SEC_CSID[n] register of the active interrupt (read in step 1 above) to the SEC_END register to signal to the SEC that the interrupt has now been serviced.
5. Return from interrupt.

This procedure allows a higher-priority interrupt raised by the SEC to be serviced by the core after step 2. The SEC knows what it passed to the core because of the write to the SEC_CSID[n] register. After the core acknowledges that write, the SEC knows whether newly raised peripheral interrupts are a higher priority than the highest-priority interrupt currently being processed by the core.

- When a higher priority, the SEC pushes the current SEC_CSID[n] value to an internal stack, writes the new SEC_CSID[n] value, and asserts a new SEC interrupt request.
- When a lower priority, the SEC queues the interrupt until the core writes to the SEC_END register with the source ID of the higher-priority interrupt, confirming that it was fully processed.

At this point the SEC_CSID[n] value is popped from the internal stack and any pending peripheral interrupt requests are arbitrated before the SEC writes the new SEC_CSID[n] value and asserts a new interrupt request. At the same time the core self-nests the latched SEC interrupt requests as needed. When a higher-priority interrupt is presented to the core the write to the SEC_END register in the SEC handler epilog code guarantees that each nested level has the required handshake to signal to the SEC block that each individual source ID interrupt request is fully serviced. See the *SHARC+ Core Programming Reference* for more details about SEC handler code.

Configuring a System Source as a Fault

Use the following procedure configure a system source as a fault.

1. Write to the SEC_GCTL register to enable the SEC.
2. Write to the SEC_FCTL register to configure specific fault actions.
3. Write to the SEC_FDLY bit field to specify fault delay.
4. Write to the control register of a specific source to enable the source as a fault.

Configuring the WDOG Expiry Event to Issue a System Reset

Use the following procedure to configure the WDOG timer to issue a system reset.

1. Configure the SEC_GCTL register to enable the SEC.

```
*pREG_SEC0_GCTL = BITM_SEC_GCTL_EN;
```

2. Configure the SEC_FCTL register to choose the Fault response mode. In the following code example, the system reset is issued.

```
*pREG_RCU0_CTL |= BITM_RCU_CTL_SRSTREQEN;
*pREG_SEC0_FCTL |= BITM_SEC_FCTL_SREN;
```

3. Configure the SEC_SCTL[n].SEN and SEC_SCTL[n].FEN bits in the registers to determine how the fault source is handled. To configure the WDOG as the fault source, program the register. The program can configure any interrupt as the fault source by programming the corresponding register.

```
 *pREG_SEC0_SCTL3 = BITM_SEC_SCTL_FEN|BITM_SEC_SCTL_SEN;
```

   ADDITIONAL INFORMATION: The SEC ID corresponding to WDOG0 is 3, as indicated in the Combined SEC and GIC Interrupt List .

4. Write to the enable bit.

```
*pREG_SEC0_FCTL |= BITM_SEC_FCTL_EN;
```

SEC Programming Restrictions

Setting the SEC_FCTL.EN bit while the SEC_FSTAT.ACT bit is high can result in unpredictable behavior. To avoid this issue, set the SEC_FCTL.EN bit while the SEC_FSTAT.ACT bit is low. The SEC_FSTAT.ACT bit is only set when the SEC_FCTL.EN bit is high. Therefore, the problem can only occur if the SEC_FCTL.EN bit transitions from 1 to 0 and then to 1 again.

Writing to SEC_FEND to end a fault with both the SEC_FCTL.FOEN bit and the SEC_FCTL.FIEN bit set can result in erroneous external fault detection. If this operation (ending a fault) and configuration (fault input and fault output enabled) are required by the application, clear the SEC_FCTL.FOEN bit prior to writing to SEC_FEND.

The recommended sequence for ending a fault with the SEC_FCTL.FIEN or SEC_FCTL.FOEN==1 is as follows:

1. Clear the SEC_FCTL.FOEN bit.
2. Write to the SEC_FEND register.
3. Set the SEC_FCTL.FOEN bit.

ADSP-SC59x SEC Register Descriptions

System Event Controller (SEC) contains the following registers.

Table 6-6: ADSP-SC59x SEC Register List

Name	Description
SEC_CACT[n]	SCI Active Register n
SEC_CCTL[n]	SCI Control Register n
SEC_CGMSK[n]	SCI Group Mask Register n
SEC_CPLVL[n]	SCI Priority Level Register n
SEC_CPMSK[n]	SCI Priority Mask Register n
SEC_CPND[n]	Core Pending Register n
SEC_CSID[n]	SCI Source ID Register n
SEC_CSTAT[n]	SCI Status Register n
SEC_END	Global End Register
SEC_FCOPP	Fault COP Period Register
SEC_FCOPP_CUR	Fault COP Period Current Register
SEC_FCTL	Fault Control Register
SEC_FDLY	Fault Delay Register
SEC_FDLY_CUR	Fault Delay Current Register
SEC_FEND	Fault End Register
SEC_FSID	Fault Source ID Register
SEC_FSRDLY	Fault System Reset Delay Register
SEC_FSRDLY_CUR	Fault System Reset Delay Current Register
SEC_FSTAT	Fault Status Register
SEC_GCTL	Global Control Register
SEC_GSTAT	Global Status Register
SEC_RAISE	Global Raise Register

Table 6-6: ADSP-SC59x SEC Register List (Continued)

Name	Description
SEC_SCTL[n]	Source Control Register n
SEC_SSTAT[n]	Source Status Register n

SCI Active Register n

The SEC SCI active interrupt register (SEC_CACT[n]) contains the source ID and priority of the highest priority active interrupt detected by the SEC prioritizer.

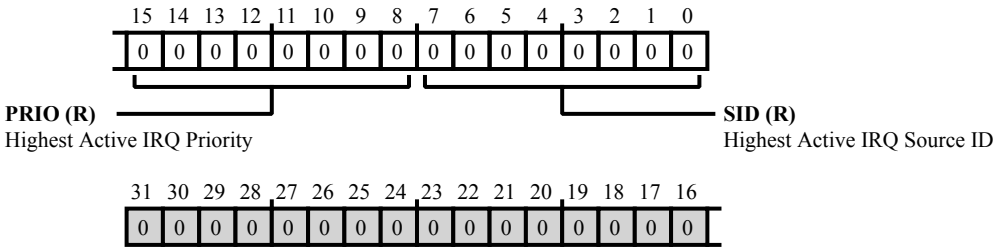


Figure 6-6: SEC_CACT[n] Register Diagram

Table 6-7: SEC_CACT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/NW)	PRIO	Highest Active IRQ Priority. The SEC_CACT[n].PRIO indicates the priority value of the highest priority active interrupt for core n.
7:0 (R/NW)	SID	Highest Active IRQ Source ID. The SEC_CACT[n].SID identifies the source ID value of the highest priority active interrupt for core n.

SCI Control Register n

The SEC control register (SEC_CCTL[n]) contains SCI control bits for all system sources.

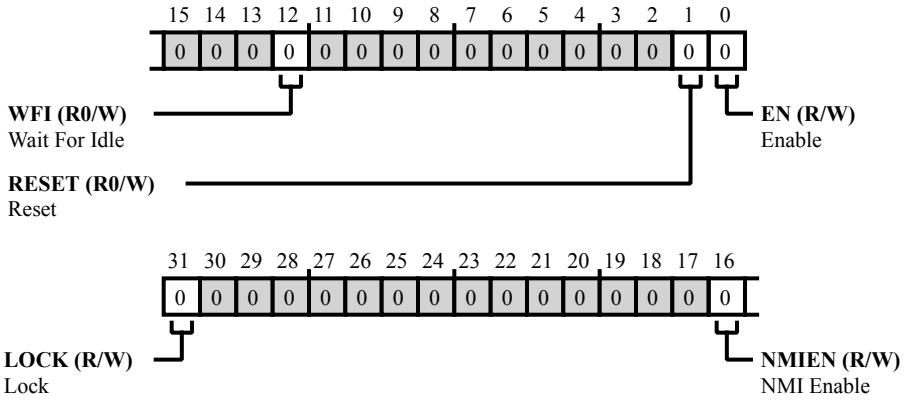


Figure 6-7: SEC_CCTL[n] Register Diagram

Table 6-8: SEC_CCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_CCTL[n].LOCK bit is enabled, the SEC_CCTL[n] register is read only.	
		0	Unlock
		1	Lock
16 (R/W)	NMIEN	NMI Enable. The SEC_CCTL[n].NMIEN bit controls NMI propagation to the core. When the SEC_CCTL[n].NMIEN bit is enabled, the SCI allows NMIs to propagate to the core for servicing.	
		0	Disable
		1	Enable
12 (R0/W)	WFI	Wait For Idle. When set, the SEC_CCTL[n].WFI bit forces the SCI to wait for indication of core idle before the SCI resumes activity.	
		0	No Action
		1	Wait for Idle

Table 6-8: SEC_CCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R0/W)	RESET	Reset. When set, the `SEC_CCTL[n].RESET` bit resets all SCI registers to their default values.	
		0	No Action
		1	Reset
0 (R/W)	EN	Enable. The `SEC_CCTL[n].EN` bit controls operation of the SCI. Clearing the `SEC_CCTL[n].EN` bit halts the execution of the SCI without resetting status registers. (The INT signal to a core is not affected.) Setting the `SEC_CCTL[n].EN` bit enables the SCI to begin or resume operation with the current configuration and status.	
		0	Disable
		1	Enable

SCI Group Mask Register n

The SEC SCI group mask register (SEC_CGMSK[n]) contains selections for a group mask, an ungroup mask, and a register lock. This register contains the system interrupt group masks for the connected core. The core uses the SEC_CGMSK[n].UGRP and SEC_CGMSK[n].GRP fields to mask (disable) interrupts from the specified groups.

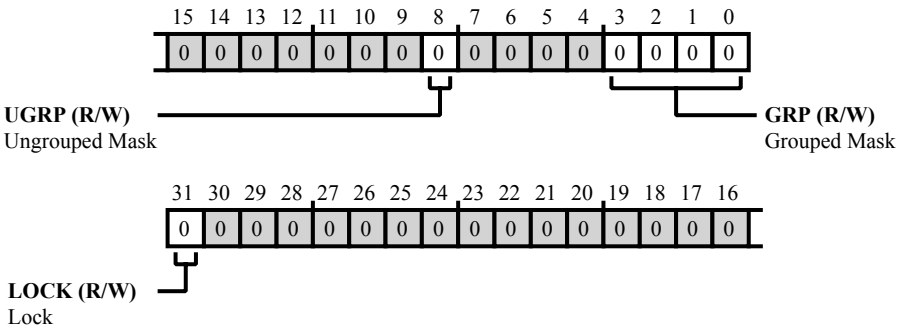


Figure 6-8: SEC_CGMSK[n] Register Diagram

Table 6-9: SEC_CGMSK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_CGMSK[n].LOCK bit is enabled, the SEC_CGMSK[n] register is read only.	
		0	Unlock
		1	Lock
8 (R/W)	UGRP	Ungrouped Mask. The SEC_CGMSK[n].UGRP bit masks interrupts (if set) for the ungrouped interrupt sources for core n.	
		0	Unmask Ungrouped Sources
		1	Mask Ungrouped Sources
3:0 (R/W)	GRP	Grouped Mask. The SEC_CGMSK[n].GRP field selects a group of interrupt sources to mask for core n. (For more information about interrupt source groups, see the SEC_SCTL[n] register description.)	
		0	No groups masked
		1	Mask group 0
		2	Mask group 1
		3	Mask groups 0, 1

Table 6-9: SEC_CGMSK[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4	Mask group 2
		5	Mask groups 0, 2
		6	Mask groups 1, 2
		7	Mask groups 0, 1, 2
		8	Mask group 3
		9	Mask groups 0, 3
		10	Mask groups 1, 3
		11	Mask groups 0, 1, 3
		12	Mask groups 2, 3
		13	Mask groups 0, 2, 3
		14	Mask groups 1, 2, 3
		15	Mask groups 0, 1, 2, 3

SCI Priority Level Register n

The SEC SCI priority level register (SEC_CPLVL[n]) contains selections for priority levels and a register lock. This register is used to divide the total number of priority levels into sub-levels. The sub-level priority resolution provides the tie breaker for simultaneously pending interrupts assigned to the same level.

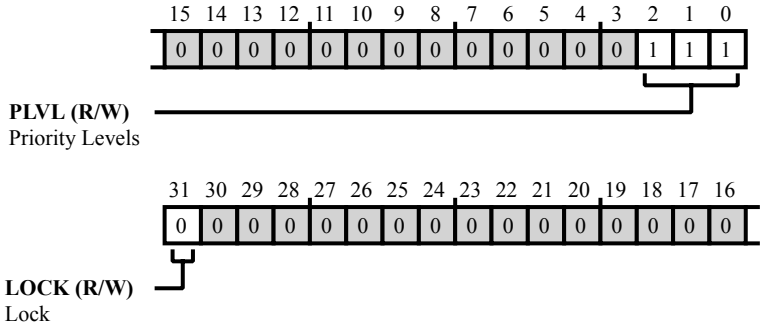


Figure 6-9: SEC_CPLVL[n] Register Diagram

Table 6-10: SEC_CPLVL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_CPLVL[n].LOCK bit is enabled, the SEC_CPLVL[n] register is read only.	
		0	Unlock
		1	Lock
2:0 (R/W)	PLVL	Priority Levels. The SEC_CPLVL[n].PLVL field serves to divide the total number of interrupt priority levels into sub-levels. The sub-level priority resolution provides the tie breaker for simultaneously pending interrupts assigned to the same interrupt level. The sub-level priority value specifies the number of MSBs (minus 1) designated to interrupt levels, while the remaining LSBs are designated for sub-level specification. For example, if the SEC_CPLVL[n].PLVL field is set to two, the result is four priority levels are specified, because only the two MSBs are used for preemption evaluation. The remaining bits of the priority setting are used for sub-level prioritization.	
		0	1 MSBs (2 priority levels)
		1	2 MSBs (4 priority levels)
		2	3 MSBs (8 priority levels)
		3	4 MSBs (16 priority levels)
		4	5 MSBs (32 priority levels)
		5	6 MSBs (64 priority levels)

Table 6-10: SEC_CPLVL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	7 MSBs (128 priority levels)
		7	8 MSBs (256 priority levels)

SCI Priority Mask Register n

The SEC SCI priority mask register (SEC_CPMSK[n]) contains the SCI priority mask for core n and includes a register lock.

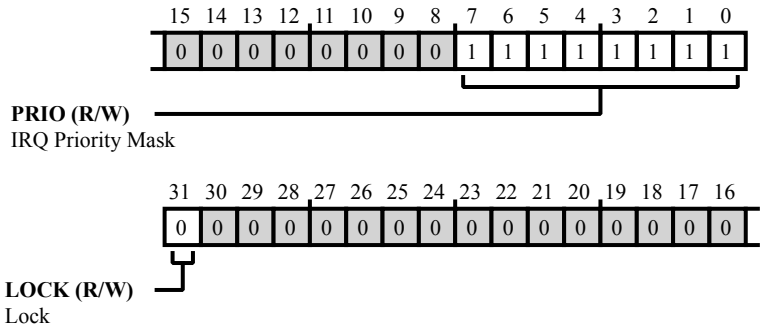


Figure 6-10: SEC_CPMSK[n] Register Diagram

Table 6-11: SEC_CPMSK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_CPMSK[n].LOCK bit is enabled, the SEC_CPMSK[n] register is read only.	
		0	Unlock
		1	Lock
7:0 (R/W)	PRIO	IRQ Priority Mask. The SEC_CPMSK[n].PRIO contains the system interrupt priority mask for core n. The core uses the SEC_CPMSK[n].PRIO field to mask (block) interrupts below the specified level.	
		0	Priority level 0 (highest)
		1-254	
		255	Priority level 255 (lowest)

Core Pending Register n

The SCI pending interrupt register (SEC_CPND[n]) contains the source ID and priority of the highest priority pending interrupt detected by the SEC prioritizer.

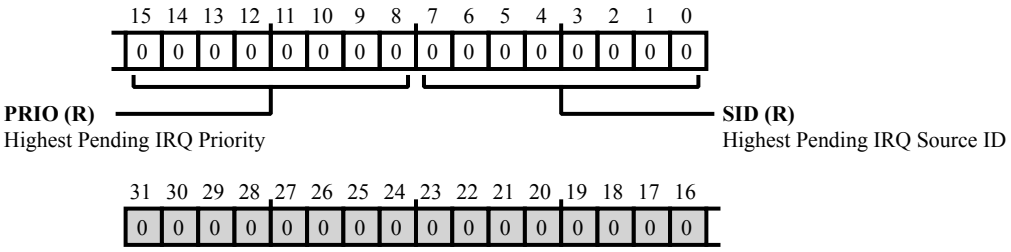


Figure 6-11: SEC_CPND[n] Register Diagram

Table 6-12: SEC_CPND[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/NW)	PRIO	Highest Pending IRQ Priority. The SEC_CPND[n].PRIO indicates the priority value of the highest priority pending interrupt for core n.
7:0 (R/NW)	SID	Highest Pending IRQ Source ID. The SEC_CPND[n].SID identifies the source ID value of the highest priority pending interrupt for core n.

SCI Source ID Register n

The SCI source ID register (SEC_CSID[n]) contains the source ID of the interrupt last issued to core n. The SEC_CSID[n] register value is loaded by the SCI when a system interrupt indication is sent to core n. The SCI does not change the SEC_CSID[n] until after the interface receives an interrupt acknowledge from core n. Writing to the SEC_CSID[n] register generates an interrupt acknowledge, but does not update the value in the register.

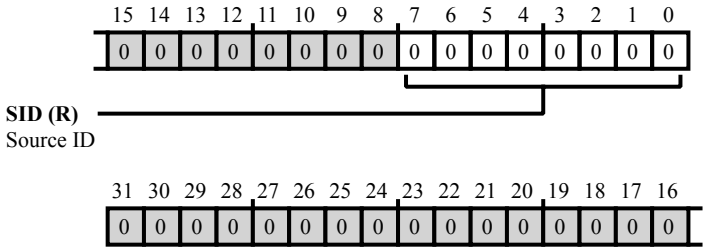


Figure 6-12: SEC_CSID[n] Register Diagram

Table 6-13: SEC_CSID[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	SID	Source ID. The SEC_CSID[n].SID bit is the source ID of the interrupt last issued to core n.

SCI Status Register n

The SCI status register (SEC_CSTAT[n]) contains status bits, indicating the operational status of the SCI.

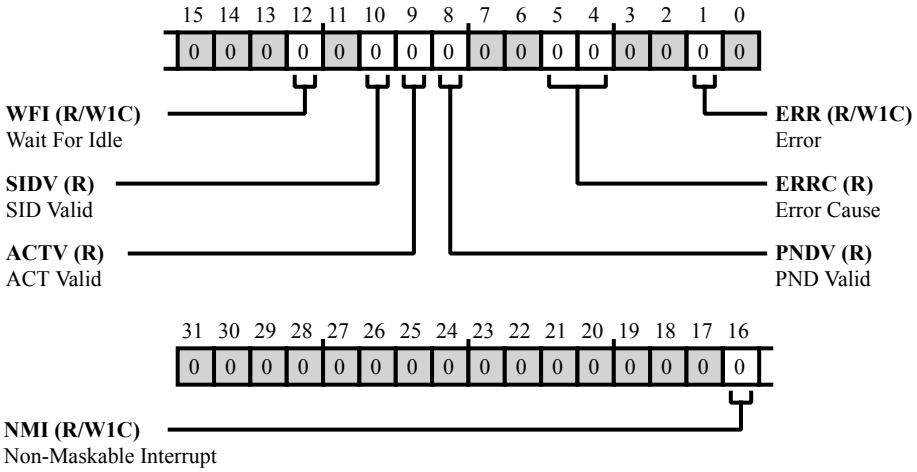


Figure 6-13: SEC_CSTAT[n] Register Diagram

Table 6-14: SEC_CSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W1C)	NMI	Non-Maskable Interrupt. The SEC_CSTAT[n].NMI bit indicates whether an NMI has occurred since the bit was last cleared.	
		0	No NMI Occurred
		1	NMI Occurred
12 (R/W1C)	WFI	Wait For Idle. The SEC_CSTAT[n].WFI bit indicates (if set) that the SCI is temporarily disabled, pending a core idle indication. This bit is set when SEC_CCTL[n].WFI is set.	
		0	Not Waiting
		1	Waiting
10 (R/NW)	SIDV	SID Valid. The SEC_CSTAT[n].SIDV bit indicates (if set) that the current value in the SEC_CSID[n] register is valid. The SCI sets the SEC_CSTAT[n].SIDV bit when the updating the SEC_CSID[n] register with a new value. The SEC_CSTAT[n].SIDV bit is cleared when the SEC_CSID[n] register is written. This status indication may be used to extract all pending interrupts in a single interrupt service routine.	
		0	Invalid
		1	Valid

Table 6-14: SEC_CSTAT[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/NW)	ACTV	ACT Valid. The SEC_CSTAT[n].ACTV bit indicates (if set) that the current value in the SEC_CACT[n] register is valid. The SCI sets the SEC_CSTAT[n].ACTV bit when updating the SEC_CACT[n] registers with a new value. The SEC_CSTAT[n].ACTV bit is cleared when the SEC_CSID[n] register is written.	
		0	Invalid
		1	Valid
8 (R/NW)	PNDV	PND Valid. The SEC_CSTAT[n].PNDV bit indicates (if set) that the current value in the SEC_CPND[n] register is valid. The SCI sets the SEC_CSTAT[n].PNDV bit when updating the SEC_CPND[n] register with a new value. The SEC_CSTAT[n].PNDV bit is cleared when the SEC_CSID[n] register is written.	
		0	Invalid
		1	Valid
5:4 (R/NW)	ERRC	Error Cause. The SEC_CSTAT[n].ERRC bits are updated on assertion of the SEC_CSTAT[n].ERR bit to indicate the SCI error type. SEC_CSTAT[n].ERRC is only updated on the assertion of SEC_CSTAT[n].ERR. Subsequent errors while SEC_CSTAT[n].ERR is asserted do not update SEC_CSTAT[n].ERRC.	
		0	Reserved
		1	Acknowledge Error. SCI has received an acknowledge without a pending, unacknowledged interrupt present.
		2	Reserved
		3	Reserved
1 (R/W1C)	ERR	Error. The SEC_CSTAT[n].ERR bit indicates that an error has occurred in the SCI. When SEC_CSTAT[n].ERR is set, the SCI updates the SEC_CSTAT[n].ERRC field to the value of the corresponding error cause.	
		0	No Error
		1	Error Occurred

Global End Register

The SEC global end register (SEC_END) contains a source ID interrupt service end field (SEC_END.SID). When a core has finished servicing an interrupt, the core writes the SEC_END.SID field in the SEC_END register. This write causes the SEC to clear the SEC_SSTAT[n].ACT bit in the SEC_SSTAT[n] register of the corresponding interrupt.

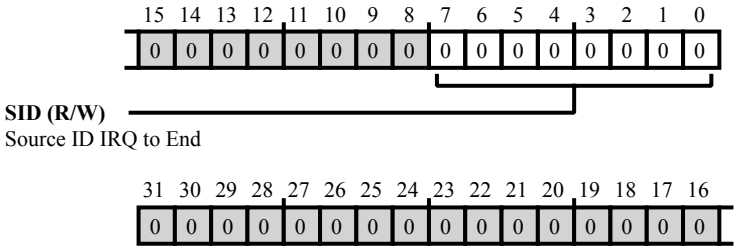


Figure 6-14: SEC_END Register Diagram

Table 6-15: SEC_END Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	SID	Source ID IRQ to End. The SEC_END.SID bit field contains the source ID interrupt service end value.

Fault COP Period Register

The SEC fault COP period register (SEC_FCOPP) contains the width value (count in (SEC) clock cycles) for the high and low phase of the computer operating properly (COP) toggled output on the COP pin. Note that the actual high/low phase value is the SEC_FCOPP.COUNT programmed value plus 1.

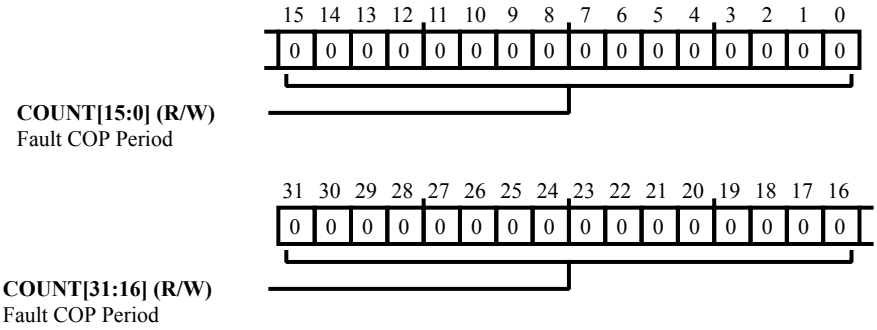


Figure 6-15: SEC_FCOPP Register Diagram

Table 6-16: SEC_FCOPP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	COUNT	Fault COP Period. The SEC_FCOPP.COUNT bit field is the width value for the high and low phase of the computer operating properly (COP) toggled output on the COP pin.

Fault COP Period Current Register

The SEC fault COP period current register (SEC_FCOPP_CUR) contains the active count (in (SEC) clock periods) for the current phase (high or low) of the computer operating properly (COP) toggled output on the COP pin. The SEC loads the SEC_FCOPP_CUR register from the SEC_FCOPP register when the SEC_FCOPP_CUR.COUNT field is cleared and the SEC is in COP mode (SEC_FCTL.CMS bit =1). The SEC decrements the SEC_FCOPP_CUR count each (SEC) clock cycle while SEC_FCTL.CMS is set and the SEC_FSTAT.ACT bit is not set.

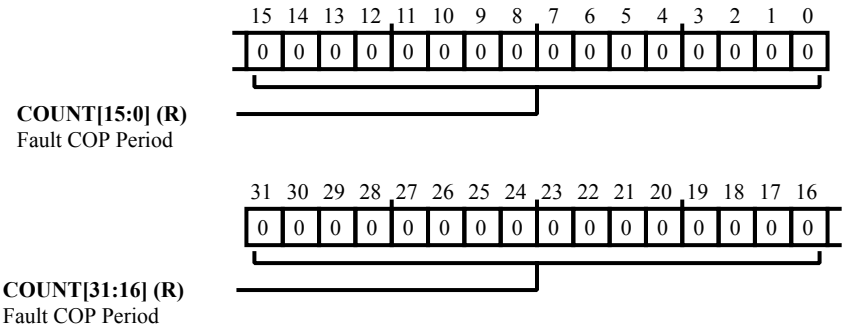


Figure 6-16: SEC_FCOPP_CUR Register Diagram

Table 6-17: SEC_FCOPP_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	COUNT	Fault COP Period. The SEC_FCOPP_CUR.COUNT bit field is the active count for the current phase (high or low) of the computer operating properly (COP) toggled output on the COP pin.

Fault Control Register

The SEC fault control register (SEC_FCTL) contains fault control bits for all SEC channels. This register controls the operation of the System Fault Management Interface (SFI).

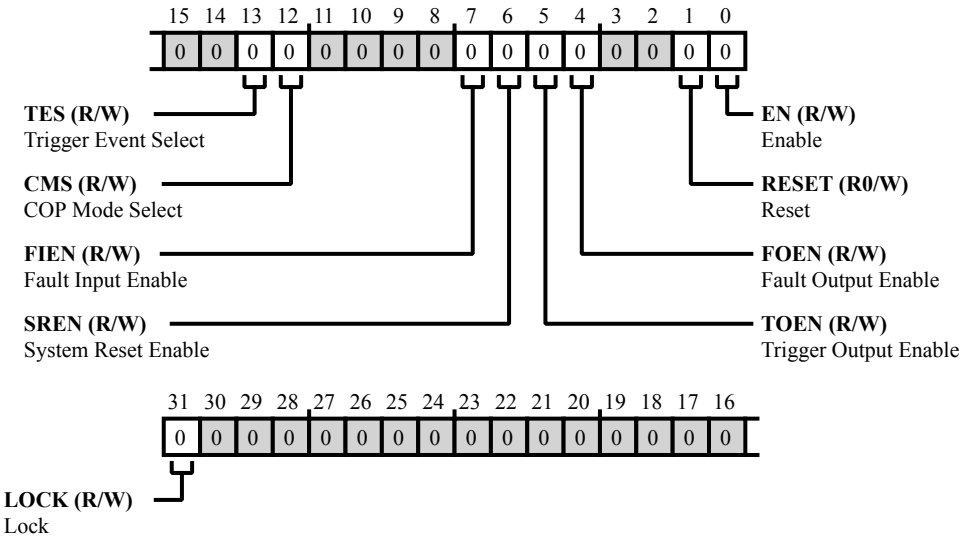


Figure 6-17: SEC_FCTL Register Diagram

Table 6-18: SEC_FCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_FCTL.LOCK bit is enabled, the SEC_FCTL register is read only.	
		0	UnLock
		1	Lock
13 (R/W)	TES	Trigger Event Select. The SEC_FCTL.TES bit selects the event that directs the SEC to assert trigger output. In fault pending mode, the SEC asserts trigger output when a fault is pending. In fault active mode, the SEC asserts trigger output when a fault is active.	
		0	Fault Active Mode
		1	Fault Pending Mode

Table 6-18: SEC_FCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	CMS	COP Mode Select. The SEC_FCTL.CMS selects the SEC mode for handling fault input. In COP mode, the SEC toggles the COP pin to indicate that no fault is active and ceases toggling the pin to indicate that a fault is active. In fault mode, the SEC deasserts the fault pin (=0) and fault_b pin (=1) when no fault is active and asserts the fault pin (=1) and fault_b pin (=0) when a fault is active. Not all processors feature both the fault and fault_b pins. Refer to the product data sheet for details.	
		0	Fault Mode
		1	COP Mode
7 (R/W)	FIEN	Fault Input Enable. The SEC_FCTL.FIEN bit enables the SEC to sample fault input. If SEC_FCTL.FIEN is set (=1), a fault indication from an external device sets the SEC_FSTAT.ACT bit and SEC_FSID.FEXT bit.	
		0	Disable
		1	Enable
6 (R/W)	SREN	System Reset Enable. The SEC_FCTL.SREN bit enables the SEC to issue a system reset request when a fault becomes active.	
		0	Disable
		1	Enable
5 (R/W)	TOEN	Trigger Output Enable. The SEC_FCTL.TOEN bit enables the SEC to produce trigger output when a fault becomes active.	
		0	Disable
		1	Enable
4 (R/W)	FOEN	Fault Output Enable. The SEC_FCTL.FOEN bit enables the SEC to indicate fault status, according to the SEC_FCTL.CMS bit configuration.	
		0	Disable
		1	Enable
1 (R0/W)	RESET	Reset. Setting the SEC_FCTL.RESET bit resets ALL SEC registers to their default values.	
		0	No Action
		1	Reset

Table 6-18: SEC_FCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Enable. The `SEC_FCTL.EN` bit controls the operational state of the SEC. Clearing the `SEC_FCTL.EN` bit halts the execution of the SEC without resetting status registers. Setting the `SEC_FCTL.EN` bit enables the SEC to begin or resume operation with the current configuration and status.	
		0	Disable
		1	Enable

Fault Delay Register

The SEC fault delay register (SEC_FDLY) contains the number (SEC_FDLY.COUNT field) of (SEC) clock periods to delay from fault pending to fault active, when actions are enabled.

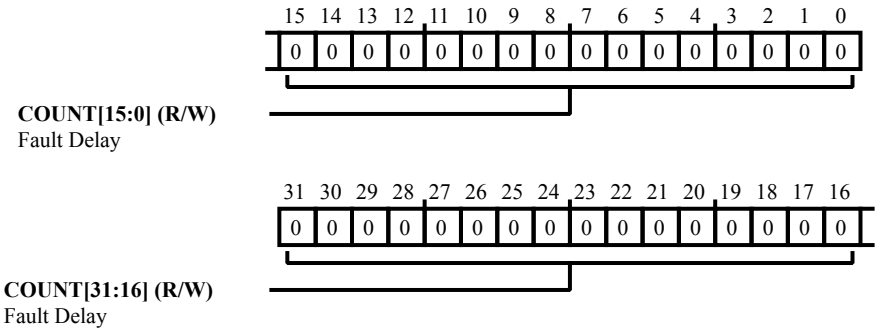


Figure 6-18: SEC_FDLY Register Diagram

Table 6-19: SEC_FDLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	COUNT	Fault Delay. The SEC_FDLY.COUNT bit field is the number of (SEC) clock periods to delay from fault pending to fault active, when actions are enabled.

Fault Delay Current Register

The SEC fault delay current register (SEC_FDLY_CUR) contains the active count (SEC_FDLY_CUR.COUNT field) in (SEC) clock periods for the delay from fault pending to fault active, when actions are enabled. The count is loaded from the SEC_FDLY register when a fault becomes pending (SEC_FSTAT.PND bit is set). The SEC decrements the value in SEC_FDLY_CUR each (SEC) clock cycle while the SEC_FSTAT.PND bit is set.

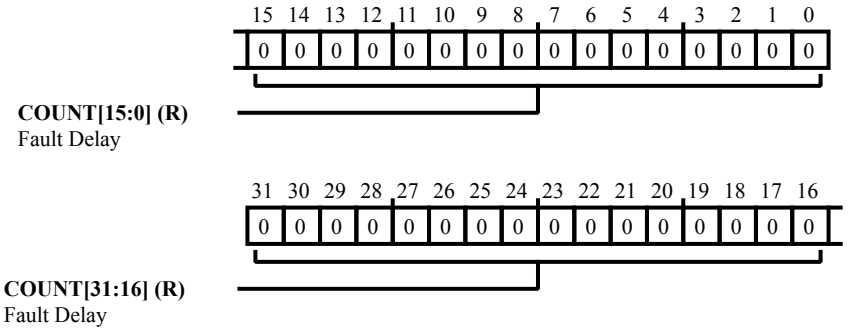


Figure 6-19: SEC_FDLY_CUR Register Diagram

Table 6-20: SEC_FDLY_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	COUNT	Fault Delay. The SEC_FDLY_CUR.COUNT bit field is the active count in (SEC) clock periods for the delay from fault pending to fault active, when actions are enabled.

Fault End Register

The SEC fault end register (SEC_FEND) contains fault source ID and internal/external fields. This register receives fault end indication from a core.

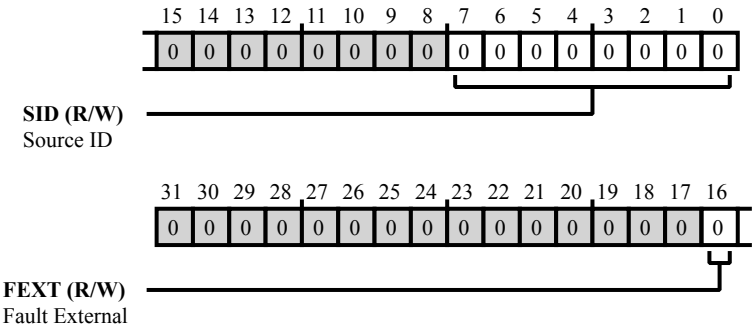


Figure 6-20: SEC_FEND Register Diagram

Table 6-21: SEC_FEND Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	FEXT	Fault External. Setting the SEC_FEND.FEXT bit, when the SEC_FEND.SID field is cleared, clears an active fault from an external source.	
		0	Fault Internal
		1	Fault External
7:0 (R/W)	SID	Source ID. The SEC_FEND.SID identifies a fault to be ended as indicated to the SEC by the core. The core loads the SEC_FEND.SID field value. If the SEC_FEND.SID value matches the SEC_FSID.SID value, the SEC_FSTAT.PND bit and SEC_FSTAT.ACT bit are cleared.	

Fault Source ID Register

The SEC fault source ID register (SEC_FSID) contains a fault source ID and internal/external fields.

NOTE:These bits are not reset by system reset so that a fault that automatically triggers a system reset to avoid a fault may be analyzed after the reset occurs.

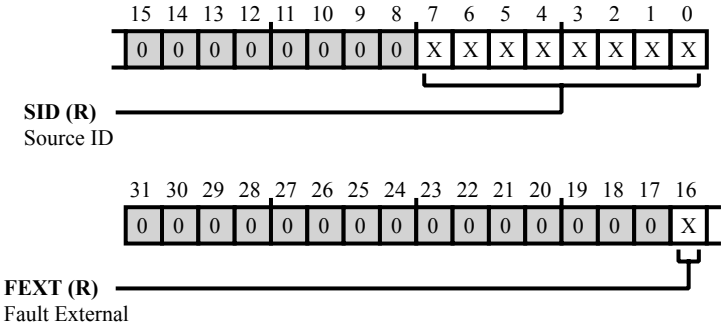


Figure 6-21: SEC_FSID Register Diagram

Table 6-22: SEC_FSID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	FEXT	Fault External. The SEC_FSID.FEXT bit indicates that the last active fault was asserted by an external device. The SEC sets the SEC_FSID.FEXT bit when the SEC_FSTAT.ACT bit is set by the fault input pins. The SEC_FSID.FEXT bit is cleared when the SEC_FSTAT.ACT bit is set by an internal fault or when the external fault is ended. When the SEC_FSID.FEXT bit is set, the SEC_FSID.SID is cleared.	
		0	Fault Internal
		1	Fault External
7:0 (R/NW)	SID	Source ID. The SEC_FSID.SID identifies the fault assertion detected by the SEC fault interface. The SEC loads the SEC_FSID.SID field value when a system fault indication is asserted. The SEC fault interface does not change the SEC_FSID.SID value until the fault is no longer pending or active, as indicated by the SEC_FSTAT.PND bit and SEC_FSTAT.ACT bit being cleared in the SEC_FSTAT register.	

Fault System Reset Delay Register

The SEC fault system reset delay register (SEC_FSRDLY) contains the number (SEC_FSRDLY.COUNT field) of (SEC) clock periods for the delay from a fault becoming active to system reset request assertion, if enabled.

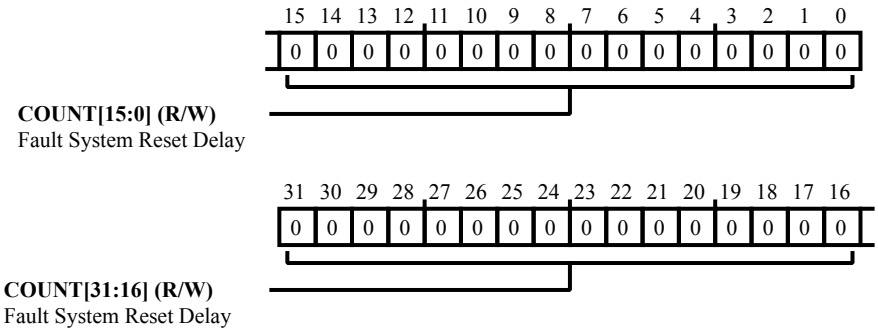


Figure 6-22: SEC_FSRDLY Register Diagram

Table 6-23: SEC_FSRDLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	COUNT	Fault System Reset Delay. The SEC_FSRDLY.COUNT bit field is the number of (SEC) clock periods for the delay from a fault becoming active to system reset request assertion.

Fault System Reset Delay Current Register

The SEC fault system reset delay current register (SEC_FSRDLY_CUR) contains the active count (SEC_FSRDLY_CUR.COUNT field) in (SEC) clock periods for the delay from fault active to system reset assertion, if enabled. The count is loaded from the SEC_FSRDLY register when a fault becomes active (SEC_FSTAT.ACT bit is set). The SEC decrements the value in SEC_FSRDLY_CUR each (SEC) clock cycle while the SEC_FSTAT.ACT bit is set.

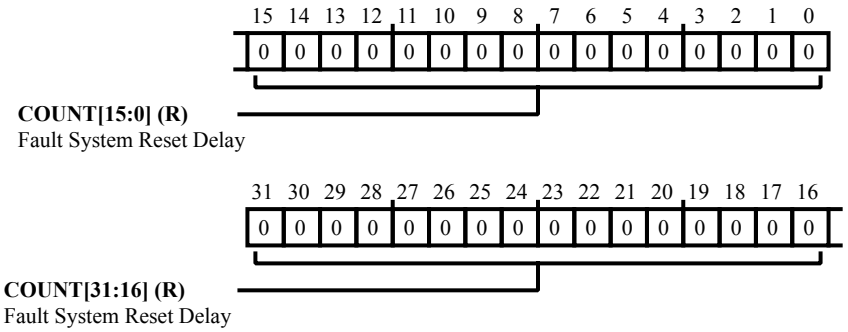


Figure 6-23: SEC_FSRDLY_CUR Register Diagram

Table 6-24: SEC_FSRDLY_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	COUNT	Fault System Reset Delay. The SEC_FSRDLY_CUR.COUNT bit field is the active count in (SEC) clock periods for the delay from fault active to system reset assertion.

Fault Status Register

The SEC fault status register (SEC_FSTAT) indicates the operational status of the SFI.

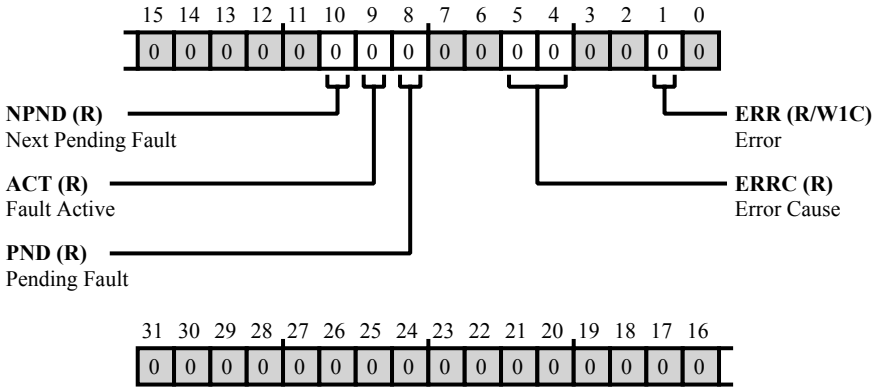


Figure 6-24: SEC_FSTAT Register Diagram

Table 6-25: SEC_FSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/NW)	NPND	Next Pending Fault. The SEC_FSTAT.NPND bit indicates that one or more sources have signaled fault assertion, but the input has not yet triggered the fault pending detection in the SEC fault interface. The SEC sets the SEC_FSTAT.NPND bit when the fault interface detects assertion of any enabled fault source input, while either the SEC_FSTAT.PND or SEC_FSTAT.ACT bits are set. The SEC clears the SEC_FSTAT.NPND bit when there are no fault sources waiting.	
		0	Not Pending
		1	Pending
9 (R/NW)	ACT	Fault Active. The SEC_FSTAT.ACT bit indicates that the SEC has received a fault source input, the current fault delay count (in the SEC_FDLY_CUR register) has expired, and the fault actions are enabled. The SEC also sets the SEC_FSTAT.ACT bit on fault input detection if the SEC_FCTL.FIEN bit is set. The SEC_FSTAT.ACT bit is cleared by writing the ID value of the asserted fault from SEC_FSID register to the SEC_FEND register.	
		0	No Fault
		1	Active Fault

Table 6-25: SEC_FSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	PND	Pending Fault. The SEC_FSTAT.PND bit indicates a fault source has signaled a fault assertion to the SEC, but the SEC has not yet triggered the event actions due to the delay selected with the SEC_FDLY register. The SEC fault interface sets the SEC_FSTAT.PND bit when the SEC_FSID is updated on assertion of a fault source input. The SEC_FSTAT.PND bit is only set when the SEC_FSTAT.ACT bit is cleared. The SEC updates the SEC_FSID register with the SID value when the SEC_FSTAT.PND bit is set. The SEC_FSTAT.PND bit is cleared *either* by the SEC fault interface when the current delay count in the SEC_FDLY_CUR register expires *or* by writing the SEC_FSID.SID field value (which indicates the ID of the asserted fault) to the SEC_FEND register.	
		0	Not Pending
		1	Pending
5:4 (R/NW)	ERRC	Error Cause. When the SEC_FSTAT.ERR bit is asserted, the SEC updates SEC_FSTAT.ERRC field to convey the interrupt source error type. When the error type is source overflow, the status indicates that a source signal assertion occurred or an SEC raise operation was attempted while pending was already set. The source overflow is detected when the source is set for edge only. When the error type is end error, the status indicates that an end was received for a source while the SEC_FSTAT.ACT bit was not set.	
		0	Source Overflow Error
		1	Reserved
		2	End Error
		3	Reserved
1 (R/W1C)	ERR	Error. The SEC_FSTAT.ERR bit indicates an SEC fault interface error. When SEC_FSTAT.ERR is set, the SEC updates the SEC_FSTAT.ERRC field to indicate the corresponding error cause. When multiple errors occur, the SEC_FSTAT register captures the status for the first error and does not capture subsequent errors until the status is cleared.	
		0	No Error
		1	Error Occurred

Global Control Register

The SEC global control register (SEC_GCTL) provides register locking, reset, and enable for the SEC module.

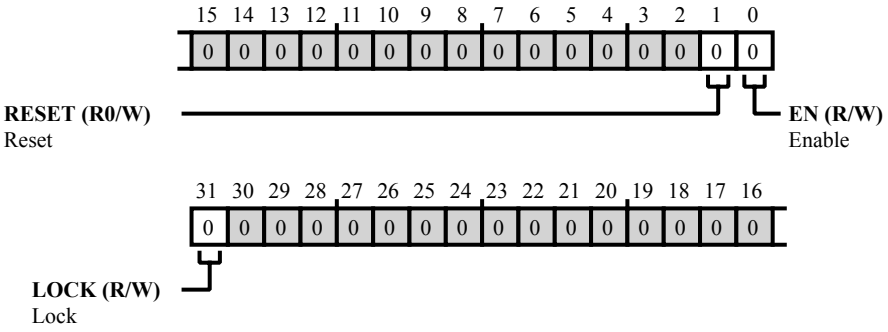


Figure 6-25: SEC_GCTL Register Diagram

Table 6-26: SEC_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_GCTL.LOCK bit is enabled, the SEC_GCTL register is read only.	
		0	Unlock
		1	Lock
1 (R0/W)	RESET	Reset. The SEC_GCTL.RESET bit is write-1-action and triggers a soft reset to all SEC registers.	
		0	No Action
		1	Reset
0 (R/W)	EN	Enable. The SEC_GCTL.EN bit is read/write and must be set for the SEC to begin/resume SEC operation with the current configuration and status. Clearing the SEC_GCTL.EN bit halts the execution of the SFI and all SCIs. All SSIs remain active, along with all error detection, without resetting status registers.	
		0	Disable
		1	Enable

Global Status Register

The SEC global status register (SEC_GSTAT) contains global status bits for the SEC.

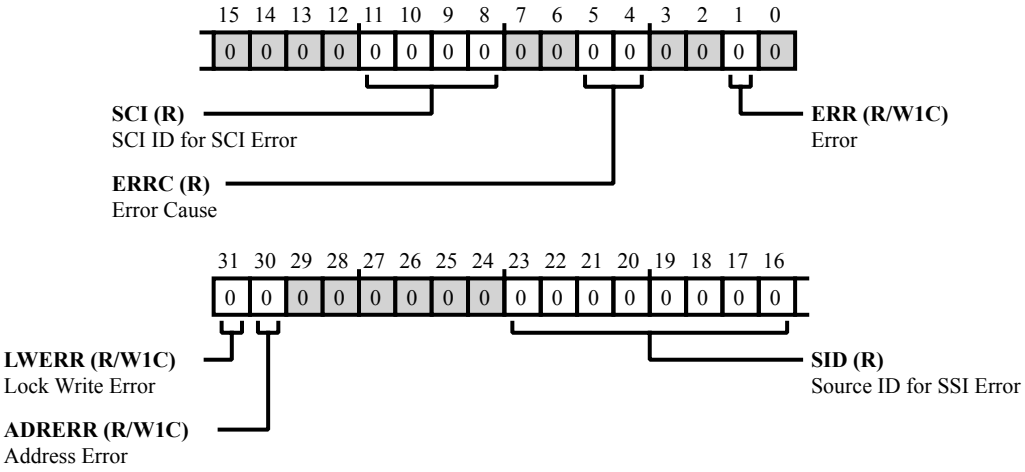


Figure 6-26: SEC_GSTAT Register Diagram

Table 6-27: SEC_GSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Lock Write Error. The SEC_GSTAT.LWERR bit indicates (when set) there was an attempted write to an SEC register while the SEC_GCTL.LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK bit =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
30 (R/W1C)	ADRERR	Address Error. The SEC_GSTAT.ADRERR bit indicates that the SEC generated and address error. This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
23:16 (R/NW)	SID	Source ID for SSI Error. The SEC_GSTAT.SID bits indicate the source ID that generated the last SSI error conveyed in the SEC_GSTAT.ERRC field.	
11:8 (R/NW)	SCI	SCI ID for SCI Error. The SEC_GSTAT.SCI bits indicate the number for the specific SCI that generated the last SCI error conveyed in the SEC_GSTAT.ERRC field.	

Table 6-27: SEC_GSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/NW)	ERRC	Error Cause. When the SEC updates the `SEC_GSTAT.ERR` bit, the SEC updates the `SEC_GSTAT.ERRC` bits to indicate the error type. Note that for SCI errors, the error status represents an OR of all the errors from each SCI. Note that for SSI errors, the error status indicates an error is active for any SSI input. This error is an OR of all the interrupt source errors.	
		0	SFI Error
		1	SCI Error
		2	SSI Error
		3	Reserved
1 (R/W1C)	ERR	Error. The `SEC_GSTAT.ERR` bit indicates an error has occurred in the SEC. When the SEC asserts this bit (=1), the SEC updates the `SEC_GSTAT.ERRC` field to indicate the corresponding error cause. Even if multiple errors occur, only the first error is captured on assertion of this bit. This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred

Global Raise Register

The SEC global raise register (SEC_RAISE) contains a source ID event set-to-pending field (SEC_RAISE.SID). When a source ID value is written to this field, the SEC raises the source's event status to pending.

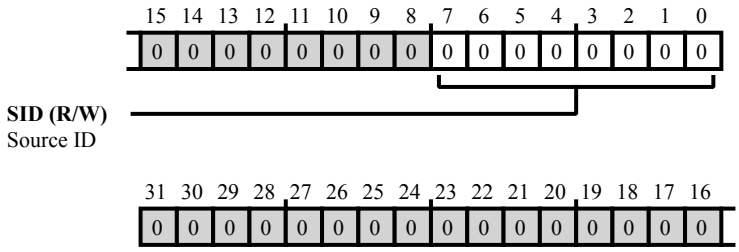


Figure 6-27: SEC_RAISE Register Diagram

Table 6-28: SEC_RAISE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	SID	Source ID. The SEC_RAISE.SID bit field is the source ID of event that is set to pending status.

Source Control Register n

The SEC source control register (SEC_SCTL[n]) contains control bits to configure the SEC event sources. This register controls the configuration of the corresponding SEC event source.

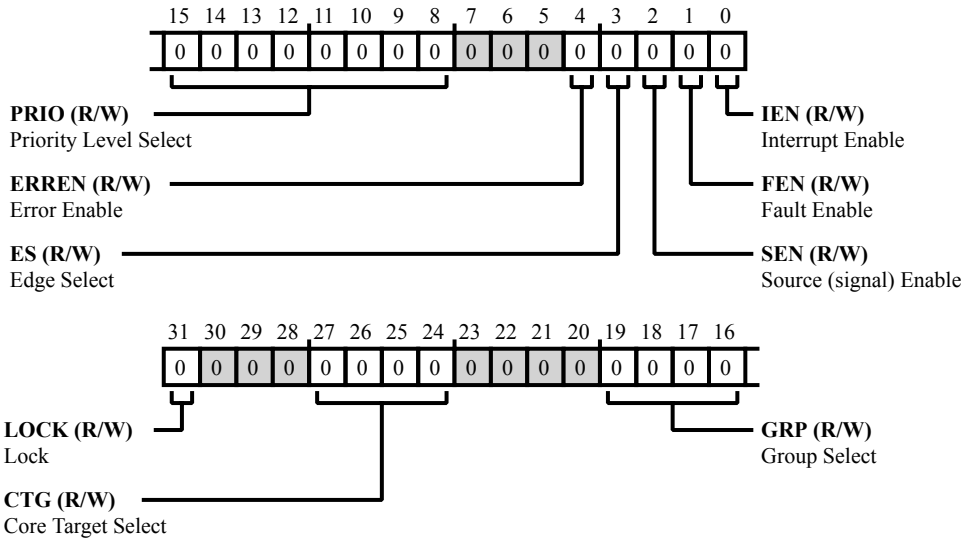


Figure 6-28: SEC_SCTL[n] Register Diagram

Table 6-29: SEC_SCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the SEC_SCTL[n].LOCK bit is enabled, the SEC_SCTL[n] register is read only.	
		0	Unlock
		1	Lock
27:24 (R/W)	CTG	Core Target Select. The SEC_SCTL[n].CTG bits selects the specific SEC core interface to which the interrupt is mapped. Each system interrupt is mapped uniquely to one specific SEC core interface and (as a result) to a specific core.	
		1	CORE1 SHARC0
		2	CORE2 SHARC1

Table 6-29: SEC_SCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	GRP	Group Select. The SEC_SCTL[n].GRP bits each select a specific group for the interrupt. Each system interrupt can be assigned to any combination of groups supported by the SEC_SCTL[n].GRP field. For example, consider the situation where SEC_SCTL[n].GRP0 represents interrupt group 0, SEC_SCTL[n].GRP1 represents interrupt group 1, and so on. One group might be used for all enabled interrupts (for example, group 0) and an additional group might be used for all wakeup interrupts (for example, group 1). This approach supports a model of all interrupts and just the wakeup subset. Before going to idle or sleep, all non-wakeup interrupts can be masked off to allow only wakeup interrupts to be enabled for service. Selecting no group (all SEC_SCTL[n].GRP bits = 0) places the interrupt source in the category of "ungrouped".	
15:8 (R/W)	PRIO	Priority Level Select. The SEC_SCTL[n].PRIO bits sets the relative priority for an interrupt request. A pending interrupt request forwards its SEC_SCTL[n].PRIO value to the SEC core interface.	
4 (R/W)	ERREN	Error Enable. The SEC_SCTL[n].ERREN bit permits the SEC_SSTAT[n].ERR status bit to be set on error detection. If SEC_SCTL[n].ERREN is cleared, no errors are detected.	
		0	Disable
		1	Enable
3 (R/W)	ES	Edge Select. The SEC_SCTL[n].ES bit selects the operational and sensitivity mode of the SEC source interface input.	
		0	Level Sensitive
		1	Edge Sensitive
2 (R/W)	SEN	Source (signal) Enable. The SEC_SCTL[n].SEN bit controls whether the system event source input signal may affect the pending status of the source. Clearing the SEC_SCTL[n].SEN bit disables the source input signal from affecting the pending status. Setting SEC_SCTL[n].SEN enables the source input signal to affect the pending status.	
		0	Disable
		1	Enable

Table 6-29: SEC_SCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	FEN	Fault Enable. The `SEC_SCTL[n].FEN` bit controls whether the SEC may forward an interrupt request to the SEC fault interface as a fault source. This bit does not affect the ability of an interrupt source to set an interrupt as pending. The `SEC_SCTL[n].FEN` bit only affects whether the pending request may be forwarded to the SEC fault interface.	
		0	Disable
		1	Enable
0 (R/W)	IEN	Interrupt Enable. The `SEC_SCTL[n].IEN` bit controls whether the SEC may forward an interrupt request to a core for servicing. This bit does not affect the ability of an interrupt source to set an interrupt as pending.	
		0	Disable
		1	Enable

Source Status Register n

The SEC event source status register (SEC_SSTAT[n]) contains bits indicating the status of the corresponding event source n. An event source may be: pending, active, active and pending, or neither pending nor active.

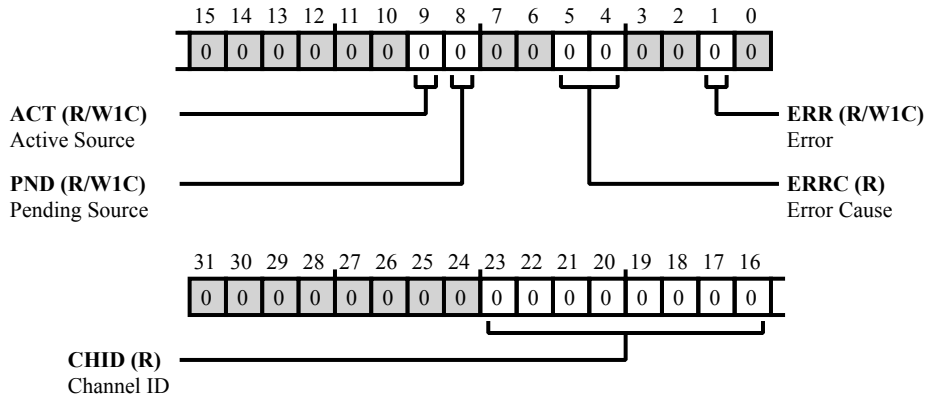


Figure 6-29: SEC_SSTAT[n] Register Diagram

Table 6-30: SEC_SSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:16 (R/NW)	CHID	Channel ID. The SEC_SSTAT[n].CHID bits indicate the ID of the specific source (from a set of sources sharing one SEC source interface input) that asserted the SEC source interface input. An SEC source interface input may support multiple system sources, in which case the assertion must be qualified by an identifier to determine the channel that generated the assertion. The SEC_SSTAT[n].CHID field provides this value in the form of a numeric reference that is mapped to a specific interrupt source. The prioritization for simultaneously asserted sources is according to ID, with 0 being the highest priority. The SEC_SSTAT[n].CHID is captured when the SEC source interface input is acknowledged.	
9 (R/W1C)	ACT	Active Source. The SEC_SSTAT[n].ACT bit indicates the source has been accepted by a core for servicing, but the service is not yet complete. An SEC_SSTAT[n].ACT bit is set by the SEC when the specific system interrupt is acknowledged by the core through the SEC core interface. An SEC_SSTAT[n].ACT bit is cleared by the SEC when the core provides interrupt service end indication for the specific system interrupt through the SEC core interface.	
		0	Not Active
		1	Active

Table 6-30: SEC_SSTAT[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	PND	Pending Source. The SEC_SSTAT[n].PND bit indicates the source has signaled an event request, but the event request has not been (or is not currently being) serviced. A SEC_SSTAT[n].PND bit is set by the SEC on detection of an assertion of the corresponding system source input. A SEC_SSTAT[n].PND bit is cleared by the SEC when the specific system event is acknowledged by the core through the SEC core interface or by a W1C operation.	
		0	Not Pending
		1	Pending
5:4 (R/NW)	ERRC	Error Cause. When the SEC_SSTAT[n].ERR bit is asserted, the SEC updates SEC_SSTAT[n].ERRC field to convey the interrupt source error type. When the error type is source overflow, the status indicates that a source signal assertion occurred or an SEC raise operation was attempted while pending was already set. The source overflow is detected when the source is set for edge only. When the error type is end error, the status indicates that an end was received for a source while the SEC_SSTAT[n].ACT bit was not set.	
		0	Source Overflow Error
		1	Reserved
		2	End Error
		3	Reserved
1 (R/W1C)	ERR	Error. The SEC_SSTAT[n].ERR bit indicates an error for a specific system interrupt source. When the SEC_SSTAT[n].ERR bit is set, the SEC updates the SEC_SSTAT[n].ERRC field to the value of the corresponding error cause. Even if multiple errors occur, only the first error is captured on assertion of the SEC_SSTAT[n].ERR bit.	
		0	No Error
		1	Error Occurred

GIC Overview

The generic interrupt controller (GIC) provides an interface to the uniprocessor Cortex A55 core in the processor and collects up to 354 interrupt requests from all processor system sources. In addition, the GIC also supports sixteen software generated interrupts and nine private peripheral interrupts that are internal to the Cortex A55 core and not connected to the SECs. All GIC interrupts are also connected to the SEC in the same order.

Each interrupt can be configured as a normal or a secure interrupt. Software force registers and software priority masking are also supported. The “Register Descriptions” section in this chapter provides brief descriptions of these Arm-based registers. For complete information refer to the *Arm® Generic Interrupt Controller Architecture version 3.0 and 4.0 Architecture Specification*.

GIC Functional Description

The GIC interrupt controller consists of three interfaces;

- Distributor interface
- Redistributor interface
- CPU interface

The distributor and the redistributor interfaces are used to configure interrupts; the CPU interface is used to handle interrupts.

GIC Distributer Interface

The distributor registers are memory-mapped. They are used to configures SPIs. The distributor block provides a programming interface to perform the following tasks.

- Interrupt prioritization and distribution of SPIs
- Enable and disable SPIs
- Set the priority level of each SPI
- Set each SPI to be level-sensitive or edge-triggered
- Assign each SPI to an interrupt group
- Control the active and pending state of SPIs.

GIC Redistributer Interface

There is one redistributor per connected core. The redistributors provide a programming interface to perform the following tasks.

- Enable and disable SGIs and PPIs
- Set the priority level of SGIs and PPIs
- Set each PPI to be level-sensitive or edge-triggered
- Assign each SGI and PPI to an interrupt group
- Control the state of SGIs and PPIs.

GIC CPU Interface

Each core contains a CPU interface that includes system registers that are used during interrupt handling. The CPU interfaces provide a programming interface to perform the following tasks.

- Provide general control and configuration to enable interrupt handling
- Acknowledge an interrupt
- Perform a priority drop and deactivation of interrupts
- Set an interrupt priority mask for the processor
- Define the preemption policy for the processor
- Determine the highest priority pending interrupt for the processor

In Arm CoreLink GICv3, the CPU interface registers are accessed as system registers: ICC_*_ELn, where “n” specifies the Exception level: EL1-EL3.

Refer to the GIC Register Descriptions in this chapter for details on the distributor and the redistributor interface registers. Refer to the *Arm® Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and 4.0* for details on the CPU interface registers.

GIC Block Diagram

The ***GIC Block Diagram*** shows the event management architecture.

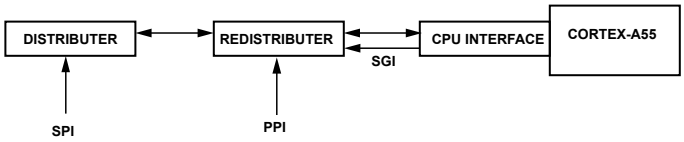


Figure 6-30: GIC Block Diagram

GIC Performance Monitoring Unit (GIC PMU)

The GIC contains a performance monitoring unit (PMU) for counting key GIC events from the distributor. Redistributor events are not tracked by the PMU. The delivery of PPI and SGI interrupts are counted by recording calls to the core interrupt service routine.

The PMU has five counters with snapshot capability and overflow interrupt. For all the GIC events, PMU count configuration and register information, please refer to the *Arm® CoreLink™ GIC 600 Generic Interrupt Controller Technical Reference Manual*.

Overflow interrupt

The overflow interrupt is enabled on a per counter basis by enabling the relevant bit of the register. Similarly, the overflow interrupt enable is disabled by corresponding writes to the register. When enabled, the interrupt activates when there is a write to the register for any counter or overflow on any enabled counter.

Snapshot

Each PMU counter, , has a corresponding snapshot register. On a snapshot event, all five counters are copied to their backup registers so that all consistent data is copied out over a longer period.

The snapshot events are as follows:

- Write of 1 to the GICP_ bit
- An overflow of an enabled counter when the bit is set

ADSP-SC59x GIC Interrupt List

Table 6-31: ADSP-SC59x GIC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
339	GIC0_SPI_SOFT0_INT	GIC0 GIC Software Interrupt 0		
340	GIC0_SPI_SOFT1_INT	GIC0 GIC Software Interrupt 1		
341	GIC0_SPI_SOFT2_INT	GIC0 GIC Software Interrupt 2		
342	GIC0_SPI_SOFT3_INT	GIC0 GIC Software Interrupt 3		
343	GIC0_SPI_SOFT4_INT	GIC0 GIC Software Interrupt 4		
344	GIC0_SPI_SOFT5_INT	GIC0 GIC Software Interrupt 5		
345	GIC0_SPI_SOFT6_INT	GIC0 GIC Software Interrupt 6		
346	GIC0_SPI_SOFT7_INT	GIC0 GIC Software Interrupt 7		
347	GIC0_SPI_SOFT8_INT	GIC0 GIC Software Interrupt 8		
348	GIC0_SPI_SOFT9_INT	GIC0 GIC Software Interrupt 9		
349	GIC0_SPI_SOFT10_INT	GIC0 GIC Software Interrupt 10		
350	GIC0_SPI_SOFT11_INT	GIC0 GIC Software Interrupt 11		
351	GIC0_SPI_SOFT12_INT	GIC0 GIC Software Interrupt 12		
352	GIC0_SPI_SOFT13_INT	GIC0 GIC Software Interrupt 13		
353	GIC0_SPI_SOFT14_INT	GIC0 GIC Software Interrupt 14		
354	GIC0_SPI_SOFT15_INT	GIC0 GIC Software Interrupt 15		

7 Trigger Routing Unit (TRU)

The TRU provides system-level sequence control without core intervention. The TRU maps trigger generators to trigger receivers. Receiver endpoints can be configured to respond to triggers in various ways. Multiple TRUs may be provided in a multiprocessor system to create a trigger network. Common applications enabled by the TRU include:

- Automatically triggering the start of a DMA sequence after a sequence from another DMA channel completes
- Software triggering
- Synchronization of concurrent activities

TRU Features

The TRU supports the following features:

- Automatically triggering the start of a DMA sequence after a sequence from another DMA channel completes. Once a DMA channel completes data transfer, it can act as a trigger generator and signal an internal trigger pulse to the programmed trigger receiver which can also be another DMA channel. The receiver trigger connected to the DMA channel kicks off the DMA transfer automatically. None of this requires core intervention once the initialization is done.
- Software triggers. The best use of triggers is to minimize core intervention. It is also possible to initiate a trigger pulse to a trigger receiver, in the software.
- Synchronization of concurrent activities. A single trigger generator can initiate a trigger pulse to multiple trigger receiver so that several system level activities can be synchronized on an internally or externally generated event.
- Configuration protection through register-level lock bits and global lock indication

TRU Functional Description

The following sections provide a description of the TRU.

ADSP-SC59x TRU Register List

The Trigger Routing Unit (TRU) provides simple sequence control of distributed modules without the penalties associated with core intervention (for example, interrupt overhead). The TRU receives trigger inputs from all trigger generator inputs (MTI) and the TRU trigger receiver register (TRU_MTR). Based on these inputs, the TRU logic generates trigger outputs that initiate receiver operations in the processor core and peripherals. A set of registers governs TRU operations. For more information on TRU functionality, see the TRU register descriptions.

Table 7-1: ADSP-SC59x TRU Register List

Name	Description
TRU_ERRADDR	Error Address Register
TRU_GCTL	Global Control Register
TRU_MTR	Generator Trigger Register
TRU_SSR[n]	Receiver Select Register
TRU_STAT	Status Information Register

ADSP-SC59x TRU Interrupt List

Table 7-2: ADSP-SC59x TRU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
242	TRU0_SLV4	TRU0 Interrupt 4 - Core 1	Edge	
243	TRU0_SLV5	TRU0 Interrupt 5 - Core 1	Edge	
244	TRU0_SLV6	TRU0 Interrupt 6 - Core 1	Edge	
245	TRU0_SLV7	TRU0 Interrupt 7 - Core 1	Edge	
246	TRU0_SLV8	TRU0 Interrupt 8 - Core 2	Edge	
247	TRU0_SLV9	TRU0 Interrupt 9 - Core 2	Edge	
248	TRU0_SLV10	TRU0 Interrupt 10 - Core 2	Edge	
249	TRU0_SLV11	TRU0 Interrupt 11 - Core 2	Edge	
334	TRU0_SLV0	TRU0 Interrupt 0 - Core 0	Edge	
335	TRU0_SLV1	TRU0 Interrupt 1 - Core 0	Edge	
336	TRU0_SLV2	TRU0 Interrupt 2 - Core 0	Edge	
337	TRU0_SLV3	TRU0 Interrupt 3 - Core 0	Edge	

ADSP-SC59x Trigger List

Table 7-3: ADSP-SC59x Trigger List Generators

Trigger ID	Name	Description	Sensitivity
0		Reserved	
1	CGU0_EVT	CGU0 Event	Edge
2	CGU1_EVT	CGU1 Event	Edge
3	CNT0_STAT	CNT0 Status	Level
4	CNT0_UD	CNT0 CNT0 Count Up and Direction	Level
5	CNT0_DG	CNT0 CNT0 Count Down and Gate	Level
6	CNT0_TO	CNT0 CNT0 Output to Timer Block	Level
7	C1_SID_ACK	Core 1 System Interface Disable Acknowledge	
8	C0_EVENTIACK	Core 0 Event Input Request Acknowledge	
9	C0_EVENTOREQ	Core 0 Event Output Request	
10	C2_SID_ACK	C2 System Interface Disable Acknowledge	
11	MDMA0_SRC	Enhanced BW DMA Channel 0 Source (CRC IN)	
12	MDMA0_DST	Enhanced BW DMA Channel 0 Source (CRC OUT)	
13	MDMA1_SRC	Enhanced BW DMA Channel 0 Source (CRC IN)	
14	MDMA1_DST	Enhanced BW DMA Channel 0 Source (CRC OUT)	
15	MDMA4_SRC	Standard BW DMA Channel 0 Source (CRC IN)	
16	MDMA4_DST	Standard BW DMA Channel 0 Source (CRC OUT)	
17	MDMA5_SRC	Standard BW DMA Channel 0 Source (CRC IN)	
18	MDMA5_DST	Standard BW DMA Channel 0 Source (CRC OUT)	
19	SOC600CTI0_MST0	SOC600CTI0 SYSCTI (CTI3) System Halt Slave 0	Edge
20	SOC600CTI0_MST1	SOC600CTI0 SYSCTI (CTI3) System Halt Slave 1	Edge
21	SOC600CTI0_MST2	SOC600CTI0 SYSCTI (CTI3) System Halt Slave 2	Edge

Table 7-3: ADSP-SC59x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
22	SOC600CTI0_MST3	SOC600CTI0 SYSCTI (CTI3) System Halt Slave 3	Edge
23	SOC600CTI0_MST4	SOC600CTI0 SYSCTI (CTI3) System Halt Slave 4	Edge
24	SOC600CTI0_MST5	SOC600CTI0 SYSCTI (CTI3) System Halt Slave 5	Edge
25	SOC600CTI0_MST6	SOC600CTI0 SYSCTI (CTI3) System Halt Slave 6	Edge
26	SOC600CTI0_MST7	SOC600CTI0 SYSCTI (CTI3) System Halt Slave 7	Edge
27	EMDMA0_DONE	EMDMA0 DMA Done	Edge
28	EMDMA1_DONE	EMDMA1 DMA Done	Edge
29	EPPI0_CH0_DMA	EPPI0 Channel 0 DMA	Edge
30	EPPI0_CH1_DMA	EPPI0 Channel 1 DMA	Edge
31	C1_FIR0_DMA	FIR0 Core 1 DMA	Edge
32	C2_FIR0_DMA	FIR1 Core 2 DMA	Edge
33	HADC0_EOC	HADC0 HADC0 End of Conversion	Edge
34	C1_IIR0_DMA	IIR0 Core 1 DMA	Edge
35	C1_IIR1_DMA	IIR1 Core 1 DMA	Edge
36	C1_IIR2_DMA	IIR2 Core 1 DMA	Edge
37	C1_IIR3_DMA	IIR3 Core 1 DMA	Edge
38	C2_IIR0_DMA	IIR4 Core 2 DMA	Edge
39	C2_IIR1_DMA	IIR5 Core 2 DMA	Edge
40	C2_IIR2_DMA	IIR6 Core 2 DMA	Edge
41	C2_IIR3_DMA	IIR7 Core 2 DMA	Edge
42	L2CTL0_EVT	L2CTL0 L2 Memory Event	Level
43	LP0_DMA	LP0 DMA Channel	
44	LP1_DMA	LP1 DMA Channel	
45	MDMA2_SRC	Enh BW DMA Channel 0	
46	MDMA2_DST	Enh BW DMA Channel 1	
47	MDMA3_SRC	Max BW DMA Channel 0	
48	MDMA3_DST	Max BW DMA Channel 1	
49	MDMA6_SRC	Enh BW DMA Channel 0	

Table 7-3: ADSP-SC59x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
50	MDMA6_DST	Enh BW DMA Channel 1	
51	MDMA7_SRC	Max BW DMA Channel 0	
52	MDMA7_DST	Max BW DMA Channel 1	
53	MEC0_EEIRQ0	MEC0 ECC Error Interrupt Request	Level
54	MEC0_MEIRQ0	MEC0 Parity Error Interrupt Request	Level
55	MEC0_PEIRQ1	MEC0 Parity Error Interrupt Request	Level
56	MEC0_PEIRQ2	MEC0 Parity Error Interrupt Request	Level
57	MEC0_PEIRQ3	MEC0 Parity Error Interrupt Request	Level
58	MEC1_EEIRQ0	MEC1 ECC Error Interrupt Request	Level
59	MEC1_MEIRQ0	MEC1 Parity Error Interrupt Request	Level
60	MEC1_PEIRQ1	MEC1 Parity Error Interrupt Request	Level
61	MEC1_PEIRQ2	MEC1 Parity Error Interrupt Request	Level
62	MEC1_PEIRQ3	MEC1 Parity Error Interrupt Request	Level
63	MEC2_EEIRQ0	MEC2 ECC Error Interrupt Request	Level
64	MEC2_MEIRQ0	MEC2 Parity Error Interrupt Request	Level
65	MEC2_PEIRQ1	MEC2 Parity Error Interrupt Request	Level
66	MEC2_PEIRQ2	MEC2 Parity Error Interrupt Request	Level
67	MEC2_PEIRQ3	MEC2 Parity Error Interrupt Request	Level
68	PINT0_BLOCK	PINT0 Pin Interrupt Block	Level
69	PINT1_BLOCK	PINT1 Pin Interrupt Block	Level
70	PINT2_BLOCK	PINT2 Pin Interrupt Block	Level
71	PINT3_BLOCK	PINT3 Pin Interrupt Block	Level
72	PINT4_BLOCK	PINT4 Pin Interrupt Block	Level
73	PINT5_BLOCK	PINT5 Pin Interrupt Block	Level
74	PINT6_BLOCK	PINT6 Pin Interrupt Block	Level
75	PINT7_BLOCK	PINT7 Pin Interrupt Block	Level
76	SEC0_FAULT	SEC0 Fault	Edge
77	SPI0_TXDMA	SPI0 TX DMA Channel	Edge
78	SPI0_RXDMA	SPI0 RX DMA Channel	Edge
79	SPI1_TXDMA	SPI1 TX DMA Channel	Edge
80	SPI1_RXDMA	SPI1 RX DMA Channel	Edge

Table 7-3: ADSP-SC59x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
81	SPI3_TXDMA	SPI3 TX DMA Channel	Edge
82	SPI3_RXDMA	SPI3 RX DMA Channel	Edge
83	SPI2_TXDMA	SPI2 TX DMA Channel	Edge
84	SPI2_RXDMA	SPI2 RX DMA Channel	Edge
85	SPORT0_A_DMA	SPORT0 Channel A DMA	Edge
86	SPORT0_B_DMA	SPORT0 Channel B DMA	Edge
87	SPORT1_A_DMA	SPORT1 Channel A DMA	Edge
88	SPORT1_B_DMA	SPORT1 Channel B DMA	Edge
89	SPORT2_A_DMA	SPORT2 Channel A DMA	Edge
90	SPORT2_B_DMA	SPORT2 Channel B DMA	Edge
91	SPORT3_A_DMA	SPORT3 Channel A DMA	Edge
92	SPORT3_B_DMA	SPORT3 Channel B DMA	Edge
93	SPORT4_A_DMA	SPORT4 Channel A DMA	Edge
94	SPORT4_B_DMA	SPORT4 Channel B DMA	Edge
95	SPORT5_A_DMA	SPORT5 Channel A DMA	Edge
96	SPORT5_B_DMA	SPORT5 Channel B DMA	Edge
97	SPORT6_A_DMA	SPORT6 Channel A DMA	Edge
98	SPORT6_B_DMA	SPORT6 Channel B DMA	Edge
99	SPORT7_A_DMA	SPORT7 Channel A DMA	Edge
100	SPORT7_B_DMA	SPORT7 Channel B DMA	Edge
101	DAI0_GBL_SPORT_TRG_O0	DAI0 SPORT GROUP0 Trigger Output	None
102	DAI0_GBL_SPORT_TRG_O1	DAI0 SPORT GROUP1 Trigger Output	None
103	DAI1_GBL_SPORT_TRG_O0	DAI1 SPORT GROUP2 Trigger Output	None
104	DAI1_GBL_SPORT_TRG_O1	DAI1 SPORT Group3 Trigger Output	None
106	SWU1_EVT	SWU1 Event	None
107	SWU2_EVT	SWU2 Event	None
108	SWU7_EVT	SWU7 Event	None
109	SWU8_EVT	SWU8 Event	None

Table 7-3: ADSP-SC59x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
110	SWU9_EVT	SWU9 Event	None
111	SWU10_EVT	SWU10 Event	None
112	SWU11_EVT	SWU11 Event	None
113	SWU12_EVT	SWU12 Event	None
114	SWU13_EVT	SWU13 Event	None
115	SWU3_EVT	SWU3 Event	None
116	SWU4_EVT	SWU4 Event	None
117	SWU5_EVT	SWU5 Event	None
119	SWU1_DBG	SWU1 Debug	Edge
120	SWU2_DBG	SWU2 Debug	Edge
121	SWU7_DBG	SWU7 Debug	Edge
122	SWU8_DBG	SWU8 Debug	Edge
123	SWU9_DBG	SWU9 Debug	Edge
124	SWU10_DBG	SWU10 Debug	Edge
125	SWU11_DBG	SWU11 Debug	Edge
126	SWU12_DBG	SWU12 Debug	Edge
127	SWU13_DBG	SWU13 Debug	Edge
128	SWU3_DBG	SWU3 Debug	Edge
129	SWU4_DBG	SWU4 Debug	Edge
130	SWU5_DBG	SWU5 Debug	Edge
131	SOFT0_MST	Software-driven Trigger 0	
132	SOFT1_MST	Software-driven Trigger 1	
133	SOFT2_MST	Software-driven Trigger 2	
134	SOFT3_MST	Software-driven Trigger 3	
135	SOFT4_MST	Software-driven Trigger 4	
136	SOFT5_MST	Software-driven Trigger 5	
137	TIMER0_TMR00_MST	TIMER0 Timer 0	Edge
138	TIMER0_TMR01_MST	TIMER0 Timer 1	Edge
139	TIMER0_TMR02_MST	TIMER0 Timer 2	Edge
140	TIMER0_TMR03_MST	TIMER0 Timer 3	Edge
141	TIMER0_TMR04_MST	TIMER0 Timer 4	Edge

Table 7-3: ADSP-SC59x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
142	TIMER0_TMR05_MST	TIMER0 Timer 5	Edge
143	TIMER0_TMR06_MST	TIMER0 Timer 6	Edge
144	TIMER0_TMR07_MST	TIMER0 Timer 7	Edge
145	TIMER0_TMR08_MST	TIMER0 Timer 8	Edge
146	TIMER0_TMR09_MST	TIMER0 Timer 9	Edge
147	TIMER0_TMR10_MST	TIMER0 Timer 10	Edge
148	TIMER0_TMR11_MST	TIMER0 Timer 11	Edge
149	TIMER0_TMR12_MST	TIMER0 Timer 12	Edge
150	TIMER0_TMR13_MST	TIMER0 Timer 13	Edge
151	TIMER0_TMR14_MST	TIMER0 Timer 14	Edge
152	TIMER0_TMR15_MST	TIMER0 Timer 15	Edge
153	TMU0_FAULT	TMU0 TM0 Fault Event	
154	TMU0_ALERT	TMU0 TM0 Alert Event	
155	UART0_TXDMA	UART0 Transmit DMA	Edge
156	UART0_RXDMA	UART0 Receive DMA	Edge
157	UART1_TXDMA	UART1 Transmit DMA	Edge
158	UART1_RXDMA	UART1 Receive DMA	Edge
159	UART2_TXDMA	UART2 Transmit DMA	Edge
160	UART2_RXDMA	UART2 Receive DMA	Edge
161	UART3_TXDMA	UART3 Transmit DMA	Edge
162	UART3_RXDMA	UART3 Receive DMA	Edge
163	USBC0_SOF_TGL	USBC0 USB start of frame generation toggle	None
164	USBC0_SOF_SENT_TGL	USBC0 USB start of frame transmission toggle	None
165	USBC0_DMAREQ	USBC0 USB Internal DMA Request for Remote Memory Support	None
166	USBC0_DMADONE	USBC0 USB Internal DMA Done for Remote Memory Support	None
167	WDOG0_EXP	WDOG0 Expiration	Level
168	WDOG1_EXP	WDOG1 Expiration	Level
169	WDOG2_EXP	WDOG2 Expiration	Level
170	CANFD0_IPD_REQ	CANFD0 CAN0 DMA request interrupt	None
171	CANFD1_IPD_REQ	CANFD1 CAN1 DMA request interrupt	None

Table 7-3: ADSP-SC59x Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
172	PCG0_HWA	PCG0 PCG-A Hardware trigger control	None
173	PCG0_HWB	PCG0 PCG-B Hardware trigger control	None
174	PCG0_HWC	PCG0 PCG-C Hardware trigger control	None
175	PCG0_HWD	PCG0 PCG-D Hardware trigger control	None
176	PCG0_HWE	PCG0 PCG-E Hardware trigger control	None
177	PCG0_HWF	PCG0 PCG-F Hardware trigger control	None
178	PCG0_HWG	PCG0 PCG-G Hardware trigger control	None
179	PCG0_HWH	PCG0 PCG-H Hardware trigger control	None
180	EMAC0_MCGR_DMA_REQ0	EMAC0 EMAC0 Media Clock Generation Recovery 0	Edge
181	EMAC0_MCGR_DMA_REQ1	EMAC0 EMAC0 Media Clock Generation Recovery 1	Edge
182	EMAC0_MCGR_DMA_REQ2	EMAC0 EMAC0 Media Clock Generation Recovery 2	Edge
183	EMAC0_MCGR_DMA_REQ3	EMAC0 EMAC0 Media Clock Generation Recovery 3	Edge
184	EMAC0_DMA0_RX	EMAC0 EMAC0 DMA0 Rx	None
185	EMAC0_DMA1_RX	EMAC0 EMAC0 DMA1 Rx	None
186	EMAC0_DMA2_RX	EMAC0 EMAC0 DMA2 Rx	None
187	EMAC0_DMA3_RX	EMAC0 EMAC0 DMA3 Rx	None
188	EMAC0_DMA4_RX	EMAC0 EMAC0 DMA4 Rx	None
189	EMAC0_DMA5_RX	EMAC0 EMAC0 DMA5 Rx	None
190	EMAC0_DMA6_RX	EMAC0 EMAC0 DMA6 Rx	None
191	EMAC0_DMA7_RX	EMAC0 EMAC0 DMA7 Rx	None
192	EMAC1_DMA0_RX	EMAC1 EMAC1 DMA0 Rx	None
193	A55MISC0_COREP_ACCEPT0	A55MISC0 A55 PREQ ACCEPT	Edge
194	A55MISC0_COREP_DENY0	A55MISC0 A55 PREQ DENY	Edge
195	A55MISC0_CLUSTERP_ACCEPT	A55MISC0 CLUSTER PREQ ACCEPT	Edge
196	A55MISC0_CLUSTERP_DENY	A55MISC0 CLUSTER PREQ DENY	Edge

Table 7-4: ADSP-SC59x Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
0	CNT0_UD	CNT0 CNT0 Count Up and Direction	Pulse
1	CNT0_DG	CNT0 CNT0 Count Down and Gate	Pulse
2	C0_EVENTIREQ	Core 0 Event Input for Wakeup from WFE State	Pulse
3	C0_EVENTOACK	Core 0 Event Output Acknowledgment for Wakeup	Pulse
4	MDMA0_SRC	Enhanced BW DMA Channel 0 Source (CRC IN)	Pulse
5	MDMA0_DST	Enhanced BW DMA Channel 0 Source (CRC OUT)	Pulse
6	MDMA1_SRC	Enhanced BW DMA Channel 0 Source (CRC IN)	Pulse
7	MDMA1_DST	Enhanced BW DMA Channel 0 Source (CRC OUT)	Pulse
8	MDMA4_SRC	Standard BW DMA Channel 0 Source (CRC IN)	Pulse
9	MDMA4_DST	Standard BW DMA Channel 0 Source (CRC OUT)	Pulse
10	MDMA5_SRC	Standard BW DMA Channel 1 Source (CRC IN)	Pulse
11	MDMA5_DST	Standard BW DMA Channel 1 Source (CRC OUT)	Pulse
12	SOC600CTI0_SLV0	SOC600CTI0 SYSCTI System Halt Master 0	Pulse
13	SOC600CTI0_SLV1	SOC600CTI0 SYSCTI System Halt Master 1	Pulse
14	SOC600CTI0_SLV2	SOC600CTI0 SYSCTI System Halt Master 2	Pulse
15	SOC600CTI0_SLV3	SOC600CTI0 SYSCTI System Halt Master 3	Pulse
16	SOC600CTI0_SLV4	SOC600CTI0 SYSCTI System Halt Master 4	Pulse
17	SOC600CTI0_SLV5	SOC600CTI0 SYSCTI System Halt Master 5	Pulse
18	SOC600CTI0_SLV6	SOC600CTI0 SYSCTI System Halt Master 6	Pulse
19	EPPI0_CH0_DMA	EPPI0 Channel 0 DMA	Pulse
20	EPPI0_CH1_DMA	EPPI0 Channel 1 DMA	Pulse
21	C1_FIR0_TRGI	FIR0 Core 1 FIR Wait on Trigger Input	Pulse
22	C2_FIR0_TRGI	FIR1 Core 2 FIR Wait on Trigger Input	Pulse
23	C1_IIR0_TRGI	IIR0 Core 1 IIR Wait on Trigger Input	Pulse
24	C1_IIR1_TRGI	IIR1 Core 1 IIR Wait on Trigger Input	Pulse

Table 7-4: ADSP-SC59x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
25	C1_IIR2_TRGI	IIR2 Core 1 IIR Wait on Trigger Input	Pulse
26	C1_IIR3_TRGI	IIR3 Core 1 IIR Wait on Trigger Input	Pulse
27	C2_IIR0_TRGI	IIR4 Core 2 IIR Wait on Trigger Input	Pulse
28	C2_IIR1_TRGI	IIR5 Core 2 IIR Wait on Trigger Input	Pulse
29	C2_IIR2_TRGI	IIR6 Core 2 IIR Wait on Trigger Input	Pulse
30	C2_IIR3_TRGI	IIR7 Core 2 IIR Wait on Trigger Input	Pulse
31	LP0_DMA	LP0 DMA Channel	Pulse
32	LP1_DMA	LP1 DMA Channel	Pulse
33	MDMA2_SRC	Enh BW DMA Channel 0	Pulse
34	MDMA2_DST	Enh BW DMA Channel 1	Pulse
35	MDMA3_SRC	Max BW DMA Channel 0	Pulse
36	MDMA3_DST	Max BW DMA Channel 1	Pulse
37	MDMA6_SRC	Enh BW DMA Channel 0	Pulse
38	MDMA6_DST	Enh BW DMA Channel 1	Pulse
39	MDMA7_SRC	Max BW DMA Channel 0	Pulse
40	MDMA7_DST	Max BW DMA Channel 1	Pulse
41	PORTA_TOGGLE	Port Toggle Trigger	Pulse
42	PORTB_TOGGLE	Port Toggle Trigger	Pulse
43	PORTC_TOGGLE	Port Toggle Trigger	Pulse
44	PORTD_TOGGLE	Port Toggle Trigger	Pulse
45	PORTE_TOGGLE	Port Toggle Trigger	Pulse
46	PORTF_TOGGLE	Port Toggle Trigger	Pulse
47	PORTG_TOGGLE	Port Toggle Trigger	Pulse
48	PORTH_TOGGLE	Port Toggle Trigger	Pulse
49	PORTI_TOGGLE	Port Toggle Trigger	Pulse
50	RCU0_SYSRST0	RCU0 System Reset Slave 0	Pulse
51	RCU0_SYSRST1	RCU0 System Reset Slave 1	Pulse
52	SPI0_TXDMA	SPI0 TX DMA Channel	Pulse
53	SPI0_RXDMA	SPI0 RX DMA Channel	Pulse
54	SPI1_TXDMA	SPI1 TX DMA Channel	Pulse
55	SPI1_RXDMA	SPI1 RX DMA Channel	Pulse

Table 7-4: ADSP-SC59x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
56	SPI3_TXDMA	SPI3 TX DMA Channel	Pulse
57	SPI3_RXDMA	SPI3 RX DMA Channel	Pulse
58	SPI2_TXDMA	SPI2 TX DMA Channel	Pulse
59	SPI2_RXDMA	SPI2 RX DMA Channel	Pulse
60	SPORT0_A_DMA	SPORT0 Channel A DMA	Pulse
61	SPORT0_B_DMA	SPORT0 Channel B DMA	Pulse
62	SPORT1_A_DMA	SPORT1 Channel A DMA	Pulse
63	SPORT1_B_DMA	SPORT1 Channel B DMA	Pulse
64	SPORT2_A_DMA	SPORT2 Channel A DMA	Pulse
65	SPORT2_B_DMA	SPORT2 Channel B DMA	Pulse
66	SPORT3_A_DMA	SPORT3 Channel A DMA	Pulse
67	SPORT3_B_DMA	SPORT3 Channel B DMA	Pulse
68	SPORT4_A_DMA	SPORT4 Channel A DMA	Pulse
69	SPORT4_B_DMA	SPORT4 Channel B DMA	Pulse
70	SPORT5_A_DMA	SPORT5 Channel A DMA	Pulse
71	SPORT5_B_DMA	SPORT5 Channel B DMA	Pulse
72	SPORT6_A_DMA	SPORT6 Channel A DMA	Pulse
73	SPORT6_B_DMA	SPORT6 Channel B DMA	Pulse
74	SPORT7_A_DMA	SPORT7 Channel A DMA	Pulse
75	SPORT7_B_DMA	SPORT7 Channel B DMA	Pulse
76	DAI0_GBL_SPORT_TRG_I0	DAI0 SPORT GROUP0 Trigger Input	Pulse
77	DAI0_GBL_SPORT_TRG_I1	DAI0 SPORT GROUP1 Trigger Input	Pulse
78	DAI1_GBL_SPORT_TRG_I0	DAI1 SPORT GROUP2 Trigger Input	Pulse
79	DAI1_GBL_SPORT_TRG_I1	DAI1 SPORT GROUP3 Trigger Input	Pulse
80	STM0_EVT0	STM0 STM0 Event 0	Pulse
81	STM0_EVT1	STM0 STM0 Event 1	Pulse
82	STM0_EVT2	STM0 STM0 Event 2	Pulse
83	STM0_EVT3	STM0 STM0 Event 3	Pulse

Table 7-4: ADSP-SC59x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
84	STM0_EVT4	STM0 STM0 Event 4	Pulse
85	STM0_EVT5	STM0 STM0 Event 5	Pulse
86	STM0_EVT6	STM0 STM0 Event 6	Pulse
87	STM0_EVT7	STM0 STM0 Event 7	Pulse
88	STM0_EVT8	STM0 STM0 Event 8	Pulse
89	STM0_EVT9	STM0 STM0 Event 9	Pulse
90	STM0_EVT10	STM0 STM0 Event 10	Pulse
91	STM0_EVT11	STM0 STM0 Event 11	Pulse
92	STM0_EVT12	STM0 STM0 Event 12	Pulse
93	STM0_EVT13	STM0 STM0 Event 13	Pulse
94	STM0_EVT14	STM0 STM0 Event 14	Pulse
95	STM0_EVT15	STM0 STM0 Event 15	Pulse
96	STM0_EVT16	STM0 STM0 Event 16	Pulse
97	STM0_EVT17	STM0 STM0 Event 17	Pulse
98	STM0_EVT18	STM0 STM0 Event 18	Pulse
99	STM0_EVT19	STM0 STM0 Event 19	Pulse
100	STM0_EVT20	STM0 STM0 Event 20	Pulse
101	STM0_EVT21	STM0 STM0 Event 21	Pulse
102	STM0_EVT22	STM0 STM0 Event 22	Pulse
103	STM0_EVT23	STM0 STM0 Event 23	Pulse
104	STM0_EVT24	STM0 STM0 Event 24	Pulse
105	STM0_EVT25	STM0 STM0 Event 25	Pulse
106	STM0_EVT26	STM0 STM0 Event 26	Pulse
107	STM0_EVT27	STM0 STM0 Event 27	Pulse
108	STM0_EVT28	STM0 STM0 Event 28	Pulse
109	STM0_EVT29	STM0 STM0 Event 29	Pulse
110	STM0_EVT30	STM0 STM0 Event 30	Pulse
111	STM0_EVT31	STM0 STM0 Event 31	Pulse
113	SWU1_EN	SWU1 Enable	Pulse
114	SWU2_EN	SWU2 Enable	Pulse
115	SWU7_EN	SWU7 Enable	Pulse

Table 7-4: ADSP-SC59x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
116	SWU8_EN	SWU8 Enable	Pulse
117	SWU9_EN	SWU9 Enable	Pulse
118	SWU10_EN	SWU10 Enable	Pulse
119	SWU11_EN	SWU11 Enable	Pulse
120	SWU12_EN	SWU12 Enable	Pulse
121	SWU13_EN	SWU13 Enable	Pulse
122	SWU3_EN	SWU3 Enable Event CL2_1	Pulse
123	SWU4_EN	SWU4 Enable Event DL2_1	Pulse
124	SWU5_EN	SWU5 Enable Event CL2_2	Pulse
125	TIMER0_TMR00_SLV0	TIMER0 Timer 0	Pulse
126	TIMER0_TMR00_SLV1	TIMER0 Timer 0	Pulse
127	TIMER0_TMR01_SLV0	TIMER0 Timer 1	Pulse
128	TIMER0_TMR01_SLV1	TIMER0 Timer 1	Pulse
129	TIMER0_TMR02_SLV0	TIMER0 Timer 2	Pulse
130	TIMER0_TMR02_SLV1	TIMER0 Timer 2	Pulse
131	TIMER0_TMR03_SLV0	TIMER0 Timer 3	Pulse
132	TIMER0_TMR03_SLV1	TIMER0 Timer 3	Pulse
133	TIMER0_TMR04_SLV0	TIMER0 Timer 4	Pulse
134	TIMER0_TMR04_SLV1	TIMER0 Timer 4	Pulse
135	TIMER0_TMR05_SLV0	TIMER0 Timer 5	Pulse
136	TIMER0_TMR05_SLV1	TIMER0 Timer 5	Pulse
137	TIMER0_TMR06_SLV0	TIMER0 Timer 6	Pulse
138	TIMER0_TMR06_SLV1	TIMER0 Timer 6	Pulse
139	TIMER0_TMR07_SLV0	TIMER0 Timer 7	Pulse
140	TIMER0_TMR07_SLV1	TIMER0 Timer 7	Pulse
141	TIMER0_TMR08_SLV0	TIMER0 Timer 8	Pulse
142	TIMER0_TMR08_SLV1	TIMER0 Timer 8	Pulse
143	TIMER0_TMR09_SLV0	TIMER0 Timer 9	Pulse
144	TIMER0_TMR09_SLV1	TIMER0 Timer 9	Pulse
145	TIMER0_TMR10_SLV0	TIMER0 Timer 10	Pulse
146	TIMER0_TMR10_SLV1	TIMER0 Timer 10	Pulse

Table 7-4: ADSP-SC59x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
147	TIMER0_TMR11_SLV0	TIMER0 Timer 11	Pulse
148	TIMER0_TMR11_SLV1	TIMER0 Timer 11	Pulse
149	TIMER0_TMR12_SLV0	TIMER0 Timer 12	Pulse
150	TIMER0_TMR12_SLV1	TIMER0 Timer 12	Pulse
151	TIMER0_TMR13_SLV0	TIMER0 Timer 13	Pulse
152	TIMER0_TMR13_SLV1	TIMER0 Timer 13	Pulse
153	TIMER0_TMR14_SLV0	TIMER0 Timer 14	Pulse
154	TIMER0_TMR14_SLV1	TIMER0 Timer 14	Pulse
155	TIMER0_TMR15_SLV0	TIMER0 Timer 15	Pulse
156	TIMER0_TMR15_SLV1	TIMER0 Timer 15	Pulse
157	TRU0_SLV0	TRU0 Interrupt Request 0	Pulse
158	TRU0_SLV1	TRU0 Interrupt Request 1	Pulse
159	TRU0_SLV2	TRU0 Interrupt Request 2	Pulse
160	TRU0_SLV3	TRU0 Interrupt Request 3	Pulse
161	TRU0_SLV4	TRU0 Interrupt Request 4	Pulse
162	TRU0_SLV5	TRU0 Interrupt Request 5	Pulse
163	TRU0_SLV6	TRU0 Interrupt Request 6	Pulse
164	TRU0_SLV7	TRU0 Interrupt Request 7	Pulse
165	TRU0_SLV8	TRU0 Interrupt Request 8	Pulse
166	TRU0_SLV9	TRU0 Interrupt Request 9	Pulse
167	TRU0_SLV10	TRU0 Interrupt Request 10	Pulse
168	TRU0_SLV11	TRU0 Interrupt Request 11	Pulse
169	UART0_TXDMA	UART0 Transmit DMA	Pulse
170	UART0_RXDMA	UART0 Receive DMA	Pulse
171	UART1_TXDMA	UART1 Transmit DMA	Pulse
172	UART1_RXDMA	UART1 Receive DMA	Pulse
173	UART2_TXDMA	UART2 Transmit DMA	Pulse
174	UART2_RXDMA	UART2 Receive DMA	Pulse
175	UART3_TXDMA	UART3 Transmit DMA	Pulse
176	UART3_RXDMA	UART3 Receive DMA	Pulse
177	PCG0_HWA	PCG0 PCG-A Hardware trigger control	Pulse

Table 7-4: ADSP-SC59x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
178	PCG0_HWB	PCG0 PCG-B Hardware trigger control	Pulse
179	PCG0_HWC	PCG0 PCG-C Hardware trigger control	Pulse
180	PCG0_HWD	PCG0 PCG-D Hardware trigger control	Pulse
181	PCG0_HWE	PCG0 PCG-E Hardware trigger control	Pulse
182	PCG0_HWF	PCG0 PCG-F Hardware trigger control	Pulse
183	PCG0_HWG	PCG0 PCG-G Hardware trigger control	Pulse
184	PCG0_HWH	PCG0 PCG-H Hardware trigger control	Pulse
185		Reserved	
186		Reserved	
187		Reserved	
188	EMAC0_DMA0_RX	EMAC0 EMAC0 DMA0 Rx	Pulse
189	EMAC0_DMA1_RX	EMAC0 EMAC0 DMA1 Rx	Pulse
190	EMAC0_DMA2_RX	EMAC0 EMAC0 DMA2 Rx	Pulse
191	EMAC0_DMA3_RX	EMAC0 EMAC0 DMA3 Rx	Pulse
192	EMAC0_DMA4_RX	EMAC0 EMAC0 DMA4 Rx	Pulse
193	EMAC0_DMA5_RX	EMAC0 EMAC0 DMA5 Rx	Pulse
194	EMAC0_DMA6_RX	EMAC0 EMAC0 DMA6 Rx	Pulse
195	EMAC0_DMA7_RX	EMAC0 EMAC0 DMA7 Rx	Pulse
196	EMAC0_DMA0_TX	EMAC0 EMAC0 DMA0 Tx	Pulse
197	EMAC0_DMA1_TX	EMAC0 EMAC0 DMA1 Tx	Pulse
198	EMAC0_DMA2_TX	EMAC0 EMAC0 DMA2 Tx	Pulse
199	EMAC0_DMA3_TX	EMAC0 EMAC0 DMA3 Tx	Pulse
200	EMAC0_DMA4_TX	EMAC0 EMAC0 DMA4 Tx	Pulse
201	EMAC0_DMA5_TX	EMAC0 EMAC0 DMA5 Tx	Pulse
202	EMAC0_DMA6_TX	EMAC0 EMAC0 DMA6 Tx	Pulse
203	EMAC0_DMA7_TX	EMAC0 EMAC0 DMA7 Tx	Pulse
204	EMAC1_DMA0_RX	EMAC1 EMAC1 DMA0 Rx	Pulse
205	EMAC1_DMA0_TX	EMAC1 EMAC1 DMA0 Tx	Pulse
206	EMAC0_DMA0_RX_STOP	EMAC0 EMAC0 DMA0 Rx Stop	Pulse
207	EMAC0_DMA1_RX_STOP	EMAC0 EMAC0 DMA1 Rx Stop	Pulse
208	EMAC0_DMA2_RX_STOP	EMAC0 EMAC0 DMA2 Rx Stop	Pulse

Table 7-4: ADSP-SC59x Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
209	EMAC0_DMA3_RX_STOP	EMAC0 EMAC0 DMA3 Rx Stop	Pulse
210	EMAC0_DMA4_RX_STOP	EMAC0 EMAC0 DMA4 Rx Stop	Pulse
211	EMAC0_DMA5_RX_STOP	EMAC0 EMAC0 DMA5 Rx Stop	Pulse
212	EMAC0_DMA6_RX_STOP	EMAC0 EMAC0 DMA6 Rx Stop	Pulse
213	EMAC0_DMA7_RX_STOP	EMAC0 EMAC0 DMA7 Rx Stop	Pulse
214	EMAC0_DMA0_TX_STOP	EMAC0 EMAC0 DMA0 Tx Stop	Pulse
215	EMAC0_DMA1_TX_STOP	EMAC0 EMAC0 DMA1 Tx Stop	Pulse
216	EMAC0_DMA2_TX_STOP	EMAC0 EMAC0 DMA2 Tx Stop	Pulse
217	EMAC0_DMA3_TX_STOP	EMAC0 EMAC0 DMA3 Tx Stop	Pulse
218	EMAC0_DMA4_TX_STOP	EMAC0 EMAC0 DMA4 Tx Stop	Pulse
219	EMAC0_DMA5_TX_STOP	EMAC0 EMAC0 DMA5 Tx Stop	Pulse
220	EMAC0_DMA6_TX_STOP	EMAC0 EMAC0 DMA6 Tx Stop	Pulse
221	EMAC0_DMA7_TX_STOP	EMAC0 EMAC0 DMA7 Tx Stop	Pulse
222	EMAC1_DMA0_RX_STOP	EMAC1 EMAC1 DMA0 Rx Stop	Pulse
223	EMAC1_DMA0_TX_STOP	EMAC1 EMAC1 DMA0 Tx Stop	Pulse
224	EMAC0_MCGR_DMA_ACK0	EMAC0 EMAC0 Media Clock Generation Recovery Ack 0	Pulse
225	EMAC0_MCGR_DMA_ACK1	EMAC0 EMAC0 Media Clock Generation Recovery Ack 1	Pulse
226	EMAC0_MCGR_DMA_ACK2	EMAC0 EMAC0 Media Clock Generation Recovery Ack 2	Pulse
227	EMAC0_MCGR_DMA_ACK3	EMAC0 EMAC0 Media Clock Generation Recovery Ack 3	Pulse

TRU Definitions

The following definitions are helpful when using the TRU module.

Trigger Generator

A trigger generator is any system module that provides trigger event indication to the TRU. Trigger generator modules define trigger events and conditions for assertion.

Trigger Generator ID

Trigger generators are assigned a unique numeric ID according to their physical connection to the TRU. Trigger receiver ID 0 is reserved and defined as null.

Trigger Receiver

A trigger receiver is any system module that receives a trigger event indication from the TRU. Trigger receiver modules define a trigger event response.

NOTE: For peripherals configured as a receiver: when the corresponding peripheral DMA is kept waiting for trigger from a generator, it can receive stale or garbage data. The peripheral receiver begins shifting in data as soon as it is enabled, and it does not wait for the trigger generator. Upon receiving the trigger, this data is moved into the DMA FIFO. Therefore, this sequence can result in the receiver getting unusable data.

TRU Block Diagram

The trigger generator and the Generator Trigger register (MTR) generate trigger assertions. Each trigger receiver has a dedicated Receiver Select register (SSR) that specifies the unique trigger generator from which it receives the trigger indication.

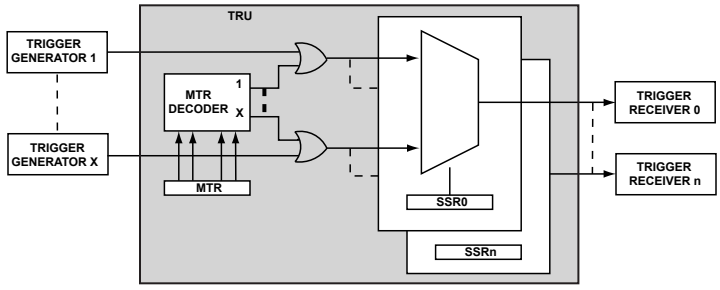


Figure 7-1: TRU Block Diagram

TRU Architectural Concepts

The TRU supports a simple trigger-in/trigger-out model for modules that comply with the triggering functional model. The TRU is the controller of the trigger system. Trigger outputs from trigger generators are mapped to trigger inputs of trigger receivers through a set of programmable registers (`TRU_SSR[n]`).

System modules are trigger generator only, trigger receiver only, or trigger generator and trigger receiver.

All of the trigger input and output signals are connected to a trigger routing unit (TRU) which manages the connections of triggers between modules.

In multi-processor systems, multiple TRU units are provided. These TRUs are networked together. Generic Trigger Ports (GTPs) are provided to forward trigger events from one TRU unit to another, forming a pathway from trigger generator to trigger receivers wherever they might lie in the system.

TRU Programming Model

Implementing sequence control using the TRU requires, at a minimum, proper configuration of a trigger receiver, a trigger generator, and the TRU module itself. The only requirement for the configuration procedure is that the trigger generator is configured and enabled as the last step.

Complete the following other steps:

- Configure the trigger receiver for response to triggers
- Configure the TRU to map the trigger generator to the trigger receiver through the `TRU_SSR[n]` registers
- Configure the trigger generator to generate trigger assertions
- Alternatively, use software triggering for trigger assertion. Writing the trigger generator ID to the MTR register generates software triggers.

Programming Concepts

The following concepts aid in programming the TRU.

- ***Trigger Sequence Configuration.*** A simple sequence consists of one trigger generator and one trigger receiver. More complex trigger sequences consist of several trigger receivers functioning as trigger receiver and trigger generator. Additionally, trigger sequences can loopback to the original generator forming a perpetual sequence.
- ***Software Triggering.*** Writing a trigger generator ID to the MTR generates a trigger within the TRU from the trigger generator ID specified.
- ***Synchronization.*** The TRU can be used to coarsely synchronize events by mapping multiple trigger receivers to the same trigger generator or by generating multiple trigger generator assertions simultaneously through the MTR.
- ***Configuration Protection.*** The `TRU_SSR[n].LOCK` bit and the `TRU_GCTL.LOCK` bit enable register level write-protection when the global lock is asserted in the SPU.

Programming Examples

The following examples shows the steps to create a single trigger.

Configuring a Simple Trigger Sequence

The following example shows the steps to create a simple trigger.

1. Write to the `TRU_GCTL` register to enable the TRU.
2. Write to the `TRU_SSR[n]` register of a specific receiver target to assign it to a specific trigger generator.
3. Enable the trigger receiver to wait for and accept a trigger.
4. Enable the trigger generator to generate a trigger.

TRU Event Control

The TRU is a major part of event control solutions. It is the center of the trigger functional model and can extend to support the interrupt and fault management models as well.

TRU Status and Error Signals

The TRU does not have dedicated status and error output signals other than the MMR interface. Receiver errors are reported to the generator over the standard peripheral bus protocol.

ADSP-SC59x TRU Register Descriptions

Trigger Routing Unit (TRU) contains the following registers.

Table 7-5: ADSP-SC59x TRU Register List

Name	Description
TRU_ERRADDR	Error Address Register
TRU_GCTL	Global Control Register
TRU_MTR	Generator Trigger Register
TRU_SSR[n]	Receiver Select Register
TRU_STAT	Status Information Register

Error Address Register

The TRU error address register (TRU_ERRADDR) holds the address from the memory-mapped register access generating an access error of TRU registers.

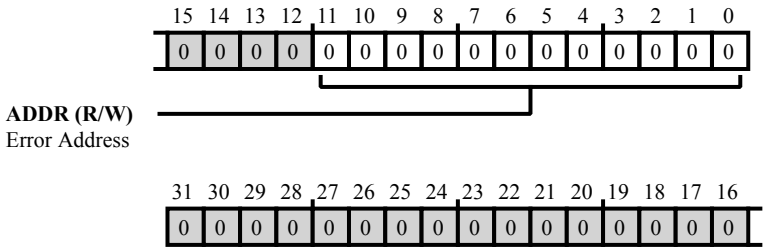


Figure 7-2: TRU_ERRADDR Register Diagram

Table 7-6: TRU_ERRADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	ADDR	Error Address. The TRU_ERRADDR.ADDR holds the address from the memory-mapped register access generating an access error of TRU registers. These errors occur on access to the TRU_SSR[n] or TRU_MTR registers when these registers are locked or on access to an invalid address. See the TRU_SSR[n] and TRU_MTR register descriptions for more information about locking. The TRU_ERRADDR register holds the address of the first error to occur. In the event of multiple errors occurring, the TRU_ERRADDR register contains the address of the first error. To re-enable the TRU_ERRADDR register for update, both status bits (TRU_STAT.LWERR and TRU_STAT.ADDRERR) in the TRU_STAT register must be cleared.

Global Control Register

The TRU global control register (TRU_GCTL) provides register locking, TRU reset, and TRU enable.

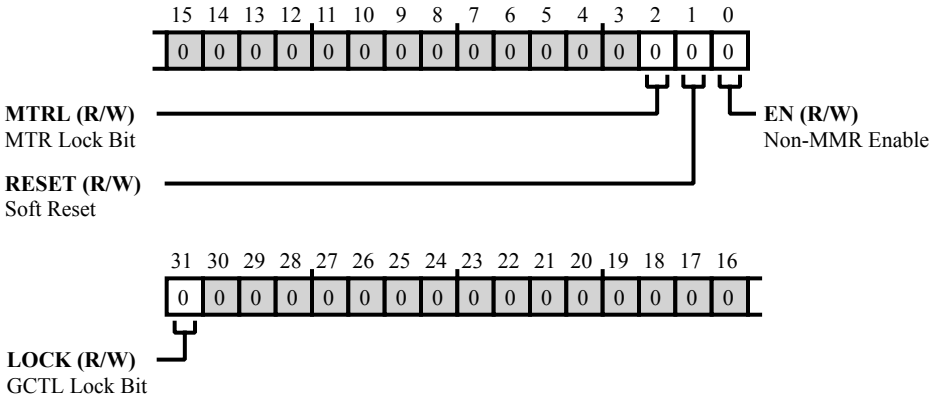


Figure 7-3: TRU_GCTL Register Diagram

Table 7-7: TRU_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	GCTL Lock Bit. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the TRU_GCTL.LOCK bit is enabled, the TRU_GCTL register is read only.	
		0	Read write
		1	Read only
2 (R/W)	MTRL	MTR Lock Bit. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the TRU_GCTL.MTRL bit is enabled, the TRU_MTR register is read only.	
		0	Read write
		1	Read only
1 (R/W)	RESET	Soft Reset. The TRU_GCTL.RESET bit is write-1-action and triggers a soft reset to all TRU registers.	
		0	No action
		1	Soft reset

Table 7-7: TRU_GCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Non-MMR Enable. The `TRU_GCTL.EN` bit is read/write and must be set for the TRU to propagate trigger events. All TRU register read/write operations continue to operate independent of the `TRU_GCTL.EN` bit.	
		0	No trigger events
		1	Propagate trigger events

Generator Trigger Register

The TRU generator trigger register (TRU_MTR) permits trigger generation through software by writing a trigger requester ID value to one of the four fields in the TRU_MTR register. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the TRU_GCTL.LOCK bit is set, the TRU_MTR register is read only. Note this register is primarily used for debug to trigger a TRU output

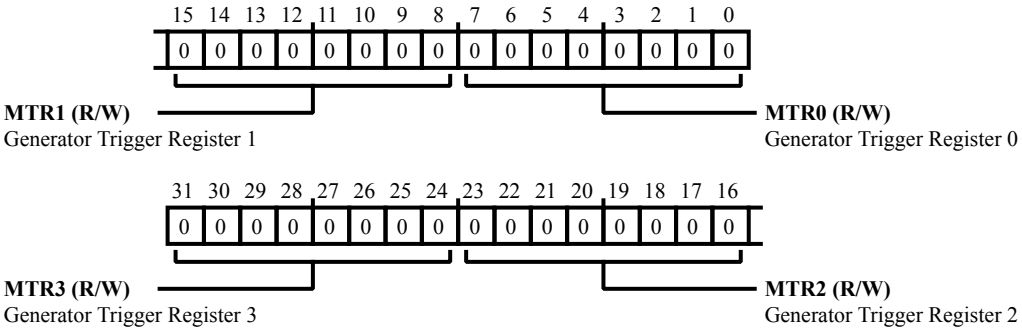


Figure 7-4: TRU_MTR Register Diagram

Table 7-8: TRU_MTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	MTR3	Generator Trigger Register 3. The TRU_MTR.MTR3 bit field is the trigger generator ID value for generator 3.	
		0	No generator specified
		1-197	Range of valid generators
23:16 (R/W)	MTR2	Generator Trigger Register 2. The TRU_MTR.MTR2 bit field is the trigger requester ID value for requester 2.	
		0	No generator specified
		1-197	Range of valid generators
15:8 (R/W)	MTR1	Generator Trigger Register 1. The TRU_MTR.MTR1 bit field is the trigger generator ID value for generator 1.	
		0	No generator specified
		1-197	Range of valid generators
7:0 (R/W)	MTR0	Generator Trigger Register 0. The TRU_MTR.MTR0 bit field is the trigger generator ID value for generator 0.	
		0	No generator specified
		1-197	Range of valid generators

Receiver Select Register

The TRU receiver select registers (TRU_SSR[n]) each provide receiver selection and register locking.

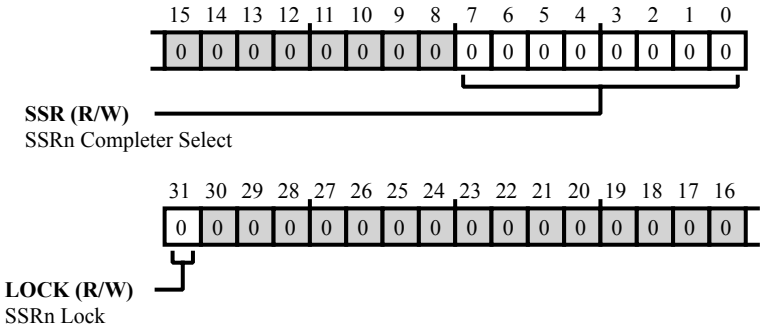


Figure 7-5: TRU_SSR[n] Register Diagram

Table 7-9: TRU_SSR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	SSRn Lock. If the global lock is enabled (SPU_CTL.GLCK bit =1) and the TRU_SSR[n].LOCK bit is enabled, the TRU_SSR[n] register is read only.	
		0	Unlock register
		1	Lock register
7:0 (R/W)	SSR	SSRn Completer Select. The TRU_SSR[n] register selects the trigger generator ID to which the trigger receiver responds. For example, when a TRU_SSR[n] register is set to respond to trigger generator ID n, a trigger that is generated by trigger generator ID n results in a trigger out to the receiver.	
		0	No generator specified
		1-197	Range of valid generators

Status Information Register

The TRU status register (TRU_STAT) contains the status of TRU_MTR and TRU_SSR[n] register writes and status of bus read/write errors.

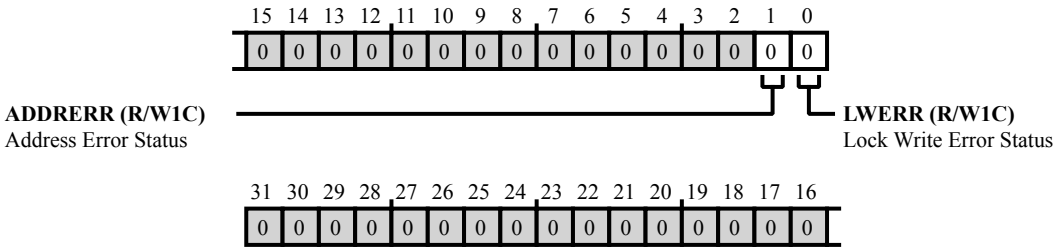


Figure 7-6: TRU_STAT Register Diagram

Table 7-10: TRU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	ADDRERR	Address Error Status. The TRU_STAT.ADDRERR bit is set when an invalid address is provided for an MMR access while the TRU is selected. Writing a one to this bit clears the error indication. The TRU_ERRADDR register also is updated when an address error occurs during an MMR access while the TRU is selected.	
		0	No error
		1	Error occurred
0 (R/W1C)	LWERR	Lock Write Error Status. If TRU_STAT.LWERR is set, a lock write error has occurred. Writing a one to this bit clears the error indication.	
		0	No error
		1	Error occurred

8 L2 System Memory

L2 system memories have significant bandwidth for core accesses, but it is important to note that L2 responds slower to core accesses than L1 memories. L2 SRAM is the ideal storage for multiple processor cores to share data and instruction resources, such as semaphores, shared buffers, and code libraries. Due to sophisticated data integrity protection and write protection, L2 SRAM is also ideal for data and instructions critical for safe operation of the application.

L2 System Memory Features

The L2 system memory features include:

- Operation at `SYSCLK` frequency
- ECC protection of SRAM area
- ECC memory refresh
- L2CTL0 contains 2M byte of RAM grouped into eight banks, 256K bytes each and 192K bytes of boot ROM.

L2 System Memory Functional Description

The L2 system memory manages all of the L2 SRAM and ROM memory banks. The system memory interface arbitrates competing accesses, write protection, and ensures SRAM data integrity. The L2 system memory domain is a unified instruction and data memory. It can hold any mixture of code and data required by the system design.

The following sections provide a functional description of the L2 system memory.

ADSP-SC59x L2CTL Register List

The L2 memory controller (L2CTL) includes the controls to manage each L2 memory bank independently. A set of registers governs L2CTL operations. For more information on L2CTL functionality, see the L2CTL register descriptions.

Table 8-1: ADSP-SC59x L2CTL Register List

Name	Description
L2CTL_CTL	Control Register
L2CTL_EADDR0	Error Type 0 Address Register
L2CTL_EADDR1	Error Type 1 Address Register
L2CTL_EADDR2	Error Type 2 Address Register
L2CTL_EADDR3	Error Type 3 Address Register
L2CTL_EADDR4	Error Type 4 Address Register
L2CTL_ERRADDR0	ECC Error Address 0 Register
L2CTL_ERRADDR1	ECC Error Address 1 Register
L2CTL_ERRADDR2	ECC Error Address 2 Register
L2CTL_ERRADDR3	ECC Error Address 3 Register
L2CTL_ERRADDR4	ECC Error Address 4 Register
L2CTL_ERRADDR5	ECC Error Address 5 Register
L2CTL_ERRADDR6	ECC Error Address 6 Register
L2CTL_ERRADDR7	ECC Error Address 7 Register
L2CTL_ERRADDR8	ECC Error Address 8 Register
L2CTL_ET0	Error Type 0 Register
L2CTL_ET1	Error Type 1 Register
L2CTL_ET2	Error Type 2 Register
L2CTL_ET3	Error Type 3 Register
L2CTL_ET4	Error Type 4 Register
L2CTL_INIT	Initialization Register
L2CTL_ISTAT	Initialization Status Register
L2CTL_PCTL	Power Control Register
L2CTL_REVID	Revision ID Register
L2CTL_RPCR0	Read Priority Count Register
L2CTL_RPCR1	Read Priority Count Register
L2CTL_SADR	Scrub Start Address Register
L2CTL_SCNT	Scrub Count Register
L2CTL_SCTL	Scrub Control Register
L2CTL_STAT	Status Register
L2CTL_STAT_1	L2 Port Error Status Register

Table 8-1: ADSP-SC59x L2CTL Register List (Continued)

Name	Description
L2CTL_WPCR0	Write Priority Count Register
L2CTL_WPCR1	Write Priority Count Register

ADSP-SC59x L2CTL Interrupt List

Table 8-2: ADSP-SC59x L2CTL Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
10	L2CTL0_ECC_ERR	L2CTL0 ECC Error	Level	
12	L2CTL0_EVT	L2CTL0 Scrub/Initialization Done	Level	

ADSP-SC59x L2CTL Trigger List

Table 8-3: ADSP-SC59x L2CTL Trigger List Generators

Trigger ID	Name	Description	Sensitivity
42	L2CTL0_EVT	L2CTL0 L2 Memory Event	Level

Table 8-4: ADSP-SC59x L2CTL Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

L2 System Memory Block Diagram

The ***ADSP-SC59x Complete L2 System Memory Block Diagram*** shows the complete L2 system memory, including the memory block instance L2CTL0. The L2CTL0 block contains boot ROM code for Arm and eight banks of L2 RAM containing 256 Kbytes each. The Arm reset ISR (location 0x00000000) is mapped to this block.

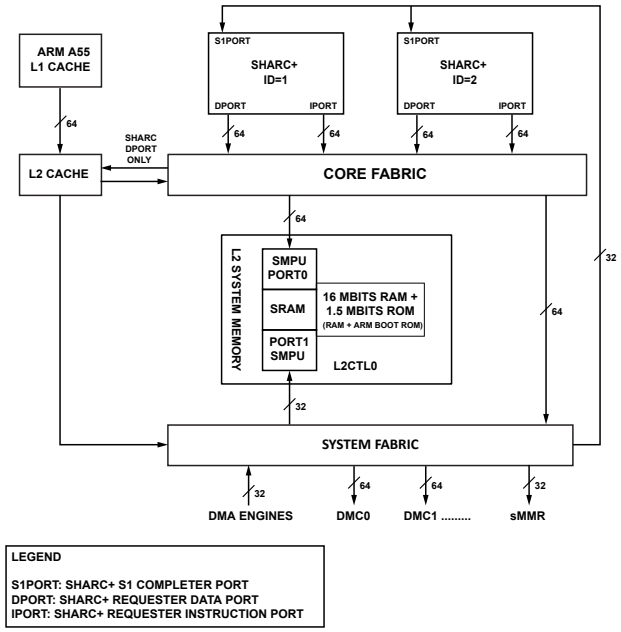


Figure 8-1: ADSP-SC59x Complete L2 System Block Diagram

L2 System Memory Architectural Concepts

The following sections describe the architecture features of the L2 system memory.

Access Characteristics

The L2 system memory interface converts all 8-bit, 16-bit, 32-bit, and 64-bit accesses to 128-bit accesses. Additionally, it converts 8-bit, 16-bit, 32-bit, and 64-bit bursts to an equivalent internal 128-bit access. For example, the L2 system memory interface converts a 128-bit address-aligned burst of 8-bit accesses of burst length 16 to a single 128-bit access.

Read/Write Latency and Throughput

The L2 memory design is optimized for burst accesses at the crossbar interface. The L2 system memory buffers and converts write data of 8/16/32/64-bit to an equivalent 128-bit access. This conversion creates modulo-32-bit writes if the starting addresses are 32-bit aligned. A single 8-bit or 16-bit access, or a non-32-bit address-aligned 8-bit or 16-bit burst access to an ECC-enabled bank creates an extra latency of two SYSCLK cycles. No extra latency is seen if the ECC is disabled.

NOTE: Continuous 8/16-bit core access to an ECC-enabled L2 bank is not recommended from a throughput perspective.

L2 Memory Controller Block Diagram (Instance)

As shown in the ***L2 System Memory Block Diagram***, the L2 controller has five ports that interface to system crossbars. Port 0, port 1, and port 2 are 128-bit interfaces dedicated to core and MDMA3 traffic. Port 3 and port 4 are 128-bit interfaces that connect through DMA. For L2 SRAM, all ports (0/1/2/3/4) have a read channel and a write

channel; for L2 ROM, all ports (0/1/2/3/4) have read channels only. The SRAM is organized in multiple banks; each bank has 256K Bytes. The 192KB ROM is divided into three banks of 64KB.

Within each bank, data is organized into 16384 words, with each word comprising 128 bits of data and 28 bits of ECC checksum. ROM memory is not protected by the ECC scheme. When the L2 controller accesses RAM and ROM cells, it always reads and writes whole 128-bit words. Despite this, the L2 controller supports 8-, 16-, 32-, and 64-bit reads and writes from cores and system by applying respective data masks.

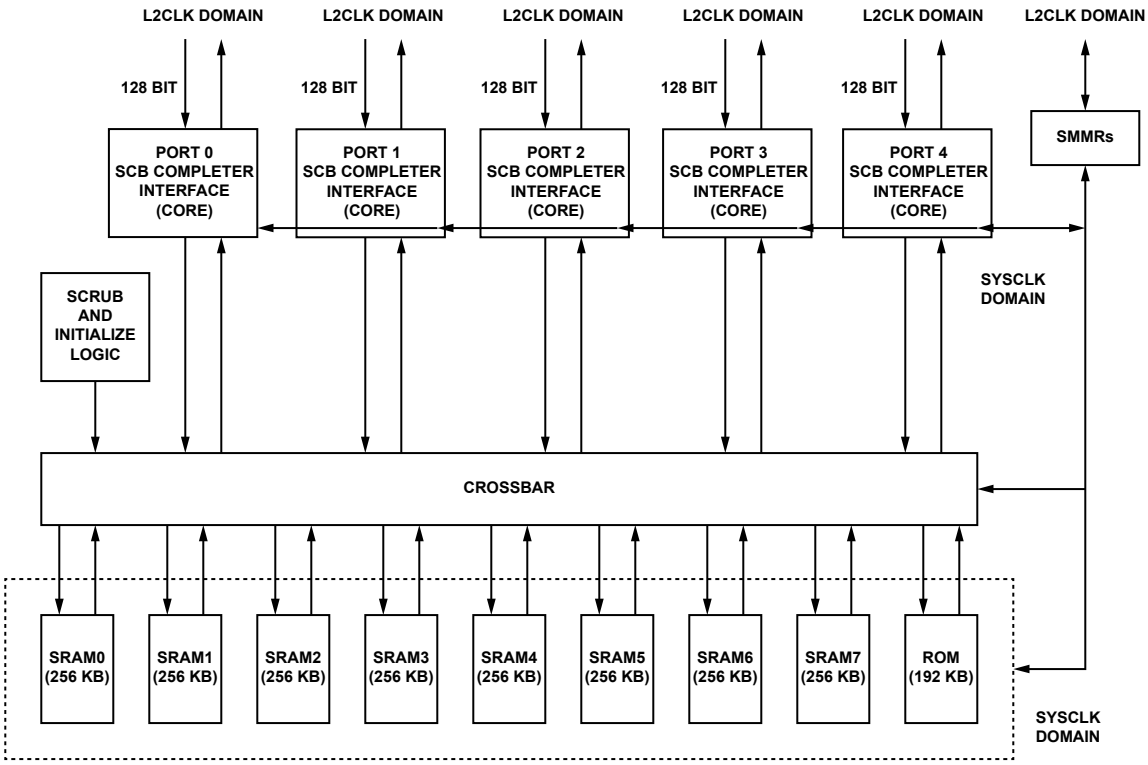


Figure 8-2: L2 System Memory Block Diagram

Arbitration and Priority

Each bank of L2 RAM or ROM has an arbiter which receives requests from the five crossbar ports.

Each arbiter follows a fixed priority scheme for giving grants when more than one channel requests the same bank. The arbiter also supports priority elevation through urgent priority requests.

NOTE: Attempting a write access to both L2 ROM spaces returns an error.

The ***Fixed Priority*** table shows the priority for fixed priority mode (with urgent priority disabled) for each SCB channel.

If two cores (or the 64-bit Max BW DMA) simultaneously try to access L2 for the same instance (both read or both write), even to different banks, software allows only one controller access at a time. One access port can support one read and write at the same time. However, if one core issues a write and the other issues a read, then access can proceed simultaneously. There is no extra latency inside L2, when the accesses are to different banks (assuming pending DMA traffic is also to a non-conflicting bank).

When a core and DMA both access the same bank via the same port (both read or both writes), the best access rate that DMA can achieve is one 128-bit access in every three SYSCLK cycles during the conflict period. This access rate is achieved by programming the read priority count register (`L2CTL_RPCR0.RPC0`) bit and the write priority count register (`L2CTL_WPCR0.WPC0`) to 0, while programming the `L2CTL_RPCR0.RPC1` and the `L2CTL_WPCR0.WPC1` bits to 1.

The arbiters also support priority elevation for a particular channel that has been starved of grants for many SYSCLK cycles. If a channel does not get a grant for *N* cycles after its request, then that channel can elevate the priority of its request by issuing an urgent priority request. This request causes that channel to become the highest priority controller for the next grant cycle (pipelined arbitration for urgent priority). The number of cycles *N*, after which the priority is elevated, can be programmed for each channel separately using the (`L2CTL_RPCR0`) and (`L2CTL_WPCR0`) registers.

Programming the bits in the (`L2CTL_RPCR0`) and (`L2CTL_WPCR0`) registers appropriately achieves the best grant rate for DMA. This grant rate of one in three SYSCLK cycles during the conflict period is achievable under the following conflict conditions:

- An access conflict between the core and DMA to the same memory bank in the fixed priority arbitration scheme with core activity always prioritized over DMA activity
- An access conflict within the pipelined implementation of urgent priority

To disable urgent priority requests, set the `L2CTL_CTL.DISURP` bit. This bit disables the urgent priority requests for all port channels. Each channel can also be prevented from raising the urgent priority request through the priority count register for the specific channel. However, there is no support for disabling urgent priority for a specific memory bank arbiter.

The ***Fixed Priority With Priority Elevation*** table provides the various priority levels for the L2 system memory.

Table 8-5: Fixed Priority With Priority Elevation

Channel	Priority Level
L2 Scrub/Initialization Request	21 (highest)
Port 0 Read Channel Urgent Request	20
Port 0 Write Channel Urgent Request	19
Port 1 Read Channel Urgent Request	18
Port 1 Write Channel Urgent Request	17
Port 2 Read Channel Urgent Request	16
Port 2 Write Channel Urgent Request	15
Port 3 Read Channel Urgent Request	14
Port 3 Write Channel Urgent Request	13
Port 4 Read Channel Urgent Request	12
Port 4 Write Channel Urgent Request	11

Table 8-5: Fixed Priority With Priority Elevation (Continued)

Channel	Priority Level
Port 0 Read Channel Normal Request	10
Port 0 Write Channel Normal Request	9
Port 1 Read Channel Normal Request	8
Port 1 Write Channel Normal Request	7
Port 2 Read Channel Normal Request	6
Port 2 Write Channel Normal Request	5
Port 3 Read Channel Normal Request	4
Port 3 Write Channel Normal Request	3
Port 4 Read Channel Normal Request	2
Port 4 Write Channel Normal Request	1 (lowest)

Data Integrity

The following sections provide information on how the L2 system memory ensures data integrity.

ECC Algorithm

Hsaio encoding calculates the ECC syndrome. A 7-bit syndrome is generated during write operation and stored as a 7-bit parity field along with the 32 data bits. Each data bit contributes to three parity bits according. Each parity bit represents the XOR value of 13 or 14 data bits according to the following mapping:

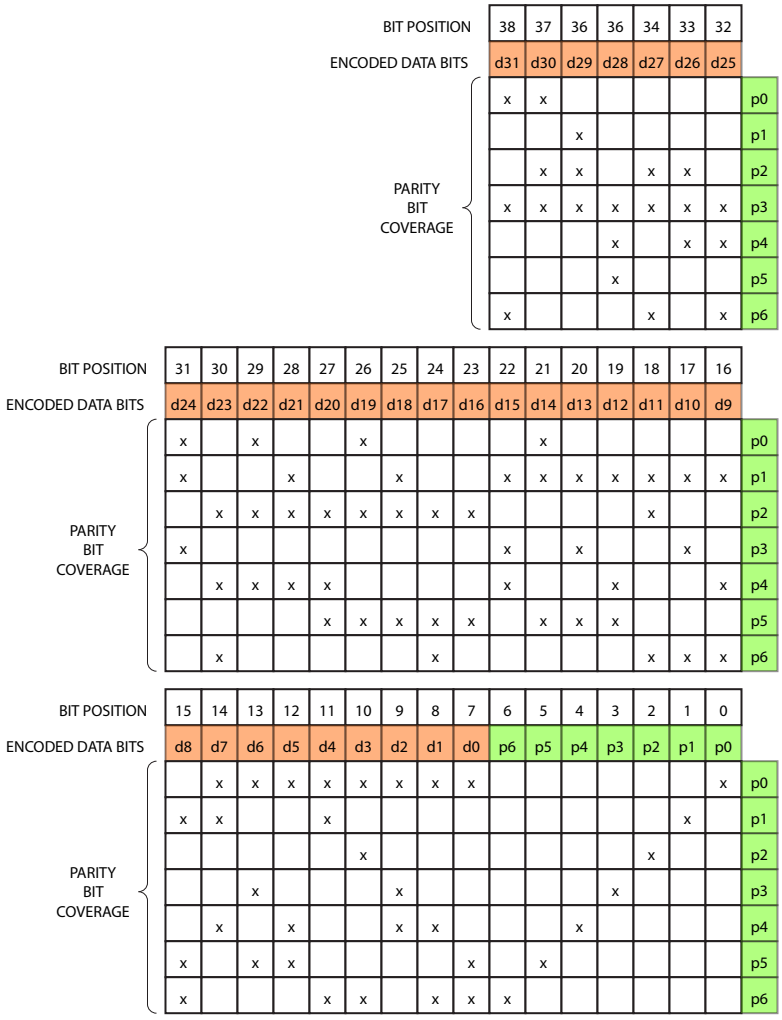


Figure 8-3: Hsaio Parity Bit Mapping

During read operation, the parity bits become part of the syndrome equation. The new syndrome bits are now the XOR values of the 13 or 14 data bits plus the respective stored parity bit. If any of the seven syndrome bits is set, an error situation is detected. An OR gate cross of the 7 bits reports the error, without specifying the type of the error.

If a single parity bit failed, the new 7-bit syndrome has 1 bit that is set. If a single data bit failed, the new syndrome has 3 bits that are set, because all three related parity bits fail. So, an XOR gate cross of all seven syndrome bits detects a single-bit error, indicating that an odd number of syndrome bits is set.

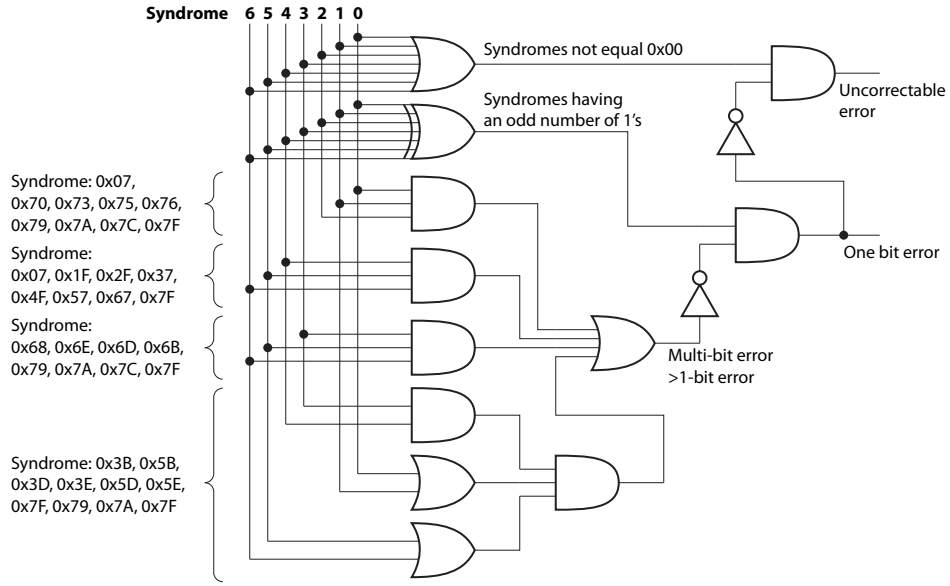


Figure 8-4: Hsiao Error Reports

The XOR gate detects single-bit errors and does not flag any dual-bit error. But the gate does flag 50% of the other multi-bit errors undesirably. Extra logic is implemented to increase the detection rate of multi-bit errors to 68.7% as shown in the figure.

If a single-bit error is detected, the failing bit can be localized and corrected. If all three syndrome bits corresponding to a specific data bit are 1, a data error is assumed. The respective data bit is toggled on its way to the system bus.

ECC Hardware Control

After reset, ECC protection is enabled. The boot code initializes all L2 SRAM data and checksum cells. ECC protection adds some cycle penalty when 8-bit and 16-bit values write L2 memory. Disable ECC protection for individual SRAM banks by setting the `L2CTL_CTL.BK0EDIS` through `L2CTL_CTL.BK7EDIS` disable bits. Due to caching mechanisms of the processor cores and data bursting of the DMA channels, 8-bit and 16-bit write accesses are uncommon. Typically, only two-dimensional DMA operations or un-cached 8-bit and 16-bit store instruction can trigger these writes.

ECC Error Management

The L2 system memory flags 2-bit and multi-bit errors to the system by:

- Raising the ECC_ERR interrupt
- Reporting a read error to the system bus
- Setting the sticky `L2CTL_STAT.ECCERR7-L2CTL_STAT.ECCERR0` status flag
- Latching the address of the failing operation into the respective `L2CTL_ERRADDR7-L2CTL_ERRADDR0` register.

There is one error status bit and one error address register per L2 SRAM bank.

Typically, ECC_ERR events are declared as system faults in the system event controller (SEC). Whether these faults are reported, the interrupt service routine can consult the L2CTL_STAT register and the L2CTL_ERRADDR0 through L2CTL_ERRADDR7 registers to determine whether:

- The data at the failing L2 address was critical enough to require an immediate reboot of the system
- The data at the failing L2 address was less critical or can be restored

The L2CTL_STAT.ECCERR0 through L2CTL_STAT.ECCERR7 flags are cleared with a W1C operation.

Memory Refresh

If data in L2 SRAM contains single-bit errors, the data is corrected on its way to the system buses. The corrected value is not written back to the SRAM location. To prevent any risk of accumulation of single-bit errors over time and to minimize likelihood of multi-bit errors, the L2 system memory provides a special memory refresh mechanism.

If there are dual-bit or multi-bit errors, the ECC_ERR interrupt is raised, and data is not written back to memory.

Program the memory refresh or scrub using:

- TheL2CTL_SADR register with the start address of the scrub
- The L2CTL_SCNT register with the total number of 128-bit addresses to be scrubbed starting from the L2CTL_SADR address

Program the number of cycles between each scrub using the L2CTL_SCTL register. During the scrub, the L2 system memory issues a read, followed by a write-back operation if there is a single-bit ECC error. Once the L2 system memory completes scrubbing the programmed memory region, it generates an interrupt and starts again from the L2CTL_SADR address unless the L2CTL_SCTL.SEN bit is disabled. If the L2CTL_SCTL.SEN bit is cleared before completing the programmed address range, the scrub stops after completing any already issued scrub access. The scrub read/writes always start from the full 128-bit equivalent of the address written into the L2CTL_SADR register. A 128-bit value is always read, and the 128-bit value is written back. The scrub access has the highest priority. Programs can configure 8-bit, 16-bit, or 32-bit addresses in the L2CTL_SADR register. The lower 4 bits of this register are treated as *do-not-care* values because the internal memory array is always accessed when using 128 bits.

Memory refresh operation is meaningless when the L2CTL_CTL.BK0EDIS through L2CTL_CTL.BK7EDIS disable bits are set.

Power Modes

Each L2 memory bank supports low-power modes.

If the bank is not in use, put each L2 system memory bank into the following low-power modes to save power.

Deep Sleep Mode

Set the L2CTL_PCTL.BK7DS through L2CTL_PCTL.BK0DS control bits to enter this mode. This mode preserves the memory contents.

Shut Down Mode

Set the L2CTL_PCTL.BK7SD through L2CTL_PCTL.BK0SD control bits to enter this mode. The memory banks do not retain the data (existing data is lost).

NOTE: It may take up to 11 SYSCLK cycles to deactivate the power mode after writing to the L2CTL_PCTL register. Therefore, L2 access must be issued after this delay.

Access to a memory bank in shut down or deep sleep mode may result in an unpredictable behavior. Ensure that initialization/refresh and memory access to such banks are not issued.

There is no deep sleep or shut down mode for ROM memories.

L2 System Memory Event Control

The following sections describe event control features of the L2 system memory, such as error response.

ECC Error Interrupt

A bus error is signaled under any of the following conditions.

- A write access to ROM address space
- A read/write access to reserved address space
- An ECC multi-bit error in an ECC-enabled bank. A non-modulo, 32-bit write to an ECC-enabled bank can also potentially create a bus error response due to an ECC multi-bit error. This response is because the L2 system memory implements a 32-bit ECC, and therefore a non-modulo, 32-bit write results in a read. This read can create multi-bit errors even if the memory was initialized.

Bus error notifications are stored in the L2CTL_STAT register, and the addresses that generated the error on a given port are stored in the L2CTL_EADDR0/L2CTL_EADDR1 register of that port. The details of the error are stored in the L2CTL_ET0/L2CTL_ET1 register of the port.

ADSP-SC59x L2CTL Register Descriptions

L2 Memory Controller (L2CTL) contains the following registers.

Table 8-6: ADSP-SC59x L2CTL Register List

Name	Description
L2CTL_CTL	Control Register
L2CTL_EADDR0	Error Type 0 Address Register
L2CTL_EADDR1	Error Type 1 Address Register
L2CTL_EADDR2	Error Type 2 Address Register
L2CTL_EADDR3	Error Type 3 Address Register

Table 8-6: ADSP-SC59x L2CTL Register List (Continued)

Name	Description
L2CTL_EADDR4	Error Type 4 Address Register
L2CTL_ERRADDR0	ECC Error Address 0 Register
L2CTL_ERRADDR1	ECC Error Address 1 Register
L2CTL_ERRADDR2	ECC Error Address 2 Register
L2CTL_ERRADDR3	ECC Error Address 3 Register
L2CTL_ERRADDR4	ECC Error Address 4 Register
L2CTL_ERRADDR5	ECC Error Address 5 Register
L2CTL_ERRADDR6	ECC Error Address 6 Register
L2CTL_ERRADDR7	ECC Error Address 7 Register
L2CTL_ERRADDR8	ECC Error Address 8 Register
L2CTL_ET0	Error Type 0 Register
L2CTL_ET1	Error Type 1 Register
L2CTL_ET2	Error Type 2 Register
L2CTL_ET3	Error Type 3 Register
L2CTL_ET4	Error Type 4 Register
L2CTL_INIT	Initialization Register
L2CTL_ISTAT	Initialization Status Register
L2CTL_PCTL	Power Control Register
L2CTL_REVID	Revision ID Register
L2CTL_RPCR0	Read Priority Count Register
L2CTL_RPCR1	Read Priority Count Register
L2CTL_SADR	Scrub Start Address Register
L2CTL_SCNT	Scrub Count Register
L2CTL_SCTL	Scrub Control Register
L2CTL_STAT	Status Register
L2CTL_STAT_1	L2 Port Error Status Register
L2CTL_WPCR0	Write Priority Count Register
L2CTL_WPCR1	Write Priority Count Register

Control Register

The L2CTL_CTL register includes a write protection bit, enables L2 banks, and selects mapping of banks (as ECC RAM or data RAM).

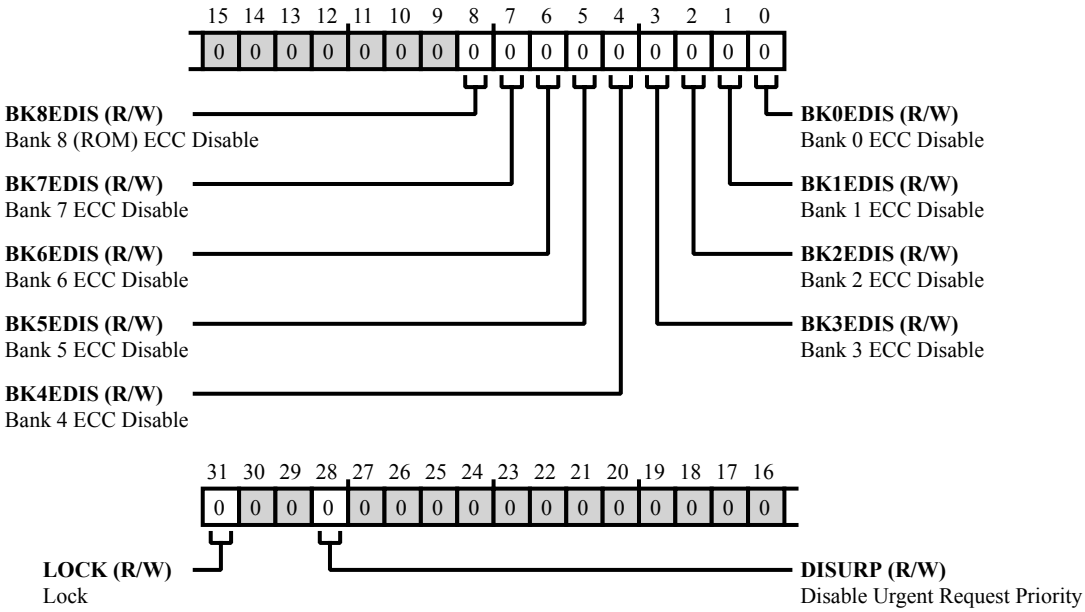


Figure 8-5: L2CTL_CTL Register Diagram

Table 8-7: L2CTL_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the L2CTL_CTL.LOCK bit is set, the L2CTL_CTL register is read only (locked).	
		0	Unlock
		1	Lock
28 (R/W)	DISURP	Disable Urgent Request Priority. The L2CTL_CTL.DISURP disables urgent request priority mode for all L2 banks.	
		0	Enable URP
		1	Disable URP
8 (R/W)	BK8EDIS	Bank 8 (ROM) ECC Disable. The L2CTL_CTL.BK8EDIS bit disables L2 bank 8 (ROM) ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC

Table 8-7: L2CTL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	BK7EDIS	Bank 7 ECC Disable. The `L2CTL_CTL.BK7EDIS` bit disables L2 bank 7 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
6 (R/W)	BK6EDIS	Bank 6 ECC Disable. The `L2CTL_CTL.BK6EDIS` bit disables L2 bank 6 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
5 (R/W)	BK5EDIS	Bank 5 ECC Disable. The `L2CTL_CTL.BK5EDIS` bit disables L2 bank 5 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
4 (R/W)	BK4EDIS	Bank 4 ECC Disable. The `L2CTL_CTL.BK4EDIS` bit disables L2 bank 4 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
3 (R/W)	BK3EDIS	Bank 3 ECC Disable. The `L2CTL_CTL.BK3EDIS` bit disables L2 bank 3 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
2 (R/W)	BK2EDIS	Bank 2 ECC Disable. The `L2CTL_CTL.BK2EDIS` bit disables L2 bank 2 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC

Table 8-7: L2CTL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	BK1EDIS	Bank 1 ECC Disable. The `L2CTL_CTL.BK1EDIS` bit disables L2 bank 1 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC
0 (R/W)	BK0EDIS	Bank 0 ECC Disable. The `L2CTL_CTL.BK0EDIS` bit disables L2 bank 0 ECC operation. ECC operation should be enabled or disable before the start of L2 bank access.	
		0	Enable ECC
		1	Disable ECC

Error Type 0 Address Register

The L2CTL_EADDR0 register holds the address that created an access error on the L2 port 0 bus interface. This register is be updated only if the corresponding error status bit (L2CTL_STAT.ERR0) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR0 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

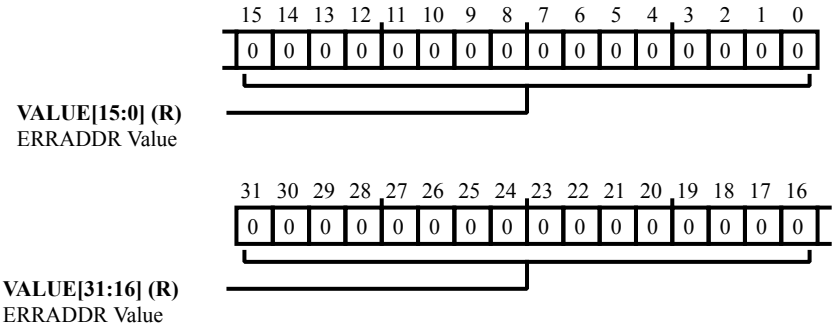


Figure 8-6: L2CTL_EADDR0 Register Diagram

Table 8-8: L2CTL_EADDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR0.VALUE bits hold the address causing the bus error.

Error Type 1 Address Register

The L2CTL_EADDR1 register holds the address that created an access error on the L2 port 1 bus interface. This register is be updated only if the corresponding error status bit (L2CTL_STAT.ERR1) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR1 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

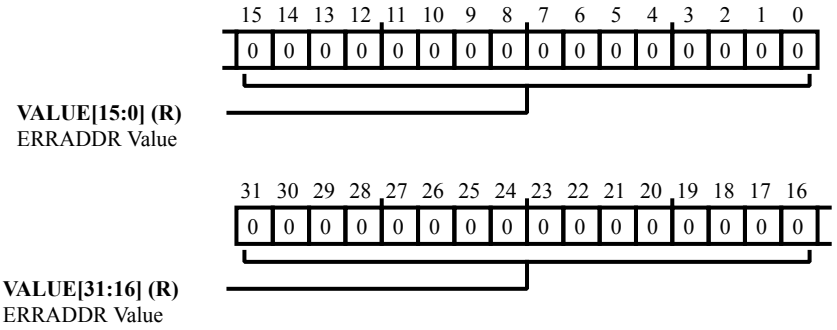


Figure 8-7: L2CTL_EADDR1 Register Diagram

Table 8-9: L2CTL_EADDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR1.VALUE bits hold the address causing the bus error.

Error Type 2 Address Register

The L2CTL_EADDR2 register holds the address that created an access error on the L2 port 2 bus interface (cores). This register is be updated only if the corresponding error status bit (L2CTL_STAT_1.ERR2) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR2 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

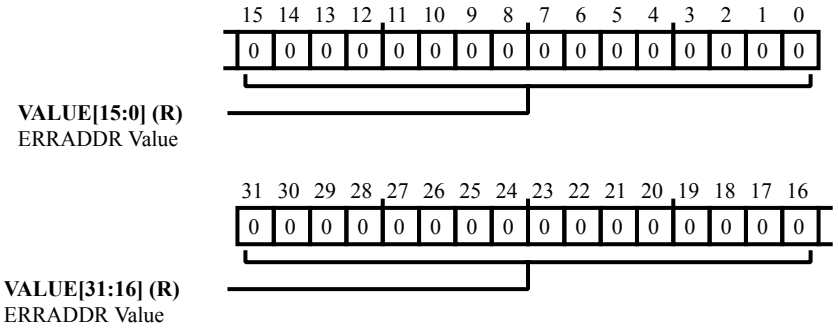


Figure 8-8: L2CTL_EADDR2 Register Diagram

Table 8-10: L2CTL_EADDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR2.VALUE bits hold the address causing the bus error.

Error Type 3 Address Register

The L2CTL_EADDR3 register holds the address that created an access error on the L2 port 3 bus interface (DMA). This register is be updated only if the corresponding error status bit (L2CTL_STAT_1.ERR3) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR3 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

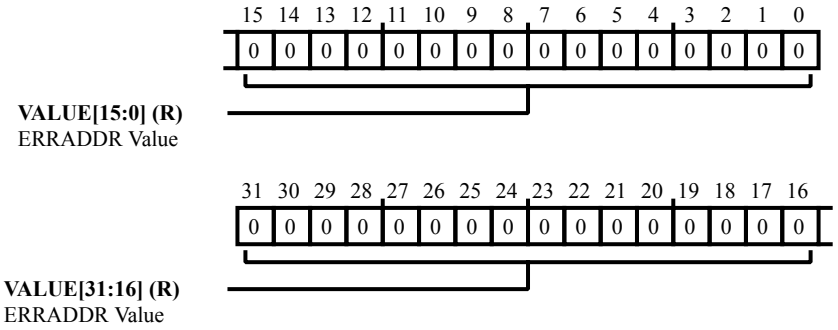


Figure 8-9: L2CTL_EADDR3 Register Diagram

Table 8-11: L2CTL_EADDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR3.VALUE bits hold the address causing the bus error.

Error Type 4 Address Register

The L2CTL_EADDR4 register holds the address that created an access error on the L2 port 4 bus interface (DMA). This register is be updated only if the corresponding error status bit (L2CTL_STAT_1.ERR4) is cleared. After the status bit is set for an error, further errors do not update the L2CTL_EADDR4 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

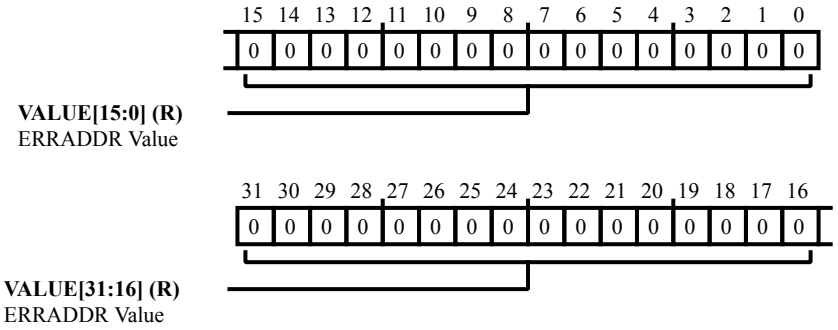


Figure 8-10: L2CTL_EADDR4 Register Diagram

Table 8-12: L2CTL_EADDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_EADDR4.VALUE bits hold the address causing the bus error.

ECC Error Address 0 Register

The L2CTL_ERRADDR0 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR0) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

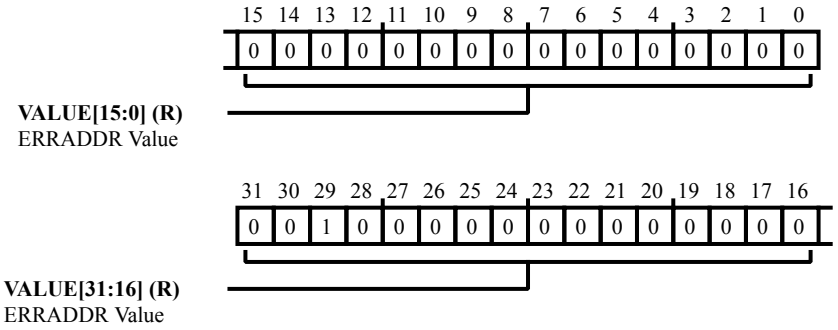


Figure 8-11: L2CTL_ERRADDR0 Register Diagram

Table 8-13: L2CTL_ERRADDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR0.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 1 Register

The L2CTL_ERRADDR1 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR1) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

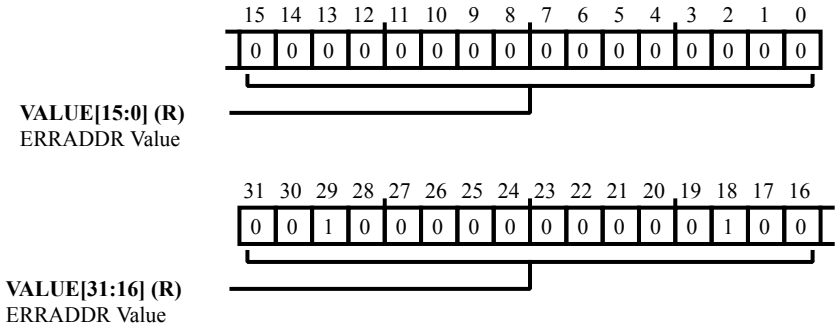


Figure 8-12: L2CTL_ERRADDR1 Register Diagram

Table 8-14: L2CTL_ERRADDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR1.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 2 Register

The L2CTL_ERRADDR2 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR2) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

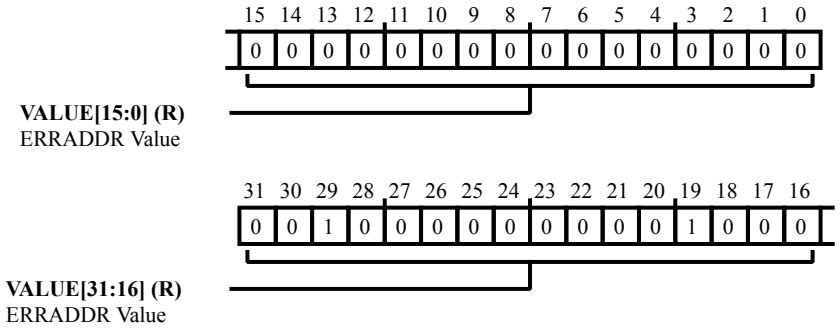


Figure 8-13: L2CTL_ERRADDR2 Register Diagram

Table 8-15: L2CTL_ERRADDR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR2.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 3 Register

The L2CTL_ERRADDR3 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR3) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

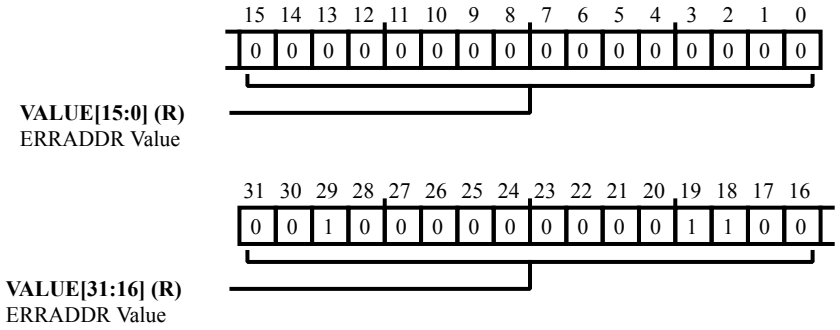


Figure 8-14: L2CTL_ERRADDR3 Register Diagram

Table 8-16: L2CTL_ERRADDR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR3.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 4 Register

The L2CTL_ERRADDR4 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR4) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

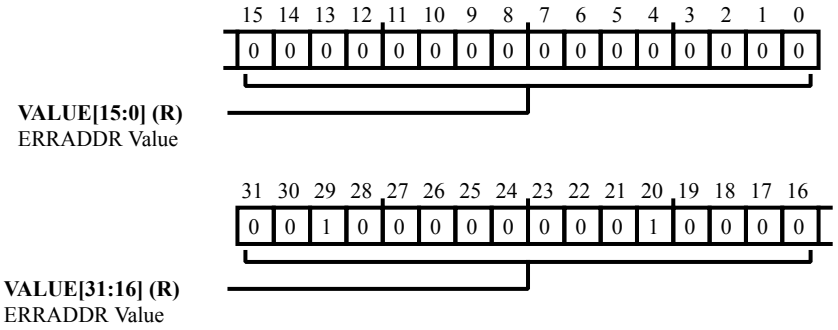


Figure 8-15: L2CTL_ERRADDR4 Register Diagram

Table 8-17: L2CTL_ERRADDR4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR4.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 5 Register

The L2CTL_ERRADDR5 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR5) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

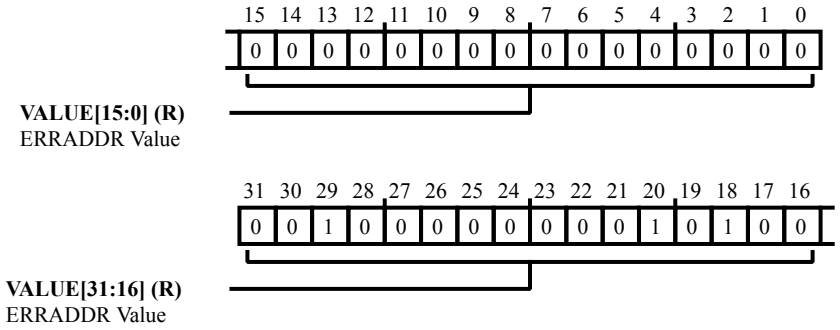


Figure 8-16: L2CTL_ERRADDR5 Register Diagram

Table 8-18: L2CTL_ERRADDR5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR5.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 6 Register

The L2CTL_ERRADDR6 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR6) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

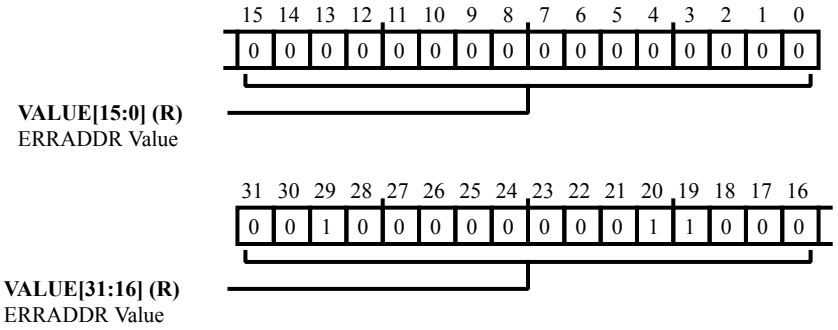


Figure 8-17: L2CTL_ERRADDR6 Register Diagram

Table 8-19: L2CTL_ERRADDR6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR6.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 7 Register

The L2CTL_ERRADDR7 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR7) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

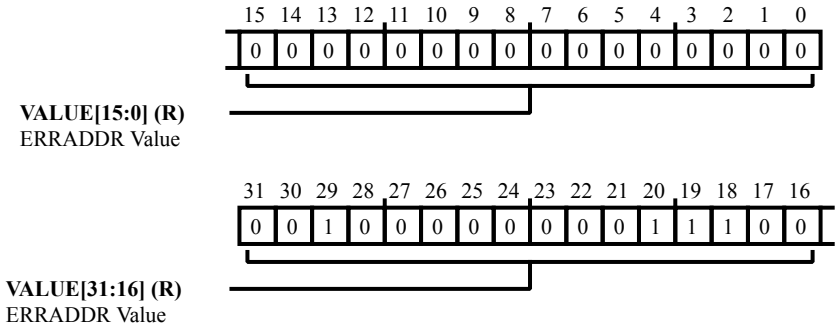


Figure 8-18: L2CTL_ERRADDR7 Register Diagram

Table 8-20: L2CTL_ERRADDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR7.VALUE bits hold the address containing the ECC double-bit error.

ECC Error Address 8 Register

The L2CTL_ERRADDR8 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL_STAT.ECCERR8) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

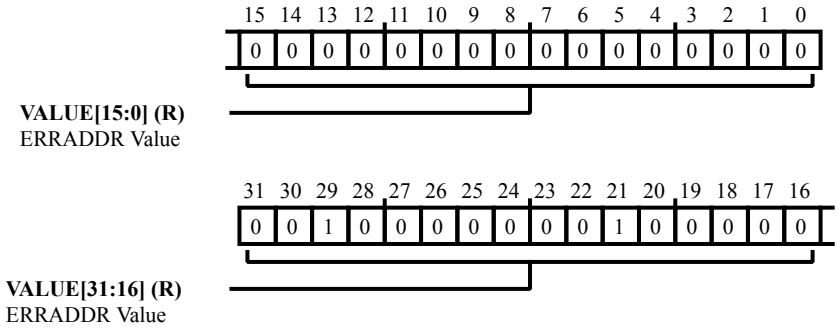


Figure 8-19: L2CTL_ERRADDR8 Register Diagram

Table 8-21: L2CTL_ERRADDR8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	ERRADDR Value. The L2CTL_ERRADDR8.VALUE bits hold the address containing the ECC double-bit error.

Error Type 0 Register

The L2CTL_ET0 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 0. This register is updated only if the corresponding error status bit L2CTL_STAT.ERR0 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET0 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET0 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR0).

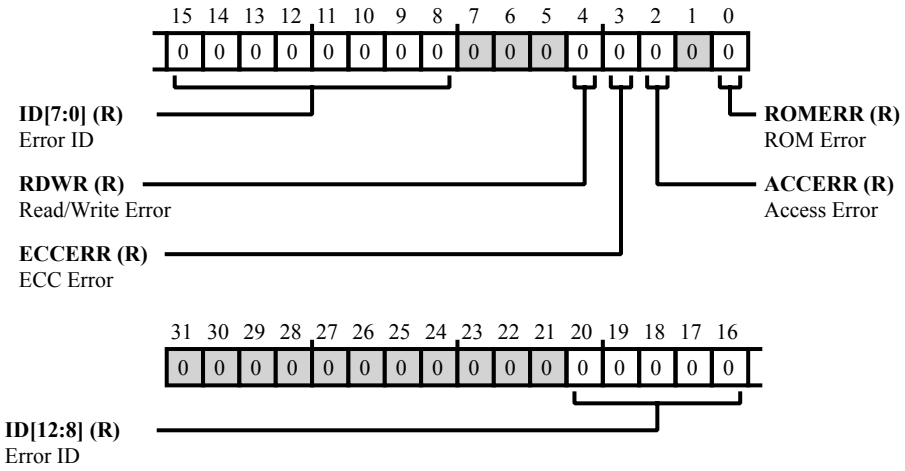


Figure 8-20: L2CTL_ET0 Register Diagram

Table 8-22: L2CTL_ET0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20:8 (R/NW)	ID	Error ID. The L2CTL_ET0.ID bits hold the bus requester ID of the access that caused an error.	
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET0.RDWR bit indicates whether a read or write access caused an error.	
		0	Read Access created Error
		1	Write Access created Error
3 (R/NW)	ECCERR	ECC Error. The L2CTL_ET0.ECCERR bit indicates whether the access had an ECC double-bit error.	
2 (R/NW)	ACCERR	Access Error. The L2CTL_ET0.ACCERR bit indicates whether the access went to a restricted bank.	

Table 8-22: L2CTL_ET0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	ROMERR	ROM Error. The `L2CTL_ET0.ROMERR` bit indicates whether a write access went to a ROM area.

Error Type 1 Register

The L2CTL_ET1 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 1. This register is updated only if the corresponding error status bit L2CTL_STAT.ERR1 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET1 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET1 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR1).

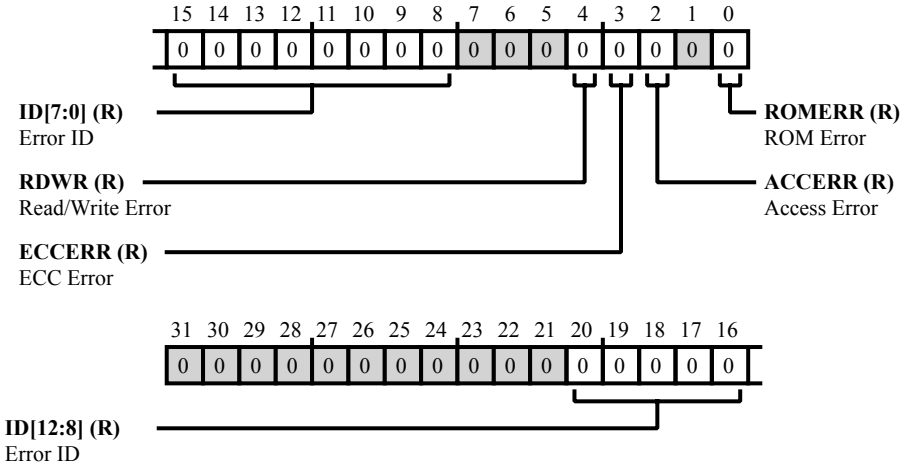


Figure 8-21: L2CTL_ET1 Register Diagram

Table 8-23: L2CTL_ET1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20:8 (R/NW)	ID	Error ID. The L2CTL_ET1.ID bits hold the bus requester ID of the access that caused an error.	
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET1.RDWR bit indicates whether a read or write access caused an error.	
		0	Read access created error
		1	Write access created error
3 (R/NW)	ECCERR	ECC Error. If the L2CTL_ET1.ECCERR bit =1, the access had an ECC double-bit error.	
2 (R/NW)	ACCERR	Access Error. If the L2CTL_ET1.ACCERR bit =1, the access went to a restricted bank.	
0 (R/NW)	ROMERR	ROM Error. If the L2CTL_ET1.ROMERR bit =1, a write access went to a ROM area.	

Error Type 2 Register

The L2CTL_ET2 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 2 (cores). This register is updated only if the corresponding error status bit L2CTL_STAT_1.ERR2 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET2 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET2 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR2).

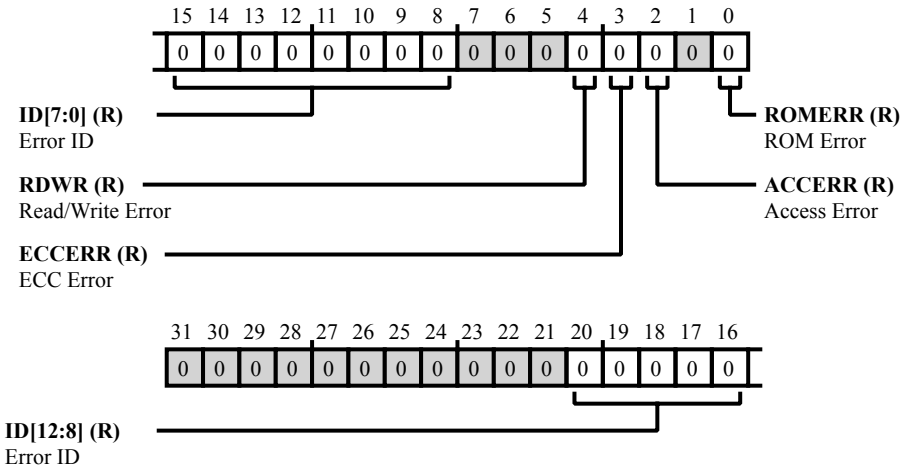


Figure 8-22: L2CTL_ET2 Register Diagram

Table 8-24: L2CTL_ET2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:8 (R/NW)	ID	Error ID. The L2CTL_ET2.ID bits hold the bus requester ID of the access that caused an error.
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET2.RDWR bit indicates whether a read or write access caused an error.
3 (R/NW)	ECCERR	ECC Error. The L2CTL_ET2.ECCERR bit indicates whether the access had an ECC double-bit error.
2 (R/NW)	ACCERR	Access Error. If the L2CTL_ET2.ACCERR bit =1, the access went to a restricted bank.
0 (R/NW)	ROMERR	ROM Error. If the L2CTL_ET2.ROMERR bit =1, a write access went to a ROM area.

Error Type 3 Register

The L2CTL_ET3 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 3 (DMA). This register is updated only if the corresponding error status bit L2CTL_STAT_1.ERR3 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET3 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET3 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR3).

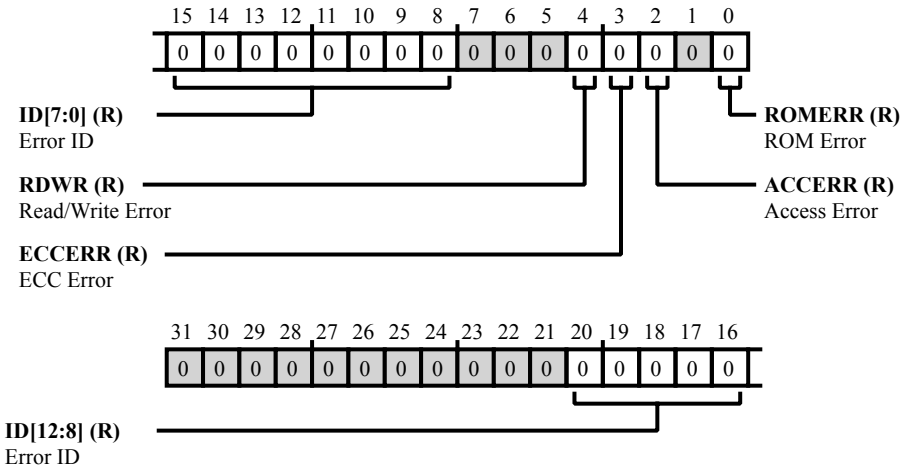


Figure 8-23: L2CTL_ET3 Register Diagram

Table 8-25: L2CTL_ET3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:8 (R/NW)	ID	Error ID. The L2CTL_ET3.ID bits hold the bus requester ID of the access that caused an error.
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET3.RDWR bit indicates whether a read or write access caused an error.
3 (R/NW)	ECCERR	ECC Error. The L2CTL_ET3.ECCERR bit indicates whether the access had an ECC double-bit error.
2 (R/NW)	ACCERR	Access Error. If the L2CTL_ET3.ACCERR bit =1, the access went to a restricted bank.
0 (R/NW)	ROMERR	ROM Error. If the L2CTL_ET3.ROMERR bit =1, a write access went to a ROM area.

Error Type 4 Register

The L2CTL_ET4 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port 4 (DMA). This register is updated only if the corresponding error status bit L2CTL_STAT_1.ERR4 is cleared. After the status bit is set for an error, further errors do not update the L2CTL_ET4 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL_ET4 captures the write access error, keeping in sync with the error address register (L2CTL_EADDR4).

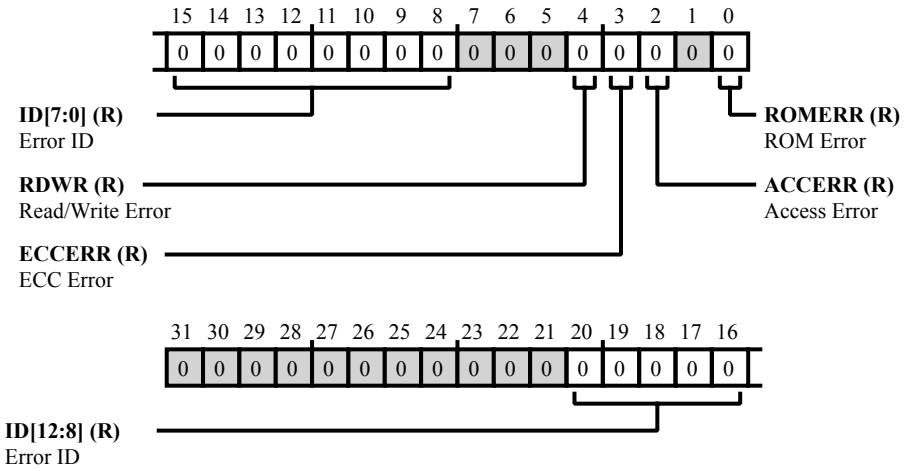


Figure 8-24: L2CTL_ET4 Register Diagram

Table 8-26: L2CTL_ET4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:8 (R/NW)	ID	Error ID. The L2CTL_ET4.ID bits hold the bus requester ID of the access that caused an error.
4 (R/NW)	RDWR	Read/Write Error. The L2CTL_ET4.RDWR bit indicates whether a read or write access caused an error.
3 (R/NW)	ECCERR	ECC Error. The L2CTL_ET4.ECCERR bit indicates whether the access had an ECC double-bit error.
2 (R/NW)	ACCERR	Access Error. If the L2CTL_ET4.ACCERR bit =1, the access went to a restricted bank.
0 (R/NW)	ROMERR	ROM Error. If the L2CTL_ET4.ROMERR bit =1, a write access went to a ROM area.

Initialization Register

The L2CTL_INIT register initializes memory banks with 128'b0 and ECC bits corresponding to 128'b0. Any writes to the bits in this register while initialization is occurring to any of the banks is ignored. All bits are W1A (Write 1 for Action).

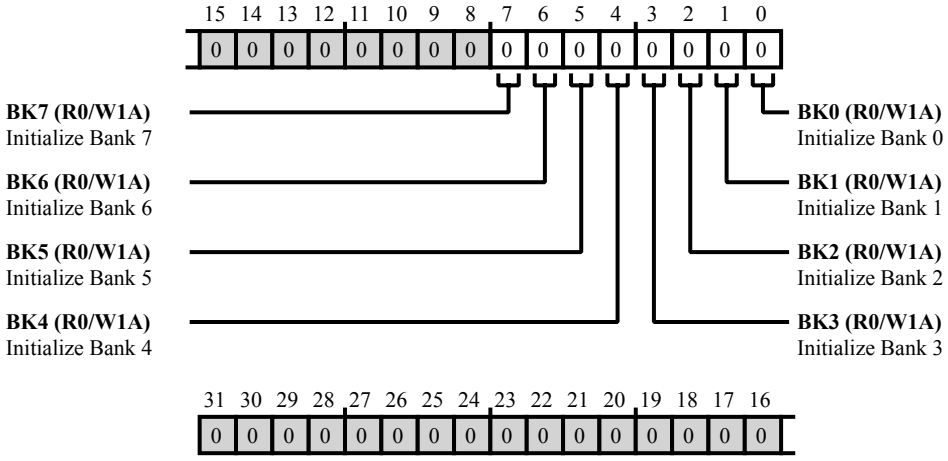


Figure 8-25: L2CTL_INIT Register Diagram

Table 8-27: L2CTL_INIT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R0/W1A)	BK7	Initialize Bank 7. The write-1 to the L2CTL_INIT.BK7 bit initializes bank 7 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
6 (R0/W1A)	BK6	Initialize Bank 6. The write-1 to the L2CTL_INIT.BK6 bit initializes bank 6 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
5 (R0/W1A)	BK5	Initialize Bank 5. The write -1 to the L2CTL_INIT.BK5 bit initializes bank 5 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
4 (R0/W1A)	BK4	Initialize Bank 4. The write-1 to the L2CTL_INIT.BK4 bit initializes bank 4 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.

Table 8-27: L2CTL_INIT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R0/W1A)	BK3	Initialize Bank 3. The write-1 to the `L2CTL_INIT.BK3` bit initializes bank 3 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
2 (R0/W1A)	BK2	Initialize Bank 2. The write-1 to the `L2CTL_INIT.BK2` bit initializes bank 2 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
1 (R0/W1A)	BK1	Initialize Bank 1. The write-1 to the `L2CTL_INIT.BK1` bit initializes bank 1 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.
0 (R0/W1A)	BK0	Initialize Bank 0. The write-1 to the `L2CTL_INIT.BK0` bit initializes bank 0 with 128b0 and ECC bits corresponding to 128b0. Any write to this bit during the initialization of the banks is ignored.

Initialization Status Register

The L2CTL_ISTAT register holds the status of the RAM bank initialization. If set, the corresponding bank is initialized.

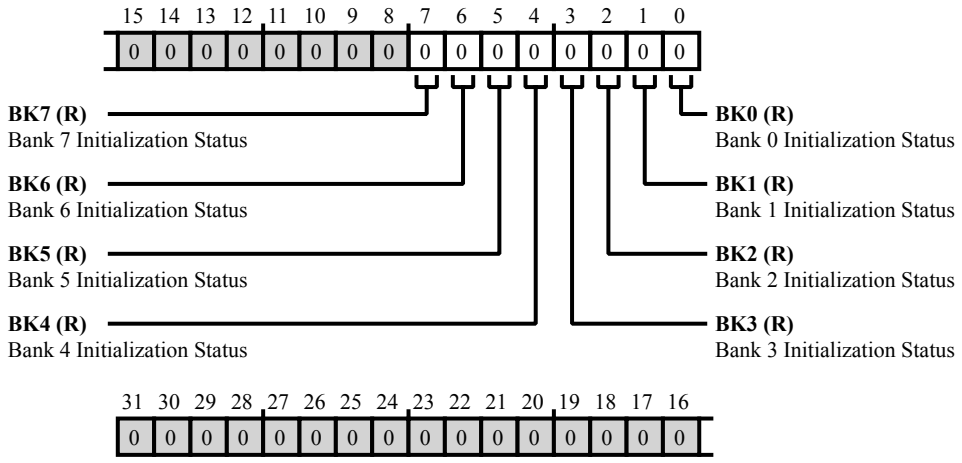


Figure 8-26: L2CTL_ISTAT Register Diagram

Table 8-28: L2CTL_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	BK7	Bank 7 Initialization Status. The L2CTL_ISTAT.BK7 bits hold the bank 7 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
6 (R/NW)	BK6	Bank 6 Initialization Status. The L2CTL_ISTAT.BK6 bits hold the bank 6 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
5 (R/NW)	BK5	Bank 5 Initialization Status. The L2CTL_ISTAT.BK5 bits hold the bank 5 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
4 (R/NW)	BK4	Bank 4 Initialization Status. The L2CTL_ISTAT.BK4 bits hold the bank 4 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
3 (R/NW)	BK3	Bank 3 Initialization Status. The L2CTL_ISTAT.BK3 bits hold the bank 3 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
2 (R/NW)	BK2	Bank 2 Initialization Status. The L2CTL_ISTAT.BK2 bits hold the bank 2 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.

Table 8-28: L2CTL_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	BK1	Bank 1 Initialization Status. The `L2CTL_ISTAT.BK1` bits hold the bank 1 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.
0 (R/NW)	BK0	Bank 0 Initialization Status. The `L2CTL_ISTAT.BK0` bits hold the bank 0 initialization status. A W1A on a BKxINIT bit clears this bit and the bit is set when initialization completes.

Power Control Register

The L2CTL_PCTL register has the various control settings for selectively enabling the deep sleep and shut down power saving features.

NOTE: The corresponding L2 bank should not be accessed if the power control feature is enabled for that bank. An access to a bank in deep sleep/shut down may create unpredictable behavior.

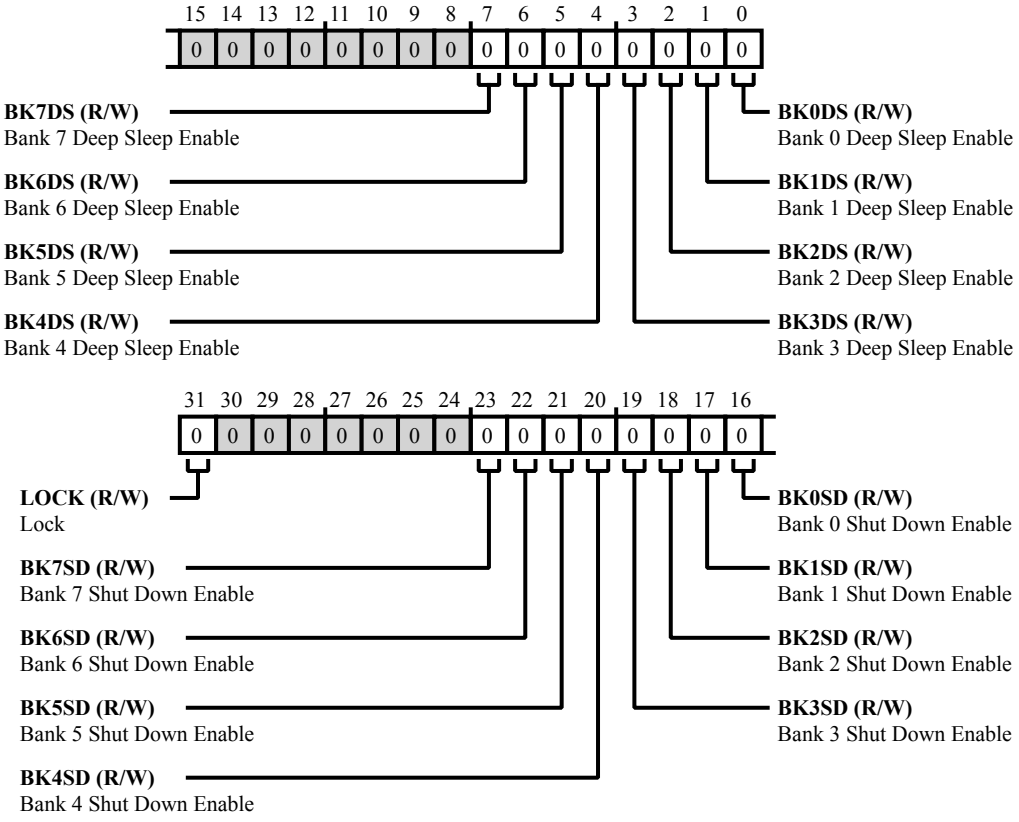


Figure 8-27: L2CTL_PCTL Register Diagram

Table 8-29: L2CTL_PCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the L2CTL_PCTL.LOCK bit is set, the L2CTL_PCTL register is read only (locked).	
		0	Unlock
		1	Lock
23 (R/W)	BK7SD	Bank 7 Shut Down Enable. The L2CTL_PCTL.BK7SD bits enables bank 7 shut down.	

Table 8-29: L2CTL_PCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
22 (R/W)	BK6SD	Bank 6 Shut Down Enable. The L2CTL_PCTL.BK6SD bits enables bank 6 shut down.
21 (R/W)	BK5SD	Bank 5 Shut Down Enable. The L2CTL_PCTL.BK5SD bits enables bank 5 shut down.
20 (R/W)	BK4SD	Bank 4 Shut Down Enable. The L2CTL_PCTL.BK4SD bits enables bank 4 shut down.
19 (R/W)	BK3SD	Bank 3 Shut Down Enable. The L2CTL_PCTL.BK3SD bits enables bank 3 shut down.
18 (R/W)	BK2SD	Bank 2 Shut Down Enable. The L2CTL_PCTL.BK2SD bits enables bank 2 shut down.
17 (R/W)	BK1SD	Bank 1 Shut Down Enable. The L2CTL_PCTL.BK1SD bits enables bank 1 shut down.
16 (R/W)	BK0SD	Bank 0 Shut Down Enable. The L2CTL_PCTL.BK0SD bits enables bank 0 shut down.
7 (R/W)	BK7DS	Bank 7 Deep Sleep Enable. The L2CTL_PCTL.BK7DS bits enables bank 7 deep sleep.
6 (R/W)	BK6DS	Bank 6 Deep Sleep Enable. The L2CTL_PCTL.BK6DS bits enables bank 6 deep sleep.
5 (R/W)	BK5DS	Bank 5 Deep Sleep Enable. The L2CTL_PCTL.BK5DS bits enables bank 5 deep sleep.
4 (R/W)	BK4DS	Bank 4 Deep Sleep Enable. The L2CTL_PCTL.BK4DS bits enables bank 4 deep sleep.
3 (R/W)	BK3DS	Bank 3 Deep Sleep Enable. The L2CTL_PCTL.BK3DS bits enables bank 3 deep sleep.
2 (R/W)	BK2DS	Bank 2 Deep Sleep Enable. The L2CTL_PCTL.BK2DS bits enables bank 2 deep sleep.
1 (R/W)	BK1DS	Bank 1 Deep Sleep Enable. The L2CTL_PCTL.BK1DS bits enables bank 1 deep sleep.
0 (R/W)	BK0DS	Bank 0 Deep Sleep Enable. The L2CTL_PCTL.BK0DS bits enables bank 0 deep sleep.

Revision ID Register

The L2CTL_REVID register provides the L2 Revision ID.

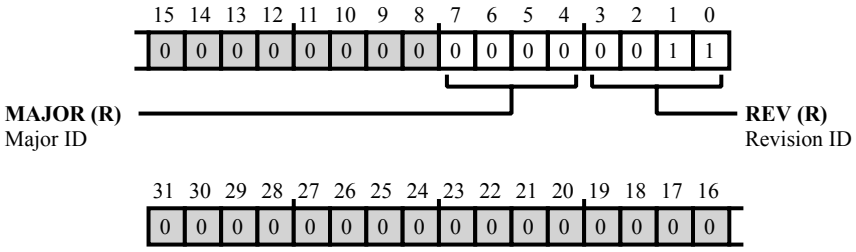


Figure 8-28: L2CTL_REVID Register Diagram

Table 8-30: L2CTL_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major ID. The L2CTL_REVID.MAJOR bit indicates L2 Major ID.
3:0 (R/NW)	REV	Revision ID. The L2CTL_REVID.REV bit indicates the L2 Revision ID.

Read Priority Count Register

The L2CTL_RPCR0 register stores the count value to be used for priority elevation for bus read channels of Port0 to Port3. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number (stored in RPCx) of SYSCLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request. Default value wait cycle for each channel is 15..

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus read channel. A best practice is to program this register before initiating an L2 access.

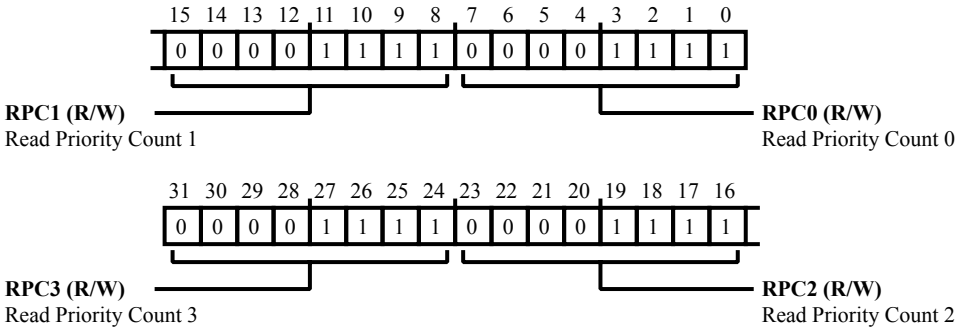


Figure 8-29: L2CTL_RPCR0 Register Diagram

Table 8-31: L2CTL_RPCR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RPC3	Read Priority Count 3. The L2CTL_RPCR0.RPC3 bits hold the priority count for L2 bus read channel 3.
23:16 (R/W)	RPC2	Read Priority Count 2. The L2CTL_RPCR0.RPC2 bits hold the priority count for L2 bus read channel 2.
15:8 (R/W)	RPC1	Read Priority Count 1. The L2CTL_RPCR0.RPC1 bits hold the priority count for L2 bus read channel 1.
7:0 (R/W)	RPC0	Read Priority Count 0. The L2CTL_RPCR0.RPC0 bits hold the priority count for L2 bus read channel 0.

Read Priority Count Register

The L2CTL_RPCR1 register stores the count value to be used for priority elevation for bus read channel 4. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number (stored in RPC4) of SYSCLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request. Default value of RPC4 is 15.

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus read channel. A best practice is to program this register before initiating an L2 access.

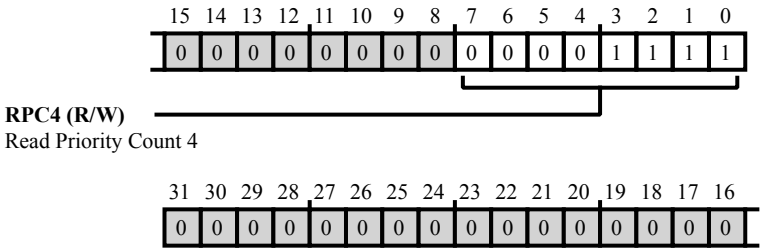


Figure 8-30: L2CTL_RPCR1 Register Diagram

Table 8-32: L2CTL_RPCR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RPC4	Read Priority Count 4. The L2CTL_RPCR1.RPC4 bits hold the priority count for L2 bus read channel 4.

Scrub Start Address Register

The L2CTL_SADR register stores the scrub start address value. Writes to this register can be prevented by setting the L2CTL_SCTL.LOCK bit and the global lock.

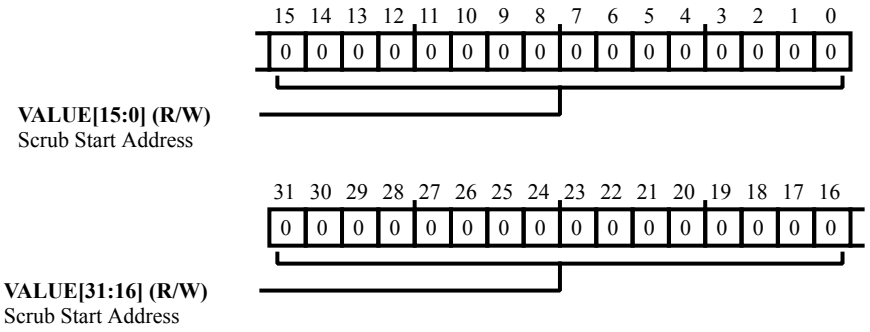


Figure 8-31: L2CTL_SADR Register Diagram

Table 8-33: L2CTL_SADR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Scrub Start Address. The L2CTL_SADR.VALUE bits hold the scrub start address. The writes to this register can be prevented by setting the L2_SCTL.LOCK and the global lock.

Scrub Count Register

The L2CTL_SCNT register determines the number of 128-bit locations scrubbed starting from the start address (L2CTL_SADR register). Writes to the L2CTL_SCNT register can be prevented by setting the L2CTL_CTL.LOCK bit and the global lock.

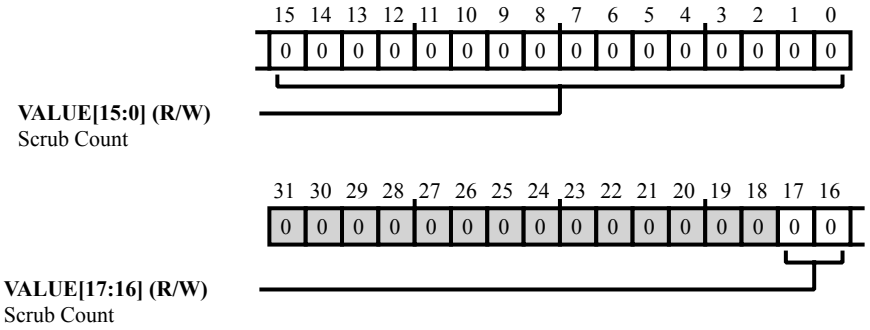


Figure 8-32: L2CTL_SCNT Register Diagram

Table 8-34: L2CTL_SCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17:0 (R/W)	VALUE	Scrub Count. The L2CTL_SCNT.VALUE bits determines the number of 128-bit locations scrubbed starting from the start address. Ensure value programmed is less than the total addressable 128-bit L2 RAM locations available.

Scrub Control Register

The L2CTL_SCTL register holds the automatic scrub related controls. This is a read/write register. Memory scrub can be performed by programming the L2CTL_SADR (scrub start address) register with the start address of the scrub and the L2CTL_SCNT register with the total number of 128 -bit addresses to be scrubbed starting from the L2CTL_SADR register. The number of cycles between each scrub can be programmed with the L2CTL_SCTL.SRT bit.

During the scrub the controller issues a read followed by write back if there is a single bit ECC error. Once the L2 controller completes scrubbing the memory region mentioned using the L2CTL_SADR and the L2CTL_SCNT registers, an interrupt is generated and scrubbing re-starts from the start unless the scrub enable bit is disabled in the control register. If scrub enable is cleared before completing the address range used in the L2CTL_SADR and L2CTL_SCNT registers, the scrub stops after completing any already issued scrub access.

The scrub read/writes always start from the full 128-bit equivalent of the address written into the L2CTL_SADR register. A 128-bit value is always read and the 128-bit value is written back. The scrub access has the highest priority. Programs can configure 8-, 16-, or 32-bit addresses in the L2CTL_SADR register but the lower 3 bits are treated as don't care because the internal memory array is always accessed in a 128-bit mode.

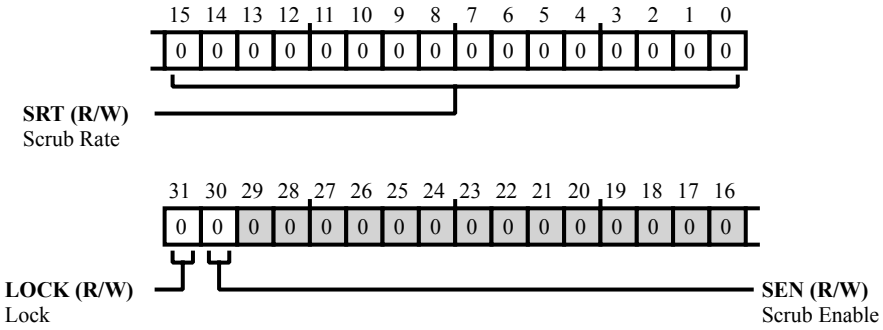


Figure 8-33: L2CTL_SCTL Register Diagram

Table 8-35: L2CTL_SCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the L2CTL_SCTL.LOCK bit is set, the L2CTL_SCTL register is read only (locked).	
		0	Unlock
		1	Lock
30 (R/W)	SEN	Scrub Enable. The L2CTL_SCTL.SEN bits enable automatic scrub.	

Table 8-35: L2CTL_SCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	SRT	Scrub Rate. The `L2CTL_SCTL.SRT` bits determines the number of clock cycles that elapsed between two automatic scrubs.

Status Register

The L2CTL_STAT register indicates ECC error status, refresh register status, and bus error status of Port0 and Port1.

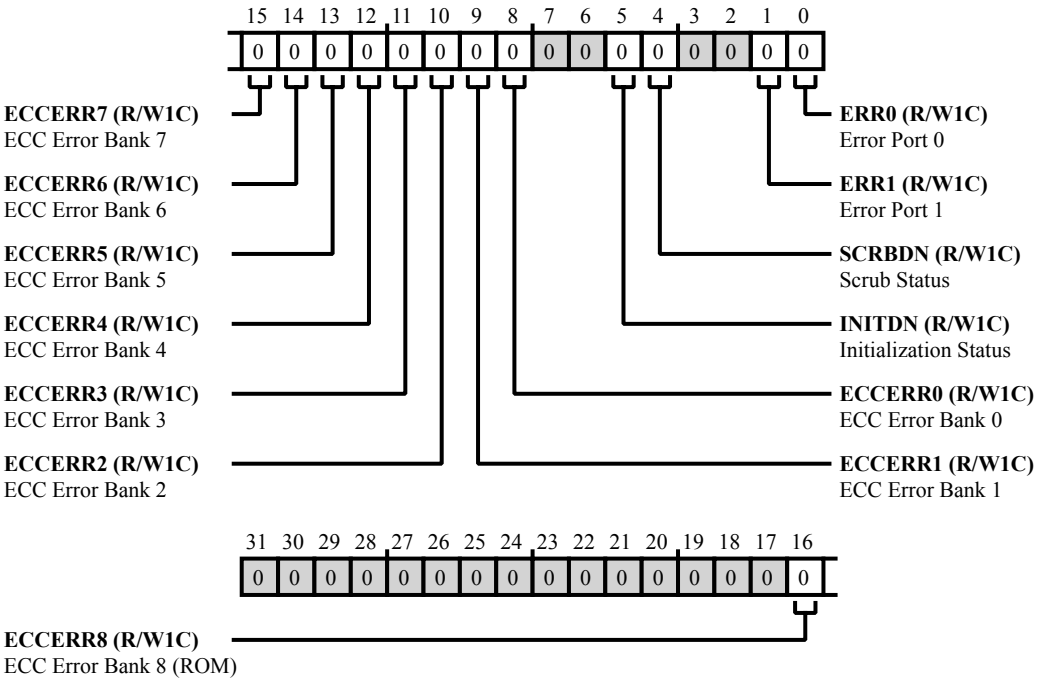


Figure 8-34: L2CTL_STAT Register Diagram

Table 8-36: L2CTL_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W1C)	ECCERR8	ECC Error Bank 8 (ROM). The L2CTL_STAT.ECCERR8 bit indicates that an ECC double-bit error occurred inside L2 bank 8 (ROM).	
		0	No Status
		1	ECC Double Bit Error
15 (R/W1C)	ECCERR7	ECC Error Bank 7. The L2CTL_STAT.ECCERR7 bit indicates that an ECC double-bit error occurred inside L2 bank 7.	
		0	No Status
		1	ECC Double Bit Error

Table 8-36: L2CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W1C)	ECCERR6	ECC Error Bank 6. The `L2CTL_STAT.ECCERR6` bit indicates that an ECC double-bit error occurred inside L2 bank 6.	
		0	No Status
		1	ECC Double Bit Error
13 (R/W1C)	ECCERR5	ECC Error Bank 5. The `L2CTL_STAT.ECCERR5` bit indicates that an ECC double-bit error occurred inside L2 bank 5.	
		0	No Status
		1	ECC Double Bit Error
12 (R/W1C)	ECCERR4	ECC Error Bank 4. The `L2CTL_STAT.ECCERR4` bit indicates that an ECC double-bit error occurred inside L2 bank 4.	
		0	No Status
		1	ECC Double Bit Error
11 (R/W1C)	ECCERR3	ECC Error Bank 3. The `L2CTL_STAT.ECCERR3` bit indicates that an ECC double-bit error occurred inside L2 bank 3.	
		0	No Status
		1	ECC Double Bit Error
10 (R/W1C)	ECCERR2	ECC Error Bank 2. The `L2CTL_STAT.ECCERR2` bit indicates that an ECC double-bit error occurred inside L2 bank 2.	
		0	No Status
		1	ECC Double Bit Error
9 (R/W1C)	ECCERR1	ECC Error Bank 1. The `L2CTL_STAT.ECCERR1` bit indicates that an ECC double-bit error occurred inside L2 bank 1.	
		0	No Status
		1	ECC Double Bit Error

Table 8-36: L2CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	ECCERR0	ECC Error Bank 0. The `L2CTL_STAT.ECCERR0` bit indicates that an ECC double-bit error occurred inside L2 bank 0.	
		0	No Status
		1	ECC Double Bit Error
5 (R/W1C)	INITDN	Initialization Status. The `L2CTL_STAT.INITDN` bit indicates whether initialization has been completed.	
		0	Initialization Not Complete
		1	Initialization Completed
4 (R/W1C)	SCRBDN	Scrub Status. The `L2CTL_STAT.SCRBDN` bit indicates whether a round of memory scrub has completed.	
		0	Scrub Not Complete
		1	Scrub Completed
1 (R/W1C)	ERR1	Error Port 1. The `L2CTL_STAT.ERR1` indicates whether the L2CTL has detected a bus access error on L2s bus port 1.	
		0	No Error
		1	Bus Access Error
0 (R/W1C)	ERR0	Error Port 0. The `L2CTL_STAT.ERR0` indicates whether the L2CTL has detected a bus access error on L2s bus port 0.	
		0	No Error
		1	Bus Access Error

L2 Port Error Status Register

The L2CTL_STAT_1 register stores the AXI Error Status from Port2 onwards. The register bits are sticky - once set it has to be explicitly reset by writing a 1 to that particular bit.

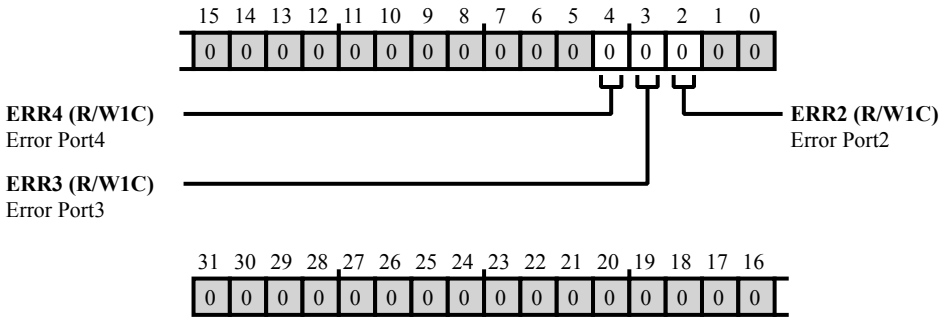


Figure 8-35: L2CTL_STAT_1 Register Diagram

Table 8-37: L2CTL_STAT_1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W1C)	ERR4	Error Port4. The L2CTL_STAT_1.ERR4 indicates whether the L2CTL has detected a bus access error on L2s bus port 4.
3 (R/W1C)	ERR3	Error Port3. The L2CTL_STAT_1.ERR3 indicates whether the L2CTL has detected a bus access error on L2s bus port 3.
2 (R/W1C)	ERR2	Error Port2. The L2CTL_STAT_1.ERR2 indicates whether the L2CTL has detected a bus access error on L2s bus port 2.

Write Priority Count Register

The L2CTL_WPCR0 register stores the count value to be used for priority elevation for bus write channels. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number of SYSCLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request.

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus write channel. A best practice is to program this register before initiating an L2 access.

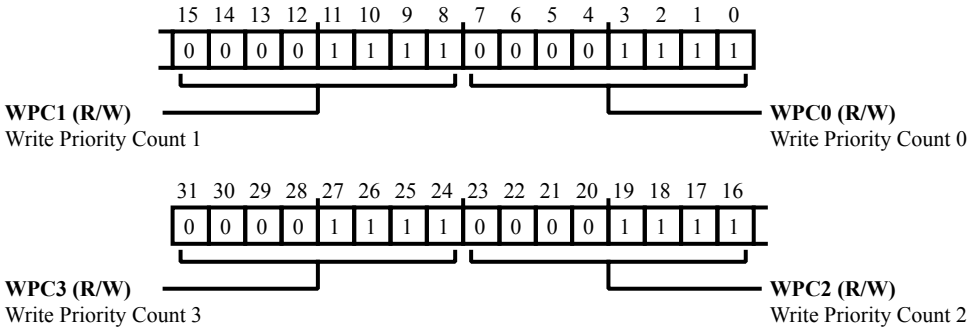


Figure 8-36: L2CTL_WPCR0 Register Diagram

Table 8-38: L2CTL_WPCR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	WPC3	Write Priority Count 3. The L2CTL_WPCR0.WPC3 bits hold the priority count for L2 bus write channel 3.
23:16 (R/W)	WPC2	Write Priority Count 2. The L2CTL_WPCR0.WPC2 bits hold the priority count for L2 bus write channel 2.
15:8 (R/W)	WPC1	Write Priority Count 1. The L2CTL_WPCR0.WPC1 bits hold the priority count for L2 bus write channel 1.
7:0 (R/W)	WPC0	Write Priority Count 0. The L2CTL_WPCR0.WPC0 bits hold the priority count for L2 bus write channel 0.

Write Priority Count Register

The L2CTL_WPCR1 register stores the count value to be used for priority elevation for bus write channels. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number of SYSCLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request.

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus write channel. A best practice is to program this register before initiating an L2 access.

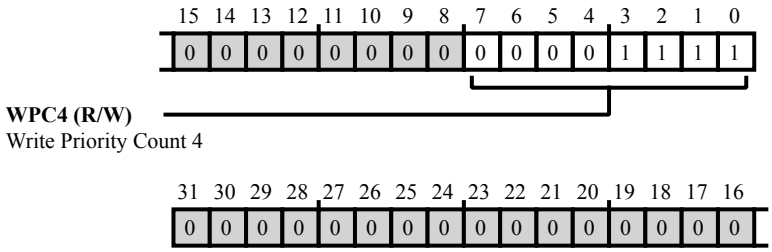


Figure 8-37: L2CTL_WPCR1 Register Diagram

Table 8-39: L2CTL_WPCR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	WPC4	Write Priority Count 4. The L2CTL_WPCR1.WPC4 bits hold the priority count for L2 bus write channel 4.

9 Dynamic Memory Controller (DMC)

The dynamic memory controller (DMC) provides a glueless interface between DDR3 SDRAMs and the system crossbar interface (SCB). The DMC enables execution of instructions from, as well as transfer of data to and from, DDR3 SDRAM respectively.

NOTE: The term DDR3 is referred to generically as DDR SDRAM in the rest of this chapter unless otherwise noted.

The DMC is partitioned in a manner that allows reconfiguration and maintainability. The memory access protocol state machine along with JEDEC standard specific logic is embedded in the *protocol controller*. An access and operation reordering mechanism is incorporated as an *efficiency controller*. An SCB bus responder interface is provided to interface with the on-chip interconnect. This interface results in an efficient completer implementation owing to its out-of-order transaction capabilities. The control and status registers present in the DMC can be accessed using the MMR access bus.

The DMC supports access to the external memory by core and DMA accesses.

DMC Features

The DMC includes a protocol controller that supports:

- JESD79-3E compatible double data rate DDR3 SDRAM devices

The features of the dynamic memory controller are:

- Provides 16-bit data only interface to SDRAM devices
- Supports a single external rank (one chip select)
- Provides page hit detection that supports multiple column accesses to the same row
- User-specified active, precharge, and refresh commands.
- Programmable SDRAM access timing parameters
- Enables automatic refresh generation with programmable refresh intervals
- Self-refresh mode to reduce system power consumption

- Efficient transaction processing to improve throughput and bandwidth using:
  - Software programmable SCB IDs to allow SCB ID-based priority
  - The ability to postpone up to eight auto-refresh commands
  - Software selectable closed page scheme on a per bank basis
  - Simple transaction scheduling mechanism to reduce read write turnaround frequency on the memory bus
  - Accesses with the same SCB ID are scheduled back-to-back to take advantage of same page access in SDRAM
  - Caching of SDRAM read data burst for specific controllers to reduce the latency for same burst accesses.

The DDR3 features are:

- 512 Mb to 8 Gb device sizes
- Burst length BL = 8
- Support for additive latency
- Support for programmable (and ZQ calibration) ODT and drive impedance
- Support for read leveling
- Support for write leveling

Feature Exclusions

The DMC exclusions are as follows:

For DDR3:

- 4-bit and 8-bit wide DDR3 DRAM memories are not supported
- Burst interleaved accesses are not supported
- Both burst chop-BC4 and BC4-on-the-fly are not supported
- Auto refresh pull-in is not supported
- DLL off mode is not supported

DMC Functional Description

The dynamic memory controller consists of controller and target interfaces, a protocol controller, and an efficiency controller. The following sections describe the function of these interfaces and controllers.

ADSP-SC59x DMC Register List

The Dynamic Memory Controller module (DMC) provides an interface to external double-data-rate SDRAM. This interface supports various DDR standards (see chapter descriptions). A set of registers governs DMC controller operations. For more information on DMC controller functionality, see the DMC Controller Register Descriptions.

Table 9-1: ADSP-SC59x DMC Register List

Name	Description
DMC_CFG	Configuration Register
DMC_CTL	Control Register
DMC_DLLCTL	DLL Control Register
DMC_DT_CALIB_ADDR	Data Calibration Address Register
DMC_DT_DATA_CALIB_DATA0	Data Calibration Data 0 Register
DMC_DT_DATA_CALIB_DATA1	Data Calibration Data 1 Register
DMC_EFFCTL	Efficiency Control Register
DMC_EMR3	Shadow EMR3 Register
DMC_MR	Shadow MR0 Register (DDR3)
DMC_MR1	Shadow MR1 Register (DDR3)
DMC_MR2	Shadow MR2 Register (DDR3)
DMC_MSK	Mask (Mode Register Shadow) Register
DMC_PM_CTL	DMC Performance Control Register
DMC_PM_IDCYDQ_CNT	Performance Monitor Idle Cycle DQ Count Register
DMC_PM_IDPG_CNT	Performance Monitor Idle Page Count Register
DMC_PM_PGHT_CNT	Performance Monitor Page Hit Count
DMC_PM_PGMS_CNT	Performance Monitor Page Miss Count
DMC_PM_RD2WRTA_CNT	Performance Monitor Read to Write Turn Around Count Register
DMC_PM_REF_CNT	Performance Monitor Refresh Count Register
DMC_PM_WR2RDTA_CNT	Performance Monitor Write to Read Turn Around Count Register
DMC_PRIO	Priority ID Register 1
DMC_PRIO2	Priority ID Register 2
DMC_PRIOMSK	Priority ID Mask Register 1
DMC_PRIOMSK2	Priority ID Mask Register 2
DMC_RDDATABUFID1	DMC Read Data Buffer ID Register 1
DMC_RDDATABUFID2	DMC Read Data Buffer ID Register 2
DMC_RDDATABUFMSK1	DMC Read Data Buffer Mask Register 1

Table 9-1: ADSP-SC59x DMC Register List (Continued)

Name	Description
DMC_RDDATABUFMSK2	DMC Read Data Buffer Mask Register 2
DMC_RDD_BUFHT_CNT	Performance Monitor Buffer Hit Count Register
DMC_SCB_PM_BRLN_CNT	SCB Performance AXI Transfer Count Register
DMC_SCB_PM_BRSZ_CNT	SCB Performance AXI Transfer Count Register
DMC_SCB_PM_IDRDDCY_CNT	SCB Performance Monitor Idle Cycle Count Register
DMC_SCB_PM_IDWRDCY_CNT	SCB Performance Monitor Idle Cycle Count Register
DMC_STAT	Status Register
DMC_TR0	Timing 0 Register
DMC_TR1	Timing 1 Register
DMC_TR2	Timing 2 Register

ADSP-SC59x DMC Register List

DMCPHY (DMC) contains the following registers.

Table 9-2: ADSP-SC59x DMC Register List

Name	Description
DMC_DDR_CA_CTL	DDR CA Lane Control Register
DMC_DDR_LANE0_CTL0	Data Lane 0 Control Register 0
DMC_DDR_LANE0_CTL1	Data Lane 0 Control Register 1
DMC_DDR_LANE1_CTL0	Data Lane 1 Control Register 0
DMC_DDR_LANE1_CTL1	Data Lane 1 Control Register 1
DMC_DDR_ROOT_CTL	DDR ROOT Module Control Register
DMC_DDR_SCRATCH_2	Scratch Register 2
DMC_DDR_SCRATCH_3	Scratch Register 3
DMC_DDR_SCRATCH_4	Scratch Register 4
DMC_DDR_SCRATCH_5	Scratch Register 5
DMC_DDR_SCRATCH_6	Scratch Register 6
DMC_DDR_SCRATCH_7	Scratch Register 7
DMC_DDR_ZQ_CTL0	DDR Calibration Control Register 0
DMC_DDR_ZQ_CTL1	DDR Calibration Control Register 1
DMC_DDR_ZQ_CTL2	DDR Calibration Control Register 2

Protocol Controller

The DDR SDRAM protocol controller translates memory access requests from the SCB (system crossbar) interface to JEDEC protocol-specific transactions used by DDR SDRAM devices.

The protocol controller ensures that the various timing parameters are met before reading and writing the SDRAM. The controller also performs the SDRAM initialization sequence as mandated by the standard. The protocol controller can:

- Issue reads and writes
- Precharge a row in a bank
- Activate a row in a bank
- Put the SDRAM devices in self-refresh and power-down modes

The protocol controller takes mode register writes from the MMR interface and translates them into mode register writes to SDRAM. Writing into the mode register is restricted through a mask register.

Efficiency Controller

The efficiency controller controls the ordering of transfers buffered in the read and write command buffers. It attempts to order transfers to optimize the available memory bandwidth. The DMC uses a number of schemes, described in the following sections, to increase the throughput.

Page-Based Scheduling

The DMC parses each write and read transaction that it buffered and gets the information of the row (page) and bank address. The protocol controller maintains the information about the pages that are opened in each bank. The efficiency controller uses the information about the opened pages while scheduling the buffered transactions. The transactions to the opened pages are given higher priority than the other outstanding transactions.

Same Controller Transaction Scheduling

The DMC also stores the ID of each transaction that it buffered. In most of the cases, the transactions related to a controller result in page hits from the locality of reference rule. The efficiency controller uses the ID information of the transactions while scheduling. When the page-based scheduling of the buffered transactions is complete, same controller transaction scheduling is triggered. If multiple transactions from a controller are received, the efficiency controller schedules the transactions back-to-back.

DMC Read Data Buffer

The DMC read data buffer contains a data buffer and an address buffer. The depth of the data buffer is equal to the burst length that is programmed in SDRAM. The address buffer holds the corresponding SDRAM burst address. When an SDRAM write address from any controller matches an address in the DMC read data buffer, the DMC invalidates the related data in the read buffer. When the DMC_RDDATABUFMSK1 or DMC_RDDATABUFMSK2 register is programmed with a value other than zero, the DMC read data buffer operation is enabled. The set of controllers whose data is buffered and retrieved are programmed in the DMC_RDDATABUFID1 or DMC_RDDATABUFID2 registers. The DMC can use the DMC_RDDATABUFMSK1 and DMC_RDDATABUFMSK2

ID registers to select a set of controllers similar to the programming of the DMC_PRIOMSK and DMC_PRIOMSK2 registers.

See the SCB ID-Based Priority section for details.

Closed Page Per Bank

The DMC_EFFCTL register provides per-bank granularity for closing pages. The software can determine that most accesses to a given bank in memory always result in a missed page. In this case, set the PREC_BANK bit corresponding to the required bank to close the row after every transfer. This proactive step can result in reduced thrashing and increases memory throughput.

SCB ID-Based Priority

The primary goal of the dynamic memory controller is to improve sustainable memory system bandwidth so that the service time for the average request can be reduced. However, to service critical requests from any controller in the system, the DMC provides a mechanism to elevate priority of a given access. The DMC priority ID registers (DMC_PRIO and DMC_PRIO2) can be programmed with up to two SCB IDs with elevated priority.

After every access in a snapshot, the command buffers are searched to determine whether an ID of a command matches with the ID programmed in the DMC_PRIO and DMC_PRIO2 registers. The priority SCB ID access is sent before the subsequent access in the snapshot if:

- A match occurs, and
- The direction of the access (for example write) is the same as the direction of the snapshot (write)

There is an alternative to providing priority to a specific SCB ID. If a number of IDs from the same controller require priority, program the DMC priority mask ID registers (DMC_PRIOMSK and DMC_PRIOMSK2) so that the corresponding bits are 0. The DMC uses a combination of the DMC_PRIO and DMC_PRIO2 registers and the DMC_PRIOMSK/DMC_PRIOMSK2 registers to elevate the priority of a select few or all IDs that belong to a controller. By default, none of the IDs are prioritized. The following are a few possibilities:

- The DMC_PRIOMSK field is set to 0x00000000. If a single ID (7234) needs priority, set the DMC_PRIOMSK field to 0xFFFFFFFF and set the DMC_PRIO field to 7234.
- If the DMC_PRIOMSK field is set to 0xFFFFFFFE, the SCB IDs 7234 and 7235 are given priority.
- If the DMC_PRIOMSK field is set to 0xFFFFFFFC, the SCB IDs 7234, 7235, 7236, and 7237 are given priority.
- If two transactions with priority, one read and the other a write, are outstanding, the priority transaction that does not change the direction of the DMC access gets priority. The other priority transaction is handled at the beginning of the next snapshot. For example, if a write snapshot is in-progress, the write priority transaction is sent. The read priority transaction is sent at the beginning of the next read snapshot.

NOTE: Use SCB ID-based priority judiciously because it can significantly reduce the throughput of the DMC.

Delaying up to Eight Auto-Refresh Commands

The DMC uses this method to ensure that auto-refresh does not interfere with any critical data transfers. Up to eight auto-refresh commands can accumulate in the DMC. The exact number of auto-refresh commands can be programmed using the DMC_EFFCTL.NUMREF bit.

After the first refresh command is accumulated, the DMC constantly looks for an opportunity to schedule a refresh command. When the SCB read and write command buffers become empty for the programmed number of clock cycles (DMC_EFFCTL.IDLECYC bit field), the accumulated number of refresh commands are sent back-to-back to the DRAM. (The empty state of the SCB command buffers implies that no access is outstanding.)

After every refresh, the SCB command buffers are checked to ensure that they stay empty. If the SCB command buffers are always full, once the programmed number of refresh commands accumulates, the refresh operation is elevated to urgent priority. One refresh command is sent immediately. After this process, the DMC continues to wait for an opportunity to send out refresh commands. If self-refresh mode is enabled, all pending refresh commands are given out only after that DMC enters self-refresh mode.

Page and Bank Interleaving

Page and bank interleaving allow consecutive row accesses to fall into the same bank (bank interleaving) or into a different bank (page interleaving). The DMC uses bank interleaving by default (DMC_CTL.ADDRMODE bit =0). If the DMC_CTL.ADDRMODE bit =1, the DMC uses page interleaving. Page misses in one addressing mode result in hits in the other addressing mode.

System Crossbar Completer Interface

The DMC uses the system crossbar completer interface to move all data. The system crossbar interface accepts interleaved write transactions and sends out-of-order responses. The read and write interfaces consist of buffers for address, data, and control information transferred to or from the system crossbar bus.

The system crossbar interface transactions are sent to the SDRAM only after the SDRAM has been initialized. However, if transactions arrive before or during initialization, they accumulate in the system crossbar interface and are sent out to the protocol controller once the initialization completes.

To increase throughput, the system crossbar write-response is sent out as soon as the final DDR burst is scheduled for transfer into the SDRAM. However, if an auto-refresh is needed, the scheduled write data is sent only after the auto-refresh. A delay can occur. The delay is a maximum of 64 clock cycles from the moment the write response is sent on the SCB to the write operation of the data into SDRAM.

The system crossbar interface performs the following operations:

- Buffers read and write command requests from the system crossbar bus
- Processes the requests by converting them to protocol controller user-interface transfers
- Sends and receives data to or from the protocol controller
- Creates a suitable read/write response and sends read data back to the system crossbar bus

The system crossbar completer interface supports the following:

- All burst lengths (1–16)
- Incremental and wrap bursts
- Data transfer sizes of 8-bit, 16-bit, or 32-bit
- Arrival of write data before write address
- Generation of error responses which include:
  - Any access to an unimplemented region of the external memory space
  - Any access when the SDRAM is in self-refresh mode/power-down mode
  - Any access when the direct command interface is in operation

Read/Write Command and Data Buffers

The system crossbar interface consists of a four-deep read command buffer and a four-deep write command buffer. Up to four write commands and four read commands can be waiting for access to the SDRAM. The system crossbar write buffer is 32 deep. It can support write data interleaving of two. The system crossbar read buffer is 32 deep.

Peripheral Bus Completer Interface

The peripheral bus completer interface connects the dynamic memory controller to the peripheral bus and provides a host controller with access to the registers. The peripheral bus completer interface supports the following features:

- Read and write word accesses
- 32-bit data bus

Architectural Concepts

The following sections provide information on the architecture of the interface.

Controller On Die Termination (ODT)

The controller ODT is enabled with the granularity of a byte lane. The description of this feature can be obtained in the description of the corresponding PHY registers. Controller ODT involves extra overhead in terms of power consumption during reads.

The DMC implements dynamic on die termination at processor pads. When controller ODT is enabled, the termination resistors in the pads are turned on when the controller reads data from the DRAM. These resistors are turned off when the controller writes to the DRAM.

Mode Register Set and Extended Mode Register Set Command

The load mode register command initializes the SDRAM operation parameters. The DMC supports the mode register set and extended mode register set commands. The controller automatically issues the mode register set command during power-on initialization and also when the DMC_MR register is written with the DMC_MSK.MR bit. The mode register set command is sent after the ongoing data transfer completes.

DDR3 Reset Functionality

DDR3 contains an additional pin corresponding to reset functionality. Reset is part of the initialization sequence, but it can be performed asynchronously when needed. The reset procedure is similar to the steps involved in the initialization except the initial part of power-up.

To perform reset on the DDR3 module:

1. Check to ensure the module is in the idle state by polling the `DMC_STAT.IDLE` bit (0x0008).
2. Set the `DMC_CTL.RESET` bit (0x0004).
3. Monitor the `DMC_STAT.RESETDONE` bit for the completion of the reset function.

Do not perform any transactions during a module reset. Wait for the `DMC_STAT.RESETDONE` signal.

DDR3 SDRAM Organization

The DMC supports DDR3 SDRAM memory modules ranging from 512 Mb to 8 Gb. The following tables list the address translation mechanism from the user interface to DDR3 memory interface. The controller also supports two types of addressing modes: bank interleaving (`DMC_CTL.ADDRMODE` =1) and page interleaving (`DMC_CTL.ADDRMODE` =0).

Bank Interleaving

The ***DDR3 Bank Interleaving*** table shows DDR3 bank interleaving.

Table 9-3: DDR3 Bank Interleaving

SDRAM size	Bank address bits	Row address bits	Column address bits
512 Mb	25:24	23:11	10:1
1 Gb	26:24	23:11	10:1
2 Gb	27:25	24:11	10:1
8 Gb	29:27	26:11	10:1

Page Interleaving

The ***DDR3 Page Interleaving*** table shows DDR3 page interleaving.

Table 9-4: DDR3 Page Interleaving

SDRAM size	Row address bits	Bank address bits	Column address bits
512 Mb	25:13	12:11	10:1
1 Gb	26:14	13:11	10:1
2 Gb	27:14	13:11	10:1
4 Gb	28:14	13:11	10.1
8 Gb	29:14	13:11	10.1

DMC Clocking

The DMC uses a divided-down version of the *PLLCLK* (PLL clock) to generate an internal clock for clocking the DMC block and interface. The specific value of the *DCLK* frequency is programmed in the CGU_DIV register.

For information on the maximum clock frequency supported for specific modes, refer to the processor data sheet.

NOTE: For details on DMC clocking, see the Clock Generation Unit (CGU) and Clock Generation Unit (CGU) chapters.

DMC DMA

The DMC supports DMA-based transfers to and from external DDR SDRAM memory and internal memory.

The DMC DMA controller, part of the distributed DMA engines (DDE) that are dispersed through the infrastructure, connects to the system crossbar fabric.

The DMC uses two DDEs for memory-to-memory DMA (MDMA). One channel is the source channel, and the second, the destination channel.

DMA transfers on the processor are descriptor-based or register-based. Register-based DMA allows the processor to program DMA control registers directly to initiate a DMA transfer. On completion, the control registers can be automatically updated with their original setup values for continuous transfer, if needed. Descriptor-based DMA transfers require a set of parameters stored within memory to initiate a DMA sequence. This transfer allows the chaining together of multiple DMA sequences. In descriptor-based DMA operations, a DMA channel can be programmed to set up and start another DMA transfer automatically after the current sequence completes.

Enhanced DMA operations (such as delay line DMA, scatter or gather DMA) are also supported to or from the DMC module.

DMC Operating Modes

DDR3 Mode

The DMC module supports JESD79-3E compatible double data rate DDR3 SDRAM. To configure this mode of operation, first set (=1) the DMC_CTL.DDR3EN bit

Self-Refresh Mode

For low-power consumption, the SDRAM can be put in self-refresh mode. When no data activity occurs, the DMC can put the SDRAM in self refresh to save power. The DMC_STAT.IDLE bit indicates the activity on the DMC. If this bit is set, there is no activity in the DMC.

Enable self-refresh mode by writing the DMC_CTL.SRREQ bit. The DMC stays in a self-refresh state when this bit is asserted. The DMC_STAT.SRACK bit indicates when the SDRAM enters self-refresh mode.

When the DMC is in self-refresh mode, the DMC generates an SCB error when any data accesses (read or write requests) is requested.

The DMC can be brought out of self-refresh mode by clearing the DMC_CTL.SRREQ bit again. The controller clears the DMC_STAT.SRACK bit after the self-refresh operation completes.

DMC Event Control

The DMC has no related interrupt or trigger event information.

DMC Programming Model

The dynamic memory controller contains five groups of memory-mapped registers. The DMC uses the MMR access bus to connect to these registers.

- Control and status registers. These registers control the various operation modes of the dynamic memory controller and provide status.
- Timing parameter registers. The value programmed in these registers depends on the speed grade of the SDRAM device used.
- Mode register mirror registers. These shadow registers are copies of the mode registers residing in the SDRAM device.
- PHY control and status registers. The DMC uses these registers to control the operation of the PHY.
- PAD control registers. The DMC uses these registers to control the various aspects of the I/O pads.

The DMC control registers contain sensitive timing parameters and settings for the DDR SDRAM. These registers are programmed with values that are in the operating range of the DDR used.

Writing to reserved fields or writing any reserved values in register bits can cause the dynamic memory controller to function erroneously.

PHY DLL Calibration

The PHY DLL calibration is performed as part of the SDRAM power-up initialization. It calibrates data against the DQS and CLK signal. However, running DLL calibration after self-refresh or at an arbitrary time is required in certain cases.

The DMC allows PHY DLL calibration to start by setting the DMC_CTL.DLLCAL bit. The DMC_STAT.DLLCALDONE bit can be used to monitor the progress of the calibration. Once calibration is over, this bit is set. Once the calibration procedure is started by writing to the DMC_CAL_PADCTL0.CALSTRT bit, the full calibration takes 300 DCLK cycles to complete.

NOTE: DLL calibration can be initiated only when the DMC is idle (DMC_STAT.IDLE= 1).

DDR3 ZQ Calibration Short CMD

The ZQ calibration short command is generally used to correct small variations in ZQ (~0.5%). To perform ZQ calibration, the controller is first checked for its idle state. Once the idle bit is obtained, a ZQCS command can be

issued by setting the DMC_STAT.ZQCSDONE bit (0x0004). The DMC_STAT.ZQCSDONE bit (0x0008) can be used to monitor the calibration sequence. When this bit is 0, it indicates that the calibration is ongoing. When it is 1, it indicates that the calibration is done. As an example, a GP timer can be used to periodically trigger a ZQCS command to address tiny variations.

NOTE: The ZQ calibration function is essential for normal operation of DDR3. With the reference of the external resistance (240 Ω ± 1%) connected to the DMC_RZQ pin, DDR3 calibrates the R_{on} and R_{tt} values of the ZQ pin against temperature and voltage variations.

DDR3 ZQ Calibration Long CMD

Several DDR3 impedance calibrations are implemented for optimal signal integrity. The long ZQ calibration is used after power-up and the short ZQ calibration is used periodically during normal operation to compensate for voltage and temperature drift. These calibration sequences improve connectivity between the SDRAM pads and the PCB trace. The DMC_RZQ pin on the SDRAM is connected to an external precision resistor that adjusts the output driver impedance Rtt and ODT values to match the trace impedance. The connection reduces impedance discontinuity and minimizes signal reflections.

The command has two variants named as ZQ calibration long (ZQCL) and ZQ calibration short (ZQCS). The ZQCL command is issued during initialization and after self-refresh exit command. It can be issued later depending on the system environment.

The DMC pads can be auto calibrated to the required driver impedance Rtt using an external resistance RZQ and the On Die Termination (ODT) value using the corresponding bits (DMC_CAL_PADCTL2). The autocalibration logic translates these values into a corresponding drive strength control inside the PHY and then routed to the PADS. Autocalibration starts as soon as the bit is programmed (set the DCLK at the required frequency before setting this DMC_CAL_PADCTL0.CALSTRT bit). Autocalibration expects the program to select two different member sets of pads (address/command pads versus CLK/Data/DQS/DM pads).

On Die Termination (DDR3)

The DMC supports dynamic On Die Termination (ODT) at the pads. When the controller ODT is set, the termination resistors in the pads are turned on when the controller reads data from the DRAM. These resistors are turned off when the controller is writing to the DRAM. Controller ODT is enabled with the granularity of a byte lane. The description of this feature can be obtained in the description of the corresponding PHY registers.

DDR3 SDRAM provides extended ODT mode.

- *Asynchronous ODT*: ODT timing in the slow exit power-down mode
- *Dynamic ODT*: Function that can dynamically switch the ODT resistance during a write operation without an MRS command. It improves signal quality during a write operation.

Leveling Techniques

The DMC controller supports read and write leveling.

Write Leveling

It is difficult to meet the specifications with the on-board parasitic, trace length variability along with high frequency. The DMC controller supports leveling to compensate the skews between clock and strobe. This skew compensation is done independently for every byte lane. Write leveling is performed by the DDR3 memory controller as a part of initialization by programming the `DMC_MR1.WL` bit.

To dynamically perform write leveling:

1. Ensure that the DMC is idle.
2. Program the `DMC_MR1.WL` bit.
3. Enable the `DMC_MSK` register for the `DMC_MR1` register.
4. Wait for 2000 DMC clock cycles to complete write leveling.

Read Leveling

Read leveling is useful for calibrating read data timing. It is compensated for imbalanced loading on read path. Read leveling can be performed on LSB or byte lane. It is performed as a part of initialization sequence by setting the `DMC_EMR3.MPR` bit.

To dynamically perform read leveling:

1. Ensure that the DMC is idle.
2. Program the `DMC_EMR3.MPR` bit.
3. Enable DMC mask register `DMC_EMR3` register.
4. Wait for 2000 DMC clock cycles to complete read leveling.

Read Leveling during DQS Strobe Gating

To ensure robust read timing, read leveling must be done individually for all data bits. Some DDR3 memory devices do not drive read leveling patterns on all data pins. The devices send pattern only information on the LSBs of each byte lane. In these cases, the `DMC_EMR3` register cannot be used for read leveling. Instead, use the `DMC_CTL.RL_DQS` bit for read leveling.

Read leveling during DQS strobe gating is dynamically supported except during initialization.

Initializing the DMC

Complete the following procedures to initialize the DMC module.

- Resetting the DMC Lane
- Performing ZQ Calibration
- Programming the DMC Controller

Resetting the DMC Lane

If there is a change in the DMC clock frequency, the DDR lane must be reset using the following procedure. Once the lane is reset, wait 9000 DDR clock cycles for the DDR DLL to lock before restarting the DDR.

1. Set the DMC_DDR_LANE0_CTL0.CB_RSTDLL and DMC_DDR_LANE1_CTL0.CB_RSTDLL bits.
2. Change the DMC clock frequency.
3. Clear the DMC_DDR_LANE0_CTL0.CB_RSTDLL and DMC_DDR_LANE1_CTL0.CB_RSTDLL bits.

Performing ZQ Calibration

Perform the following ZQ calibration procedure for the proper impedance matching.

1. Program the DMC_DDR_ZQ_CTL0.IMPWRADD bit field with the drive strength of the address and command signals. A 100Ω drive strength is recommended for the drive strength of address and command signals. When programming a drive strength of 100Ω, write 0x64 into this bit field.
2. Program the DMC_DDR_ZQ_CTL0.IMPWRDQ bit field with the drive strength of the DQ, DQS, DM and clock signals. A 100Ω drive strength is recommended for the drive strength of address and command signals. When programming a drive strength of 100Ω, write 0x64 into this bit field.
3. Program the DMC_DDR_ZQ_CTL0.IMPRTT bit field with the adjusted On Die Termination (ODT) for the Data and DQS signals for the read operation. Due to a correction factor, program this field to 80% of the equivalent ODT.

   DMC_DDR_ZQ_CTL0.IMPRTT value = ODT*2*0.8

   For example, if a 50Ω terminating resistance is required on the data pads to match the trace impedance to the board impedance, there will be two 50Ω resistance data pads in parallel. The value is programmed to 100 × 0.8 = 80.
4. Write 0 to the DMC_DDR_ZQ_CTL1 register.
5. Program 0x70000000 to the DMC_DDR_ZQ_CTL2 register.
6. Program 0x0000000 to the DMC_DDR_CA_CTL register.
7. Program 0x0000000 to the DMC_DDR_ROOT_CTL register.
8. Program 0x00010000 to the DMC_DDR_ROOT_CTL register.
9. Wait for 8000 DMC clock cycles.
10. Program 0x0C000001 to the DMC_DDR_CA_CTL register.
11. Wait for 8000 DMC clock cycles.
12. Program 0x0000000 to the DMC_DDR_CA_CTL register.
13. Program 0x0000000 to the DMC_DDR_ROOT_CTL register.

Programming Duty Cycles

Perform the following procedure to program the DQS duty cycle and clock duty cycle trim.

1. Program the DMC_DDR_LANE0_CTL0 and DMC_DDR_LANE1_CTL0 registers with DMC_DDR_LANE0_CTL0.BYPCODE/DMC_DDR_LANE0_CTL1.BYPCODE , DMC_DDR_LANE1_CTL0.BYPSELP/DMC_DDR_LANE0_CTL1.BYPCODE and DMC_DDR_LANE0_CTL0.BYPENB/DMC_DDR_LANE1_CTL0.BYPENB to trim the DQS duty cycle adjustment.
2. Program DMC_DDR_CA_CTL registers with DMC_DDR_CA_CTL.BYPCODE0, DMC_DDR_CA_CTL.BYPCODE1, DMC_DDR_CA_CTL.BYPSELP and DMC_DDR_CA_CTL.BYPENB to trim the clock duty cycle adjustment.

Programming the DMC Controller

Perform the following procedure to program the DMC controller.

1. Program the DMC_CFG register with the size of the DDR memory device. Program the DMC_CFG.IFWID and DMC_CFG.SDRWID bit fields to 16-bits wide.
2. Program the DMC_TR0, DMC_TR1, DMC_TR2 registers with the timing parameters as required by the DDR memory device.
3. Program the DMC_MR and DMC_MR1 registers with the appropriate values. A memory ODT of 120Ω is recommended. Write leveling is recommended for a 666 MHz operating frequency.
4. Program the DMC_MR2 register. If read leveling is required, program the DMC_EMR3 register. Read leveling is recommended for a 666 MHz operating frequency.
5. Program the DMC_DLLCTL register. Program the DMC_DLLCTL.DATACYC bit field with 15ns/ddr clock period in ns. Program the DMC_DLLCTL.DLLCALRDCNT bit field with 240.
6. Wait for 2000 DMC clock cycles.
7. Set the DMC_DDR_CA_CTL.SW_REFRESH bit, keeping the other bits unchanged.
8. Wait for five DMC clock cycles.
9. Set the DMC_DDR_ROOT_CTL.SW_REFRESH bit. Program the DMC_DDR_ROOT_CTL.PIPE_OFSTDCYCLE bit field with 2.
10. Program the DMC_CTL register.
    - DMC_CTL.RDTOWR = 5
    - Set the DMC_CTL.DDR3EN bit.
    - Set DMC_CTL.AL_EN if the operating frequency is above 667 MHz.

- Set DMC_CTL.RL_DQS if operating frequency above 667 MHz and to enable read leveling during data strobing.
- Set the DMC_CTL.INIT bit.

11. If write leveling is enabled, wait for 600 DMC clock cycles or until the DMC_MR1.WL bit auto clears.
12. If read leveling is enabled wait for 2000 DMC clock cycles or until the DMC_EMR3.MPR bit auto clears.
13. If read leveling during data strobe gating is enabled, wait for 600 DMC clock cycles or until the DMC_CTL.RL_DQS bit auto clears.
14. Wait for 722000 DMC clock cycles to complete DMC initialization or until the DMC_STAT.INITDONE bit is set.
15. Set the DMC_DDR_LANE0_CTL1.COMP_DCYCLE and DMC_DDR_LANE1_CTL1.COMP_DCYCLE bits, without affecting other bits.
16. Wait for 10 DMC clock cycles.
17. Clear the DMC_DDR_LANE0_CTL1.COMP_DCYCLE and DMC_DDR_LANE1_CTL1.COMP_DCYCLE bits, without affecting other bits.
18. Set and then clear the DMC_DDR_LANE0_CTL0.CB_RSTDAT bit, without affecting other bits.
19. Set and then clear DMC_DDR_LANE1_CTL0.CB_RSTDAT bit, without affecting other bits.
20. Wait for 2500 DMC clock cycles.
21. Read the DMC_STAT.PHYRDPHASE bit field. Program this value into the DMC_DLLCTL.DATACYC bit field, without affecting other bits in the DMC_DLLCTL register.
22. Reprogram the DMC_CTL.DDR3EN and DMC_CTL.RDTOWR bits. If the operation frequency is above 667 MHz set the DMC_CTL.AL_EN bit.

Programming DQ Delay Trim

To trim or fine tune the delay on the data lane, the write leveling code should be read first from the data lanes. The write leveling code can be incremented to increase setup time or decremented to increase hold time before writing it back to the data lanes.

For lane 0, complete the following steps:

1. Write 0x000000D0 to the DMC_DDR_LANE0_CTL1 register without changing other bits.
2. Wait 2500 DMC clock cycles.
3. Write 0x00400000 to the DMC_DDR_ROOT_CTL register.
4. Wait 2500 DMC clock cycles.
5. Write 0x00000000 to the DMC_DDR_ROOT_CTL register.

6. Bits [20:16] of the DMC_DDR_SCRATCH_4 holds the write leveling code for lane 0.
7. Increment or decrement the write leveling code as needed. Write it to the DMC_DDR_LANE0_CTL1.BYPCODE bit. Set the DMC_DDR_LANE0_CTL1.BYPDELCHAINEN
8. Wait 2500 DMC clock cycles.

For lane 1, complete the following steps:

1. Write 0x000000D0 to the DMC_DDR_LANE1_CTL1 register without changing other bits.
2. Wait 2500 DMC clock cycles.
3. Write 0x00800000 to the DMC_DDR_ROOT_CTL register.
4. Wait 2500 DMC clock cycles.
5. Write 0x00000000 to the DMC_DDR_ROOT_CTL register.
6. Bits [20:16] of the DMC_DDR_SCRATCH_5 holds the write leveling code for lane 1.
7. Increment or decrement the write leveling code as needed. Write it to the DMC_DDR_LANE1_CTL1.BYPCODE bit. Set the DMC_DDR_LANE1_CTL1.BYPDELCHAINEN
8. Wait 2500 DMC clock cycles.

DMC Performance Monitor (DMCPM)

The dynamic memory controller includes performance monitors to obtain the performance index for SDRAM and SCB fabric. These counters can be used to analyze the DMC access patterns and bus utilization (both DMC and SCB bus) at a system level. The DMCPM provides seven DMC performance counters and five SCB performance counters. The DMCPM has 32-bit counters that (when enabled) increment based on events.

To enable performance monitoring, set the DMC_PM_CTL.PMCNT bit = 1.

ADSP-SC59x DMC Register Descriptions

Dynamic Memory Controller (DMC) contains the following registers.

Table 9-5: ADSP-SC59x DMC Register List

Name	Description
DMC_CFG	Configuration Register
DMC_CTL	Control Register
DMC_DLLCTL	DLL Control Register
DMC_DT_CALIB_ADDR	Data Calibration Address Register
DMC_DT_DATA_CALIB_DATA0	Data Calibration Data 0 Register

Table 9-5: ADSP-SC59x DMC Register List (Continued)

Name	Description
DMC_DT_DATA_CALIB_DATA1	Data Calibration Data 1 Register
DMC_EFFCTL	Efficiency Control Register
DMC_EMR3	Shadow EMR3 Register
DMC_MR	Shadow MR0 Register (DDR3)
DMC_MR1	Shadow MR1 Register (DDR3)
DMC_MR2	Shadow MR2 Register (DDR3)
DMC_MSK	Mask (Mode Register Shadow) Register
DMC_PM_CTL	DMC Performance Control Register
DMC_PM_IDCYDQ_CNT	Performance Monitor Idle Cycle DQ Count Register
DMC_PM_IDPG_CNT	Performance Monitor Idle Page Count Register
DMC_PM_PGHT_CNT	Performance Monitor Page Hit Count
DMC_PM_PGMS_CNT	Performance Monitor Page Miss Count
DMC_PM_RD2WRTA_CNT	Performance Monitor Read to Write Turn Around Count Register
DMC_PM_REF_CNT	Performance Monitor Refresh Count Register
DMC_PM_WR2RDTA_CNT	Performance Monitor Write to Read Turn Around Count Register
DMC_PRIO	Priority ID Register 1
DMC_PRIO2	Priority ID Register 2
DMC_PRIOMSK	Priority ID Mask Register 1
DMC_PRIOMSK2	Priority ID Mask Register 2
DMC_RDDATABUFID1	DMC Read Data Buffer ID Register 1
DMC_RDDATABUFID2	DMC Read Data Buffer ID Register 2
DMC_RDDATABUFMSK1	DMC Read Data Buffer Mask Register 1
DMC_RDDATABUFMSK2	DMC Read Data Buffer Mask Register 2
DMC_RDD_BUFHT_CNT	Performance Monitor Buffer Hit Count Register
DMC_SCB_PM_BRLN_CNT	SCB Performance AXI Transfer Count Register
DMC_SCB_PM_BRSZ_CNT	SCB Performance AXI Transfer Count Register
DMC_SCB_PM_IDRDDCY_CNT	SCB Performance Monitor Idle Cycle Count Register
DMC_SCB_PM_IDWRDCY_CNT	SCB Performance Monitor Idle Cycle Count Register
DMC_STAT	Status Register
DMC_TR0	Timing 0 Register
DMC_TR1	Timing 1 Register

Table 9-5: ADSP-SC59x DMC Register List (Continued)

Name	Description
DMC_TR2	Timing 2 Register

Configuration Register

The DMC_CFG register selects SDRAM device specific parameters and selects the SDRAM interface width.

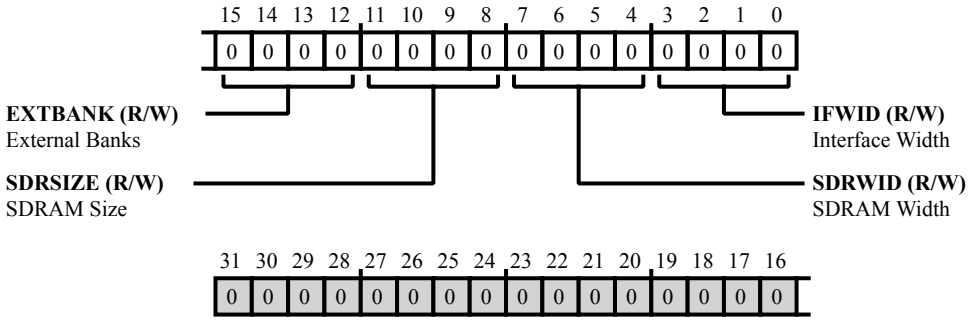


Figure 9-1: DMC_CFG Register Diagram

Table 9-6: DMC_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	EXTBANK	External Banks. The DMC_CFG.EXTBANK bits select the number of external banks connected to the DMC. Note that all values other than those shown are reserved.	
		0	1 External Bank
		1-15	Reserved
11:8 (R/W)	SDRSIZE	SDRAM Size. The DMC_CFG.SDRSIZE bits select the size of individual SDRAM connected to the DMC. Note that all values other than those shown are reserved.	
		3	512M Bit SDRAM
		4	1G Bit SDRAM
		5	2G Bit SDRAM
		6	4G Bit SDRAM
		7	8G Bit SDRAM
7:4 (R/W)	SDRWID	SDRAM Width. The DMC_CFG.SDRWID bits select the width of the individual SDRAM connected to the DMC. Note that all values other than those shown are reserved.	
		0-1	Reserved
		2	16-Bit Wide SDRAM
		3-15	Reserved

Table 9-6: DMC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:0 (R/W)	IFWID	Interface Width. The `DMC_CFG.IFWID` bits select the width of the interface between the DMC and SDRAM. Note that all values other than those shown are reserved.	
		0-1	Reserved
		2	16-Bit Wide Interface. All other values are reserved. This field specifies the interface width between the controller and the SDRAM.
		3-15	Reserved

Control Register

The DMC_CTL register controls DMC modes, DLL calibration, and DRAM initialization.

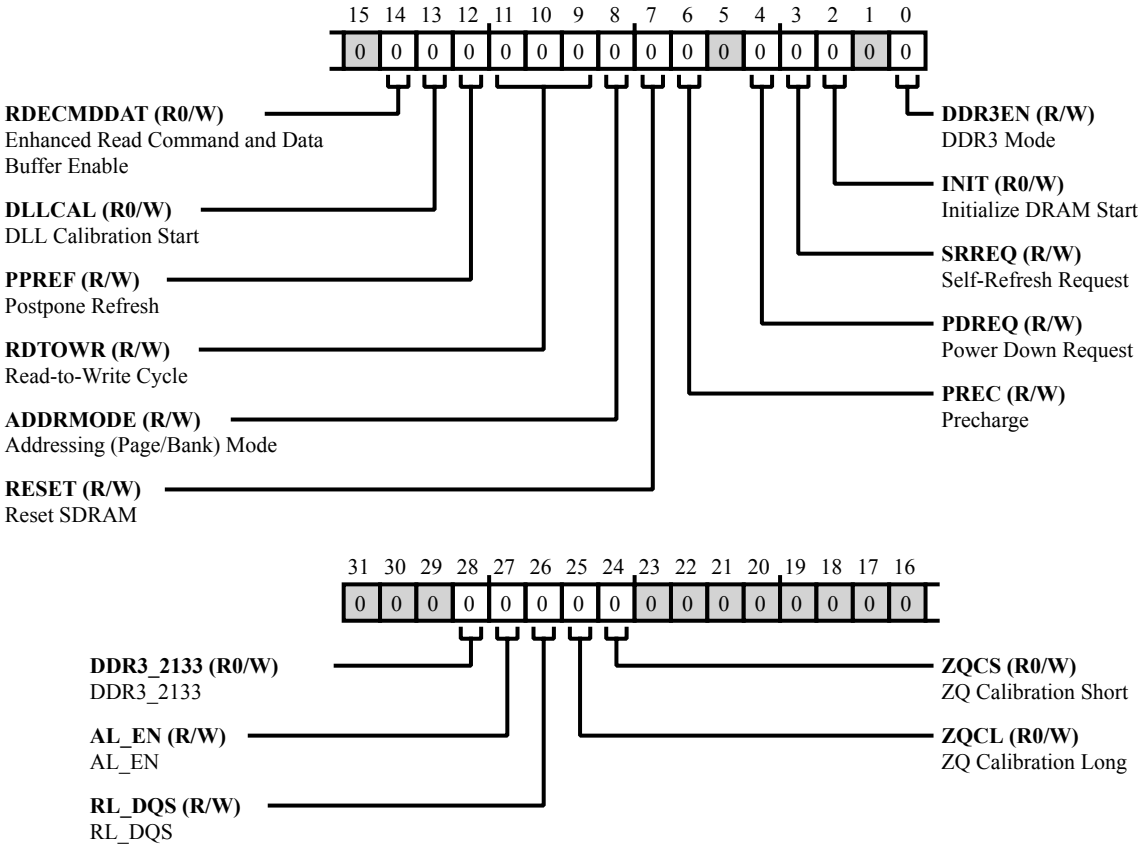


Figure 9-2: DMC_CTL Register Diagram

Table 9-7: DMC_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R0/W)	DDR3_2133	DDR3_2133. The DMC_CTL.DDR3_2133 bit starts the ZQ calibration long sequence. Note that this bit always reads as 0.	
		0	No effect
		1	Triggers ZQ calibration long sequence
27 (R/W)	AL_EN	AL_EN. When set, enables 800MHz operation	
		0	Disables greater than 667MHz operation
		1	Enables greater than 667MHz operation

Table 9-7: DMC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	RL_DQS	RL_DQS. When set, enables read leveling during DQS gating training. This bit auto-clears on read leveling completion	
		0	Enable
		1	Disable
25 (R0/W)	ZQCL	ZQ Calibration Long. The `DMC_CTL.ZQCL` bit starts the ZQ calibration long sequence. Note that this bit always reads as 0.	
		0	No effect
		1	Triggers ZQ calibration long sequence
24 (R0/W)	ZQCS	ZQ Calibration Short. The `DMC_CTL.ZQCS` bit starts the ZQ calibration short sequence. Note that this bit always reads as 0.	
		0	No effect
		1	Triggers ZQ calibration short sequence
14 (R0/W)	RDECMDDAT	Enhanced Read Command and Data Buffer Enable. Enhanced Read Command and Data Buffer Enable. 0 DDR AXI interface works with legacy Read Data and Command Buffer depth. (default state) 1 DDR AXI interface works with enhanced (double) Read Data and Command Buffer depth. 0 - No effect 1 - Enable	
		0	No effect
		1	Start PHY DLL calibration
13 (R0/W)	DLLCAL	DLL Calibration Start. The `DMC_CTL.DLLCAL` bit starts the PHY DLL calibration sequence. Note that this bit always reads as 0.	
		0	No effect
		1	Start PHY DLL calibration

Table 9-7: DMC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	PPREF	Postpone Refresh. The DMC_CTL.PPREF bit enables postponing the DMCs sending of auto-refresh commands. When enabled, the DMC accumulates refresh commands. The DMC_EFFCTL.NUMREF field selects the number of refresh commands that the DMC may accumulate. When disabled, the DMC_TR1.TREF field selects the interval for auto-refresh command distribution. A maximum of eight auto-refresh commands can be accumulated in DDR3 mode.	
		0	Disable Postpone Refresh
		1	Enable Postpone Refresh
11:9 (R/W)	RDTOWR	Read-to-Write Cycle. The DMC_CTL.RDTOWR bits select the number of cycles that the DMC adds when a write operation follows a read operation. For proper operation, it should be programmed with the value of 010.	
		0	1 Cycle Added from JEDEC Spec Value
		1	2 Cycles Added from JEDEC Spec Value
		2	3 Cycles Added from JEDEC Spec Value
		3	4 Cycles Added from JEDEC Spec Value
		4	5 Cycles Added from JEDEC Spec Value
		5	6 Cycles Added from JEDEC Spec Value
		6	7 Cycles Added from JEDEC Spec Value
		7	8 Cycles Added from JEDEC Spec Value
8 (R/W)	ADDRMODE	Addressing (Page/Bank) Mode. The DMC_CTL.ADDRMODE bit selects whether the DMC uses page or bank interleaving for addressing. When using page interleaving, the bank address bits follow the most significant column address bits. When using bank interleaving, the bank address bits follow the most significant row address bits.	
		0	Bank Interleaving
		1	Page Interleaving
7 (R/W)	RESET	Reset SDRAM. The DMC_CTL.RESET bit starts the reset sequence. Note that this bit always reads as 0.	
		0	No effect
		1	Starts reset sequence

Table 9-7: DMC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	PREC	Precharge. The DMC_CTL.PREC bit enables precharge, which closes DRAM rows immediately after access. When disabled, all accesses result in the respective DRAM rows remaining open, until the DMC needs to close them.	
		0	No Effect
		1	Enable Precharge
4 (R/W)	PDREQ	Power Down Request. The DMC_CTL.PDREQ bit enables power-down mode. When the DMC is in power-down mode, any data accesses cause the DMC to generate a bus error. The DRAM remains in power-down mode as along as this bit is 1.	
		0	Disable Power-Down
		1	Enable Power-Down
3 (R/W)	SRREQ	Self-Refresh Request. The DMC_CTL.SRREQ bit enables self-refresh mode. When the DMC is in self-refresh mode, any data accesses cause the DMC to generate a bus error. The DRAM remains in self-refresh mode as along as this bit is 1.	
		0	Disable Self-Refresh
		1	Enable Self-Refresh
2 (R0/W)	INIT	Initialize DRAM Start. The DMC_CTL.INIT bit starts the power up DRAM initialization sequence and DLL calibration sequence. Note that this bit always reads as 0.	
		0	No Effect
		1	Start DRAM Initialization
0 (R/W)	DDR3EN	DDR3 Mode. The DMC_CTL.DDR3EN bit selects whether the DMC operates in DDR3 mode.	
		0	Reserved
		1	Enable DDR3 mode

DLL Control Register

The DMC_DLLCTL register holds the programmable parameters associated with the DLLs within the DMC PHY.

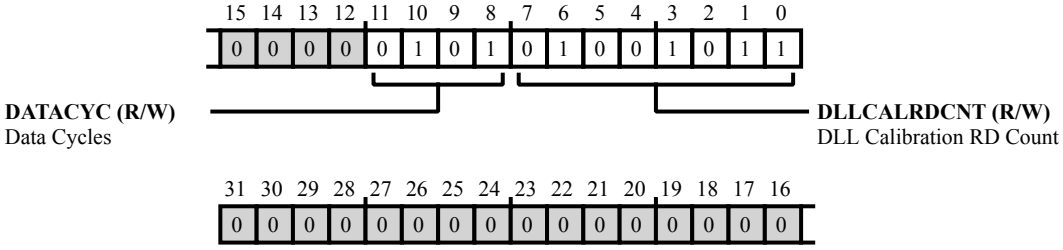


Figure 9-3: DMC_DLLCTL Register Diagram

Table 9-8: DMC_DLLCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11:8 (R/W)	DATACYC	Data Cycles. The DMC_DLLCTL.DATACYC bits select the latency after which the DMC reads data from the PHY. This field must be written with the value indicated in the DMC_STAT.PHYRDPHASE field, or data corruption occurs on all SDRAM reads.	
		2	2 Clock Cycles Latency
		3	3 Clock Cycles Latency
		4	4 Clock Cycles Latency
		5	5 Clock Cycles Latency
		6	6 Clock Cycles Latency
		7	7 Clock Cycles Latency
		8	8 Clock Cycles Latency
		9	9 Clock Cycles Latency
		10	10 Clock Cycles Latency
		11	11 Clock Cycles Latency
		12	12 Clock Cycles Latency
		13	13 Clock Cycles Latency
		14	14 Clock Cycles Latency
		15	15 Clock Cycles Latency
7:0 (R/W)	DLLCALRDCNT	DLL Calibration RD Count. The DMC_DLLCTL.DLLCALRDCNT field selects the number of read operations that the PHY uses for DLL calibration.	

Data Calibration Address Register

The DMC_DT_CALIB_ADDR register provides the address used for the data calibration for read and write. During the DMC PHY DLL calibration, a particular set of locations in the DRAM is written and a series of reads are performed back to back to calibrate the PHY. The DMC PHY needs prior information about the data that would be read during the PHY DLL calibration. The controller performs one burst write operation to the address programmed in DMC_DT_CALIB_ADDR (0x0090).

Note: While the exact address chosen does not matter much during memory initialization, if calibration of the PHY is performed when the DRAM contains valid data, care needs to be taken to ensure that this address points to an unused address. Else, this operation will modify application data stored at the address selected.

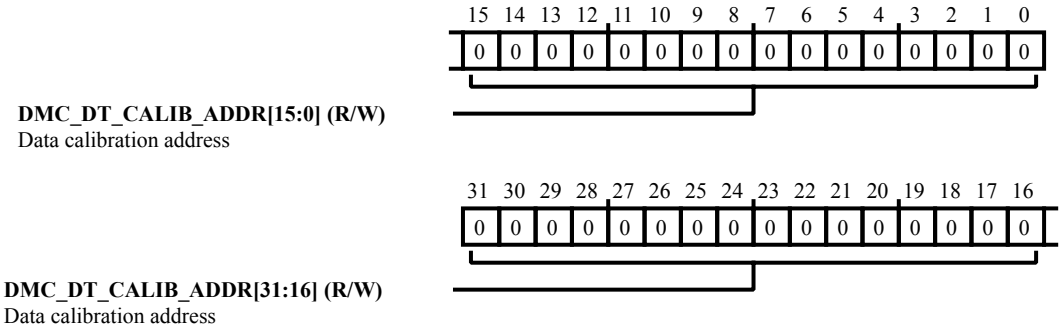


Figure 9-4: DMC_DT_CALIB_ADDR Register Diagram

Table 9-9: DMC_DT_CALIB_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DMC_DT_CALIB_ADDR	Data calibration address. The DMC_DT_CALIB_ADDR.DMC_DT_CALIB_ADDR bit field contains the address to be programmed for the data calibration for read and write.

Data Calibration Data 0 Register

The DMC_DT_DATA_CALIB_DATA0 register contains the first 32-bit data used for the write during the data calibration.

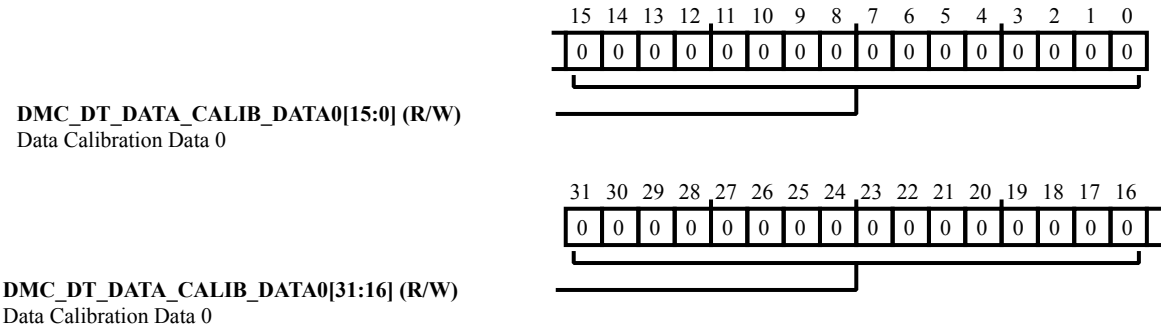


Figure 9-5: DMC_DT_DATA_CALIB_DATA0 Register Diagram

Table 9-10: DMC_DT_DATA_CALIB_DATA0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DMC_DT_DATA_CALIB_DATA0	Data Calibration Data 0. The DMC_DT_DATA_CALIB_DATA0.DMC_DT_DATA_CALIB_DATA0 bit field contains the first 32 bit data used for the write during the data calibration.

Data Calibration Data 1 Register

The DMC_DT_DATA_CALIB_DATA1 register contains the second 32-bit data used for the write during the data calibration.

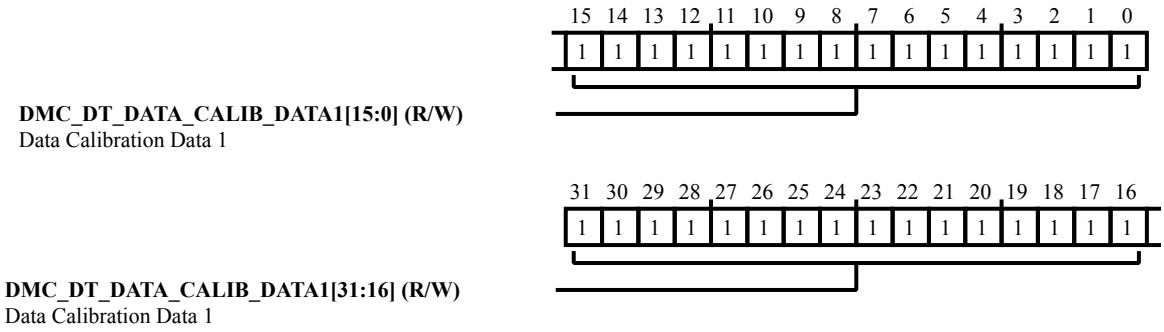


Figure 9-6: DMC_DT_DATA_CALIB_DATA1 Register Diagram

Table 9-11: DMC_DT_DATA_CALIB_DATA1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DMC_DT_DATA_CALIB_DATA1	Data Calibration Data 1. The DMC_DT_DATA_CALIB_DATA1.DMC_DT_DATA_CALIB_DATA1 bit field contains the second 32 bit data used for the write during the data calibration.

Efficiency Control Register

The DMC_EFFCTL register control DMC features that improve throughput efficiency. These include features such as auto-refresh management, precharge options, and write data options.

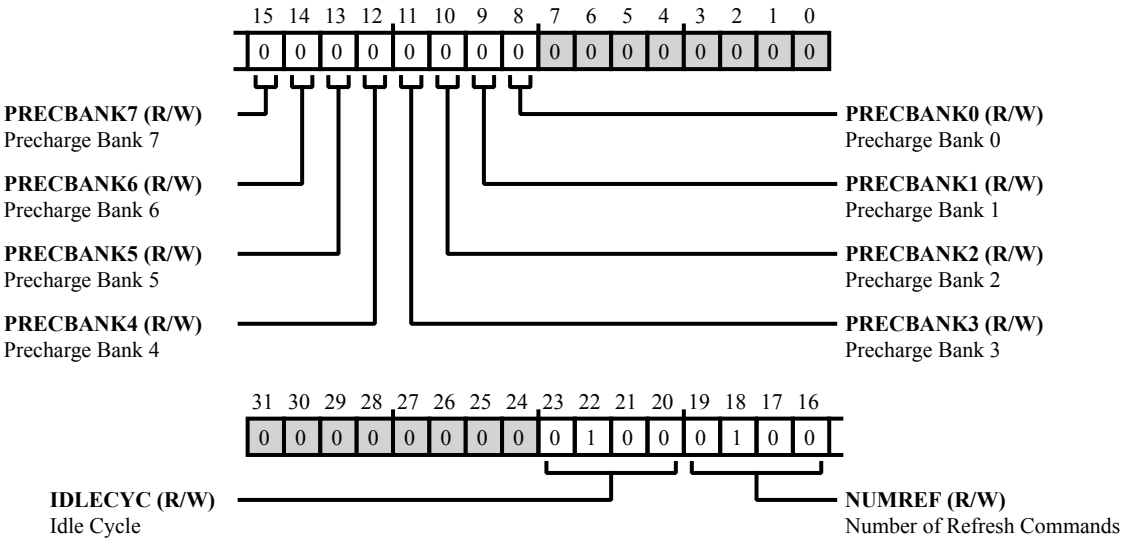


Figure 9-7: DMC_EFFCTL Register Diagram

Table 9-12: DMC_EFFCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:20 (R/W)	IDLECYC	Idle Cycle. The DMC_EFFCTL.IDLECYC bits select the number of cycles after which the DMC issues any accumulated auto-refresh commands if postpone refresh is enabled (DMC_CTL.PPREF =1). When DMC_EFFCTL.IDLECYC is set to 0, the DMC ignores the DMC_CTL.PPREF selection and does not accumulate/postpone periodic auto-refresh commands. Note 1: By default, accumulated auto-refresh commands are issued after counting four idle cycles. Note 2: This value is ignored if DMC_CTL.PPREF is not set. Note 3: Setting this value to 0000 overrides the "postpone refresh" feature and does not accumulate/postpone periodic auto refreshes.	
		0-15	0 to 15 Idle Cycles to Postpone Refresh Commands

Table 9-12: DMC_EFFCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	NUMREF	Number of Refresh Commands. The DMC_EFFCTL.NUMREF bits select the number of auto-refresh commands that the DMC can accumulate if postpone refresh is enabled (DMC_CTL.PPREF =1). In DDR3 mode, the DMC may accumulate up to eight auto-refresh commands. Note 1: By default, accumulated auto-refresh commands are issued after counting four idle cycles. Note 2: This value is ignored if DMC_CTL.PPREF is not set.	
		0	No Refresh Commands Accumulate
		1	1 Refresh Command May Accumulate
		2	2 Refresh Commands May Accumulate
		3	3 Refresh Commands May Accumulate
		4	4 Refresh Commands May Accumulate
		5	5 Refresh Commands May Accumulate
		6	6 Refresh Commands May Accumulate
		7	7 Refresh Commands May Accumulate
		8	8 Refresh Commands May Accumulate
15 (R/W)	PRECBANK7	Precharge Bank 7. The DMC_EFFCTL.PRECBANK7 bit enables precharge (closes the page) of bank 7 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). Note: The (DMC_CTL.PREC) takes precedence over value in this register. If (DMC_CTL.PREC =1) then all banks are precharged.	
		0	Disable Precharge Bank 7
		1	Enable Precharge Bank 7
14 (R/W)	PRECBANK6	Precharge Bank 6. The DMC_EFFCTL.PRECBANK6 bit enables precharge (closes the page) of bank 6 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). Note: The (DMC_CTL.PREC) takes precedence over value in this register. If (DMC_CTL.PREC =1) then all banks are precharged.	
		0	Disable Precharge Bank 6
		1	Enable Precharge Bank 6

Table 9-12: DMC_EFFCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	PRECBANK5	Precharge Bank 5. The DMC_EFFCTL.PRECBANK5 bit enables precharge (closes the page) of bank 5 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). Note: The (DMC_CTL.PREC) takes precedence over value in this register. If (DMC_CTL.PREC =1) then all banks are precharged.	
		0	Disable Precharge Bank 5
		1	Enable Precharge Bank 5
12 (R/W)	PRECBANK4	Precharge Bank 4. The DMC_EFFCTL.PRECBANK4 bit enables precharge (closes the page) of bank 4 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). Note: The (DMC_CTL.PREC) takes precedence over value in this register. If (DMC_CTL.PREC =1) then all banks are precharged.	
		0	Disable Precharge Bank 4
		1	Enable Precharge Bank 4
11 (R/W)	PRECBANK3	Precharge Bank 3. The DMC_EFFCTL.PRECBANK3 bit enables precharge (closes the page) of bank 3 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). Note: The (DMC_CTL.PREC) takes precedence over value in this register. If (DMC_CTL.PREC =1) then all banks are precharged.	
		0	Disable Precharge Bank 3
		1	Enable Precharge Bank 3
10 (R/W)	PRECBANK2	Precharge Bank 2. The DMC_EFFCTL.PRECBANK2 bit enables precharge (closes the page) of bank 2 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). Note: The (DMC_CTL.PREC) takes precedence over value in this register. If (DMC_CTL.PREC =1) then all banks are precharged.	
		0	Disable Precharge Bank 2
		1	Enable Precharge Bank 2
9 (R/W)	PRECBANK1	Precharge Bank 1. The DMC_EFFCTL.PRECBANK1 bit enables precharge (closes the page) of bank 1 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). Note: The (DMC_CTL.PREC) takes precedence over value in this register. If (DMC_CTL.PREC =1) then all banks are precharged.	
		0	Disable Precharge Bank 1
		1	Enable Precharge Bank 1

Table 9-12: DMC_EFFCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	PRECBANK0	Precharge Bank 0. The DMC_EFFCTL.PRECBANK0 bit enables precharge (closes the page) of bank 0 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). Note: The (DMC_CTL.PREC) takes precedence over value in this register. If (DMC_CTL.PREC =1) then all banks are precharged.	
		0	Disable Precharge Bank 0
		1	Enable Precharge Bank 0

Shadow EMR3 Register

The DMC_EMR3 register in the DMC shadows the EMR3 register in the SDRAM when the DMC is in DDR3 mode (DMC_CTL.DDR3EN =1). If unmasked by the corresponding bit in the shadow mask register (DMC_MSK.EMR3 =1), a write to DMC_EMR3 triggers an extended “mode register set” command on the memory interface. If masked, a write to DMC_EMR3 only updates the register in the DMC, not the register in the SDRAM.

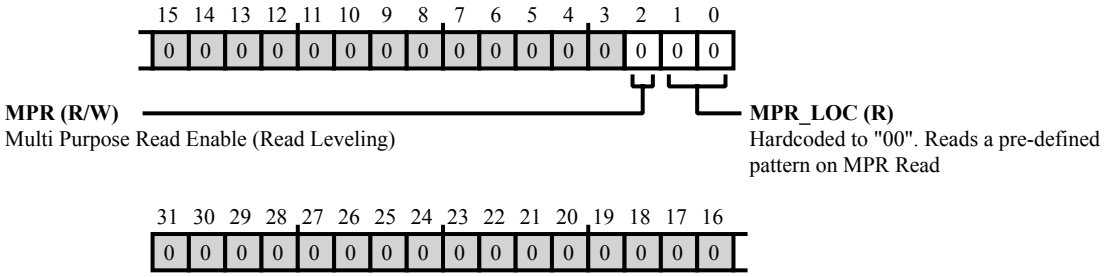


Figure 9-8: DMC_EMR3 Register Diagram

Table 9-13: DMC_EMR3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	MPR	Multi Purpose Read Enable (Read Leveling). The DMC_EMR3.MPR bit enables high temperature self-refresh rate.	
		0	Disable
		1	Enable
1:0 (R/NW)	MPR_LOC	Hardcoded to "00". Reads a pre-defined pattern on MPR Read.	

Shadow MR0 Register (DDR3)

The DMC_MR register in the DMC shadows the MR register in the SDRAM when the DMC is DDR3 mode (DMC_CTL.DDR3EN =1). If unmasked by the corresponding bit in the shadow mask register (DMC_MSK.MR =1), a write to DMC_MR triggers a “mode register set” command on the memory interface. If masked, a write to DMC_MR only updates the register in the DMC, not the register in the SDRAM.

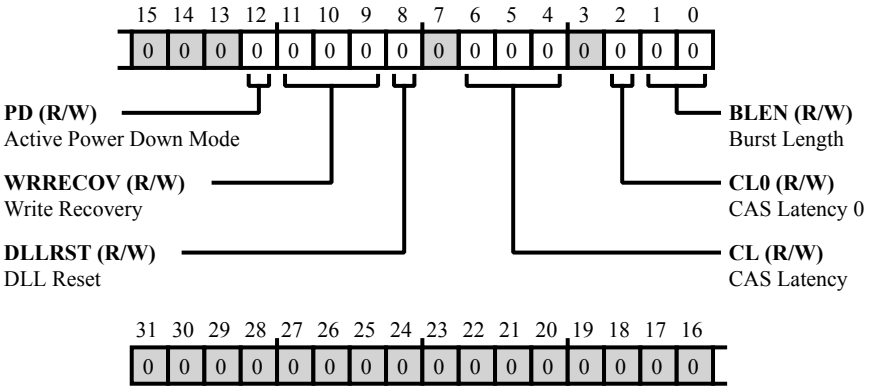


Figure 9-9: DMC_MR Register Diagram

Table 9-14: DMC_MR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	PD	Active Power Down Mode. The DMC_MR.PD bit selects the active power-down mode. Note that this parameter applies only for DDR3 mode. For more information about this mode, see the data sheet for the SDRAM being used in your system.	
		0	Fast Exit (normal)
		1	Slow Exit (low power)
11:9 (R/W)	WRRECOV	Write Recovery. The DMC_MR.WRRECOV bit selects the write recovery time in terms of clock cycles (t_{CK}). Note that this parameter applies only for DDR3 mode. For more information about this mode, see the data sheet for the SDRAM being used in your system.	
		0	16 clock cycles for DDR3
		1	5 clock cycles for DDR3
		2	6 clock cycles for DDR3
		3	7 clock cycles for DDR3
		4	8 clock cycles for DDR3
		5	10 clock cycles for DDR3
		6	12 clock cycles for DDR3

Table 9-14: DMC_MR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		7	14 clock cycles for DDR3
8 (R/W)	DLLRST	DLL Reset. The DMC_MR.DLLRST bit initiates a DLL reset on the SDRAM. Note that this parameter applies only for DDR3 mode. For more information about this operation, see the data sheet for the SDRAM being used in your system.	
		0	Normal Operation
		1	Reset DLL
6:4 (R/W)	CL	CAS Latency. The DMC_MR.CL bit field select latency from the assertion of a read/write signal to the SDRAM until the first valid data on the output from the SDRAM in terms of clock cycles. For more information about this operation, see the data sheet for the SDRAM being used in your system. For DDR3 only, Bit[2] of this register must be used along with [6:4]. Following CAS values are seen. "0010": 5 "0100": 6 "0110": 7 "1000": 8 "1010": 9 "1100": 10 "1110": 11 "0001": 12 "0011": 13 "0101": 14 All other combinations are reserved.	
2 (R/W)	CL0	CAS Latency 0. The DMC_MR.CL0 bit is applicable for DDR3 only and is used in conjunction with the DMC_MR.CL bits.	
1:0 (R/W)	BLEN	Burst Length. The DMC_MR.BLEN bits select burst length for transfers. For more information about this operation, see the data sheet for the SDRAM being used in your system. Note that values other than those shown are not supported.	
		0	8-Bit Burst Length - DDR3 only
		2	Reserved
		3	Reserved

Shadow MR1 Register (DDR3)

The DMC_MR1 register is a mirror of the DDR3 SDRAM Mode register 1. This register is used only when the DMC is operating in DDR3 mode. A write to this register triggers an extended "mode register 1 set" command on the memory interface provided the corresponding mask bit is set in the mask register. Else, only the mirror register is updated.

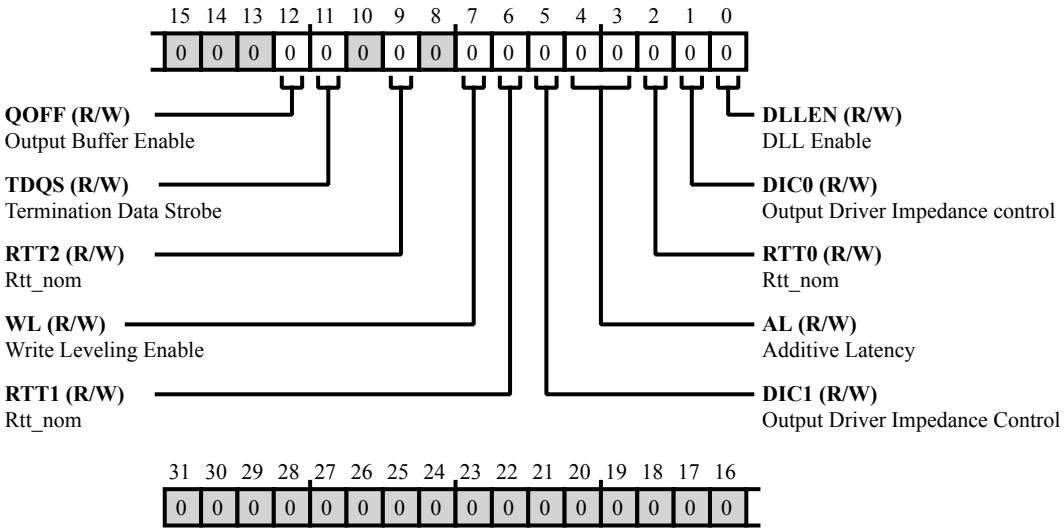


Figure 9-10: DMC_MR1 Register Diagram

Table 9-15: DMC_MR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	QOFF	Output Buffer Enable. The DMC_MR1.QOFF bit enables the SDRAM output pins. For more information about this operation, see the data sheet for the SDRAM being used in your system.	
		0	Output buffer enabled
		1	Output buffer disabled
11 (R/W)	TDQS	Termination Data Strobe. The DMC_MR1.TDQS bit provides additional termination resistance outputs that may be useful in some system configurations. The DMC_MR1.TDQS bit is not supported in x4 or x16 configurations. When enabled via the mode register, the same termination resistance function is applied to the TDQS/TDQS# pins that is applied to the DQS/DQS# pins.	
		0	Enable
		1	Disable

Table 9-15: DMC_MR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	RTT2	Rtt_nom. The DMC_MR1.RTT2 bit is used in conjunction with the DMC_MR1.RTT0 and DMC_MR1.RTT1 bits. (9 6 2) 0 0 0 Rtt_Nom disabled 0 0 1 RZQ/4 0 1 0 RZQ/2 0 1 1 RZQ/6 1 0 0 RZQ/12 (reserved if Rtt_Nom is used during writes) 1 0 1 RZQ/8 (reserved if Rtt_Nom is used during writes) 1 1 0 Reserved 1 1 1 Reserved
7 (R/W)	WL	Write Leveling Enable. The DMC_MR1.WL bit enables the SDRAM output pins. For more information about this operation, see the data sheet for the SDRAM being used in your system. 0 Disable 1 Enable
6 (R/W)	RTT1	Rtt_nom. The DMC_MR1.RTT1 bit combines with the DMC_MR1.RTT0 bit to set the termination resistance. See the DMC_MR1.RTT2 and DMC_MR1.RTT0 bit description for more information.
5 (R/W)	DIC1	Output Driver Impedance Control. The DMC_MR1.DIC1 bit is used in conjunction with the DMC_MR1.DIC0 bit. (5, 1) 0 0 RZQ/6 0 1 RZQ/7 1 0 Reserved 1 1 Reserved

Table 9-15: DMC_MR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4:3 (R/W)	AL	Additive Latency. The DMC_MR1.AL bits select a number of added latency time for CAS operations in terms of clock cycles (t_{CK}). For more information about this operation, see the data sheet for the SDRAM being used in your system.	
		0	AL disabled
		1	CL-1
		2	CL-2
		3	Reserved
2 (R/W)	RTT0	Rtt_nom. The DMC_MR1.RTT0 bit combines with the DMC_MR1.RTT1 and DMC_MR1.RTT1 bits to set the termination resistance. See the DMC_MR1.RTT1 and DMC_MR1.RTT2 bit descriptions for more information.	
1 (R/W)	DIC0	Output Driver Impedance control. The DMC_MR1.DIC0 bit is used with the DMC_MR1.DIC1 bit.	
0 (R/W)	DLLEN	DLL Enable. The DMC_MR1.DLLEN bit enables the DLL in the SDRAM. For more information about this operation, see the data sheet for the SDRAM being used in your system.	
		0	Enable
		1	Disable

Shadow MR2 Register (DDR3)

The DMC_MR2 register mirrors DDR3 SDRAM device Mode register 2 when the controller is operating in DDR3 mode. A write to this register triggers an extended "mode register set" command on the memory interface provided the corresponding mask bit is set in the mask register. Else, only the mirror register is updated.

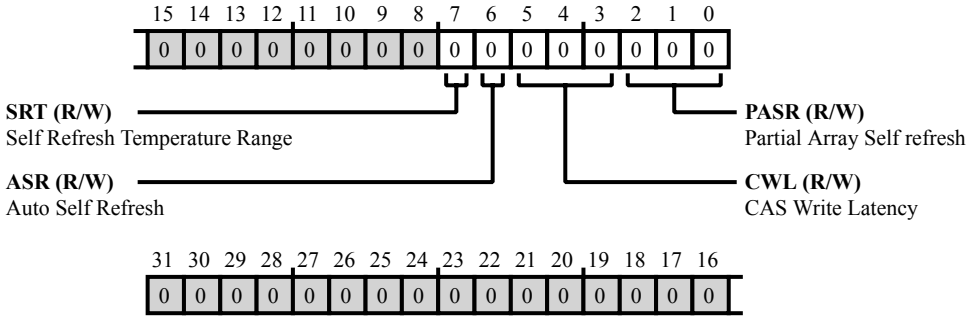


Figure 9-11: DMC_MR2 Register Diagram

Table 9-16: DMC_MR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	SRT	Self Refresh Temperature Range. The DMC_MR2.SRT bit enables high temperature self-refresh rate.	
		0	Disable
		1	Enable
6 (R/W)	ASR	Auto Self Refresh.	
		0	Manual SR Reference (SRT)
		1	ASR enable (Optional)
5:3 (R/W)	CWL	CAS Write Latency.	
		0	5 clock cycles
		1	6 clock cycles
		2	7 clock cycles
		3	8 clock cycles
		4	9 clock cycles
		5	10 clock cycles
		6	11 clock cycles
		7	12 clock cycles

Table 9-16: DMC_MR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	PASR	Partial Array Self refresh. The `DMC_MR2.PASR` bits select the amount of memory to be refreshed during self refresh. For more information about this operation, see the data sheet for the SDRAM being used in your system.	
		0	4 banks: full array, 8 banks: full array
		1	4 banks: Half Array (BA[1:0]=00&01), 8 banks: Half Array (BA[2:0] = 000, 001, 010, &011)
		2	4 banks: Quarter Array (BA[1:0]=00), 8 banks: Quarter Array (BA[2:0] = 000&001)
		3	4 banks: not defined, 8 banks: 1/8th array (BA[2:0] = 000)
		4	4 banks: 3/4 Array (BA[1:0]=01, 10&11), 8 banks: 3/4 Array (BA[2:0] = 010, 011, 100, 101, 110, &111)
		5	4 banks: Half Array (BA[1:0]=10&11), 8 banks: Half Array (BA[2:0] = 100, 101, 110, &111)
		6	4 banks: Quarter Array (BA[1:0]=11), 8 banks: Quarter Array (BA[2:0] =110 & 111)
		7	4 banks: not defined, 8 banks: 1/8th array (BA[2:0] = 111)

Mask (Mode Register Shadow) Register

The DMC_MSK register permits masking (disabling) writes to the MR and EMRn registers in the SDRAM in DDR3 Mode. When masked, writes to these registers go instead to shadow copies of these registers (DMC_MR, DMC_MR1, DMC_MR2), which are maintained within the DMC. When a shadow register's corresponding bit is unmasked (enabled), the DMC generates the MRS or EMRS command to transfer the contents of the shadow register (in the DMC) to the actual register (in the SDRAM).

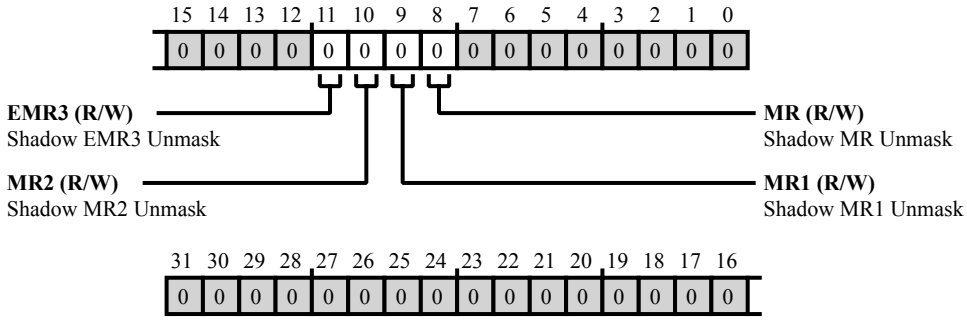


Figure 9-12: DMC_MSK Register Diagram

Table 9-17: DMC_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	EMR3	Shadow EMR3 Unmask. The DMC_MSK.EMR3 bit masks or unmasks writes to the EMR3 register in the SDRAM. When masked, writes to this register instead go to the EMR3 register. When unmasked, the DMC writes the EMR3 value to the EMR3 register in the SDRAM. After completing the write, the DMC clears this bit.	
		0	Mask (Disable) Write to EMR3
		1	Unmask (Enable) Write to EMR3
10 (R/W)	MR2	Shadow MR2 Unmask. The DMC_MSK.MR2 bit masks or unmasks writes to the MR2 register (in DDR3 mode) in the SDRAM. When masked, writes to this register instead go to the DMC_MR2 register. When unmasked, the DMC writes the DMC_MR2 value to the MR2 register (in DDR3 mode) in the SDRAM. After completing the write, the DMC clears this bit.	
		0	Mask (Disable) Write to EMR2
		1	Unmask (Enable) Write to EMR2

Table 9-17: DMC_MSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	MR1	Shadow MR1 Unmask. The DMC_MSK.MR1 bit masks or unmasks writes to the MR1 register in the DDR3 SDRAM. When masked, writes to this register instead go to the DMC_MR1 register. When unmasked, the DMC writes the DMC_MR1 value to the MR1 register in the SDRAM. After completing the write, the DMC clears this bit. Note that this bit is valid only for DDR3 Mode of operation.	
		0	Mask (Disable) Write to EMR1
		1	Unmask (Enable) Write to EMR1
8 (R/W)	MR	Shadow MR Unmask. The DMC_MSK.MR bit masks or unmasks writes to the MR register in the SDRAM. When masked, writes to this register instead go to the DMC_MR register. When unmasked, the DMC writes the DMC_MR value to the MR register in the SDRAM. After completing the write, the DMC clears this bit.	
		0	Mask (Disable) Write to MR
		1	Unmask (Enable) Write to MR

DMC Performance Control Register

The DMC_PM_CTL register generates a performance index. It does this by obtaining a count of various performance indicators.

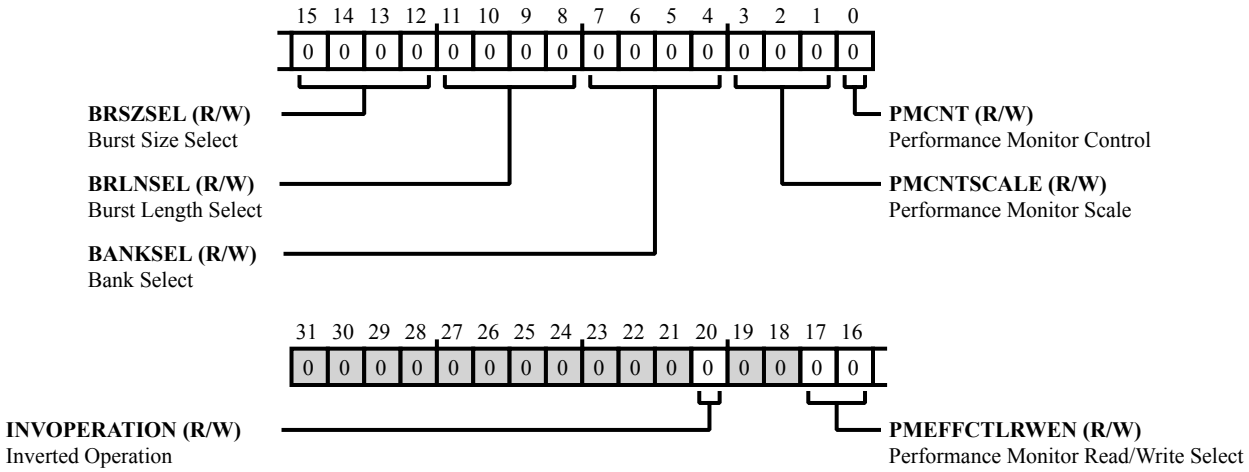


Figure 9-13: DMC_PM_CTL Register Diagram

Table 9-18: DMC_PM_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	INVOPERATION	Inverted Operation. The DMC_PM_CTL.INVOPERATION bit is inverts the operation of the idle cycle counter. When enabled the counter reports the number of cycles for which the corresponding bus is NOT idle. Used for the DQ bus, SCB Read data bus and SCB Write data bus.	
		0	Normal operation
		1	Inverted operation
17:16 (R/W)	PMEFFCTLRWEN	Performance Monitor Read/Write Select. The DMC_PM_CTL.PMEFFCTLRWEN bit selects whether the performance monitor efficiency counter is used for read transactions or write transactions	
		1	Read
		2	Write
		3	Read and Write

Table 9-18: DMC_PM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	BRSZSEL	Burst Size Select. The DMC_PM_CTL.BRSZSEL bit field reports the value of number of SCB transfers corresponding to the burst size shown by these bits.	
		0	Burst size 8
		1	Burst size 16
		2	Burst size 32
11:8 (R/W)	BRLNSEL	Burst Length Select. The DMC_PM_CTL.BRLNSEL bit field reports the value of number of SCB transfers corresponding to the burst length shown by these bits.	
		0	Burst length 1
		1	Burst length 2
		3	Burst length 4
		7	Burst length 8
		15	Burst length 16
7:4 (R/W)	BANKSEL	Bank Select. The DMC_PM_CTL.BANKSEL bit field chooses which bank or all banks are being selected for the page hit or page miss counters.	
		0	Bank 0
		1	Bank 1
		2	Bank 2
		3	Bank 3
		4	Bank 4
		5	Bank 5
		6	Bank 6
		7	Bank 7
		8	All Banks

Table 9-18: DMC_PM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:1 (R/W)	PMCNTSCALE	Performance Monitor Scale. The `DMC_PM_CTL.PMCNTSCALE` bit bit field reports the scale of the main to sub-counter.	
		0	1:1
		1	1:10
		2	1:100
		4	1:1000
0 (R/W)	PMCNT	Performance Monitor Control. The `DMC_PM_CTL.PMCNT` bit enables performance monitoring.	
		0	Disable the counters
		1	Enable the counters

Performance Monitor Idle Cycle DQ Count Register

The DMC_PM_IDCYDQ_CNT register reports the number of idle cycles on the DQ bus.

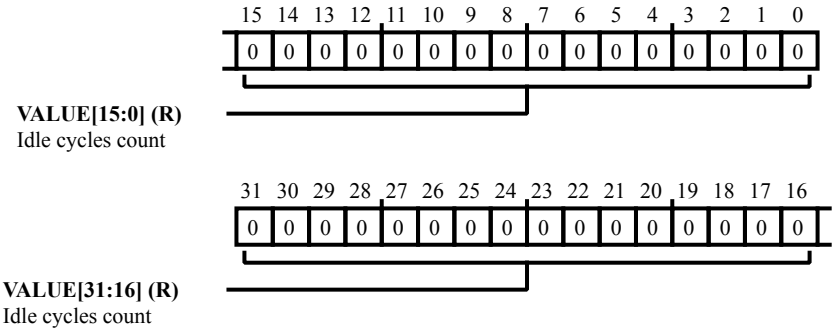


Figure 9-14: DMC_PM_IDCYDQ_CNT Register Diagram

Table 9-19: DMC_PM_IDCYDQ_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Idle cycles count.

Performance Monitor Idle Page Count Register

The DMC_PM_IDPG_CNT register reports the number of accesses to idle pages.

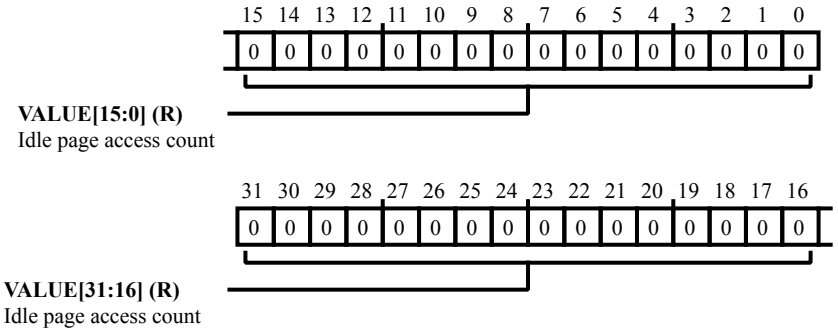


Figure 9-15: DMC_PM_IDPG_CNT Register Diagram

Table 9-20: DMC_PM_IDPG_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Idle page access count.

Performance Monitor Page Hit Count

The DMC_PM_PGHT_CNT register reports the number of page hit accesses.

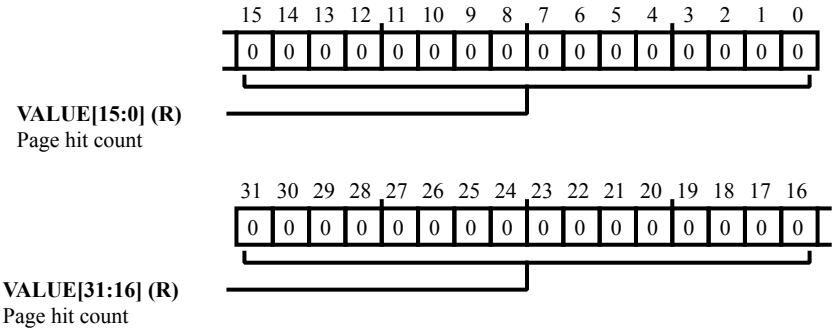


Figure 9-16: DMC_PM_PGHT_CNT Register Diagram

Table 9-21: DMC_PM_PGHT_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Page hit count.

Performance Monitor Page Miss Count

The DMC_PM_PGMS_CNT register reports the number of missed page accesses.

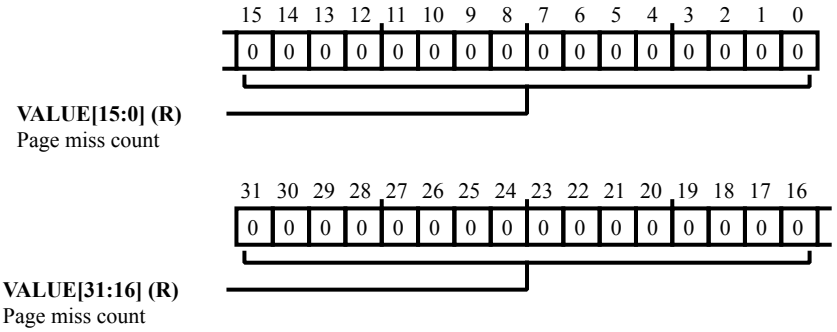


Figure 9-17: DMC_PM_PGMS_CNT Register Diagram

Table 9-22: DMC_PM_PGMS_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Page miss count.

Performance Monitor Read to Write Turn Around Count Register

The DMC_PM_RD2WRTA_CNT register reports the number of times the read to write turn around occurs.

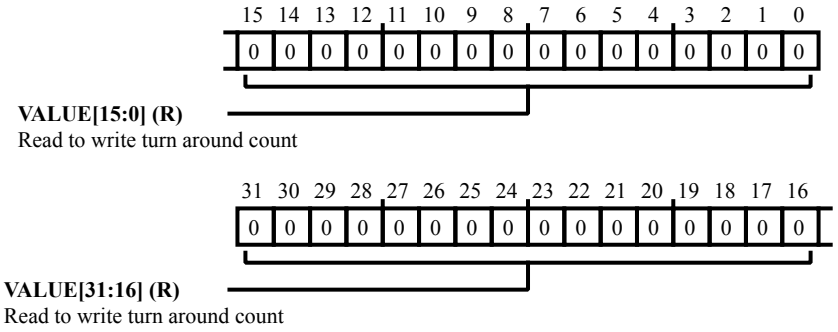


Figure 9-18: DMC_PM_RD2WRTA_CNT Register Diagram

Table 9-23: DMC_PM_RD2WRTA_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Read to write turn around count.

Performance Monitor Refresh Count Register

The DMC_PM_REF_CNT register reports the number of times a refresh occurs.

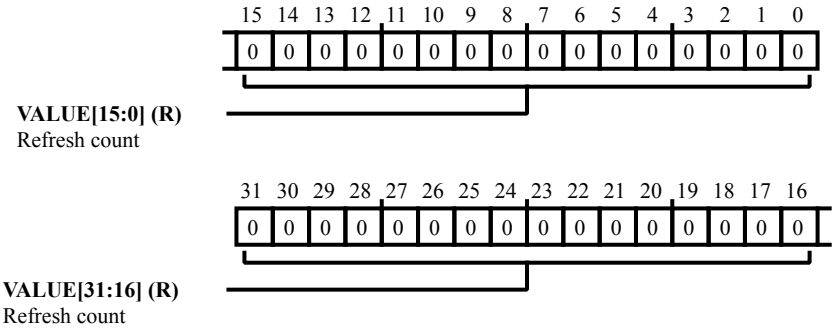


Figure 9-19: DMC_PM_REF_CNT Register Diagram

Table 9-24: DMC_PM_REF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Refresh count.

Performance Monitor Write to Read Turn Around Count Register

The DMC_PM_WR2RDTA_CNT register reports the number of times the write to read turn around occurs.

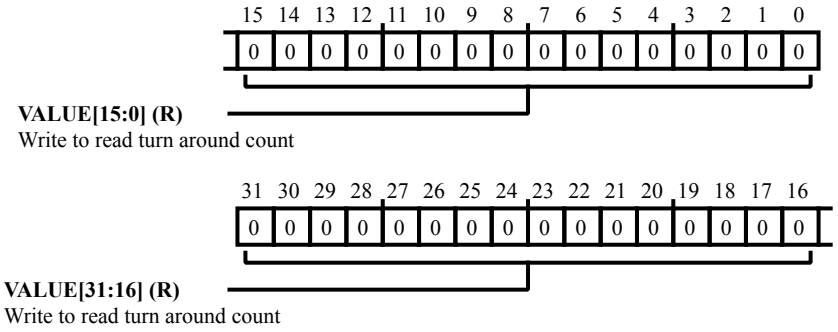


Figure 9-20: DMC_PM_WR2RDTA_CNT Register Diagram

Table 9-25: DMC_PM_WR2RDTA_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Write to read turn around count.

Priority ID Register 1

The DMC_PRIO register allows transactions from selected requesters that generate specific SCB IDs to obtain higher priority than the transactions proceeding in the usual fashion. The contents of the register are masked with the contents of the DMC_PRIOMSK register to obtain a single SCB ID or a range of IDs that get elevated priority.

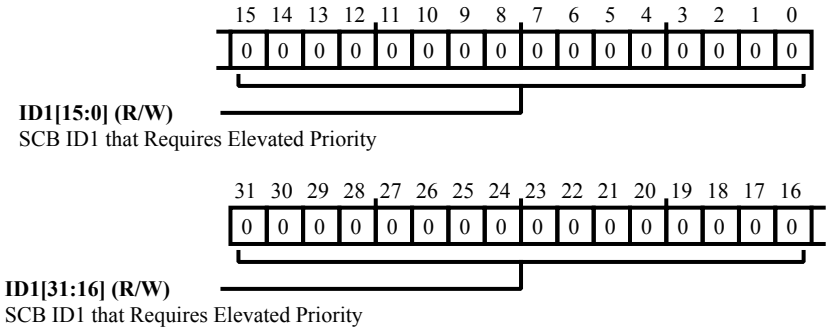


Figure 9-21: DMC_PRIO Register Diagram

Table 9-26: DMC_PRIO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ID1	SCB ID1 that Requires Elevated Priority.

Priority ID Register 2

The DMC_PRIO2 register is another register which allows transactions from selected requesters that generate specific SCB IDs to obtain higher priority than the transactions proceeding in the usual fashion. The contents of the register are masked with the contents of the DMC_PRIOMSK2 register to obtain a single SCB ID or a range of IDs that get elevated priority.

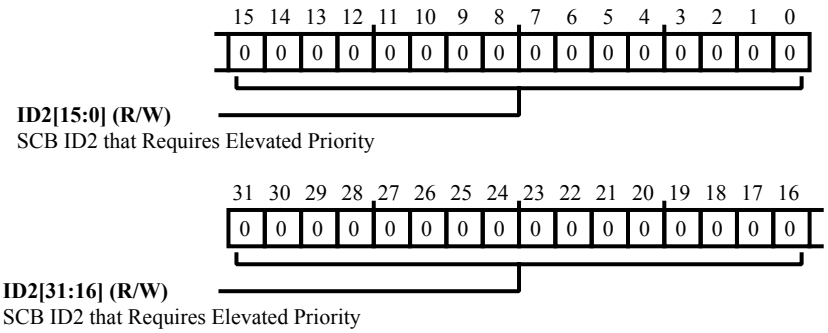


Figure 9-22: DMC_PRIO2 Register Diagram

Table 9-27: DMC_PRIO2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ID2	SCB ID2 that Requires Elevated Priority.

Priority ID Mask Register 1

The DMC_PRIOMSK register masks the respective ID bits in the DMC_PRIOMSK register. This masking provides for elevating the access priority of either a single ID or a range of IDs.

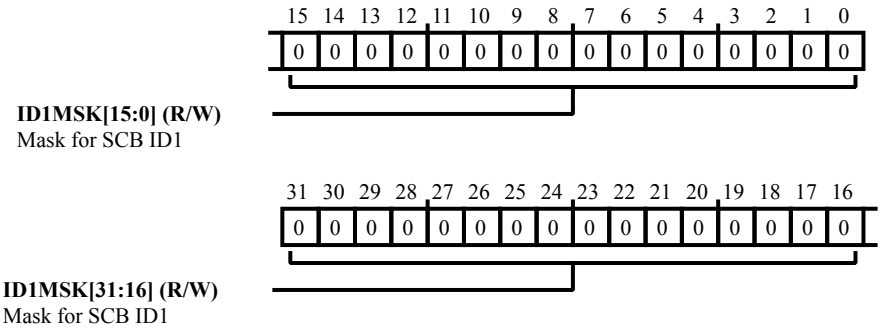


Figure 9-23: DMC_PRIOMSK Register Diagram

Table 9-28: DMC_PRIOMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ID1MSK	Mask for SCB ID1.

Priority ID Mask Register 2

The DMC_PRIOMSK2 register bits mask the respective ID bits in the DMC_PRIO2 register. This masking provides for elevating the access priority of either a single ID or a range of IDs.

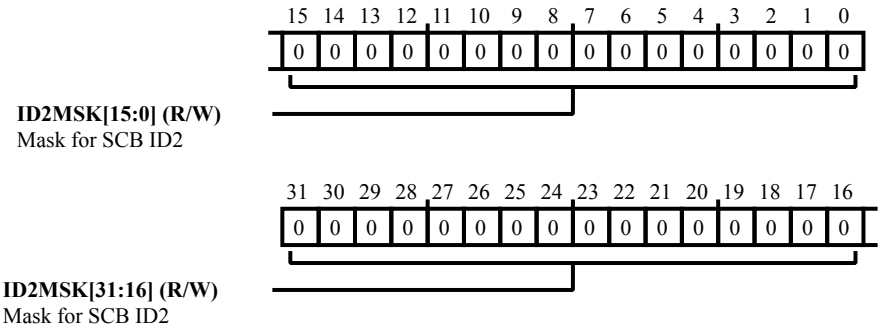


Figure 9-24: DMC_PRIOMSK2 Register Diagram

Table 9-29: DMC_PRIOMSK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ID2MSK	Mask for SCB ID2.

DMC Read Data Buffer ID Register 1

The DMC_RDDATABUFID1 register allows read transactions from selected requesters to make use of DMC read data buffer. The contents of the register are masked with the contents of the DMC_RDDATABUFMSK1 register to obtain a single SCB ID or a range of IDs that get elevated priority.

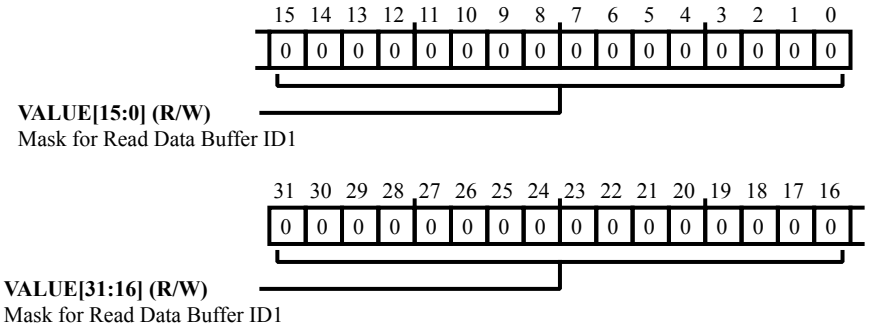


Figure 9-25: DMC_RDDATABUFID1 Register Diagram

Table 9-30: DMC_RDDATABUFID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Mask for Read Data Buffer ID1.

DMC Read Data Buffer ID Register 2

The DMC_RDDATABUFID2 register allows read transactions from selected requesters to make use of DMC read data buffer. The contents of the register are masked with the contents of the DMC_RDDATABUFMSK2 register to obtain a single SCB ID or a range of IDs that get elevated priority.

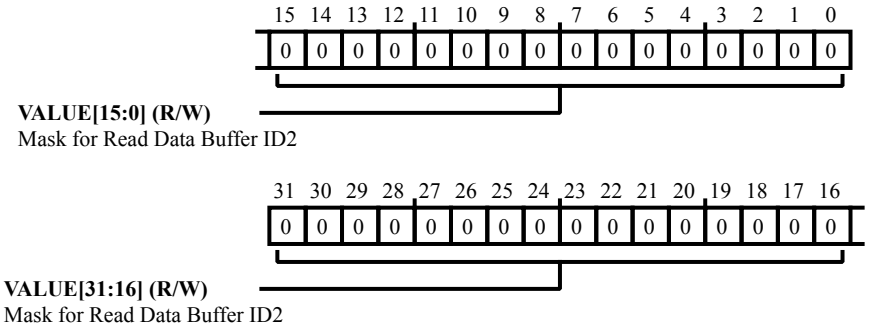


Figure 9-26: DMC_RDDATABUFID2 Register Diagram

Table 9-31: DMC_RDDATABUFID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Mask for Read Data Buffer ID2.

DMC Read Data Buffer Mask Register 1

The DMC_RDDATABUFMSK1 register bits mask the respective ID bits in the DMC Priority Mask ID register.

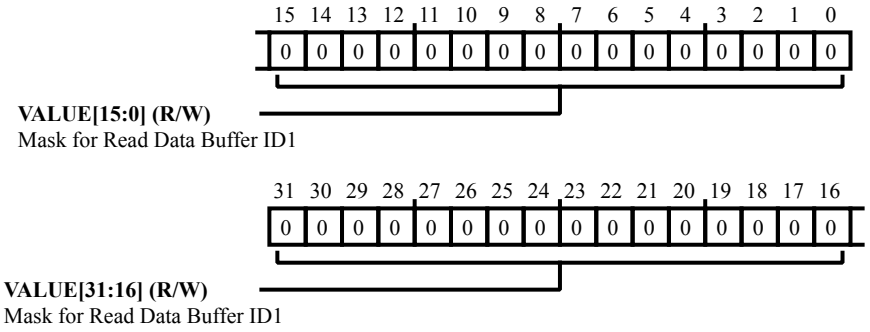


Figure 9-27: DMC_RDDATABUFMSK1 Register Diagram

Table 9-32: DMC_RDDATABUFMSK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Mask for Read Data Buffer ID1.

DMC Read Data Buffer Mask Register 2

The DMC_RDDATABUFMSK2 register bits mask the respective ID bits in the DMC Priority Mask ID register.

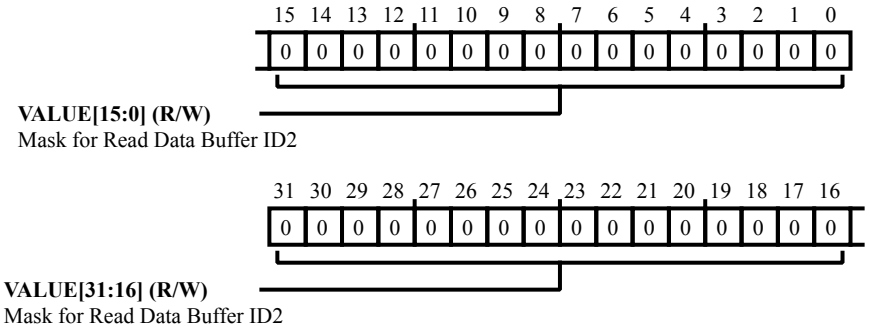


Figure 9-28: DMC_RDDATABUFMSK2 Register Diagram

Table 9-33: DMC_RDDATABUFMSK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Mask for Read Data Buffer ID2.

Performance Monitor Buffer Hit Count Register

The DMC_RDD_BUFHT_CNT register reports the count of the number of read data buffer hits.

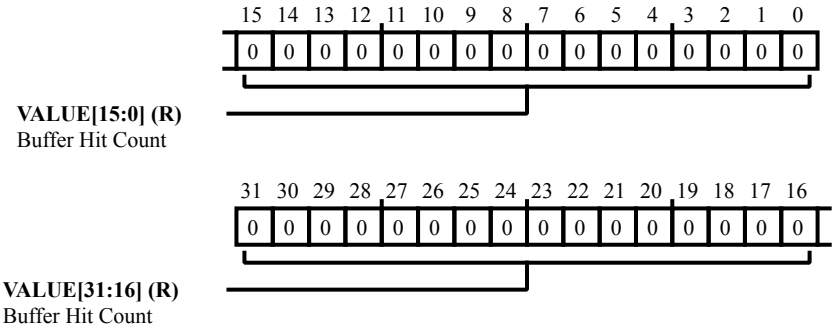


Figure 9-29: DMC_RDD_BUFHT_CNT Register Diagram

Table 9-34: DMC_RDD_BUFHT_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Buffer Hit Count.

SCB Performance AXI Transfer Count Register

The DMC_SCB_PM_BRLN_CNT register reports the count of number of times AXI transfer of burst length selected by performance control register is encountered.

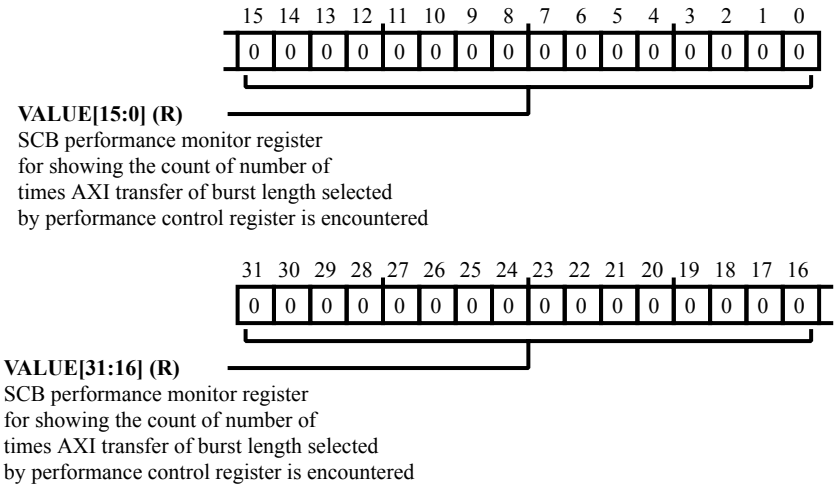


Figure 9-30: DMC_SCB_PM_BRLN_CNT Register Diagram

Table 9-35: DMC_SCB_PM_BRLN_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	SCB performance monitor register for showing the count of number of times AXI transfer of burst length selected by performance control register is encountered.

SCB Performance AXI Transfer Count Register

The DMC_SCB_PM_BRSZ_CNT register reports the count of number of times AXI transfer of burst size selected by performance control register is encountered.

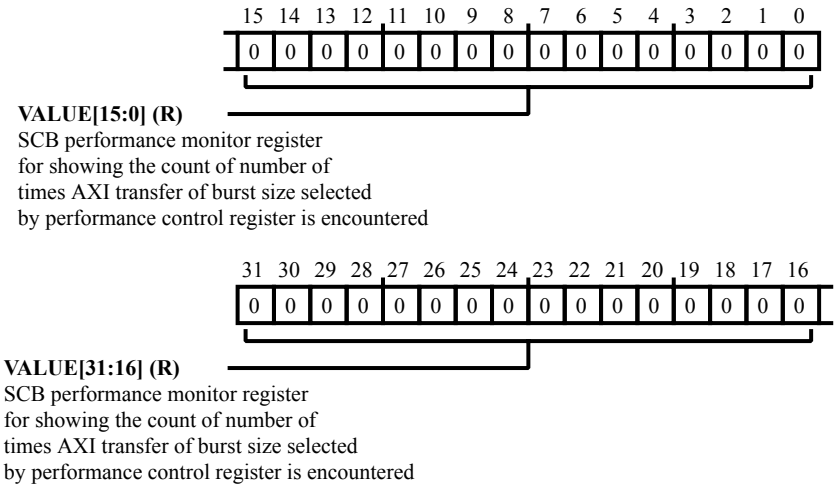


Figure 9-31: DMC_SCB_PM_BRSZ_CNT Register Diagram

Table 9-36: DMC_SCB_PM_BRSZ_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	SCB performance monitor register for showing the count of number of times AXI transfer of burst size selected by performance control register is encountered.

SCB Performance Monitor Idle Cycle Count Register

The DMC_SCB_PM_IDRDDCY_CNT register reports the count of the number idle cycles on the read data bus.

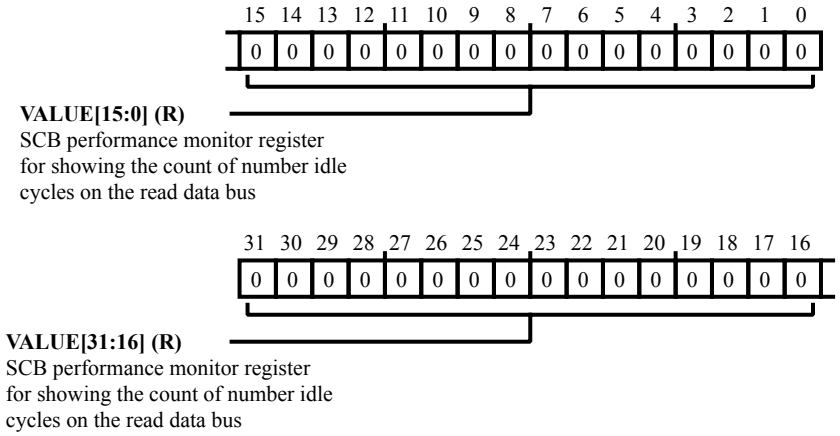


Figure 9-32: DMC_SCB_PM_IDRDDCY_CNT Register Diagram

Table 9-37: DMC_SCB_PM_IDRDDCY_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	SCB performance monitor register for showing the count of number idle cycles on the read data bus.

SCB Performance Monitor Idle Cycle Count Register

The DMC_SCB_PM_IDWRDCY_CNT register reports the count of number idle cycles on the write data bus.

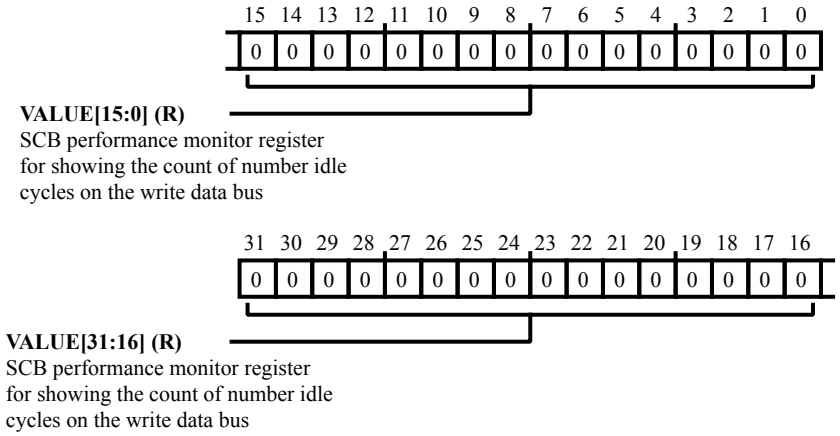


Figure 9-33: DMC_SCB_PM_IDWRDCY_CNT Register Diagram

Table 9-38: DMC_SCB_PM_IDWRDCY_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	SCB performance monitor register for showing the count of number idle cycles on the write data bus.

Status Register

The DMC_STAT register indicates status for modes selected with the DMC_CTL register and indicates status DMC operations.

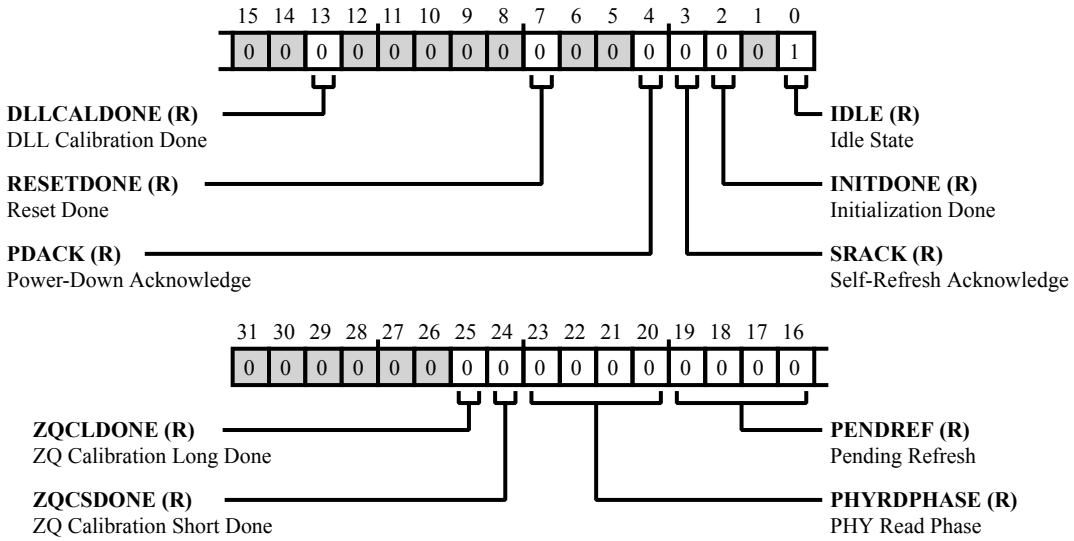


Figure 9-34: DMC_STAT Register Diagram

Table 9-39: DMC_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/NW)	ZQCLDONE	ZQ Calibration Long Done. The DMC_STAT.ZQCLDONE bit checks if the ZQ calibration long sub routine is done.	
		0	ZQ Calibration long is ongoing
		1	ZQ Calibration long is done
24 (R/NW)	ZQCSDONE	ZQ Calibration Short Done. The DMC_STAT.ZQCSDONE bit checks if the ZQ calibration short sub routine is done.	
		0	ZQ Calibration short is ongoing
		1	ZQ Calibration Short is done
23:20 (R/NW)	PHYRDPHASE	PHY Read Phase. The DMC_STAT.PHYRDPHASE bits indicate the latency after which the DMC may read from the PHY. Taking round trip delay into account, the DLL indicates the exact number of clock cycles after which the controller needs to read data. Values other than those shown are reserved.	
		2	2 Clock Cycles Latency

Table 9-39: DMC_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	3 Clock Cycles Latency
		4	4 Clock Cycles Latency
		5	5 Clock Cycles Latency
		6	6 Clock Cycles Latency
		7	7 Clock Cycles Latency
		8	8 Clock Cycles Latency
		9	9 Clock Cycles Latency
		10	10 Clock Cycles Latency
		11	11 Clock Cycles Latency
		12	12 Clock Cycles Latency
		13	13 Clock Cycles Latency
		14	14 Clock Cycles Latency
		15	15 Clock Cycles Latency
19:16 (R/NW)	PENDREF	Pending Refresh. The DMC_STAT.PENDREF bits indicate the number of pending auto-refresh commands whose value can be from "0000" to "0111".	
13 (R/NW)	DLLCALDONE	DLL Calibration Done. The DMC_STAT.DLLCALDONE indicates that the PHY DLL calibration sequence is complete.	
		0	No Status
		1	Completed PHY DLL Calibration
7 (R/NW)	RESETDONE	Reset Done. The DMC_STAT.RESETDONE bit indicates that the reset sequence is complete.	
		0	SDRAM Reset is ongoing
		1	SDRAM Reset is done
4 (R/NW)	PDACK	Power-Down Acknowledge. The DMC_STAT.PDACK bit indicates that power-down mode is active.	
		0	Not in Power-Down Mode
		1	Power-Down Mode Active

Table 9-39: DMC_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	SRACK	Self-Refresh Acknowledge. The `DMC_STAT.SRACK` bit indicates that self-refresh mode is active.	
		0	Not in Self-Refresh Mode
		1	Self-Refresh Mode Active
2 (R/NW)	INITDONE	Initialization Done. The `DMC_STAT.INITDONE` bit indicates that the initialization sequence is complete.	
		0	No Status
		1	Initialize Done
0 (R/NW)	IDLE	Idle State. The `DMC_STAT.IDLE` bit indicates whether the DMC is idle or busy.	
		0	Busy
		1	Idle

Timing 0 Register

The DMC_TR0 register selects timing parameters for DMC operation to corresponding with parameters of the SDRAM device that is used in the system. The timing registers must be programmed to match the device for correct operation of the SDRAM and must be programmed before initializing the SDRAM. Note that all values for bit fields in DMC_TR0 are in increments of clock cycle time (t_{CK}).

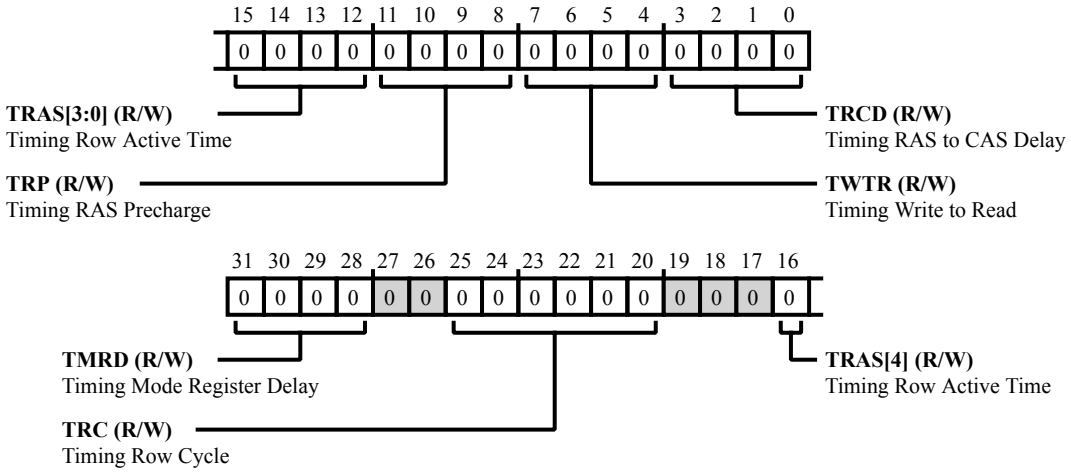


Figure 9-35: DMC_TR0 Register Diagram

Table 9-40: DMC_TR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/W)	TMRD	Timing Mode Register Delay. The DMC_TR0.TMRD field selects the set-to-active timing parameter (t_{MRD}), which is the number of clock cycles that occur after the mode registers in the SDRAM are set and before the next command is issued.
25:20 (R/W)	TRC	Timing Row Cycle. The DMC_TR0.TRC field selects the active-to-active time (t_{RC}), which is the minimum number of clock cycles that occur from an active command to the next active command in the same bank.
16:12 (R/W)	TRAS	Timing Row Active Time. The DMC_TR0.TRAS field selects the active-to-precharge time (t_{RAS}), which is the number of clock cycles that occur from an active command until a precharge command is allowed.
11:8 (R/W)	TRP	Timing RAS Precharge. The DMC_TR0.TRP field selects the precharge-to-active time (t_{RP}), which is the number of clock cycles that occur while the SDRAM recovers from a precharge command and becomes ready to accept the next active command.

Table 9-40: DMC_TR0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/W)	TWTR	Timing Write to Read. The `DMC_TR0.TWTR` field selects the write-to-read delay time (t_{WTR}), which is the number of clock cycles that occur from the last write data to the next read command.
3:0 (R/W)	TRCD	Timing RAS to CAS Delay. The `DMC_TR0.TRCD` field selects the RAS to CAS delay time (t_{RCD}), which is the number of clock cycles that occur from an active command to a read/write assertion.

Timing 1 Register

The DMC_TR1 register selects timing parameters for DMC operation to corresponding with parameters of the SDRAM device that is used in the system. The timing registers must be programmed to match the device for correct operation of the SDRAM and must be programmed before initializing the SDRAM. Note that all values for bit fields in DMC_TR1 are in increments of clock cycle time (t_{CK}).

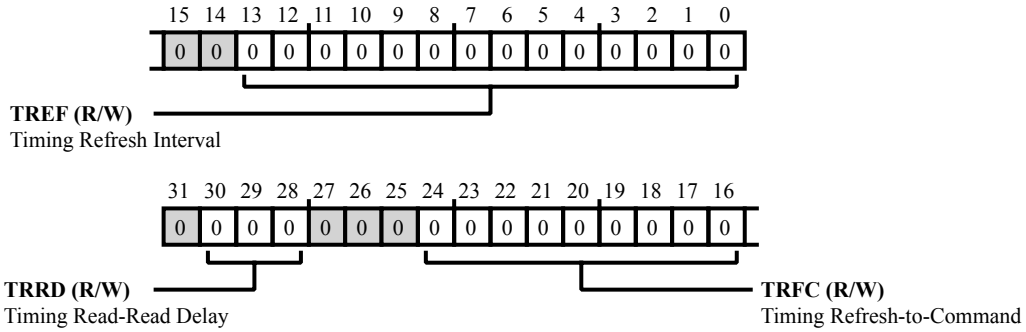


Figure 9-36: DMC_TR1 Register Diagram

Table 9-41: DMC_TR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30:28 (R/W)	TRRD	Timing Read-Read Delay. The DMC_TR1.TRRD field selects the active-to-active time (t_{RRD}), which is the minimum number of clock cycles occurring from a bank x active command to a bank y active command.
24:16 (R/W)	TRFC	Timing Refresh-to-Command. The DMC_TR1.TRFC field selects the refresh-to-active command delay (t_{RFC}), which is the number of clock cycles required for the SDRAM to recover from a refresh signal to be ready to take the next command. It is also the number of clock cycles needed for the SDRAM to recover from executing one active command and ready to accept the next active command.
13:0 (R/W)	TREF	Timing Refresh Interval. The DMC_TR1.TREF field selects the refresh interval time (t_{REF}), which is the number of clock cycles occurring from one refresh command to the next refresh command. The actual timing of issuing a precharge command may be delayed by if the SDRAM is processing a normal access. However, the delay is not accumulative so there is no need to shorten the refresh interval to account for the memory access time. The non-accumulative refresh delay typically increases memory bandwidth by a few percentage points.

Timing 2 Register

The DMC_TR2 register selects timing parameters for DMC operation to corresponding with parameters of the SDRAM device that is used in the system. The timing registers must be programmed to match the device for correct operation of the SDRAM and before initializing the SDRAM.

Note that all values for bit fields in DMC_TR2 are in increments of clock cycle time (t_{CK}).

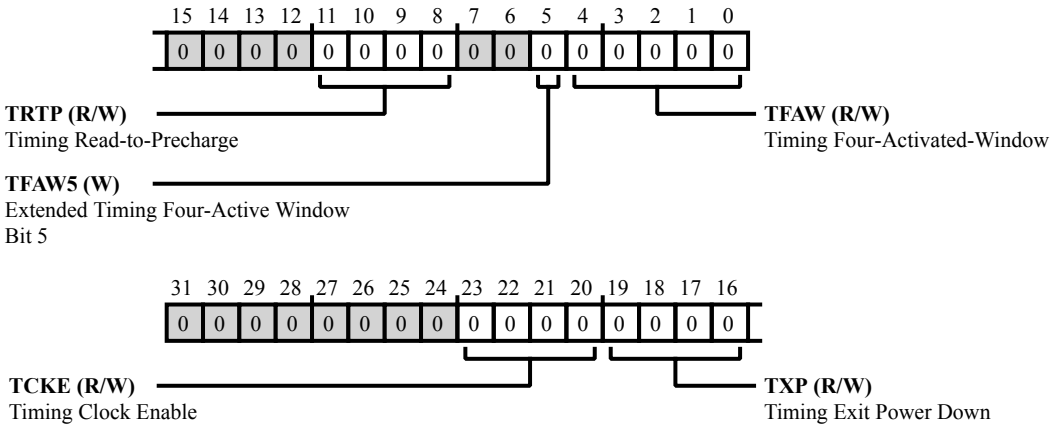


Figure 9-37: DMC_TR2 Register Diagram

Table 9-42: DMC_TR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:20 (R/W)	TCKE	Timing Clock Enable. The DMC_TR2.TCKE field selects the CKE minimum pulsewidth (t_{CKE}).
19:16 (R/W)	TXP	Timing Exit Power Down. The DMC_TR2.TXP field selects the exit power down to next valid command time (t_{XP}).
11:8 (R/W)	TRTP	Timing Read-to-Precharge. The DMC_TR2.TRTP field selects the internal read to precharge time (t_{RTP}). If the resulting value is less than 2, the register needs to be programmed with two. Note: Minimum t_{RTP} that needs to be programmed is 2.
5 (RX/W)	TFAW5	Extended Timing Four-Active Window Bit 5. The DMC_TR2.TFAW5 bit is the extended bit for FAW timing for 800 MHz operation for values greater than 31. It is only applicable when DMC_CTL. AL_EN= 1. When DMC_CTL. AL_EN = 0, the DMC_TR2.TFAW5 bit is reserved. When DMC_CTL. AL_EN = 1, the total FAW[5:0] = {TFAW5,TFAW[4:0]}. NOTE: This bit is write-only. Read as zero.

Table 9-42: DMC_TR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	TFAW	Timing Four-Activated-Window. The DMC_TR2.TFAW field selects the four-banks activated window time (t_{FAW}). No more than four SDRAM banks should be activated within this window.

ADSP-SC59x DMC Register Descriptions

1600MHz DMCPHY (DDR3/DDR3L) (DMC) contains the following registers.

Table 9-43: ADSP-SC59x DMC Register List

Name	Description
DMC_DDR_CA_CTL	DDR CA Lane Control Register
DMC_DDR_LANE0_CTL0	Data Lane 0 Control Register 0
DMC_DDR_LANE0_CTL1	Data Lane 0 Control Register 1
DMC_DDR_LANE1_CTL0	Data Lane 1 Control Register 0
DMC_DDR_LANE1_CTL1	Data Lane 1 Control Register 1
DMC_DDR_ROOT_CTL	DDR ROOT Module Control Register
DMC_DDR_SCRATCH_2	Scratch Register 2
DMC_DDR_SCRATCH_3	Scratch Register 3
DMC_DDR_SCRATCH_4	Scratch Register 4
DMC_DDR_SCRATCH_5	Scratch Register 5
DMC_DDR_SCRATCH_6	Scratch Register 6
DMC_DDR_SCRATCH_7	Scratch Register 7
DMC_DDR_ZQ_CTL0	DDR Calibration Control Register 0
DMC_DDR_ZQ_CTL1	DDR Calibration Control Register 1
DMC_DDR_ZQ_CTL2	DDR Calibration Control Register 2

DDR CA Lane Control Register

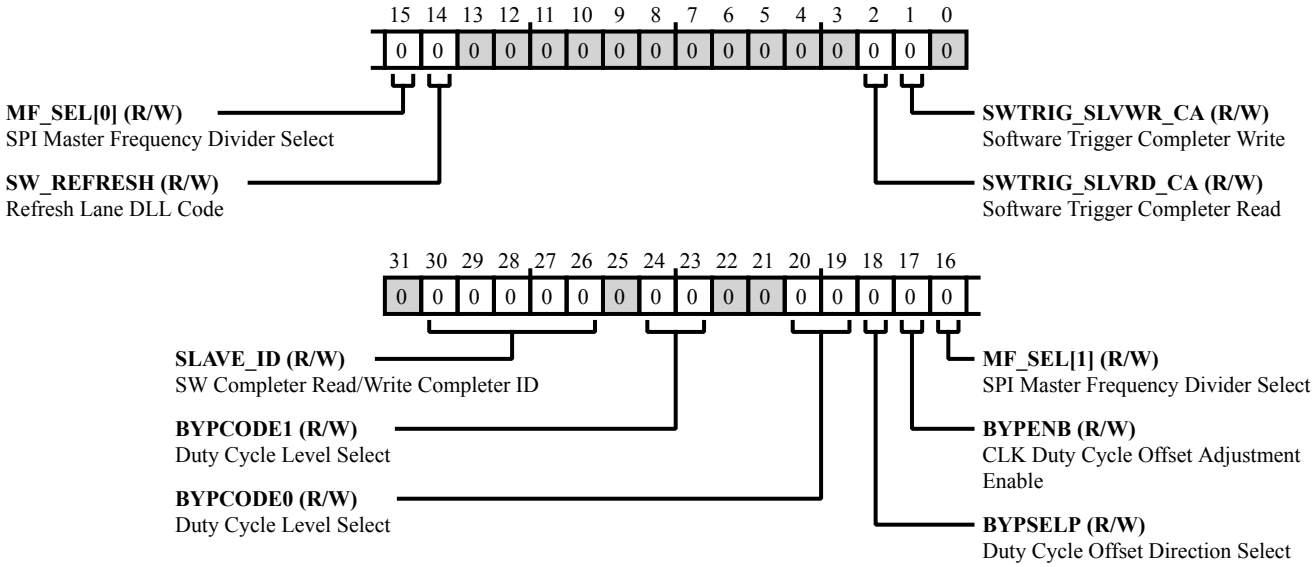


Figure 9-38: DMC_DDR_CA_CTL Register Diagram

Table 9-44: DMC_DDR_CA_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:26 (R/W)	SLAVE_ID	SW Completer Read/Write Completer ID. The `DMC_DDR_CA_CTL.SLAVE_ID` bit field is the completer ID for SW Completer read or write transactions.	
24:23 (R/W)	BYPCODE1	Duty Cycle Level Select. The `DMC_DDR_CA_CTL.BYPCODE1` bit field combines with BYPCODE0 to select one of four levels of duty cycle adjustment. Valid values for this bit field are 8,4,2,1, and 0. A larger value implies a larger offset. The default setting is 0.	
20:19 (R/W)	BYPCODE0	Duty Cycle Level Select. The `DMC_DDR_CA_CTL.BYPCODE0` bit field combines with BYPCODE1 to select one of four levels of duty cycle adjustment. Valid values for this bit field are 8,4,2,1, and 0. A larger value implies a larger offset. The default setting is 0.	
18 (R/W)	BYPSELP	Duty Cycle Offset Direction Select. The `DMC_DDR_CA_CTL.BYPSELP` bit selects the direction of the duty cycle offset.	
		0	Negative
		1	Positive

Table 9-44: DMC_DDR_CA_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	BYPENB	CLK Duty Cycle Offset Adjustment Enable. The `DMC_DDR_CA_CTL.BYPENB` bit enables the CLK duty cycle offset adjustment.	
		0	Disable
		1	Enable
16:15 (R/W)	MF_SEL	SPI Master Frequency Divider Select. The `DMC_DDR_CA_CTL.MF_SEL` bit selects the SPI master frequency divider for root to lane transfers.	
		0	Divide by 2
		1	Divide by 4
		2	Divide by 8
		3	Divide by 16
14 (R/W)	SW_REFRESH	Refresh Lane DLL Code.	
2 (R/W)	SWTRIG_SLVRD_CA	Software Trigger Completer Read.	
1 (R/W)	SWTRIG_SLVWR_CA	Software Trigger Completer Write.	

Data Lane 0 Control Register 0

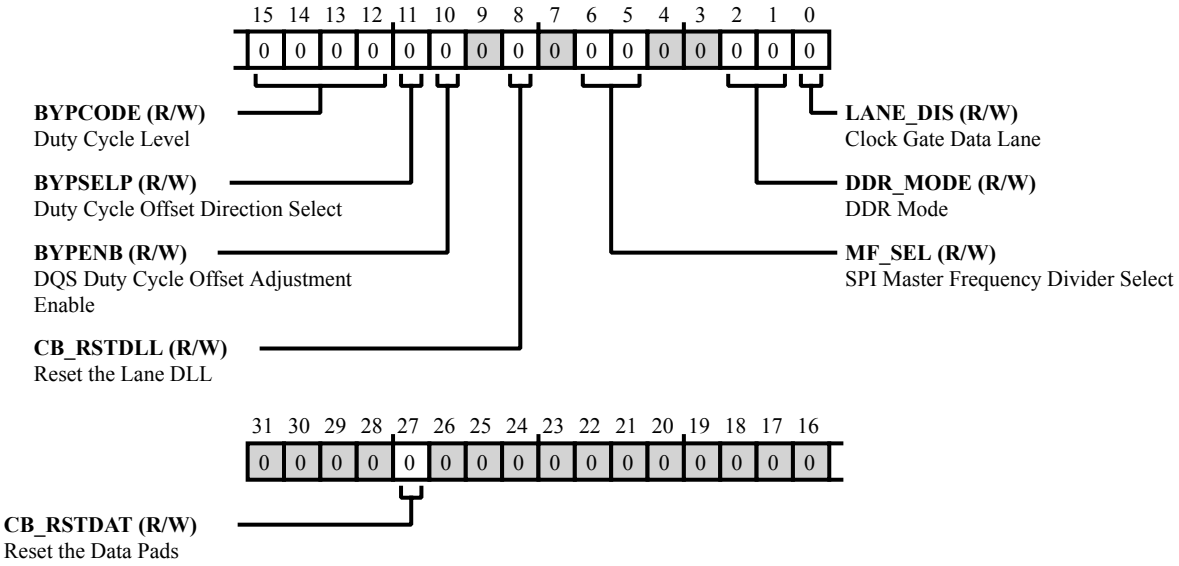


Figure 9-39: DMC_DDR_LANE0_CTL0 Register Diagram

Table 9-45: DMC_DDR_LANE0_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	CB_RSTDAT	Reset the Data Pads.	
15:12 (R/W)	BYPCODE	Duty Cycle Level. Valid values for this bit field are 8,4,2,1, and zero. A larger value implies a larger offset. The default setting is 0.	
11 (R/W)	BYPSELP	Duty Cycle Offset Direction Select. The `DMC_DDR_LANE0_CTL0.BYPSELP` bit selects the direction of the duty cycle offset.	
		0	Negative
		1	Positive
10 (R/W)	BYPENB	DQS Duty Cycle Offset Adjustment Enable. The `DMC_DDR_LANE0_CTL0.BYPENB` bit enables the DQS duty cycle offset adjustment.	
		0	Disable
		1	Enable
8 (R/W)	CB_RSTDLL	Reset the Lane DLL.	

Table 9-45: DMC_DDR_LANE0_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:5 (R/W)	MF_SEL	SPI Master Frequency Divider Select. The `DMC_DDR_LANE0_CTL0.MF_SEL` bit selects the SPI master frequency divider for root to lane transfers.	
		0	Divide by 2
		1	Divide by 4
		2	Divide by 8
		3	Divide by 16
2:1 (R/W)	DDR_MODE	DDR Mode.	
		0	DDR3/3L Mode
0 (R/W)	LANE_DIS	Clock Gate Data Lane.	
		0	Lane is Enabled
		1	Lane is Clock Gated

Data Lane 0 Control Register 1

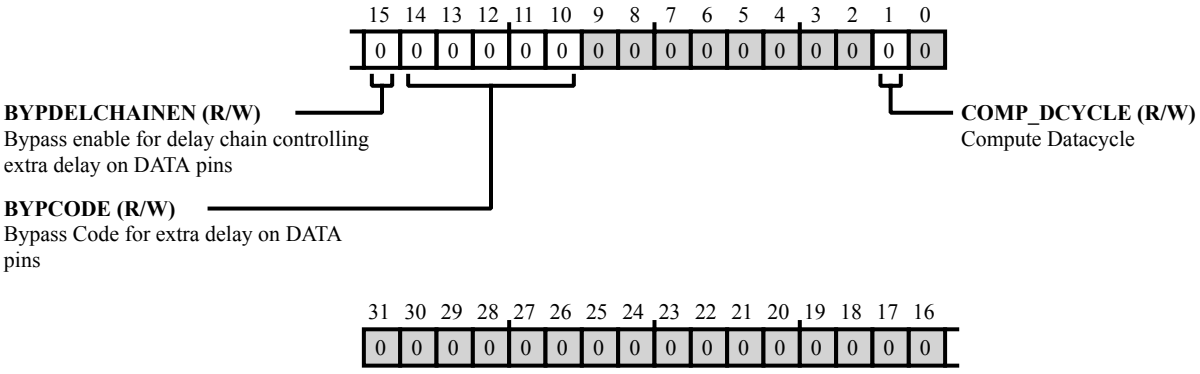


Figure 9-40: DMC_DDR_LANE0_CTL1 Register Diagram

Table 9-46: DMC_DDR_LANE0_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	BYPDELCHAINEN	Bypass enable for delay chain controlling extra delay on DATA pins.
14:10 (R/W)	BYPCODE	Bypass Code for extra delay on DATA pins.
1 (R/W)	COMP_DCYCLE	Compute Datacycle.

Data Lane 1 Control Register 0

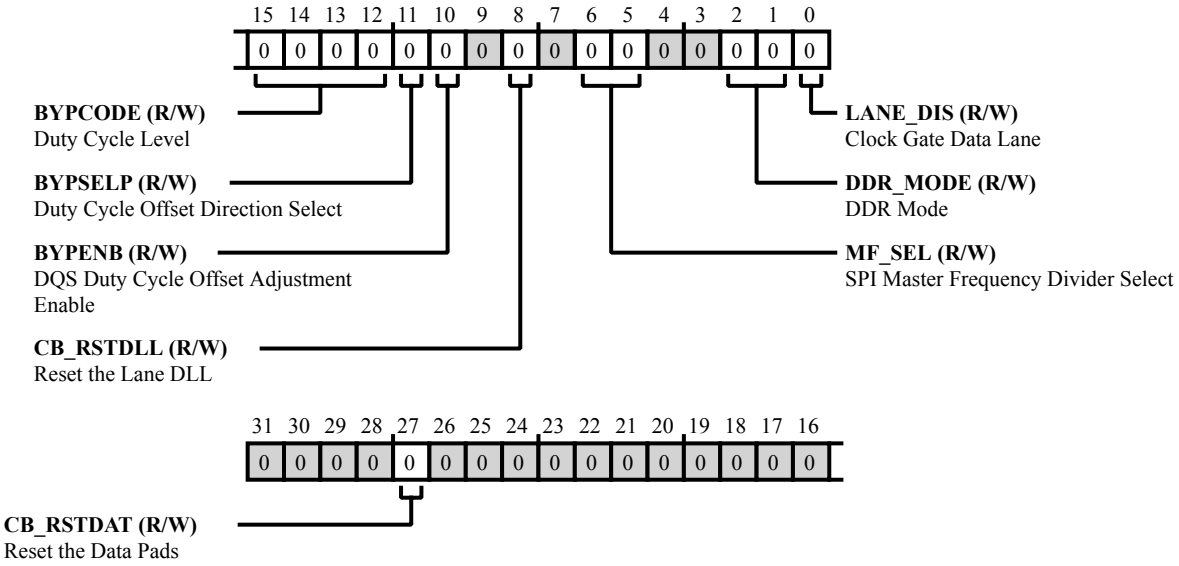


Figure 9-41: DMC_DDR_LANE1_CTL0 Register Diagram

Table 9-47: DMC_DDR_LANE1_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	CB_RSTDAT	Reset the Data Pads.	
15:12 (R/W)	BYPCODE	Duty Cycle Level. Valid values for this bit field are 8,4,2,1, and zero. A larger value implies a larger offset. The default setting is 0.	
11 (R/W)	BYPSELP	Duty Cycle Offset Direction Select. The `DMC_DDR_LANE1_CTL0.BYPSELP` bit selects the direction of the duty cycle offset.	
		0	Negative
		1	Positive
10 (R/W)	BYPENB	DQS Duty Cycle Offset Adjustment Enable. The `DMC_DDR_LANE1_CTL0.BYPENB` bit enables the DQS duty cycle offset adjustment.	
		0	Disable
		1	Enable
8 (R/W)	CB_RSTDLL	Reset the Lane DLL.	

Table 9-47: DMC_DDR_LANE1_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:5 (R/W)	MF_SEL	SPI Master Frequency Divider Select. The `DMC_DDR_LANE1_CTL0.MF_SEL` bit selects the SPI master frequency divider for root to lane transfers.	
		0	Divide by 2
		1	Divide by 4
		2	Divide by 8
		3	Divide by 16
2:1 (R/W)	DDR_MODE	DDR Mode.	
		0	DDR3/3L Mode
0 (R/W)	LANE_DIS	Clock Gate Data Lane.	
		0	Lane is Enabled
		1	Lane is Clock Gated

Data Lane 1 Control Register 1

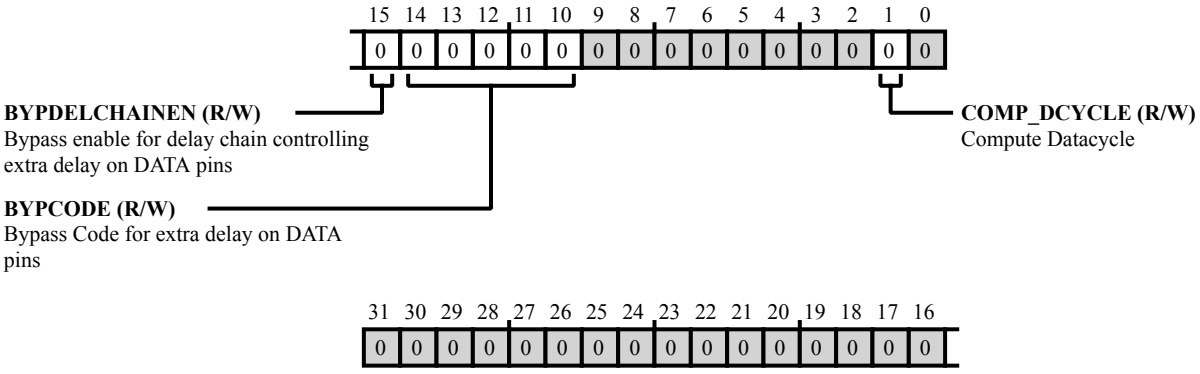


Figure 9-42: DMC_DDR_LANE1_CTL1 Register Diagram

Table 9-48: DMC_DDR_LANE1_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	BYPDELCHAINEN	Bypass enable for delay chain controlling extra delay on DATA pins.
14:10 (R/W)	BYPCODE	Bypass Code for extra delay on DATA pins.
1 (R/W)	COMP_DCYCLE	Compute Datacycle.

DDR ROOT Module Control Register

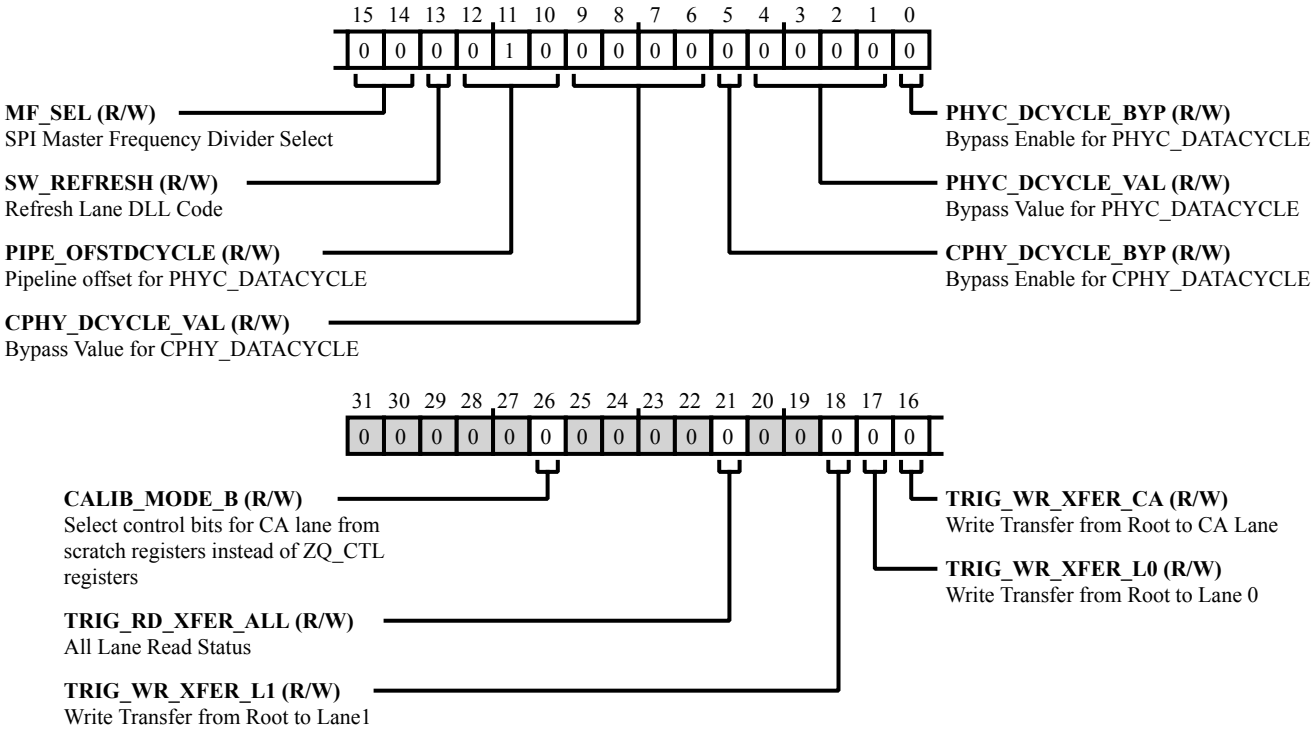


Figure 9-43: DMC_DDR_ROOT_CTL Register Diagram

Table 9-49: DMC_DDR_ROOT_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
26 (R/W)	CALIB_MODE_B	Select control bits for CA lane from scratch registers instead of ZQ_CTL registers.
21 (R/W)	TRIG_RD_XFER_ALL	All Lane Read Status. Software trigger to read the status data from all lanes.
18 (R/W)	TRIG_WR_XFER_L1	Write Transfer from Root to Lane1. Software trigger for a write transfer from the root to lane 1.
17 (R/W)	TRIG_WR_XFER_L0	Write Transfer from Root to Lane 0. Software trigger for a write transfer from the root to lane 0.
16 (R/W)	TRIG_WR_XFER_CA	Write Transfer from Root to CA Lane. Software trigger for a write transfer from the root to CA lane.

Table 9-49: DMC_DDR_ROOT_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:14 (R/W)	MF_SEL	SPI Master Frequency Divider Select. The `DMC_DDR_ROOT_CTL.MF_SEL` bit selects the SPI master frequency divider for root to lane transfers.	
		0	Divide by 2
		1	Divide by 4
		2	Divide by 8
		3	Divide by 16
13 (R/W)	SW_REFRESH	Refresh Lane DLL Code.	
12:10 (R/W)	PIPE_OFSTDCYCLE	Pipeline offset for PHYC_DATACYCLE.	
9:6 (R/W)	CPHY_DCYCLE_VAL	Bypass Value for CPHY_DATACYCLE.	
5 (R/W)	CPHY_DCYCLE_BYP	Bypass Enable for CPHY_DATACYCLE.	
4:1 (R/W)	PHYC_DCYCLE_VAL	Bypass Value for PHYC_DATACYCLE.	
0 (R/W)	PHYC_DCYCLE_BYP	Bypass Enable for PHYC_DATACYCLE.	

Scratch Register 2

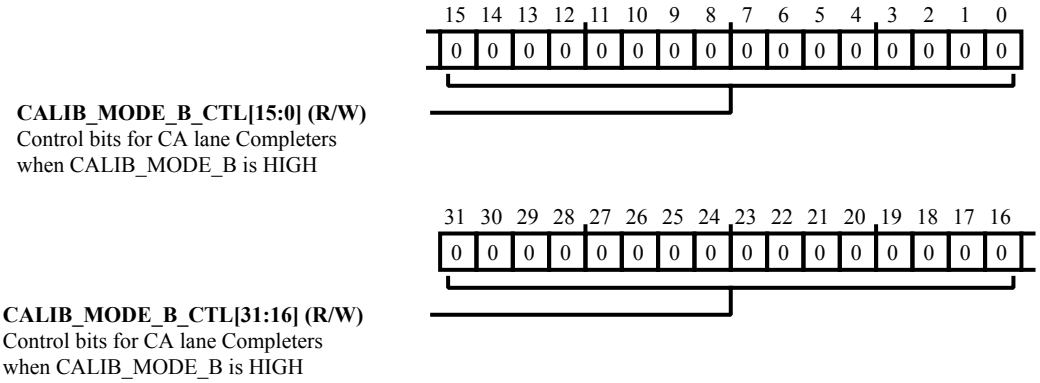


Figure 9-44: DMC_DDR_SCRATCH_2 Register Diagram

Table 9-50: DMC_DDR_SCRATCH_2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CALIB_MODE_B_CTL	Control bits for CA lane Completers when CALIB_MODE_B is HIGH.

Scratch Register 3

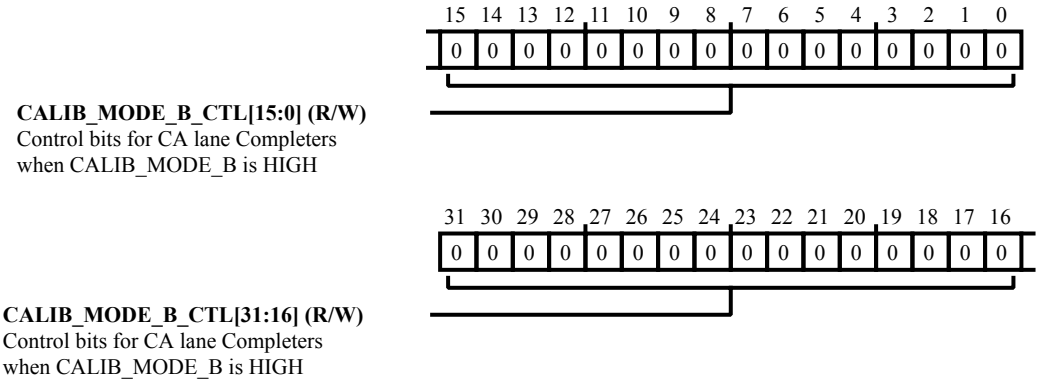


Figure 9-45: DMC_DDR_SCRATCH_3 Register Diagram

Table 9-51: DMC_DDR_SCRATCH_3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CALIB_MODE_B_CTL	Control bits for CA lane Completers when CALIB_MODE_B is HIGH.

Scratch Register 4

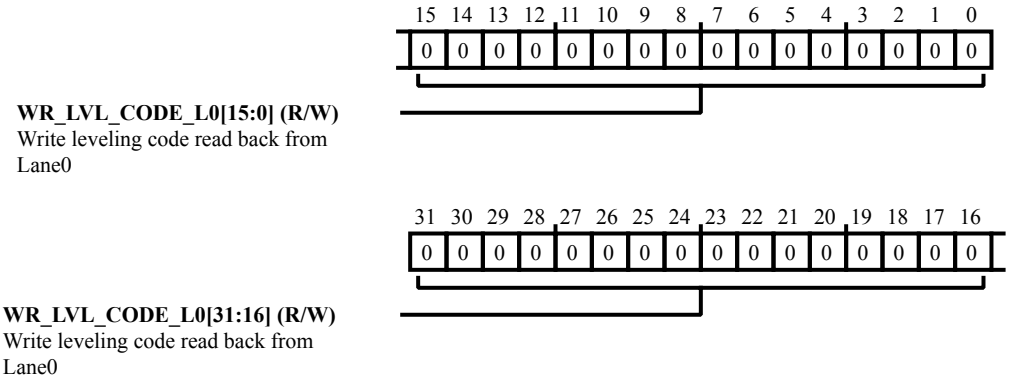


Figure 9-46: DMC_DDR_SCRATCH_4 Register Diagram

Table 9-52: DMC_DDR_SCRATCH_4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	WR_LVL_CODE_L0	Write leveling code read back from Lane0.

Scratch Register 5

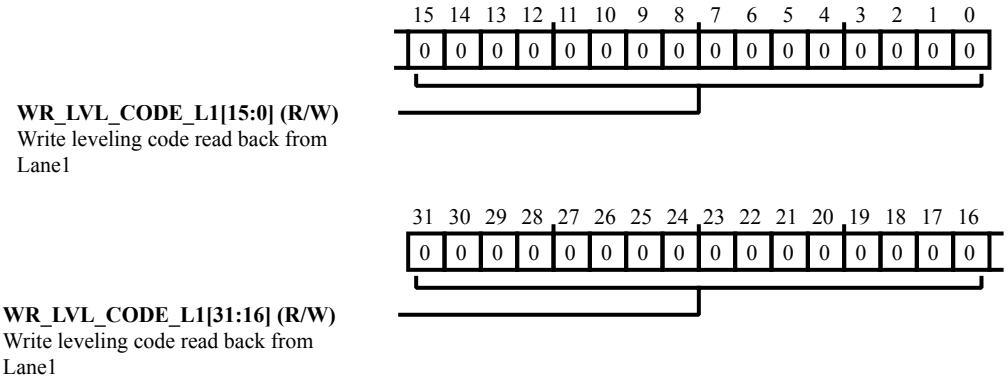


Figure 9-47: DMC_DDR_SCRATCH_5 Register Diagram

Table 9-53: DMC_DDR_SCRATCH_5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	WR_LVL_CODE_L1	Write leveling code read back from Lane1.

Scratch Register 6

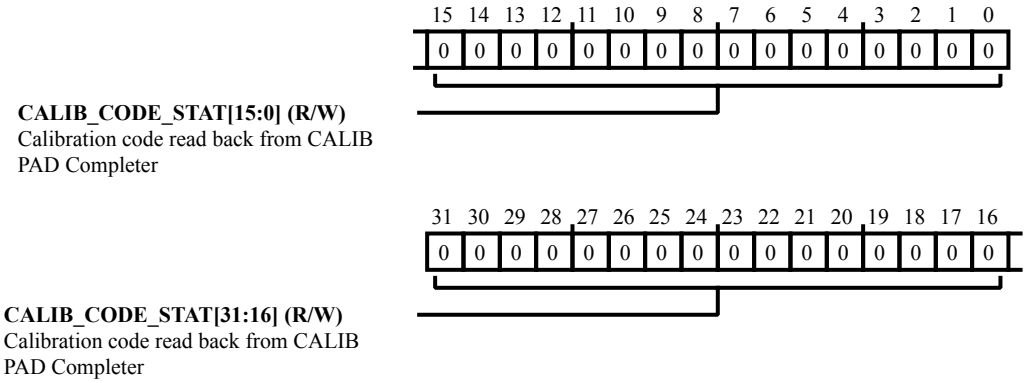


Figure 9-48: DMC_DDR_SCRATCH_6 Register Diagram

Table 9-54: DMC_DDR_SCRATCH_6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CALIB_CODE_STAT	Calibration code read back from CALIB PAD Completer.

Scratch Register 7

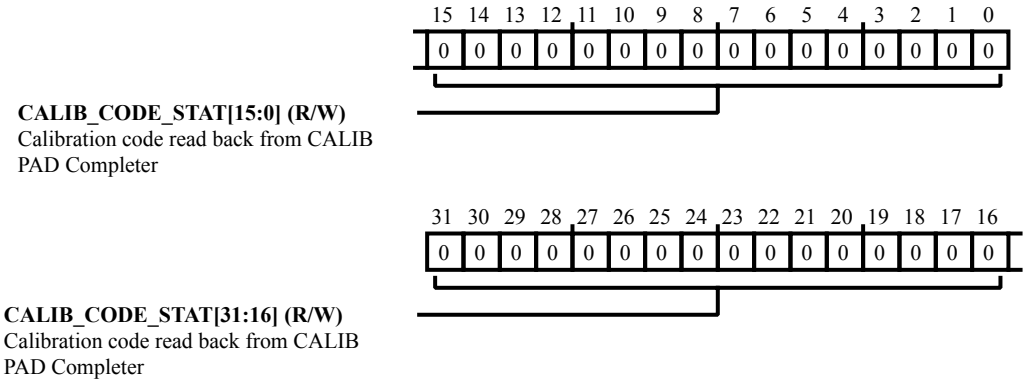


Figure 9-49: DMC_DDR_SCRATCH_7 Register Diagram

Table 9-55: DMC_DDR_SCRATCH_7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CALIB_CODE_STAT	Calibration code read back from CALIB PAD Completer.

DDR Calibration Control Register 0

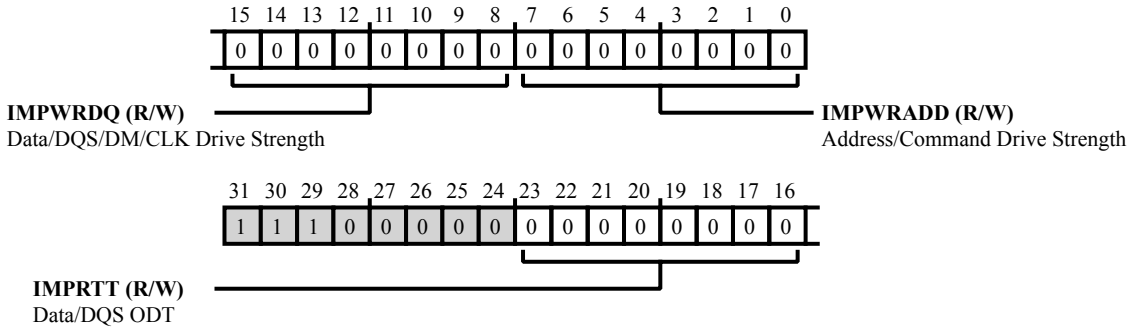


Figure 9-50: DMC_DDR_ZQ_CTL0 Register Diagram

Table 9-56: DMC_DDR_ZQ_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	IMPRTT	Data/DQS ODT. Desired ODT in ohms for Data and DQS pads.
15:8 (R/W)	IMPWRDQ	Data/DQS/DM/CLK Drive Strength. Desired drive strength in ohms for Data, DQS, DM, and clock pads. A drive strength of 100 ohms is recommended.
7:0 (R/W)	IMPWRADD	Address/Command Drive Strength. Desired drive strength in ohms for address and command pads. A drive strength of 100 ohms is recommended.

DDR Calibration Control Register 1

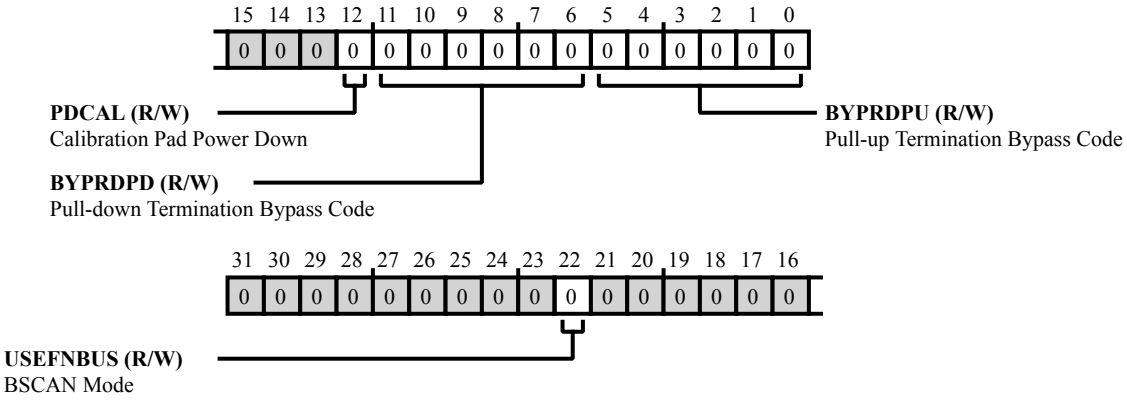


Figure 9-51: DMC_DDR_ZQ_CTL1 Register Diagram

Table 9-57: DMC_DDR_ZQ_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
22 (R/W)	USEFNBUS	BSCAN Mode. The bit controls whether to use bypass codes or functional codes in BSCAN mode.
12 (R/W)	PDCAL	Calibration Pad Power Down. Software bypass to power down the calibration pad.
11:6 (R/W)	BYPRDPD	Pull-down Termination Bypass Code.
5:0 (R/W)	BYPRDPU	Pull-up Termination Bypass Code.

DDR Calibration Control Register 2

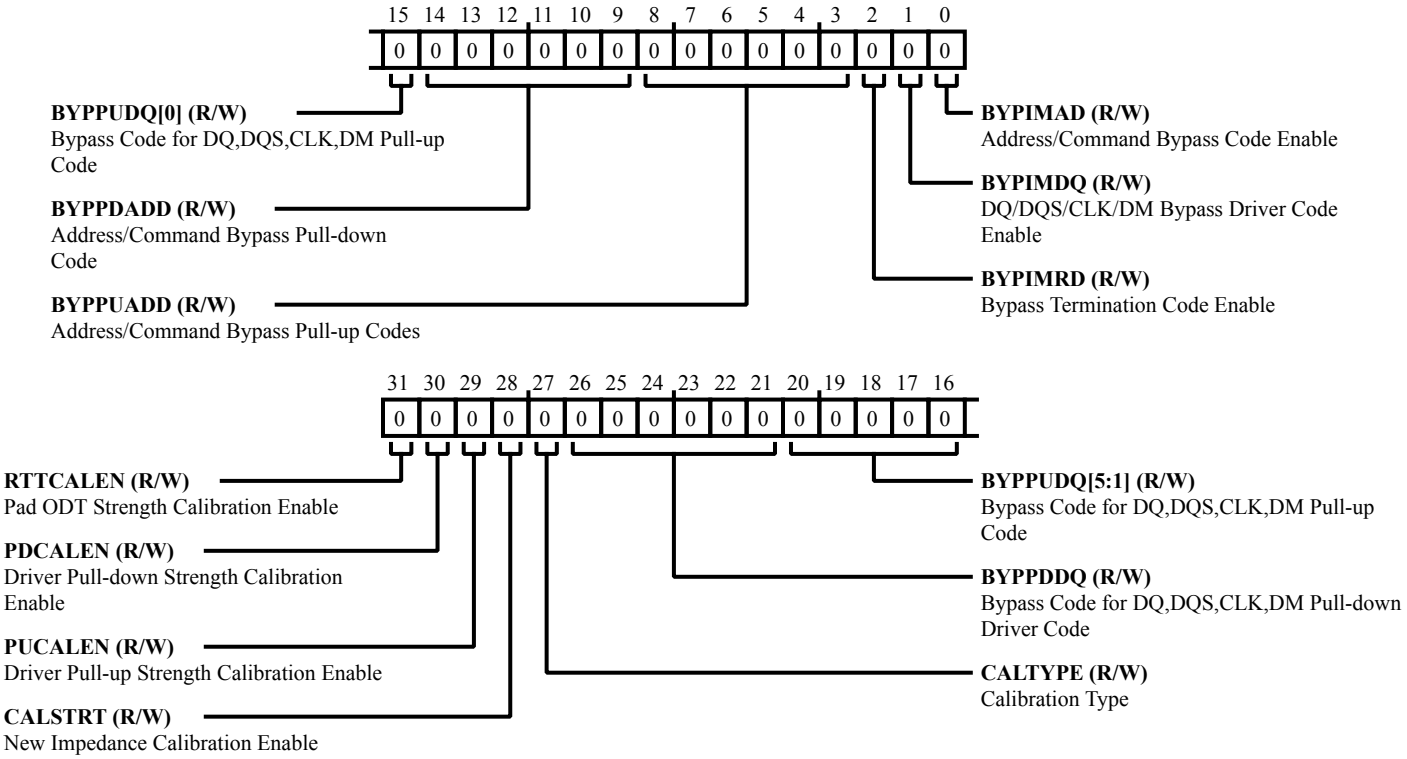


Figure 9-52: DMC_DDR_ZQ_CTL2 Register Diagram

Table 9-58: DMC_DDR_ZQ_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	RTTCALEN	Pad ODT Strength Calibration Enable.
30 (R/W)	PDCALEN	Driver Pull-down Strength Calibration Enable.
29 (R/W)	PUCALEN	Driver Pull-up Strength Calibration Enable.
28 (R/W)	CALSTRT	New Impedance Calibration Enable.
27 (R/W)	CALTYPE	Calibration Type. Short or Long Calibration
26:21 (R/W)	BYPPDDQ	Bypass Code for DQ,DQS,CLK,DM Pull-down Driver Code.

Table 9-58: DMC_DDR_ZQ_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20:15 (R/W)	BYPPUDQ	Bypass Code for DQ,DQS,CLK,DM Pull-up Code.
14:9 (R/W)	BYPPDADD	Address/Command Bypass Pull-down Code.
8:3 (R/W)	BYPPUADD	Address/Command Bypass Pull-up Codes. Bypass pull-up codes for address and command pads
2 (R/W)	BYPIMRD	Bypass Termination Code Enable.
1 (R/W)	BYPIMDQ	DQ/DQS/CLK/DM Bypass Driver Code Enable.
0 (R/W)	BYPIMAD	Address/Command Bypass Code Enable. Bypass code enable for address and command pads.

10 DDR Pre-fetch Buffer (DDRPFB)

The DDR Pre-fetch buffer is designed to reduce the latency for fetching instructions or reading data from DDR memory. The Pre-fetch buffer works in the background to reduce the latency of a subsequent data or instruction access by pre-fetching the next data for next address (using the principle of locality). The DDR pre-fetch buffer is integrated in the DDR path and uses the following features.

- SMART pre-fetching decision which is a direct access or pre-fetch based on the pattern seen in the previous read transactions issued
- ID based direct access where in a multi-controller scenario with more than four controllers avoids a slow controller (for example the SPORT) to flush out a buffer line that is being used by a fast requester (for example HSMDMA)
- Range based return zero feature

The *Pre-fetch Buffer Block Diagram* figure shows the functional blocks within the DDRPFB.

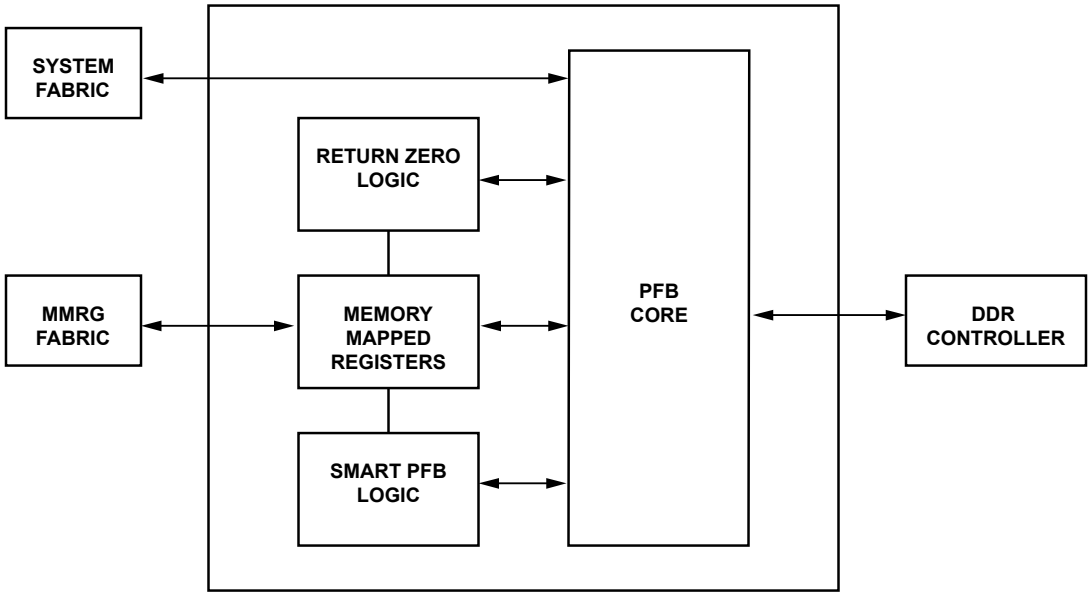


Figure 10-1: Pre-fetch Buffer Block Diagram

DDRPFB Features

The DDRPFB supports the following features:

- Smart Pre-fetch
- Direct DMC access
- Return Zero
- Controller ID based direct access
- Range based pre-fetch

Smart Pre-fetch

Smart pre-fetch is used for the worst case scenario where the controller fetches data at non continuous addresses which results no hits in the DDRPFB. To avoid fetching excess data for such non continuous read transactions, smart pre-fetching choses to start to fetch and/or pre-fetch requests from the DDRPFB to the DMC or gives direct access for such read requests. The decision is made based on the history of read transactions received at the input of the prefetch buffer. The `DDRPFB_CTL1.SMART_FEATURE_EN` bit is used to enable this feature.

Data Invalidation

When the `DDRPFB_CTL0.DATA_INVALIDATION` bit is set (=1), the complete pre-fetch buffer is invalidated. Clear this bit manually after setting it to make use of pre-fetch buffer.

Direct Access

In direct access, there are no pre-fetch accesses and the accesses are forwarded to the DMC directly. There is a one cycle latency while the pre-fetch buffer decides whether the access is direct or not. The read data from the DMC is directly forwarded to the controller without any latency impact and is not stored in the buffer.

Return Zero

This feature allows the program to configure a range of addresses range in L3 memory to always returns zero. To enable this feature, set (=1) the `DDRPFB_CTL0.RETURN_ZERO` bit. The address range which require all zeros to be read is configured using the `DDRPFB_ZERO_START.VAL` and `DDRPFB_ZERO_END.VAL` registers. All the requests which that fall between the start and end addresses (that are aligned to 256 bytes) are returned with all zeros.

Controller ID Based Direct Access

This section explains how to configure a specific SCB controller for direct access or DDR pre-fetch access. Set the `DDRPFB_CTL1.RQSTR_ID_FEATURE_EN` bit = 1 to use this feature. The following registers and bits are used to manage ID based direct access.

- The SCB controller ID is the ID of the SCB requester that is requesting data from the DDR. For ID information, see .

- The DDRPFB_RQSTR_ID_EN.VAL bit field has four bits. It can take all 16 combinations. If this bit is cleared (=0) the corresponding SCB requester has direct access.
- The DDRPFB_RQSTR0_ID, DDRPFB_RQSTR1_ID, DDRPFB_RQSTR2_ID and DDRPFB_RQSTR3_ID registers are four IDs that are preprogrammed with the SCB ID. The configuration of these registers determine whether the DDRPFB gives a normal PFB enabled access or a direct access (if the corresponding DDRPFB_RQSTR_ID_EN.VAL bit is set). If the SCB ID of the requester matches with the DDRPFB_MSTRx_ID, then the DDR pre-fetch operation is enabled.
- The DDRPFB_RQSTR0_ID_MSK, DDRPFB_RQSTR1_ID_MSK, DDRPFB_RQSTR2_ID_MSK and DDRPFB_RQSTR3_ID_MSK: When the DDRPFB_RQSTR_ID_EN bit is set for some DDRPFB_MSTRx_ID, then the PFB enabled access is provided for a partial ID match of the SCB ID with the corresponding preprogrammed DDRPFB_MSTRx_ID. Clearing a bit masks the corresponding bit and the masked bits are not checked for a match.

The *DDRPFB Access Configurations* table shows the register configurations for different access types.

Table 10-1: DDRPFB Access Configurations

SCB Requester ID	DDRPFB_MSTR_ID_EN[0]	DDRPFB_MSTR0_ID	DDRPFB_MSTR0_ID_MASK	Access Type
XXXX	0	XXXX	XXXX	Direct access Irrespective of SCB requester ID, DDRPFB_RQSTR0_ID and DDRPFB_RQSTR0_ID_MSK, direct access is given to all the requests.
MMMM	1	NNNN	0000	DDRPFB enabled access DDRPFB does not check any bits of the SCB requester ID if it matches with DDRPFB_RQSTR0_ID. Therefore, DDRPFB performs prefetch operation for all IDs. This is an example where DDR prefetch is enabled, though none of the bits of SCB ID matches with DDRPFB_RQSTR0_ID.
MMXX	1	MMXX	1100	DDRPFB enabled access DDRPFB looks for only two MSBs from SCB ID matching the two MSBs of DDRPFB_RQSTR0_ID. If they match, the DDRPFB performs a prefetch operation. Otherwise, direct access is given to such requests. In this example the two MSBs matching with the two MSBs of DDRPFB_RQSTR0_ID configures a normal DDRPFB enabled access.
MMXX	1	NNXX	1100	Direct access The PFB looks for only the two MSBs from the SCB ID that match with the two MSBs of DDRPFB_RQSTR0_ID. If they match, the PFB performs a normal operation. Otherwise, direct access is given to such requests. In this example, since the two MSBs do not match with the two MSBs of DDRPFB_RQSTR0_ID, it is a direct access.

Range Based Pre-fetch

Range based prefetch allows an entire address range to be divided into 16 address ranges to selectively disable the pre-fetch buffer (while pre-fetching is enabled) for this address range using the DDRPFB_RANGE_SELECT register. For example, the *Range Based Pre-fetch* table shows that if DDRPFB_RANGE_SELECT [9] = 0, then the PFB is disabled for address range 90000000 to 9FFFFFFF.

The DDRPFB_CTL1.RANGE_SHIFT bit field is used to configure pre-fetching 16 sections of the address map based on the most significant nibble in the input address. In the example DDRPFB_CTL1.RANGE_SHIFT is 28.

Table 10-2: Range Based Pre-fetch

Sr. No.		PFB Operation
1	DDRPFB_RANGE_SELECT [ARADDR_IN[31:28]] = 1 && DDRPFB_CTL0.PREFETCH_EN = 1	PFB enabled operation
2	DDRPFB_RANGE_SELECT [ARADDR_IN[31:28]] = 0 && DDRPFB_CTL0.PREFETCH_EN = 1	PFB disabled operation
3	DDRPFB_RANGE_SELECT [ARADDR_IN[31:28]] = X && DDRPFB_CTL0.PREFETCH_EN = 0	PFB disabled operation

Table 10-3: Range Based Address Range Selections

Input bit	Address range for Pre-fetch
RANGE_SELECT[0]	00000000 – 0FFFFFFF
RANGE_SELECT[1]	10000000 – 1FFFFFFF
RANGE_SELECT[2]	20000000 – 2FFFFFFF
RANGE_SELECT[3]	30000000 – 3FFFFFFF
RANGE_SELECT[4]	40000000 – 4FFFFFFF
RANGE_SELECT[5]	50000000 – 5FFFFFFF
RANGE_SELECT[6]	60000000 – 6FFFFFFF
RANGE_SELECT[7]	70000000 – 7FFFFFFF
RANGE_SELECT[8]	80000000 – 8FFFFFFF
RANGE_SELECT[9]	90000000 – 9FFFFFFF
RANGE_SELECT[10]	A0000000 – AFFFFFFF
RANGE_SELECT[11]	B0000000 – BFFFFFFF
RANGE_SELECT[12]	C0000000 – CFFFFFFF
RANGE_SELECT[13]	D0000000 – DFFFFFFF
RANGE_SELECT[14]	E0000000 – EFFFFFFF
RANGE_SELECT[15]	F0000000 – FFFFFFFF

Programming Notes

The following guidelines should be noted when using range based pre-fetch.

- DDRPFB_RANGE_SELECT.VAL is used to change the prefetch range
- When DDRPFB_RANGE_SELECT.VAL = 5'bN, then ARADDR_IN[N+3:N] are used for range selection (here, N is programmable number)
  - When DDRPFB_CTL1.RANGE_SHIFT = 28, logic behavior is same as above table
  - When DDRPFB_CTL1.RANGE_SHIFT = 26, then ARADDR_IN[29:26] are used for range selection

For finer control of a small range the configurations in the following table can be used.

Sr. No.		PFB Operation
1	DDRPFB_RANGE_SELECT [ARADDR_IN[N+3:N]] = 1 && DDRPFB_CTL0.PREFETCH_EN = 1	PFB enabled operation
2	DDRPFB_RANGE_SELECT [ARADDR_IN[N+3:N]] = 0 && DDRPFB_CTL0.PREFETCH_EN = 1	PFB disabled operation
3	DDRPFB_RANGE_SELECT [ARADDR_IN[N+3:N]] = X(don't care) && DDRPFB_CTL0.PREFETCH_EN = 0	PFB disabled operation

ADSP-SC59x DDRPFB Register List

PFB

Table 10-4: ADSP-SC59x DDRPFB Register List

Name	Description
DDRPFB_CTL0	Control Register 0
DDRPFB_CTL1	Control Register 1
DDRPFB_RANGE_SELECT	Range Select Register
DDRPFB_RQSTR0_ID	Requester ID 0 Register
DDRPFB_RQSTR0_ID_MSK	Requester ID 0 Mask Register
DDRPFB_RQSTR1_ID	Requester ID 1 Register
DDRPFB_RQSTR1_ID_MSK	Requester ID 1 Mask Register
DDRPFB_RQSTR2_ID	Requester ID 2 Register
DDRPFB_RQSTR2_ID_MSK	Requester ID 2 Mask Register
DDRPFB_RQSTR3_ID	Requester ID 3 Register
DDRPFB_RQSTR3_ID_MSK	Requester ID 3 Mask Register
DDRPFB_RQSTR_ID_EN	Requester ID Enable Register

Table 10-4: ADSP-SC59x DDRPFB Register List (Continued)

Name	Description
DDRPFB_STS0	Prefetch Buffer First Status Register
DDRPFB_ZERO_END	Return Zero End Address Register
DDRPFB_ZERO_START	Return Zero Start Address Register

DDRPFB Programming Model

The following sections describe how to program the DDRPFB.

Enabling the Pre-fetch Buffer and PFB Idle State

Set the DDRPFB_CTL0.PREFETCH_EN bit (=1) to enable the DDR pre-fetch buffer.

When the DDRPFB_STS0.PFB_IDLE bit = 1 there are no pending transactions between the DDRPFB and the DMC. When this bit = 0, there is at least one pending transaction between the DDRPFB and the DMC.

General Programming Guidelines

The following are few guidelines which need to be followed for proper functioning of DDR PFB.

- The DDRPFB_STS0.PFB_IDLE indicates whether the DDRPFB is IDLE or not and can be polled to add delay to allow pending requests from the PFB to completer get completed.
- To issue a DMC reset during operation, the following steps are used to maintain data coherence and prevent system hangs.
  1. Block all requesters that are currently accessing DDR
  2. Check that the PFB is in the IDLE state by checking DDRPFB_STS0.PFB_IDLE
  3. Issue a DMC reset
  4. Invalidate the PFB
  5. Unblock all requesters that need access to DDR
- During DDR initialization the PFB should be disabled. If the program needs to have the PFB enabled during DDR initialization, perform a PFB invalidation after DDR initialization is completed.
- The maximum possible value of DDRPFB_CTL1.RANGE_SHIFT is ‘b11100. Any value that is more than ‘b11100 has no significance because the address bus width is 32-bits.

Guidelines For Achieving Maximum Performance

To achieve maximum performance, please note the following:

- Each DDR PFB line can hold 256 bytes of data. To achieve maximum performance the starting address of sequential memory accesses should be 256 bytes aligned

- Enable DDRPFB for requesters that support burst mode
- Burst mode with a burse size of 128 bits provide good performance compared to other combinations of burst size less than 128 bits
- Use the Requester ID based direct access feature for the SPORTs and IIR when there are more than four requesters acting at a time.
- Multiple requesters can be combined as a one requester when they are fetching data in interleaved fashion. This is accomplished by placing data in DDR in such a way that (for example), address A is for requester 1, A+x for requester 2, A+2x for again requester 1 and A +3x for requester 2. So in this example for requester 1 addresses A + 2nx are used and for requester 2 addresses A + (2n+1)x are used, where n = 0,1,2, and so on.
- Avoiding wrap access – Data defined in source code can be placed in memory in such a way that all requests from requester A55 is cache line aligned so that all requests from cache onwards are incremental accesses.
- The decision should be made on what is most suitable with respect to performance out of the various DDRPFB configurations, including DDRPFB disabled. In most cases PFB = enabled and the smart feature = enabled has better performance. For few use-cases PFB = enabled, smart feature disabled work best.

ADSP-SC59x DDRPFB Register Descriptions

Prefetch buffer mmr registers (DDRPFB) contains the following registers.

Table 10-5: ADSP-SC59x DDRPFB Register List

Name	Description
DDRPFB_CTL0	Control Register 0
DDRPFB_CTL1	Control Register 1
DDRPFB_RANGE_SELECT	Range Select Register
DDRPFB_RQSTR0_ID	Requester ID 0 Register
DDRPFB_RQSTR0_ID_MSK	Requester ID 0 Mask Register
DDRPFB_RQSTR1_ID	Requester ID 1 Register
DDRPFB_RQSTR1_ID_MSK	Requester ID 1 Mask Register
DDRPFB_RQSTR2_ID	Requester ID 2 Register
DDRPFB_RQSTR2_ID_MSK	Requester ID 2 Mask Register
DDRPFB_RQSTR3_ID	Requester ID 3 Register
DDRPFB_RQSTR3_ID_MSK	Requester ID 3 Mask Register
DDRPFB_RQSTR_ID_EN	Requester ID Enable Register
DDRPFB_STS0	Prefetch Buffer First Status Register
DDRPFB_ZERO_END	Return Zero End Address Register
DDRPFB_ZERO_START	Return Zero Start Address Register

Control Register 0

The DDRPFB_CTL0 register disables prefetching for up to sixteen address ranges. Bits[31:16] bits control the range selection for the instruction cache, and bits [15:0] control the range selection for the data cache.

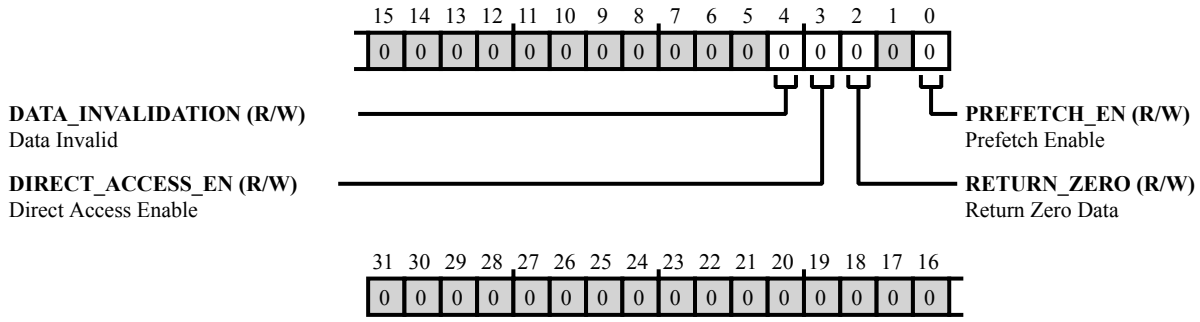


Figure 10-2: DDRPFB_CTL0 Register Diagram

Table 10-6: DDRPFB_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	DATA_INVALIDATION	Data Invalid. When the DDRPFB_CTL0.DATA_INVALIDATION bit is set the prefetch buffer is invalid. The program must clear this bit manually.	
		0	Buffer valid
		1	Buffer invalid
3 (R/W)	DIRECT_ACCESS_EN	Direct Access Enable. When the DDRPFB_CTL0.DIRECT_ACCESS_EN bit is set (=1) the prefetch buffer passes requests as is from a requester to a completer. No excess data is fetched and stored in prefetch buffer. When the DDRPFB_CTL0.DIRECT_ACCESS_EN bit is cleared (=0) the prefetch buffer behaves as if it is disabled.	
		0	Direct access disabled
		1	Direct access enabled
2 (R/W)	RETURN_ZERO	Return Zero Data. When the DDRPFB_CTL0.RETURN_ZERO bit is set, the prefetch buffer returns zero data for the address range configured in the DDRPFB_ZERO_START and DDRPFB_ZERO_END registers. The addresses in the DDRPFB_ZERO_START and DDRPFB_ZERO_END registers should be 128 byte aligned.	
		0	Return zero disabled
		1	Return zero data in selected address range

Table 10-6: DDRPFB_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	PREFETCH_EN	Prefetch Enable. When the `DDRPFB_CTL0.PREFETCH_EN` bit is set (=1), the prefetch buffer is enabled. Otherwise, the prefetch buffer is disabled.	
		0	Prefetch disabled
		1	Prefetch enabled

Control Register 1

The DDRPFB_CTL1 register enables smart prefetch, range shifting, and direct access.

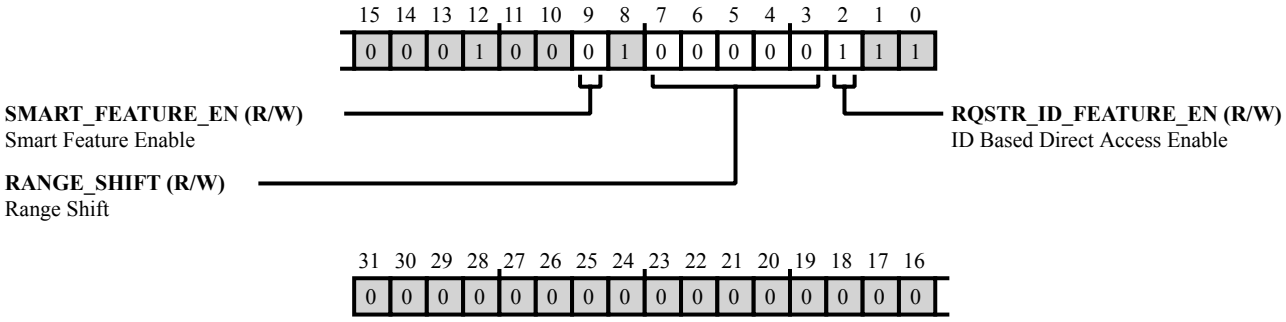


Figure 10-3: DDRPFB_CTL1 Register Diagram

Table 10-7: DDRPFB_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	SMART_FEATURE_EN	Smart Feature Enable. Setting the DDRPFB_CTL1.SMART_FEATURE_EN bit enables the SMART feature.	
		0	Smart prefetch disabled
		1	Smart prefetch enabled
7:3 (R/W)	RANGE_SHIFT	Range Shift. The DDRPFB_CTL1.RANGE_SHIFT bit field determines which bits of the read address (that appear at the input of prefetch buffer) are used for the range select operation. By default this is the four MSBs are used. When this bit field is programmed to 5hN then [N+3:N] address bits are used for the range select operation.	
2 (R/W)	RQSTR_ID_FEATURE_EN	ID Based Direct Access Enable. The DDRPFB_CTL1.RQSTR_ID_FEATURE_EN bit, when set (=1) enables the ID based direct access feature.	
		0	ID based direct access disabled
		1	ID based direct access enabled

Range Select Register

The DDRPFB_RANGE_SELECT register determines whether the prefetch buffer is enabled or disabled in the specified address range.

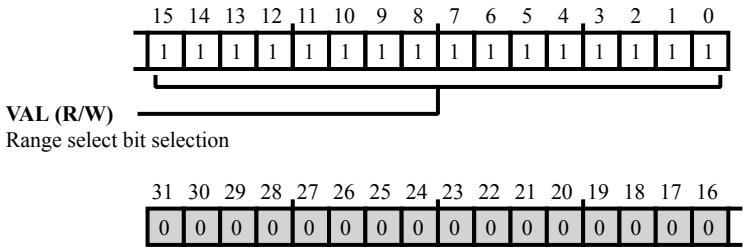


Figure 10-4: DDRPFB_RANGE_SELECT Register Diagram

Table 10-8: DDRPFB_RANGE_SELECT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VAL	Range select bit selection. Each bit in this bit field determines prefetch buffer should be enabled or disabled in that address range. This bit field is used in conjunction with RANGE_SHIFT bit field.

Requester ID 0 Register

The DDRPFB_RQSTR0_ID register is used to configure the first requester ID for controller ID based direct access operations.

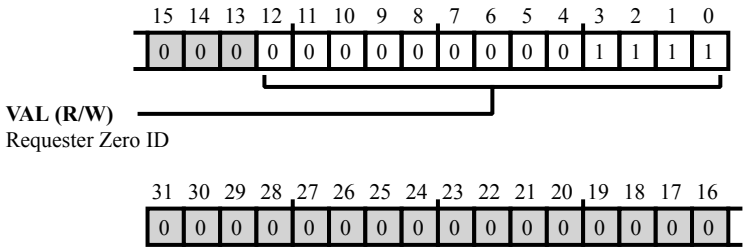


Figure 10-5: DDRPFB_RQSTR0_ID Register Diagram

Table 10-9: DDRPFB_RQSTR0_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Zero ID. The DDRPFB_RQSTR0_ID.VAL bit field identifies the first requester ID to be considered for ID based direct access.

Requester ID 0 Mask Register

The DDRPFB_RQSTR0_ID_MSK register is used to mask the first requester ID for controller ID based direct access operations. The masked bits are not used in direct access operations.

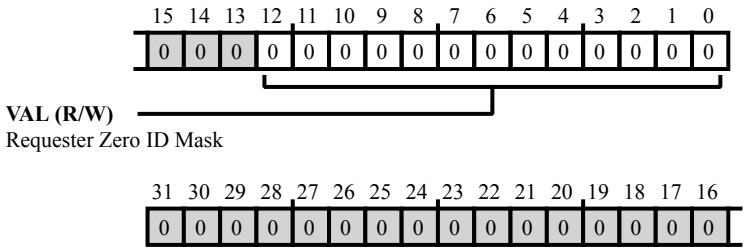


Figure 10-6: DDRPFB_RQSTR0_ID_MSK Register Diagram

Table 10-10: DDRPFB_RQSTR0_ID_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Zero ID Mask. The DDRPFB_RQSTR0_ID_MSK.VAL bit field masks the first requester ID to exclude from ID based direct access.

Requester ID 1 Register

The DDRPFB_RQSTR1_ID register is used to configure the first requester ID for controller ID based direct access operations.

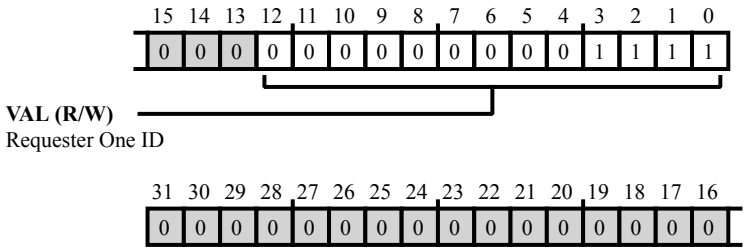


Figure 10-7: DDRPFB_RQSTR1_ID Register Diagram

Table 10-11: DDRPFB_RQSTR1_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester One ID. The DDRPFB_RQSTR1_ID.VAL bit field identifies the second requester ID to be considered for ID based direct access.

Requester ID 1 Mask Register

The DDRPFB_RQSTR1_ID_MSK register is used to mask the second requester ID for controller ID based direct access operations. The masked bits are not used in direct access operations.

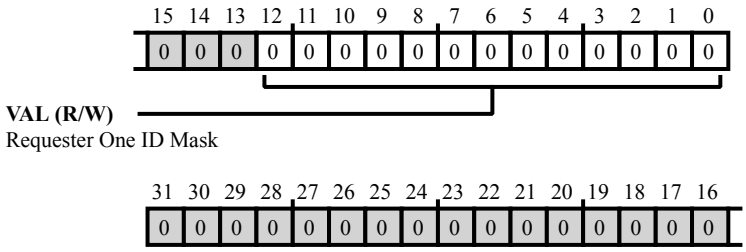


Figure 10-8: DDRPFB_RQSTR1_ID_MSK Register Diagram

Table 10-12: DDRPFB_RQSTR1_ID_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester One ID Mask. The DDRPFB_RQSTR1_ID_MSK.VAL bit field masks the second requester ID to exclude from ID based direct access.

Requester ID 2 Register

The DDRPFB_RQSTR2_ID register is used to configure the third requester ID for controller ID based direct access operations.

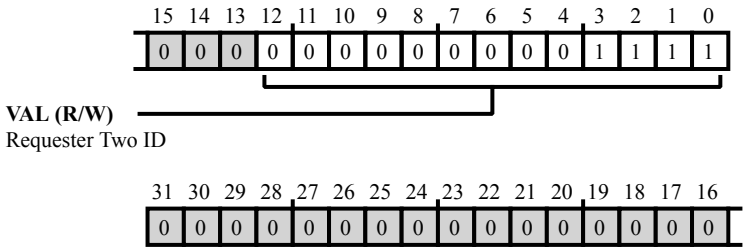


Figure 10-9: DDRPFB_RQSTR2_ID Register Diagram

Table 10-13: DDRPFB_RQSTR2_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Two ID. The DDRPFB_RQSTR2_ID.VAL bit field identifies the third requester ID to be considered for ID based direct access.

Requester ID 2 Mask Register

The DDRPFB_RQSTR2_ID_MSK register is used to mask the third requester ID for controller ID based direct access operations. The masked bits are not used in direct access operations.

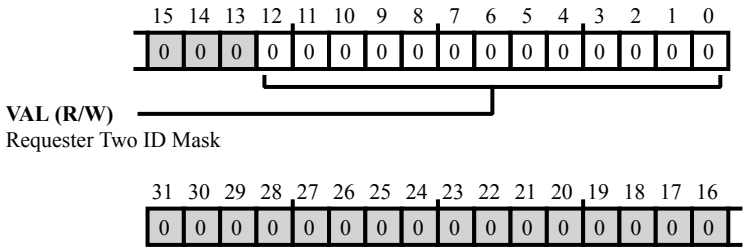


Figure 10-10: DDRPFB_RQSTR2_ID_MSK Register Diagram

Table 10-14: DDRPFB_RQSTR2_ID_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Two ID Mask. The DDRPFB_RQSTR2_ID_MSK.VAL bit field masks the third requester ID to exclude from ID based direct access.

Requester ID 3 Register

The DDRPFB_RQSTR3_ID register is used to configure the fourth requester ID for controller ID based direct access operations.

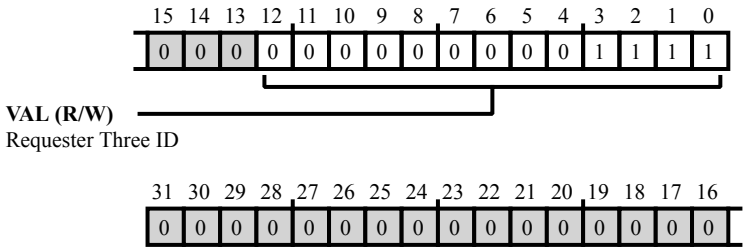


Figure 10-11: DDRPFB_RQSTR3_ID Register Diagram

Table 10-15: DDRPFB_RQSTR3_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Three ID. The DDRPFB_RQSTR3_ID.VAL bit field identifies the fourth requester ID to be considered for ID based direct access.

Requester ID 3 Mask Register

The DDRPFB_RQSTR3_ID_MSK register is used to mask the fourth requester ID for controller ID based direct access operations. The masked bits are not used in direct access operations.

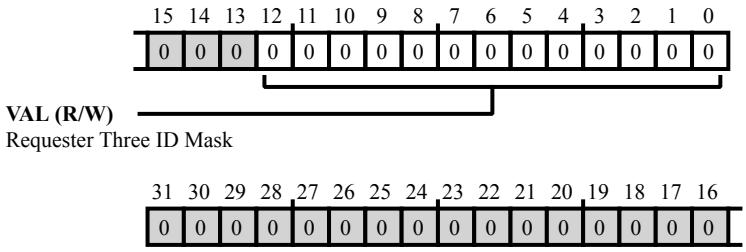


Figure 10-12: DDRPFB_RQSTR3_ID_MSK Register Diagram

Table 10-16: DDRPFB_RQSTR3_ID_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	VAL	Requester Three ID Mask. The DDRPFB_RQSTR3_ID_MSK.VAL bit field masks the fourth requester ID to exclude from ID based direct access.

Requester ID Enable Register

The DDRPFB_RQSTR_ID_EN register is used to configure controller based direct access. As many as four controller IDs can be programmed for direct access. For more information, see "Controller ID Based Direct Access".

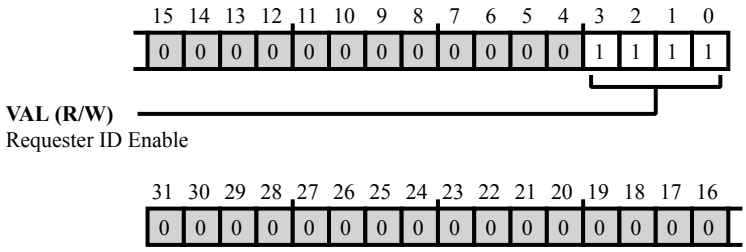


Figure 10-13: DDRPFB_RQSTR_ID_EN Register Diagram

Table 10-17: DDRPFB_RQSTR_ID_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	VAL	Requester ID Enable. The DDRPFB_RQSTR_ID_EN.VAL bits are used to configure controller based direct access. As many as four controller IDs can be programmed for direct access. For more information, see "Controller ID Based Direct Access".

Prefetch Buffer First Status Register

The DDRPFB_STS0 register indicates whether all fetch and prefetch operations from the PFB to the DDR are complete.

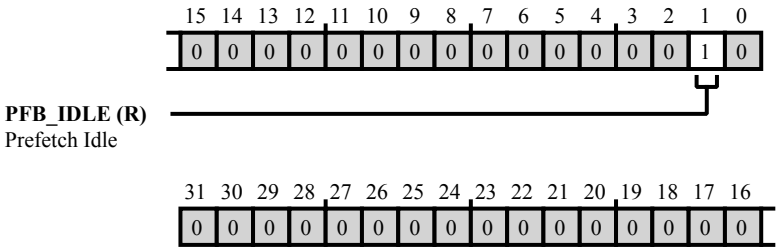


Figure 10-14: DDRPFB_STS0 Register Diagram

Table 10-18: DDRPFB_STS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	PFB_IDLE	Prefetch Idle. The DDRPFB_STS0.PFB_IDLE bit, when set (=1), indicates all fetch and prefetch operations from the PFB to the DDR are complete.

Return Zero End Address Register

The DDRPFB_ZERO_END register is used with the DDRPFB_ZERO_START register to configure an address range in L3 memory to always returns zero.

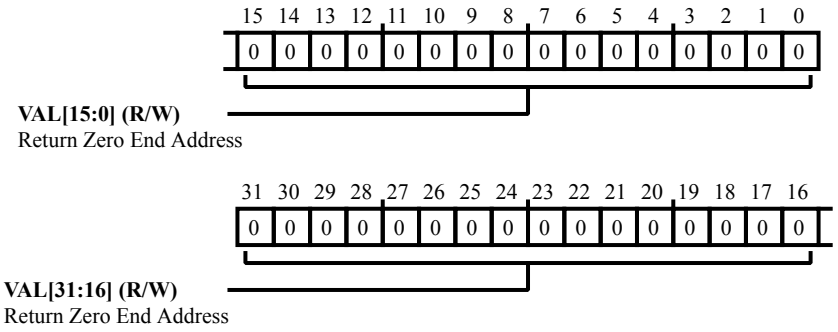


Figure 10-15: DDRPFB_ZERO_END Register Diagram

Table 10-19: DDRPFB_ZERO_END Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VAL	Return Zero End Address. The DDRPFB_ZERO_END.VAL bit field is used to configure the end address (that are aligned to 256 bytes) that are returned with all zeros.

Return Zero Start Address Register

The DDRPFB_ZERO_START register is used with the DDRPFB_ZERO_END register to configure a range of addresses in L3 memory to always returns zero.

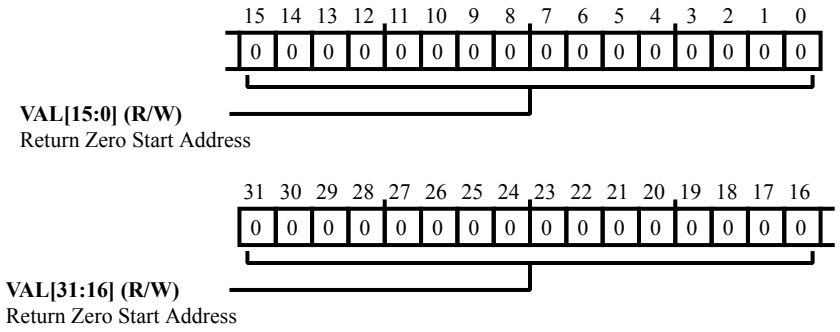


Figure 10-16: DDRPFB_ZERO_START Register Diagram

Table 10-20: DDRPFB_ZERO_START Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VAL	Return Zero Start Address. The DDRPFB_ZERO_START.VAL bit field is used to configure the start address (that are aligned to 256 bytes) that are returned with all zeros.

11 One-Time Programmable Memory Controller (OTPC)

This chapter describes the operation of the OTP controller. The OTP module is a complete system integrating an OTP memory core with a programming controller, charge pump, and voltage regulator. A built-in Hamming Code Error Correction (ECC), and a fully implemented double-redundant program or read scheme protect the OTP data.

CAUTION: OTP memory does not support burst transfers, which are required to support cache line fills. As such, OTP memory should not be made cacheable. If it is, the OTP controller returns an error when a read access is attempted.

OTPC Features

The OTP memory and controller have the following features:

- Built-in redundant read mode
- Built-in integrated power supply
- Built-in Hamming Code Error Correction (ECC)
- Full word serial (single bit at a time) programming with internal VPP

NOTE: The OTP module is protected by the System Memory Protection Unit (SMPU) and is always secure by default. Non secure access to the OTP by any module will fail unless allowed by the SMPU.

OTPC Functional Description

ADSP-SC59x OTPC Register List

The One-Time-Programmable Memory controller (OTPC) supports programming the OTP memory. A set of registers governs OTPC operations. For more information on OTPC functionality, see the OTPC register descriptions.

Table 11-1: ADSP-SC59x OTPC Register List

Name	Description
OTPC_SECU_STATE	OTP Security State Register
OTPC_STAT	OTP Status Register

ADSP-SC59x OTPC Interrupt List

Table 11-2: ADSP-SC59x OTPC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
6	OTPC0_ERR	OTPC0 Dual-bit Error	Level	

OTPC Memory Layout

This section details the memory layout of the OTPC memory.

Table 11-3: OTPC Memory Layout

Name	32 bit aligned Byte Address	Size (bits)	Description
huk	0x5c	256	Hardware Unique Key
rkek	0x7c	128	Root Key Encryption Key
dek	0x8c	128	Local Encryption Key
oem_public_key	0x9c	512	OEM Public Key
pvt_128key0	0xdc	128	Customer Privatekey 0 128bits
pvt_128key1	0xec	128	Customer Privatekey 1 128bits
pvt_128key2	0xfc	128	Customer Privatekey 2 128bits
pvt_128key3	0x10c	128	Customer Privatekey 3 128bits
ek	0x11c	256	Endorsement Key
secure_emu_key0	0x13c	128	Secure Emulation Key0
secure_emu_key1	0x14c	128	Secure Emulation Key1
reserved	0x160	928	reserved
emu_key_disable0	0x1d4	16	Secure Emulation Key0 Disable
emu_key_disable1	0x1d4	16	Secure Emulation Key1 Disable
public_key0	0x1d8	512	Customer Public Key0
public_key1	0x218	512	Customer Public Key1
reserved	0x258	128	reserved
cgu_ctl_WEN	0x268	1	CGU Config ctl_WEN

Table 11-3: OTPC Memory Layout (Continued)

Name	32 bit aligned Byte Address	Size (bits)	Description
cgu_div_WEN	0x268	1	CGU Config div_WEN
cgu_reserved0	0x268	1	CGU Config reserved0
cgu_div_DSEL	0x268	5	CGU Config div_DSEL
cgu_div_CSEL	0x268	5	CGU Config div_CSEL
cgu_div_S0SEL	0x268	3	CGU Config div_S0SEL
cgu_div_SYSSEL	0x268	5	CGU Config div_SYSSEL
cgu_div_S1SEL	0x268	3	CGU Config div_S1SEL
cgu_div_OSEL	0x268	7	CGU Config div_OSEL
cgu_ctl_DF	0x268	1	CGU Config ctl_DF
cgu_ctl_MSEL	0x26c	7	CGU Config ctl_MSEL
cgu_ctl_S0SELEXEN	0x26c	1	CGU Config ctl_S0SELEXEN
cgu_ctl_S1SELEXEN	0x26c	1	CGU Config ctl_S1SELEXEN
cgu_auto_disable	0x26c	1	CGU Disable Auto Alignment
cgu_clkoutsel_CLKOUTSEL	0x26c	5	CGU Config clkoutsel_CLKOUTSEL
cgu_clkoutsel_WEN	0x26c	1	CGU Config clkoutsel_WEN
cgu_divex_S0SELEX	0x26c	8	CGU Config divex_S0SELEX
cgu_divex_S1SELEX	0x26c	8	CGU Config divex_S1SELEX
cgu_oscwctl0_WEN	0x270	1	CGU Config oscwctl0_WEN
cgu_oscwctl0_HODF	0x270	6	CGU Config oscwctl0_HODF
cgu_oscwctl0_HODEN	0x270	1	CGU Config oscwctl0_HODEN
cgu_oscwctl0_CNGEN	0x270	1	CGU Config oscwctl0_CNGEN
cgu_oscwctl0_BOUF	0x270	5	CGU Config oscwctl0_BOUF
cgu_oscwctl0_BOUEN	0x270	1	CGU Config oscwctl0_BOUEN
cgu_oscwctl0_FAULTEN	0x270	1	CGU Config oscwctl0_FAULTEN
cgu_oscwctl0_MONDIS	0x270	1	CGU Config oscwctl0_MONDIS
cgu_oscwctl0_FAULTPINDIS	0x270	1	CGU Config oscwctl0_FAULTPINDIS
cgu_reserved3	0x270	14	CGU Config reserved3
cgu_oscwctl1_WEN	0x274	1	CGU Config oscwctl1_WEN
cgu_oscwctl1_HODF	0x274	6	CGU Config oscwctl1_HODF
cgu_oscwctl1_HODEN	0x274	1	CGU Config oscwctl1_HODEN
cgu_oscwctl1_CNGEN	0x274	1	CGU Config oscwctl1_CNGEN

Table 11-3: OTPC Memory Layout (Continued)

Name	32 bit aligned Byte Address	Size (bits)	Description
cgu_oscwctl1_BOUF	0x274	5	CGU Config oscwctl1_BOUF
cgu_oscwctl1_BOUEN	0x274	1	CGU Config oscwctl1_BOUEN
cgu_oscwctl1_FAULTEN	0x274	1	CGU Config oscwctl1_FAULTEN
cgu_oscwctl1_MONDIS	0x274	1	CGU Config oscwctl1_MONDIS
cgu_oscwctl1_FAULTPINDIS	0x274	1	CGU Config oscwctl1_FAULTPINDIS
cgu_reserved4	0x274	14	CGU Config reserved4
flashStartAddress	0x278	32	Flash Start Address Override
spiFlashBootCmd	0x27c	32	SPI Master Boot Command Override
spiHostBootCmd	0x280	32	SPI Slave Boot Command Override
lpBootCmd	0x284	32	LinkPort Boot Command Override
uartBootCmd	0x288	32	UART Boot Command Override
ospiHostBootCmd	0x28c	32	OSPI Boot Command Override
emmcHostBootCmd	0x290	32	EMSI Boot Command Override
emmcStartAddress	0x294	32	EMSI Start Address Override
lpddrBootCmd	0x298	32	LinkPort DDR Boot Command Override
ospi_read_data_capture	0x29c	16	OSPI Read Data Capture Register
ospi_rdcr_reserved0	0x29c	16	OSPI RDCR Reserved0
bcfg_lockMonitor	0x2a0	1	Boot Config lockMonitor
bcfg_reserved0	0x2a0	15	Boot Config reserved0
bcfg_dstCompare	0x2a0	1	Boot Config dstCompare
bcfg_reserved1	0x2a0	15	Boot Config reserved1
bcfg_pubkey0Inv	0x2a4	1	Boot Config pubkey0Inv
bcfg_reserved2	0x2a4	15	Boot Config reserved2
bcfg_pubkey1Inv	0x2a4	1	Boot Config pubkey1Inv
bcfg_reserved3	0x2a4	15	Boot Config reserved3
bcfg_privkey0Inv	0x2a8	1	Boot Config privkey0Inv
bcfg_reserved4	0x2a8	15	Boot Config reserved4
bcfg_privkey1Inv	0x2a8	1	Boot Config privkey1Inv
bcfg_reserved5	0x2a8	15	Boot Config reserved5
bcfg_privkey2Inv	0x2ac	1	Boot Config privkey2Inv
bcfg_reserved6	0x2ac	15	Boot Config reserved6

Table 11-3: OTPC Memory Layout (Continued)

Name	32 bit aligned Byte Address	Size (bits)	Description
bcfg_privkey3Inv	0x2ac	1	Boot Config privkey3Inv
bcfg_reserved7	0x2ac	15	Boot Config reserved7
bcfg_dmcEn	0x2b0	1	Boot Config dmcEn
bcfg_reserved8	0x2b0	15	Boot Config reserved8
bcfg_dmcInv	0x2b0	1	Boot Config dmcInv
bcfg_reserved9	0x2b0	15	Boot Config reserved9
bcfg_dmcDecDis	0x2b4	1	Boot Config dmcDecDis
bcfg_reserved10	0x2b4	31	Boot Config reserved10
reserved	0x2b8	64	reserved
antiroll_nv_cntr	0x2c0	512	AntiRollback NV Counter
gp1	0x300	512	General Purpose 1
reserved	0x340	24	reserved
bootModeDisable	0x340	8	Boot Mode Disable Bits
preboot_ddr_cfg	0x344	384	User PrebootDDR configuration
stageID	0x374	48	StageID
reserved	0x378	16	reserved
fsn	0x380	128	Factory Serial Number
lock	0x48c	1	lockbit
reserved	0x48c	15	reserved
FA_EN	0x48c	1	Field Analysis Enable
reserved	0x48c	15	reserved

Error Correction

The OTP memory features a Hamming error correction implementation. Signal bit errors are automatically corrected, and dual-bit errors are detected. Refer to OTPC Interrupt Signals.

ECC is always enabled. ECC applies to each 16-bit segment. Because of this functionality, each 16-bit location can only be written to once. Writing to a 16-bit location a second time results in unexpected behavior.

OTPC Event Control

The following sections provide information on OTP events and error management.

OTPC Interrupt Signals

When making 32-bit accesses to OTP memory, a double-bit error in any 16-bit segment triggers the OTPC_INT interrupt. The OTPC also has the OTPC dual bit error (OTPC0_ERR) with the interrupt ID of 6. See the System Event Controller (SEC) chapter for more information.

OTPC Status and Error Signals

The OTP controller does not produce error signals.

OTPC Programming Model

OTP programming is supported through the SSLDD OTP library which is a part of CCES 2.11.0.

ADSP-SC59x OTPC Register Descriptions

OTP Memory Controller (OTPC) contains the following registers.

Table 11-4: ADSP-SC59x OTPC Register List

Name	Description
OTPC_SECU_STATE	OTP Security State Register
OTPC_STAT	OTP Status Register

OTP Security State Register

The OTPC_SECU_STATE register indicates the secure state. The register is updated only after the part is rebooted and locked.

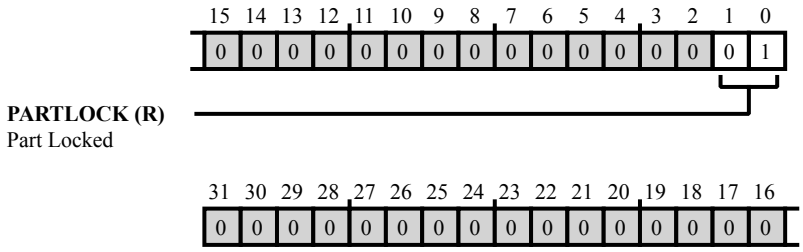


Figure 11-1: OTPC_SECU_STATE Register Diagram

Table 11-5: OTPC_SECU_STATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/NW)	PARTLOCK	Part Locked. The OTPC_SECU_STATE.PARTLOCK field indicates a locked part.	
		0	OPEN part
		1	Locked part
		2	Reserved

OTP Status Register

The OTPC_STAT register bits indicate errors and flag status and control the protection bits.

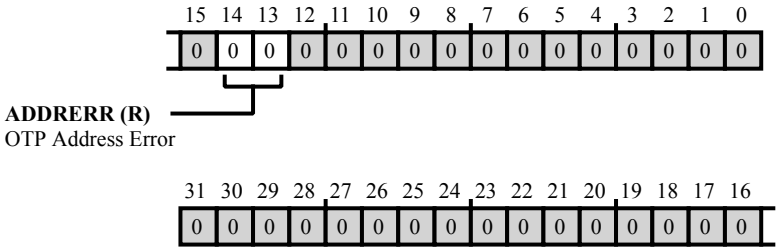


Figure 11-2: OTPC_STAT Register Diagram

Table 11-6: OTPC_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
14:13 (R/NW)	ADDRERR	OTP Address Error. The OTPC_STAT.ADDRERR bit field indicates errors which occur when the OTP programming address is out of range or tries to access protected space.	
		0	No error - proper OTP address
		1	OTP address out of range
		2	8-bit OTP address
		3	Protected OTP address

12 System Memory Protection Unit (SMPU)

The SMPU provides a flexible way of protecting memory regions against read or write access from any or all bus requester in the system. In addition, it can guard against memory access depending on security privileges of the system bus requester.

SMPU Features

The system memory protection unit has the following features.

- On reset, secure transactions are only allowed; non secure transactions are disabled.
- Each SMPU instance can be configured to monitor multiple regions. Each can be individually enabled.
- Each region can be configured with its own protection settings.
- Provides general read or write protection.
- Read and write transactions are restricted or allowed depending on the transaction ID.

Table 12-1: SMPU Instances

Module	SMPU Instance	Controller Allowed	L2 Memory Address Range
L2-Core Port 0 (CL2_0)	2	Data Port of both SHARC+ and ARM and IIR Accelerator Port	0x20000000 – 0x200FFFFF (L2 SRAM Banks 0-3) 0x20200000 – 0x2020FFFF (L2 Boot ROM0)
L2 DMA Port 0 (DL2_0)	3	HSMDMA	0x20000000 – 0x201FFFFF (L2 SRAM Banks 0-7) 0x20200000 – 0x2022FFFF (L2 Boot ROM 0-2)
L2-Core Port 1 (CL2_1)	4	Data Port of both SHARC+ and ARM and IIR AcceleratorPort	0x20100000 – 0x201FFFFF (L2 SRAM Banks 4-7) 0x20210000 – 0x2020FFFF (L2 Boot ROM 1-2)
L2 DMA Port 1 (DL2_1)	5	Peripheral and remaining DMA	0x20000000 – 0x201FFFFF (L2 SRAM Banks 0-7) 0x20200000 – 0x2022FFFF (L2 Boot ROM 0-2)
L2-Core Port 2 (CL2_2)	6	Instruction Port of both SH+ and FIR Accelerator Port	0x20000000 – 0x201FFFFF (L2 SRAM Banks 0-7) 0x20200000 – 0x2022FFFF(L2 Boot ROM 0-2)
DMC0	9	All Allowed Requesters	Entire Range

Table 12-1: SMPU Instances (Continued)

Module	SMPU Instance	Controller Allowed	L2 Memory Address Range
SPI2/OSPI0 Flash Address Space	11	All Allowed Requesters	Entire Range
OTP Memory	12	All Allowed Requesters	Entire Range

All the SMPU instances can be configured up to eight regions.

- Up to eight outstanding read or write transactions supported on SMPU instances for Core_L2 and DMC0. Up to four outstanding read or write transactions supported on the SMPU instances for DMA_L2 and SPI flash address space.

SMPU Functional Description

The following sections provide details on the function of the SMPU module. If the region security settings allow transactions to go through, the ID in the ID-based region protection settings can still filter the transactions.

For the memory that an SMPU protects, programs can configure region-based settings with the `SMPU_RCTL[n]` registers. (There can be multiple SMPUs in a system.) The `SMPU_RCTL[n]` registers define the ID-based protection for memory regions.

If the target address does not reside in any configured memory region, the transaction permission resorts back to the global configuration setting.

Protection Units

Each SMPU provides two protection units, A and B for ID-based matching in the region-based memory protection. This feature provides a degree of flexibility for the user to match against multiple IDs.

Instruction Fetches

When the core executes instructions from memory, this operation is also considered a memory transaction. If the SMPU is configured to protect a memory region from read accesses that contain instructions, the core cannot fetch and execute these instructions.

Using Cache

When the processor uses both cache and the SMPU, there are a few issues to be aware of. If the SMPU is configured to protect a memory region from write accesses, instruction fetches from a core are still possible since instructions are not updated and replaced during run time.

NOTE: The debugger typically replaces an instruction with a breakpoint instruction for software breakpoints. If a memory region is protected against write accesses, software breakpoints are not possible unless the SMPU is configured with the appropriate system controller ID of the debugger. The configuration allows it to perform a write-access.

In the case where memory is used for data, a read access or cache fill is not possible if the memory is blocked from read accesses. If read accesses are allowed but write accesses are disallowed, then there is an issue with coherency. The cache is filled but when the cache is updated and must be written back to the SMPU protected memory, the write-access is blocked.

In general, exercise caution when using both the SMPU and cache.

Speculative Reads

If speculative reads are enabled (`SMPU_CTL.RSDIS` =0), the SMPU forwards the read transaction directly to the memory before checking the protection setting corresponding to the addressed memory region. This functionality saves one clock cycle in the clock domain of the SMPU. The SMPU checks the protection setting while the read transaction occurs with the memory. If the protection setting dictates that the target memory address is blocked, the SMPU blocks the read to the controller.

If speculative reads are disabled (`SMPU_CTL.RSDIS` =1), the SMPU checks the protection settings first and forwards the transaction to memory only if it passes the configured protection settings. This functionality incurs a one-cycle latency per read.

NOTE: Reads affect certain memory operations such as automatic clearing of the memory (that is, FIFOs). When the SMPU protects this type of memory, disable read speculation since the blocking can occur without the read transaction reaching the target memory.

ADSP-SC59x SMPU Register List

The System Memory Protection Unit (SMPU) provides selective protection of the processor's memory resources. The SMPU includes a set of processor events that can be monitored during program execution. A set of registers governs SMPU operations. For more information on SMPU functionality, see the SMPU register descriptions.

Table 12-2: ADSP-SC59x SMPU Register List

Name	Description
`SMPU_BADDR`	Bus Error Address Register
`SMPU_BDTLS`	Bus Error Details Register
`SMPU_CTL`	SMPU Control Register
`SMPU_EXACADD[n]`	Exclusive Access IDn Address
`SMPU_EXACSTAT[n]`	Exclusive Access Status
`SMPU_IADDR`	Interrupt Address Register
`SMPU_IDTLS`	Interrupt Details Register
`SMPU_RADDR[n]`	Region n Address Register
`SMPU_RCTL[n]`	Region n Control Register
`SMPU_REVID`	SMPU Revision ID Register

Table 12-2: ADSP-SC59x SMPU Register List (Continued)

Name	Description
SMPU_RIDA[n]	Region n ID A Register
SMPU_RIDB[n]	Region n ID B Register
SMPU_RIDMSKA[n]	Region n ID Mask A Register
SMPU_RIDMSKB[n]	Region n ID Mask B Register
SMPU_SECURECTL	SMPU Control Secure Accesses Register
SMPU_SECURERCTL[n]	Region n Control Secure Accesses Register
SMPU_STAT	SMPU Status Register

SMPU Interrupts

The SMPU has one interrupt with the SEC ID = 217.

See the SEC chapter for complete information on interrupt generation and use.

Memory Writes

A write transaction to address n is prevented when the following is true.

Address n is in memory region m and memory region m is write-protected (SMPU_RCTL[n].WPROTEN =1) and ID is not a match. (See ID Comparison). The block occurs because the memory region is configured for write-protection and the ID comparison does not result in a match. If an ID comparison results in a match, the write transaction is allowed through.

Memory Reads

A read transaction from address n is prevented when the following is true:

- Address n is in memory region and memory region m is read-protected (SMPU_RCTL[n].RPROTEN =1) and
- ID is not a match

The block occurs because the memory region is configured for read protection and the ID comparison does not result in a match. If the ID comparison results in a match, the read transaction is permitted. (See ID Comparison).

ID Comparison

ID comparison automatically occurs during region-based memory protection. ID matches allow the transaction to bypass the configured memory protection for that region. The following sections describe the calculation of a write ID match and read ID match.

Write Transaction

The state of the following values determines the ID value that is compared with the ID of an incoming write transaction:

- The SMPU_RCTL[n].WIDCINV bit
- The SMPU_RIDA[n].ID and SMPU_RIDB[n].ID bit fields
- The SMPU_RIDMSKA[n].MSK and SMPU_RIDMSKB[n].MSK bit fields

Write IDA match = ((ID of incoming write transaction AND SMPU_RIDMSKA[n].MSK) == (SMPU_RIDA[n].ID AND SMPU_RIDMSKA[n].MSK))

Write IDB match = ((ID of incoming write transaction AND SMPU_RIDMSKB[n].MSK) == (SMPU_RIDB[n].ID AND SMPU_RIDMSKB[n].MSK))

Write ID match = (Write IDA match OR Write IDB match) XOR SMPU_RCTL[n].WIDCINV bit

Read Transaction

The state of the following values determines the ID value that is compared with the ID of an incoming read transaction:

- The SMPU_RCTL[n].RIDCINV bit
- The SMPU_RIDA[n].ID and SMPU_RIDB[n].ID bit fields
- The SMPU_RIDMSKA[n].MSK and SMPU_RIDMSKB[n].MSK bit fields

Read IDA match = ((ID of incoming read transaction AND SMPU_RIDMSKA[n].MSK) == (SMPU_RIDA[n].ID AND SMPU_RIDMSKA[n].MSK))

Read IDB match = ((ID of incoming read transaction AND SMPU_RIDMSKB[n].MSK) == (SMPU_RIDB[n].ID AND SMPU_RIDMSKB[n].MSK))

Read ID match = (Read IDA match OR Read IDB match) XOR SMPU_RCTL[n].RIDCINV

In the two cases described above, the incoming transaction (either write or read) ID is AND'ed with the configured mask value in protection unit A. It is then compared to the value of the configured ID value which is also AND'ed with the configured mask value in protection unit A. The mask provides a method to allow a group of IDs to match. This process is also performed for protection unit B. The two outcomes (from A and B) are then OR'ed together.

Depending on the setting of the SMPU_RCTL[n].RIDCINV or the SMPU_RCTL[n].WIDCINV bits, the ID match comparison is inverted or not. The final result after applying the inversion, SMPU_RCTL[n].RIDCINV, or SMPU_RCTL[n].WIDCINV, determines whether the transaction bypasses the protection.

Usage

The masks, SMPU_RIDMSKA[n] and SMPU_RIDMSKB[n], are AND'ed with both the incoming transaction ID and the configured ID in SMPU_RIDA[n].ID and SMPU_RIDB[n].ID, respectively. By default the masks

are zero. If ID-based region protection is enabled by setting the SMPU_RCTL[n].WPROTEN or SMPU_RCTL[n].RPROTEN bit fields and the masks are not set, the ID comparison essentially compares zeros. The comparison allows all transactions to bypass (if the region-based security setting is also configured in a way to allow transactions to go through for the region). To have the ID-based region protection to function, the mask registers and ID registers must also be set.

System IDs

The ***System controller IDs*** table provides the IDs for the system controllers. An x means that the bit can be a 0 or a 1. There are multiple IDs associated with that system controller.

Table 12-3: System Controller IDs

ASIB Name	IID	SIID	ID
SP0A	0	0	13'b0000x00000000
SP0B	1	0	13'b0000x00010000
SP1A	2	0	13'b0000x00100000
SP1B	3	0	13'b0000x00110000
SP2A	4	0	13'b0000x01000000
SP2B	5	0	13'b0000x01010000
SP3A	6	0	13'b0000x01100000
SP3B	7	0	13'b0000x01110000
CRC0_CH0	3	1	13'b0000x00110001
CRC0_CH1	2	1	13'b0000x00100001
MLB	4	2	13'b0000001000010
UART0_TX	0	3	13'b0000x00000011
UART0_RX	4	3	13'b0000x01000011
UART2_TX	3	3	13'b0000x00110011
SPI0TX	0	4	13'b0000x00000100
SPI0RX	1	4	13'b0000x00010100
SPI1TX	2	4	13'b0000x00100100
SPI1RX	3	4	13'b0000x00110100
SPI2TX	5	4	13'b0000x01010100
SPI2RX	4	4	13'b0000x01000100
LP0	0	5	13'b0000x00000101
UART1_TX	1	3	13'b0000x00010011
UART1_RX	2	3	13'b0000x00100011

Table 12-3: System Controller IDs (Continued)

ASIB Name	IID	SIID	ID
LP1	1	5	13'b0000x00010101
CRYPTO	5	6	13'b0000001010110
SH0_FIR_CH0	0	7	13'b0xxxx00000111
SH0_FIR_CH1	1	7	13'b0xxxx00010111
DLDMA0_CH0	0	6	13'b0000000000110
DLDMA0_CH1	1	6	13'b0000000010110
DLDMA1_CH0	2	6	13'b0000000100110
DLDMA1_CH1	3	6	13'b0000000110110
MSMDMA_CH0	0	8	13'b0000x00001000
MSMDMA_CH1	1	8	13'b0000x00011000
DBG	4	8	13'b0000001001000
ETR	4	6	13'b0000001000110
CRC1_CH0	1	1	13'b0000x00010001
CRC1_CH1	0	1	13'b0000x00000001
UART2_RX	5	3	13'b0000x01010011
SH0_DPORT	0	9	13'b0xxxx00001001
SH0_IPORT	4	7	13'b0000001000111
HSMDMA_CH0	0	10	13'b0000x00001010
HSMDMA_CH1	1	10	13'b0000x00011010
SP4A	0	11	13'b0000x00001011
SP4B	1	11	13'b0000x00011011
SP5A	2	11	13'b0000x00101011
SP5B	3	11	13'b0000x00111011
SP6A	4	11	13'b0000x01001011
SP6B	5	11	13'b0000x01011011
SP7A	6	11	13'b0000x01101011
SP7B	7	11	13'b0000x01111011
SH0_MMR	0	12	13'b0xxxx00001100
UART3_TX	6	3	13'b0000x01100011
UART3_RX	7	3	13'b0000x01110011
SPI3TX	6	4	13'b0000x01100100

Table 12-3: System Controller IDs (Continued)

ASIB Name	IID	SIID	ID
SPI3RX	7	4	13'b0000x01110100
CRC2_CH0	1	13	13'b0000x00011101
CRC2_CH1	0	13	13'b0000x00001101
CRC3_CH0	3	13	13'b0000x00111101
CRC3_CH1	2	13	13'b0000x00101101
MSMDMA1_CH0	3	8	13'b0000x00111000
MSMDMA1_CH1	2	8	13'b0000x00101000
HSMDMA1_CH0	2	10	13'b0000x00101010
HSMDMA1_CH1	3	10	13'b0000x00111010
PPI_F0	8	3	13'b0000x10000011
PPI_F1	9	3	13'b0000x10010011
USB0	2	14	13'b0000000101110
SH1_IPORT	5	7	13'b0000001010111
SH1_DPORT	1	9	13'b0xxxx00011001
SH1_MMR	2	12	13'b0xxxx00101100
PL310_M0	2	9	13'bxxxxx00101001
PL310_M1	3	9	13'bxxxxx00111001
GIGE	0	15	13'b0xxxx00001111
EMAC	1	15	13'b0xxxx00011111
SH1_FIR_CH0	2	7	13'b0xxxx00100111
SH1_FIR_CH1	3	7	13'b0xxxx00110111
SH0_IIR_CH0	4	9	13'b0xxxx01001001
SH0_IIR_CH1	5	9	13'b0xxxx01011001
SH1_IIR_CH0	6	9	13'b0xxxx01101001
SH1_IIR_CH1	7	9	13'b0xxxx01111001

Memory Region

Memory regions can start at address 0x00000000 or at any address that is a multiple of its size. The ***Supported Memory Region Size and Alignment*** table shows the memory region sizes that the processor supports and the alignment of the memory region. (X values are do-not-care).

SMPU supports a maximum of eight regions.

Table 12-4: Supported Memory Region Size and Alignment

Size	SMPU_RCTLn.SIZE	Address	Possible Values for N
4KB	0b00000	0xXXXXX000	-
8KB	0b00001	0xXXXXN000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
16KB	0b00010	0xXXXXN000	0x0, 0x4, 0x8, 0xC
32KB	0b00011	0xXXXXN000	0x0, 0x8
64KB	0b00100	0xXXXX0000	-
128KB	0b00101	0xXXXN0000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
256KB	0b00110	0xXXXN0000	0x0, 0x4, 0x8, 0xC
512KB	0b00111	0xXXXN0000	0x0, 0x8
1MB	0b01000	0xXXX00000	-
2MB	0b01001	0xXXN00000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
4MB	0b01010	0xXXN00000	0x0, 0x4, 0x8, 0xC
8MB	0b01011	0xXXN00000	0x0, 0x8
16MB	0b01100	0xXX000000	-
32MB	0b01101	0xXN000000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
64MB	0b01110	0xXN000000	0x0, 0x4, 0x8, 0xC
128MB	0b01111	0xXN000000	0x0, 0x8
256MB	0b10000	0xX0000000	-
512MB	0b10001	0xN0000000	0x0, 0x2, 0x4, 0x8, 0xA, 0xC, 0xE
1GB	0b10010	0xN0000000	0x0, 0x4, 0x8, 0xC
2GB	0b10011	0xN0000000	0x0, 0x8
4GB	0b10100	0x00000000	-

For the case where the region size is selected as 4 GB, the region address must be at address 0x00000000.

NOTE: If a memory region address is not aligned to its size, the memory region start address protected by the SMPU is the configured address with the corresponding least significant bits masked. For example, if the size is configured for 16 KB (`SMPU_RCTL[n].SIZE` =0b00010), and the base address is configured for `SMPU_RADDR[n].BADDR`=0x00005018, the actual base address used by the SMPU is 0x00004000. When `SMPU_RADDR[n].BADDR` is read back, the program reads 0x00005000. This functionality is because only bits [11:0] are reserved as 0's. Programs must use care when setting the base address as it is not always the true base address.

SMPU Definitions

To make the best use of the SMPU, it is useful to understand the terms in this section.

Global Protection

Guarding of the entire memory space for the SMPU instantiation.

Region-Based Protection

Guarding individual segments of memory inside the memory space for the SMPU instantiation.

ID Match

A successful comparison of the ID associated with the incoming transaction and the ID and MASK configured in the SMPU.

SMPU Block Diagram

The *SMPU Top-Level Block Diagram* shows the SMPU block.

As seen in the diagram, the SMPU sits between the memory port (SCB requester port) and the SCB fabric (SCB completer port). It acts as a gateway analyzing the transaction requests. It either rejects the transaction request or allows access based on the user-programmed configuration of the SMPU.

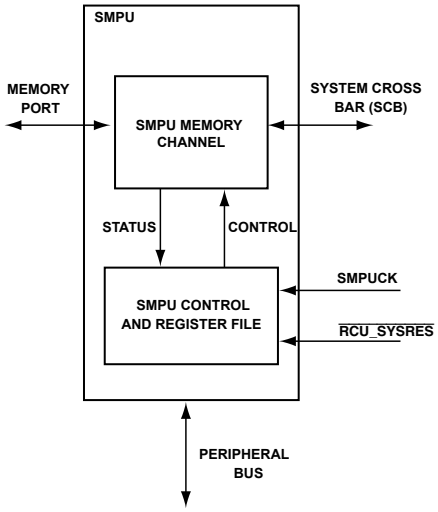


Figure 12-1: SMPU Top-Level Block Diagram

SMPU Architectural Concepts

The following sections provide brief descriptions of the architecture of the SMPU module.

Default Setting

At reset, the default state of the system is fully open. The SMPUs admit any access to memory spaces.

Latency

The SMPU adds latency to all the transactions to the memory except reads when read speculation is enabled (SMPU_CTL.RSDIS =0). In this case, read accesses are always forwarded to the memory and read responses are generated according to the SMPU settings. If read speculation is disabled (SMPU_CTL.RSDIS =1), reads are blocked if they cause a security or protection violation. The SMPU generates the SCB read response that corresponds to a blocked transaction.

If read speculation is enabled, the SMPU adds 1 clock cycle latency to the read transaction. If read speculation is disabled, the SMPU adds 2 clock cycles latency to the read transaction.

SMPU Operating Modes

The SMPU does not have any strict modes of operation. However, it can be configured for region-based protection where a controller with a particular ID can be blocked or allowed based on settings in the SMPU_RCTL[n] register.

Region-based protection is programmed with registers:

- SMPU_RCTL[n]
- SMPU_RADDR[n]
- SMPU_RIDA[n]
- SMPU_RIDMSKA[n]
- SMPU_RIDB[n]
- SMPU_RIDMSKB[n]

SMPU Interrupt Signals

There is one interrupt signal associated with the SMPU. If interrupts are enabled, the SMPU_STAT.IRQ bit is set. The SMPU_IRQ signal is asserted when the SMPU detects a memory access violation. The target address triggering the interrupt is found in the SMPU_IADDR register. The SMPU_IDTLS register provides further details about the cause of the interrupt.

Write errors are prioritized over read errors.

Protection violations (an ID-based violation) can trigger the SMPU interrupt and can be enabled independently. The protection violation interrupt is enabled by setting the SMPU_CTL.PINTEN bit.

The SMPU interrupt is asserted for any of the following conditions:

If a second memory access violation occurs while the SMPU_STAT.IRQ bit is set, the SMPU_STAT.IOVR (interrupt overrun) bit is set. The SMPU_IADDR and the SMPU_IDTLS registers are not updated until the SMPU_STAT.IRQ bit is cleared. Any information on the subsequent interrupt is lost. Once the

SMPU_STAT.IRQ bit and the SMPU_STAT.IOVR bit are cleared, any new memory access violations can trigger an interrupt and its details can be captured.

NOTE: When a blocked access occurs, the SMPU triggers an interrupt when interrupt generation is enabled. The SMPU can also be configured to generate a bus error that propagates back to the system controller. The system controller can also trigger an interrupt due to this bus error.

NOTE: On the processor, each SMPU instance has an interrupt. All of the SMPU interrupts are OR'ed and mapped to a single SMPU interrupt on the SEC. While servicing the SMPU interrupt, check all of the SMPU_STAT registers to determine which triggered the interrupt. The interrupt service routine clears the SMPU_STAT.IRQ bit of all of the SMPU_STAT registers for which the interrupt is triggered.

SMPU Status and Error Signals

If bus errors are enabled (SMPU_CTL.PBEDIS =0), the SMPU generates and returns a bus error to the controller initiating the blocked access. This bit also sets the SMPU_STAT.BERR bit. The SMPU_BADDR and SMPU_BDTLS registers can be read to get the address and details of the transaction that caused the SMPU to generate the error.

Write errors are prioritized over read errors.

A bus error status is returned to the system controller if:

- an ID-based violation happened and the SMPU_CTL.PBEDIS bit =0

If a second memory access violation occurs while the SMPU_STAT.BERR bit is set, the SMPU_STAT.BEOVR bit (bus error overrun) is set. The SMPU_BADDR and the SMPU_BDTLS registers are not updated until the SMPU_STAT.IRQ bit is cleared. The information about the transaction that caused the SMPU_STAT.BEOVR bit to be set is lost.

NOTE: If both the protection violation interrupt is not enabled (SMPU_CTL.PINTEN =0) and the protection bus error is disabled (SMPU_CTL.PBEDIS =1), the SMPU blocks invalid transactions. However, it does not provide any status or interrupt information indicating that a transaction is blocked.

ADSP-SC59x SMPU Register Descriptions

The System Memory Protection Unit (SMPU) contains the following registers.

Table 12-5: ADSP-SC59x SMPU Register List

Name	Description
SMPU_BADDR	Bus Error Address Register
SMPU_BDTLS	Bus Error Details Register
SMPU_CTL	SMPU Control Register

Table 12-5: ADSP-SC59x SMPU Register List (Continued)

Name	Description
SMPU_EXACADD[n]	Exclusive Access IDn Address
SMPU_EXACSTAT[n]	Exclusive Access Status
SMPU_IADDR	Interrupt Address Register
SMPU_IDTLS	Interrupt Details Register
SMPU_RADDR[n]	Region n Address Register
SMPU_RCTL[n]	Region n Control Register
SMPU_REVID	SMPU Revision ID Register
SMPU_RIDA[n]	Region n ID A Register
SMPU_RIDB[n]	Region n ID B Register
SMPU_RIDMSKA[n]	Region n ID Mask A Register
SMPU_RIDMSKB[n]	Region n ID Mask B Register
SMPU_SECURECTL	SMPU Control Secure Accesses Register
SMPU_SECURERCTL[n]	Region n Control Secure Accesses Register
SMPU_STAT	SMPU Status Register

Bus Error Address Register

Programs read the SMPU_BADDR and the SMPU_BDTLS registers to determine the cause of a bus error. Write errors are prioritized over read errors.

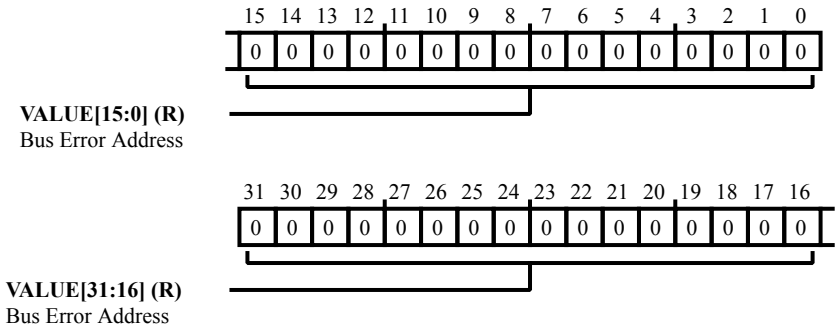


Figure 12-2: SMPU_BADDR Register Diagram

Table 12-6: SMPU_BADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Bus Error Address. The SMPU_BADDR.VALUE bit field contains the address of the bus error.

Bus Error Details Register

The SMPU_BDTLS register indicates the ID of the bus error transaction, whether the transaction that caused the last bus error was a read, a write, secure or non-secure.

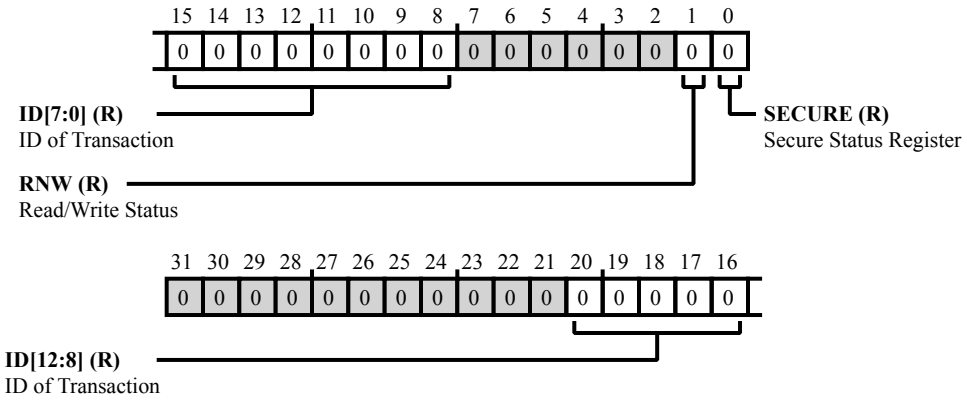


Figure 12-3: SMPU_BDTLS Register Diagram

Table 12-7: SMPU_BDTLS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20:8 (R/NW)	ID	ID of Transaction. The SMPU_BDTLS.ID bit field provides the ID of the transaction that caused the bad address error.	
1 (R/NW)	RNW	Read/Write Status. The SMPU_BDTLS.RNW bit indicates whether the last transaction that caused the bad address error was a read or write.	
		0	Transaction that caused last bus error was a write
		1	Transaction that caused last bus error was a read
0 (R/NW)	SECURE	Secure Status Register. The SMPU_BDTLS.SECURE bit indicates whether the last transaction that caused the bad address error was secure or non-secure.	
		0	Transaction that caused last bus error was non-secure
		1	Transaction that caused last bus error was secure

SMPU Control Register

The SMPU_CTL register provides access to the locking control, error interrupts and SMPU violations.

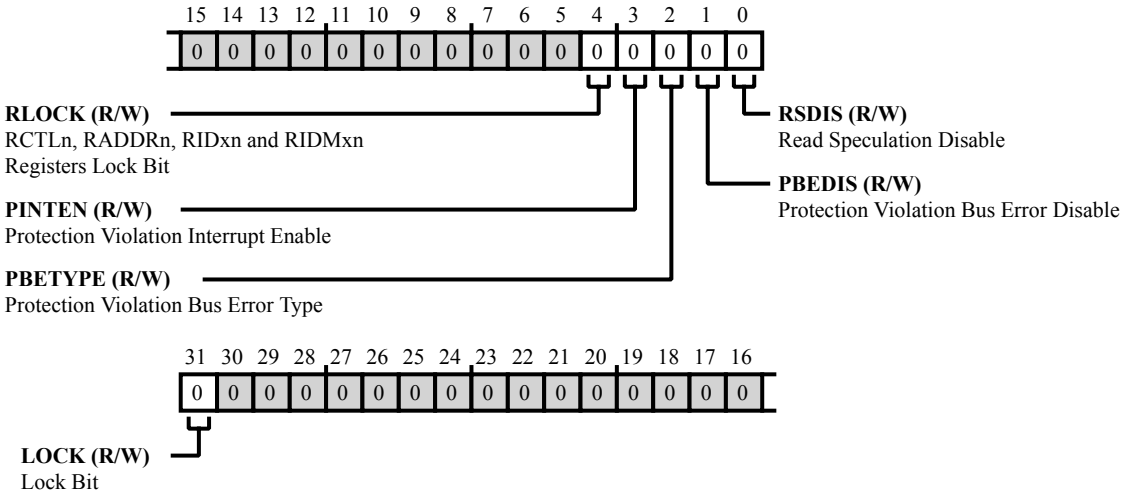


Figure 12-4: SMPU_CTL Register Diagram

Table 12-8: SMPU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit. When the SMPU_CTL.LOCK bit is set and the global lock signal is asserted from the SPU, the SMPU_CTL register is write-protected. Write-protection is disabled only when the global lock signal becomes deasserted again.	
		0	CTL Global Lock Disable. The SMPU_CTL register is not write-protected.
		1	CTL Global Lock Enable. The SMPU_CTL register is write-protected.
4 (R/W)	RLOCK	RCTLn, RADDRn, RIDxn and RIDMxn Registers Lock Bit. When the SMPU_CTL.RLOCK bit is set, all the registers associated with region-based control (SMPU_RCTL[n], SMPU_RADDR[n], SMPU_RIDA[n], SMPU_RIDB[n], SMPU_RIDMSKA[n] and SMPU_RIDMSKB[n]) are write-protected when the global lock signal is active from the SPU. Write access is allowed again when the global lock signal is deasserted.	
		0	Region Registers Write-Protect Enable. All region registers are not write-protected.
		1	Region Registers Write-Protect Disable. All region registers are write-protected.

Table 12-8: SMPU_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	PINTEN	Protection Violation Interrupt Enable. The SMPU_CTL.PINTEN bit controls whether or not an interrupt is generated when a protection violation occurs.	
		0	Protection Violation IRQ Disable. The protection violation interrupt is disabled.
		1	Protection Violation IRQ Enable. The protection violation interrupt is enabled.
2 (R/W)	PBETYPE	Protection Violation Bus Error Type. The SMPU_CTL.PBETYPE bit controls whether a protection violation produces a decode error or a completer error.	
		0	Decode Error Type. Decode error for transactions that violate the configured protection.
		1	Completer Error Type. Completer error for transactions which violate the configured protection.
1 (R/W)	PBEDIS	Protection Violation Bus Error Disable. If set, the SMPU_CTL.PBEDIS bit blocks protection violations, but does not cause a bus error.	
		0	Bus Error Generation Enable. Transactions which violate the configured protection are blocked and cause a bus error.
		1	Bus Error Generation Disable. Transactions which violate the configured protection are blocked but do not cause a bus error.
0 (R/W)	RSDIS	Read Speculation Disable. The SMPU_CTL.RSDIS bit controls whether or not the read addresses are checked before being sent to the completer.	
		0	Read Speculation Enable. Read addresses are sent to the completer without checking.
		1	Read Speculation Disable. Read addresses are checked before being sent to the completer.

Exclusive Access IDn Address

The SMPU_EXACADD[n] register provides the address ID of an exclusive access.

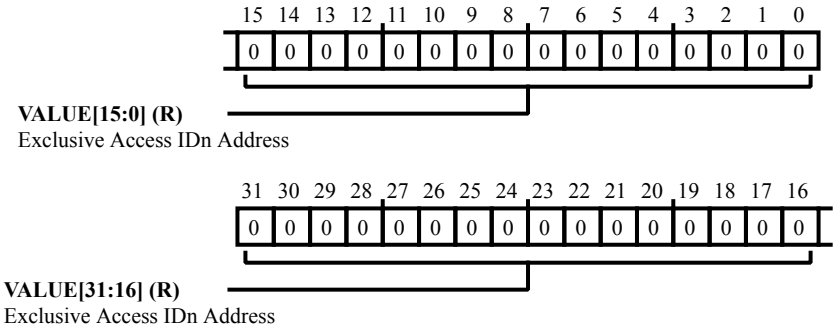


Figure 12-5: SMPU_EXACADD[n] Register Diagram

Table 12-9: SMPU_EXACADD[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Exclusive Access IDn Address.

Exclusive Access Status

The SMPU_EXACSTAT[n] register provides the exclusive access ID, read size and read length as well as the indication that the access was valid.

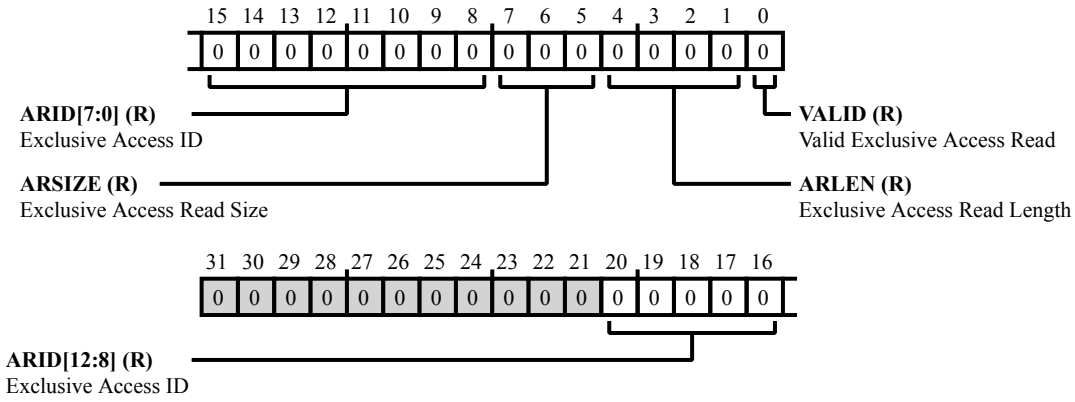


Figure 12-6: SMPU_EXACSTAT[n] Register Diagram

Table 12-10: SMPU_EXACSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:8 (R/NW)	ARID	Exclusive Access ID.
7:5 (R/NW)	ARSIZE	Exclusive Access Read Size.
4:1 (R/NW)	ARLEN	Exclusive Access Read Length.
0 (R/NW)	VALID	Valid Exclusive Access Read.

Interrupt Address Register

The SMPU_IADDR register indicates an attempt to make a read or write access to unimplemented addresses or accesses are non-aligned. The SMPU issues a bus error for this condition.

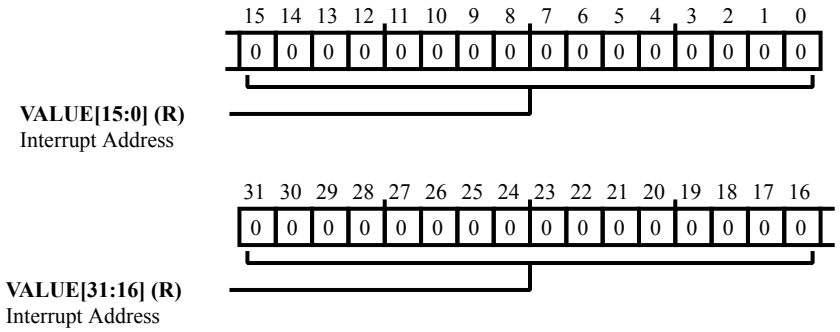


Figure 12-7: SMPU_IADDR Register Diagram

Table 12-11: SMPU_IADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Interrupt Address. The SMPU_IADDR.VALUE bit field is the address where an attempt to access an unimplemented address or a non-aligned access has occurred.

Interrupt Details Register

The SMPU_IDTLS register provides the ID of the last signaled interrupt, whether the interrupt was caused by a read or write, and whether the transaction that caused the last signaled interrupt was secure.

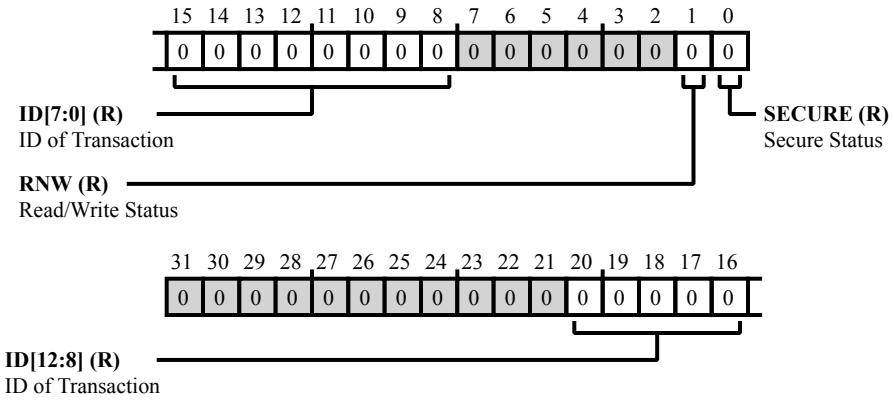


Figure 12-8: SMPU_IDTLS Register Diagram

Table 12-12: SMPU_IDTLS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20:8 (R/NW)	ID	ID of Transaction. The SMPU_IDTLS.ID bit field provides the ID of the transaction that caused the interrupt.	
1 (R/NW)	RNW	Read/Write Status. The SMPU_IDTLS.RNW bit indicates whether the last transaction that caused the interrupt was a read or write.	
		0	Transaction that caused last signaled interrupt was a write
		1	Transaction that caused last signaled interrupt was a read
0 (R/NW)	SECURE	Secure Status. The SMPU_IDTLS.SECURE bit indicates whether the last transaction that caused the interrupt was secure or non-secure.	
		0	Transaction that caused last signaled interrupt was non-secure
		1	Transaction that caused last signaled interrupt was secure

Region n Address Register

The SMPU_RADDR[n] register is used to define the base address for a memory region to be protected.

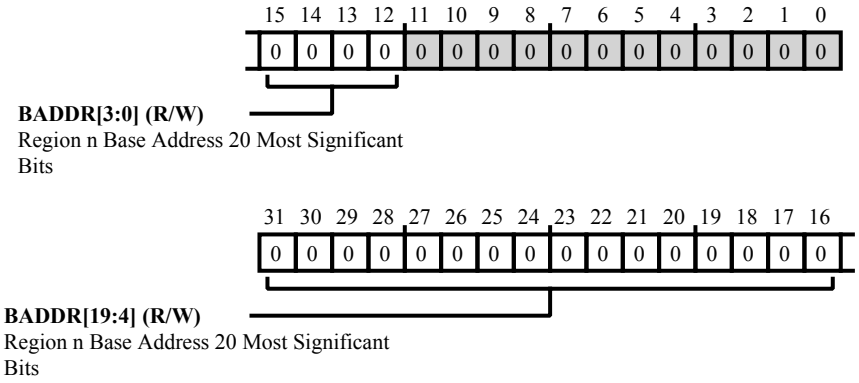


Figure 12-9: SMPU_RADDR[n] Register Diagram

Table 12-13: SMPU_RADDR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:12 (R/W)	BADDR	Region n Base Address 20 Most Significant Bits. The SMPU_RADDR[n].BADDR bit field defines the base address for a memory region to be protected.

Region n Control Register

The SMPU_RCTL[n] register is used to define the level of protection for a region of memory. The protection of a region is controlled and defined by this register and the SMPU_RADDR[n], SMPU_RIDA[n], SMPU_RIDB[n], SMPU_RIDMSKA[n], and SMPU_RIDMSKB[n] registers.

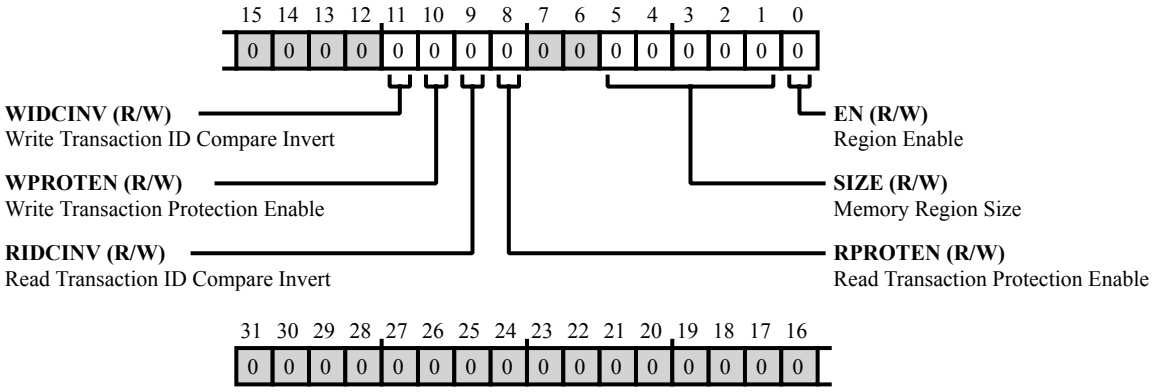


Figure 12-10: SMPU_RCTL[n] Register Diagram

Table 12-14: SMPU_RCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	WIDCINV	Write Transaction ID Compare Invert. The SMPU_RCTL[n].WIDCINV bit inverts the write ID match result.	
		0	Write transaction ID comparison result not inverted
		1	Write transaction ID comparison result inverted
10 (R/W)	WPROTEN	Write Transaction Protection Enable. The SMPU_RCTL[n].WPROTEN bit enables protection against ID-based write transactions for the memory region.	
		0	Write transaction ID-based protection disabled
		1	Write transaction ID-based protection enabled
9 (R/W)	RIDCINV	Read Transaction ID Compare Invert. When the SMPU_RCTL[n].RIDCINV bit is set, the read ID match result is inverted.	
		0	Read transaction ID comparison result not inverted
		1	Read transaction ID comparison result inverted

Table 12-14: SMPU_RCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	RPROTEN	Read Transaction Protection Enable. The `SMPU_RCTL[n].RPROTEN` bit enable bit to turn on protection against ID-based read transactions for the memory region.	
		0	Read transaction ID-based protection disabled
		1	Read transaction ID-based protection enabled
5:1 (R/W)	SIZE	Memory Region Size. The `SMPU_RCTL[n].SIZE` bit defines the size of the memory region to be protected.	
		0	4 KB
		1	8 KB
		2	16 KB
		3	32 KB
		4	64 KB
		5	128 KB
		6	256 KB
		7	512 KB
		8	1 MB
		9	2 MB
		10	4 MB
		11	8 MB
		12	16 MB
		13	32 MB
		14	64 MB
		15	128 MB
		16	256 MB
		17	512 MB
		18	1 GB
		19	2 GB
		20	4 GB
		21-31	Reserved

Table 12-14: SMPU_RCTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Region Enable. The `SMPU_RCTL[n].EN` bit enables the protection of a region.	
		0	Disabled
		1	Enabled

SMPU Revision ID Register

The SMPU_REVID register provides the major and minor revision numbers of this module.

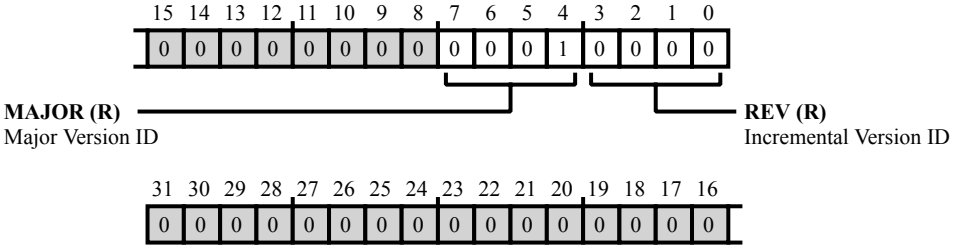


Figure 12-11: SMPU_REVID Register Diagram

Table 12-15: SMPU_REVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	MAJOR	Major Version ID.
3:0 (R/NW)	REV	Incremental Version ID.

Region n ID A Register

The SMPU_RIDA[n] register is used for ID comparison 'A'. This comparison is performed after a mask is applied to both the transaction ID (from either the read or write IDs) and the register value. An ID match means that the ID is the exception to the rule and the read or write is allowed even if the region is read or write-protected. For more detail, refer to the ID Comparison section.

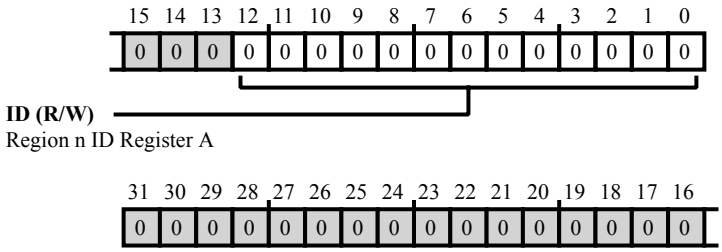


Figure 12-12: SMPU_RIDA[n] Register Diagram

Table 12-16: SMPU_RIDA[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	ID	Region n ID Register A. The SMPU_RIDA[n].ID bit field, combined with the mask provides the means to bypass the configured memory protection for a region.

Region n ID B Register

The SMPU_RIDB[n] register is used for ID comparison 'B'. This comparison is performed after a mask is applied to both the transaction ID (from either the read or write IDs) and the register value. An ID match means that the ID is the exception to the rule and the read or write is allowed even if the region is read or write-protected. For more details, refer to the ID Comparison section.

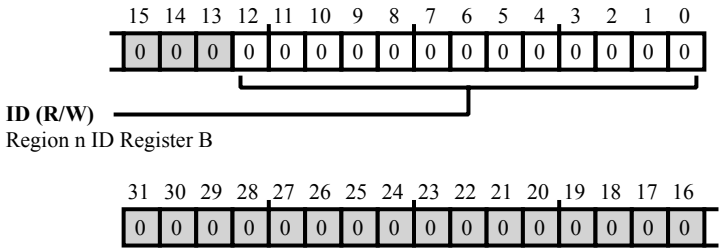


Figure 12-13: SMPU_RIDB[n] Register Diagram

Table 12-17: SMPU_RIDB[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	ID	Region n ID Register B. The SMPU_RIDB[n].ID bit field, combined with the mask provides the means to bypass the configured memory protection for a region.

Region n ID Mask A Register

The SMPU_RIDMSKA[n] register is used for ID comparison 'A'. The mask allows or disallows certain IDs from affecting the final result of the ID match. For more details, refer to the ID Comparison section.

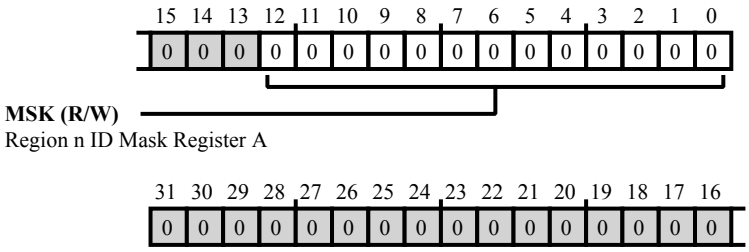


Figure 12-14: SMPU_RIDMSKA[n] Register Diagram

Table 12-18: SMPU_RIDMSKA[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	MSK	Region n ID Mask Register A. The SMPU_RIDMSKA[n].MSK bit field, combined with the incoming transaction, provides the means to bypass the configured memory protection for a region.

Region n ID Mask B Register

The SMPU_RIDMSKB[n] register is used for ID comparison 'B'. The mask allows or disallows certain IDs from affecting the final result of the ID match. For more details, refer to the ID Comparison section.

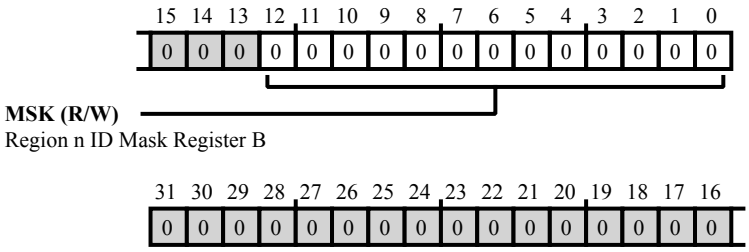


Figure 12-15: SMPU_RIDMSKB[n] Register Diagram

Table 12-19: SMPU_RIDMSKB[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12:0 (R/W)	MSK	Region n ID Mask Register B. The SMPU_RIDMSKB[n].MSK bit field, combined with the incoming transaction provides the means to bypass the configured memory protection for a region.

SMPU Control Secure Accesses Register

The SMPU_SECURECTL register provides the bits required to set up the security settings for the processor. These settings includes error generation and read/write security.

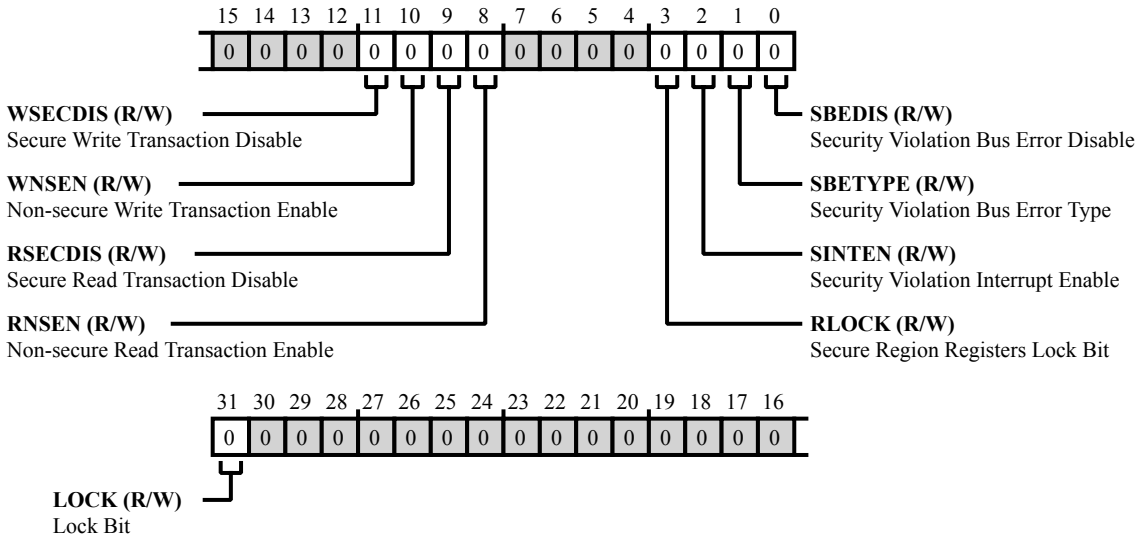


Figure 12-16: SMPU_SECURECTL Register Diagram

Table 12-20: SMPU_SECURECTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock Bit. When the SMPU_SECURECTL.LOCK bit is set and the global lock signal is asserted from the SPU, the SMPU_SECURECTL register is write-protected. Write-protection is disabled only when the global lock signal becomes deasserted again.	
		0	SMPU_SECURECTL is not write-protected
		1	SMPU_SECURECTL is write-protected
11 (R/W)	WSECDIS	Secure Write Transaction Disable. The SMPU_SECURECTL.WSECDIS bit disables secure write transactions.	
		0	Enable secure write transactions
		1	Disable secure write transactions
10 (R/W)	WNSEN	Non-secure Write Transaction Enable. The SMPU_SECURECTL.WNSEN bit enables non-secure write transactions.	
		0	Disable non-secure writes
		1	Enable non-secure writes

Table 12-20: SMPU_SECURECTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	RSECDIS	Secure Read Transaction Disable. The SMPU_SECURECTL.RSECDIS bit disables secure read transactions.	
		0	Enable secure read transactions
		1	Disable secure read transactions
8 (R/W)	RNSEN	Non-secure Read Transaction Enable. The SMPU_SECURECTL.RNSEN bit enables non-secure read transactions.	
		0	Disable non-secure read transactions
		1	Enable non-secure read transactions
3 (R/W)	RLOCK	Secure Region Registers Lock Bit. When the SMPU_SECURECTL.RLOCK bit is set, the secure region control registers, SMPU_SECURERCTL[n], are write-protected when the global lock signal is active from the SPU. When the global lock signal is deasserted, write access is allowed again.	
		0	Disable write-protection on secure region registers
		1	Enable write-protection on secure region registers
2 (R/W)	SINTEN	Security Violation Interrupt Enable. The SMPU_SECURECTL.SINTEN bit enables interrupt generation when a security violation occurs.	
		0	Disable security settings violation interrupt
		1	Enable security settings violation interrupt
1 (R/W)	SBETYPE	Security Violation Bus Error Type. The SMPU_SECURECTL.SBETYPE bit controls whether a decode error or a completer error is returned when a security violation occurs.	
		0	Decode error
		1	Completer error
0 (R/W)	SBEDIS	Security Violation Bus Error Disable. The SMPU_SECURECTL.SBEDIS bit controls whether or not a bus error is caused when a security violation occurs.	
		0	Enable bus error
		1	Disable bus error

Region n Control Secure Accesses Register

The SMPU_SECURERCTL[n] register contains bits that configure read/write security for a specific region.

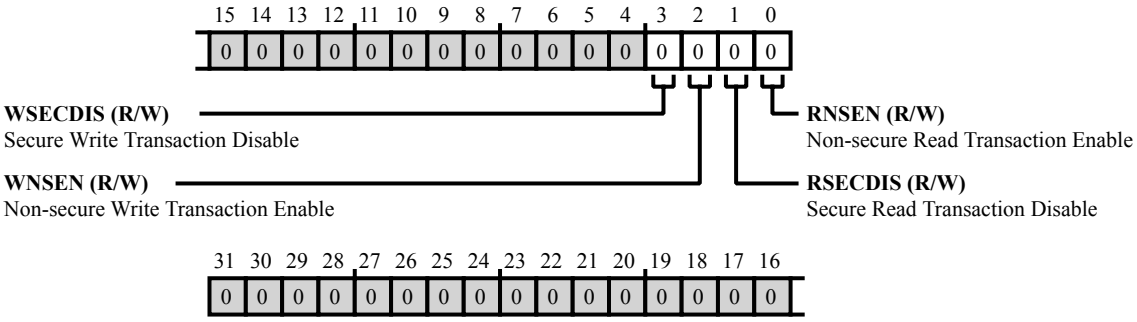


Figure 12-17: SMPU_SECURERCTL[n] Register Diagram

Table 12-21: SMPU_SECURERCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	WSECDIS	Secure Write Transaction Disable. The SMPU_SECURERCTL[n].WSECDIS bit disables secure write transactions for the memory region.	
		0	Enable secure write transactions to this region
		1	Disable secure write transactions to this region
2 (R/W)	WNSEN	Non-secure Write Transaction Enable. This SMPU_SECURERCTL[n].WNSEN bit enables non-secure write transactions for the memory region.	
		0	Disable non-secure write transactions to this region
		1	Enable non-secure write transactions to this region
1 (R/W)	RSECDIS	Secure Read Transaction Disable. The SMPU_SECURERCTL[n].RSECDIS bit disables secure read transactions for the memory region.	
		0	Enable secure read transactions to this region
		1	Disable secure read transactions to this region
0 (R/W)	RNSEN	Non-secure Read Transaction Enable. The SMPU_SECURERCTL[n].RNSEN bit enables non-secure read transactions for the memory region.	
		0	Disable non-secure read transactions to this region
		1	Enable non-secure read transactions to this region

SMPU Status Register

The SMPU_STAT register provides the state of the SMPU and indicates various errors. All bits in this register are write 1 to clear.

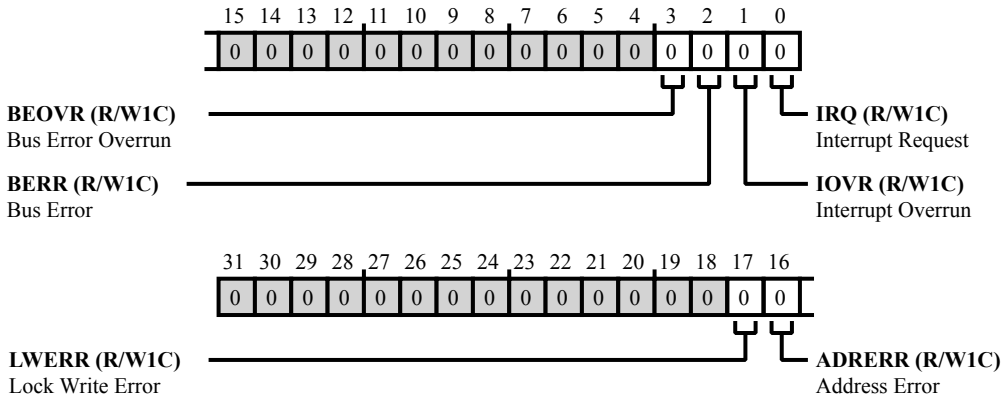


Figure 12-18: SMPU_STAT Register Diagram

Table 12-22: SMPU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	LWERR	Lock Write Error. The SMPU_STAT.LWERR bit is set when SMPU_CTL.LOCK bit =1, the global lock signal is asserted from the SPU and a read or write attempt was made to the SMPU_CTL MMR.	
		0	No Lock Write Error
		1	Lock Write Error
16 (R/W1C)	ADRERR	Address Error. The SMPU_STAT.ADRERR bit is set when the SMPU MMR is accessed as an un-aligned address, or when a read-only MMR is written to.	
		0	No Address Error
		1	Address Error
3 (R/W1C)	BEOVR	Bus Error Overrun. The SMPU_STAT.BEOVR bit indicates that another bus error had occurred. Any new information about the most recent violation which caused the bus error is not captured.	
		0	No Bus Error overrun
		1	Bus Error overrun has occurred

Table 12-22: SMPU_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	BERR	Bus Error. This `SMPU_STAT.BERR` bit indicates if a bus error was generated.	
		0	No Bus Error since this bit has been cleared
		1	Bus Error has been generated
1 (R/W1C)	IOVR	Interrupt Overrun. The `SMPU_STAT.IOVR` bit indicates if another violation occurred while the previous violation interrupt was not finished being serviced. Information about the most recent violation is then not captured.	
		0	No Interrupt overrun
		1	Interrupt overrun has occurred
0 (R/W1C)	IRQ	Interrupt Request. The `SMPU_STAT.IRQ` bit provides an indication that an interrupt has been generated.	
		0	No Interrupt since this bit has been cleared
		1	Interrupt has been generated

13 General-Purpose Ports (PORT)

This section describes general-purpose ports, pin multiplexing, general-purpose input/output (GPIO) functionality, and pin interrupts. The general-purpose ports provide the following three functions:

- Pin multiplexing scheme
- GPIO functionality
- Pin interrupt requests

NOTE: In this chapter, the naming convention for registers and bits omits the alphabetic group enumeration to refer to any and all of the ports. For example, `PORT_FER` represents registers `PORTA_FER`, `PORTB_FER`, and so on. Likewise `PORT_FER.PX1` represents bits PA1, PB1, and so on.

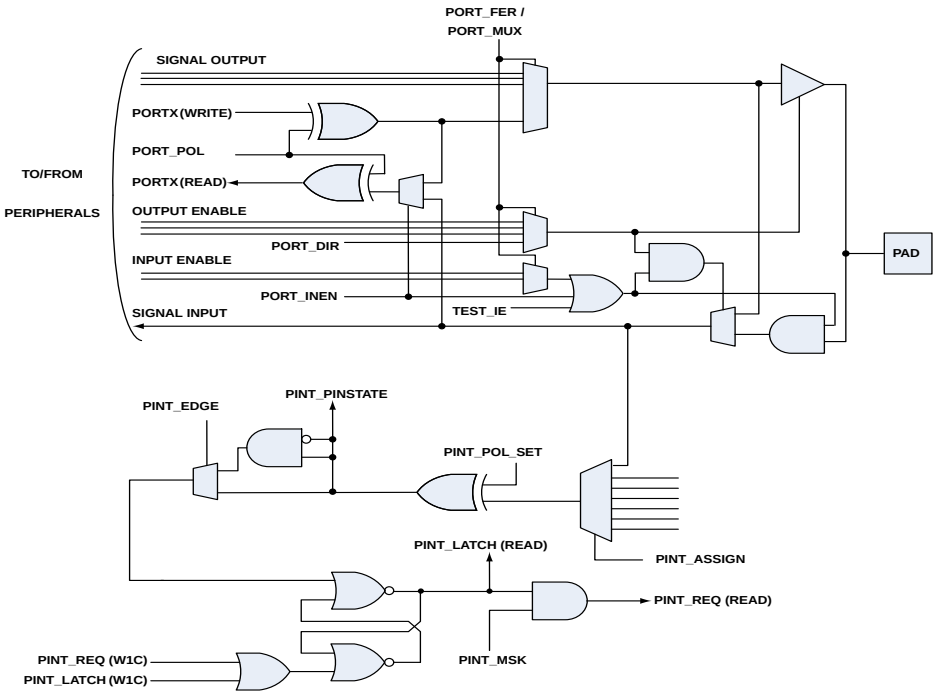


Figure 13-1: Simplified GPIO and Pin Interrupt Signal Flow

PORT Features

The PORTs include the following features:

- Input mode, output mode, and open-drain mode of GPIO operation
- Port multiplexing controlled on a pin-by-pin basis
- No external glue hardware required for unused pins
- All port pins provide interrupt request functionality
- Byte-wide pin-to-interrupt request assignment

PORT Functional Description

The number of ports and each's composition are defined in the processor data sheet. Each port has a dedicated set of MMR registers that control pin functions and operates in general-purpose I/O (GPIO) mode by default, as controlled by the port-specific PORT_FER register. Each bit in this register, as well as the other PORT MMRs, represents a specific GPIO pin on the specified port.

Input Mode, Output Mode, and Open-Drain Mode of GPIO Operation

At reset, every GPIO pin defaults to input mode with the input drivers disabled. To enable any GPIO input driver, set the bits corresponding to the individual pins in the appropriate input enable register (PORT_INEN).

The GPIO output drivers are enabled by setting the corresponding bits in the direction registers (PORT_DIR).

The PORT can use every GPIO in open-drain mode by clearing the respective bit in the PORT_DATA register or setting the respective bit in the PORT_DATA_CLR register. Then, set the corresponding bit in the PORT_INEN register. Read from the PORT_DATA register to obtain the status from the pin.

Port Multiplexing Controlled on Pin-by-Pin Basis

Each port has two dedicated MMRs that control the port multiplexing, the 16-bit function enable (PORT_FER) registers and the 32-bit port multiplexing (PORT_MUX) registers.

All Port Pins Provide Interrupt Functionality

Pin interrupts are completely decoupled from GPIO functionality. Pins are connected to the system event controller (SEC) via the PINTx modules, each of which is configurable in terms of which port pins are sensed for interrupt generation.

ADSP-SC59x PORT Register List

The PORT module (PORT) regulates the use of the multiplexable processor pins. Every port pin can operate in general-purpose I/O (GPIO) mode or as an alternate function. This GPIO operation is the default after processor reset and is controlled by a set of registers that control GPIO functionality. Every bit in these registers represents a certain GPIO pin of a specific port. For more information on PORT functionality, see the PORT register descriptions.

Table 13-1: ADSP-SC59x PORT Register List

Name	Description
PORT_DATA	Port x GPIO Data Register
PORT_DATA_CLR	Port x GPIO Data Clear Register
PORT_DATA_SET	Port x GPIO Data Set Register
PORT_DATA_TGL	Port x GPIO Output Toggle Register
PORT_DIR	Port x GPIO Direction Register
PORT_DIR_CLR	Port x GPIO Direction Clear Register
PORT_DIR_SET	Port x GPIO Direction Set Register
PORT_FER	Port x Function Enable Register
PORT_FER_CLR	Port x Function Enable Clear Register
PORT_FER_SET	Port x Function Enable Set Register
PORT_INEN	Port x GPIO Input Enable Register
PORT_INEN_CLR	Port x GPIO Input Enable Clear Register
PORT_INEN_SET	Port x GPIO Input Enable Set Register
PORT_LOCK	Port x GPIO Lock Register
PORT_MUX	Port x Multiplexer Control Register
PORT_POL	Port x GPIO Polarity Invert Register
PORT_POL_CLR	Port x GPIO Polarity Invert Clear Register
PORT_POL_SET	Port x GPIO Polarity Invert Set Register
PORT_TRIG_TGL	Port x GPIO Trigger Toggle Register

ADSP-SC59x PORT Trigger List

Table 13-2: ADSP-SC59x PORT Trigger List Generators

Trigger ID	Name	Description	Sensitivity
None			

Table 13-3: ADSP-SC59x PORT Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
41	PORTA_TOGGLE	Port Toggle Trigger	Pulse
42	PORTB_TOGGLE	Port Toggle Trigger	Pulse
43	PORTC_TOGGLE	Port Toggle Trigger	Pulse
44	PORTD_TOGGLE	Port Toggle Trigger	Pulse
45	PORTE_TOGGLE	Port Toggle Trigger	Pulse
46	PORTF_TOGGLE	Port Toggle Trigger	Pulse
47	PORTG_TOGGLE	Port Toggle Trigger	Pulse
48	PORTH_TOGGLE	Port Toggle Trigger	Pulse
49	PORTI_TOGGLE	Port Toggle Trigger	Pulse

ADSP-SC59x PINT Register List

The Pin Interrupt module (PINT) controls the pin-to-interrupt assignment in a byte-wide manner. The pin-interrupt assignment registers do not consist of 32 individual bits. They consist of four control bytes, each functioning as a multiplexer control. For more information, see the PINT register descriptions.

All PINT registers are 32 bits wide and can be accessed by 32-bit load/store instructions. They also support 16-bit operation where the upper 16 bits are ignored and the application uses the lower 16 bits only. Consequently, all PINT registers support 32-bit accesses as well as 16-bit accesses for the lower half words. Applications may use faster 16-bit accesses as long as they do not require functionality of upper register halves.

Table 13-4: ADSP-SC59x PINT Register List

Name	Description
PINT_ASSIGN	PINT Assign Register
PINT_EDGE_CLR	PINT Edge Clear Register
PINT_EDGE_SET	PINT Edge Set Register
PINT_INV_CLR	PINT Invert Clear Register
PINT_INV_SET	PINT Invert Set Register
PINT_LATCH	PINT Latch Register
PINT_MSK_CLR	PINT Mask Clear Register
PINT_MSK_SET	PINT Mask Set Register
PINT_PINSTATE	PINT Pin State Register
PINT_REQ	PINT Request Register

ADSP-SC59x PINT Interrupt List

Table 13-5: ADSP-SC59x PINT Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
65	PINT0_BLOCK	PINT0 Pin Interrupt Block 0	Level	
66	PINT1_BLOCK	PINT1 Pin Interrupt Block 1	Level	
67	PINT2_BLOCK	PINT2 Pin Interrupt Block 2	Level	
68	PINT3_BLOCK	PINT3 Pin Interrupt Block 3	Level	
69	PINT4_BLOCK	PINT4 Pin Interrupt Block 4	Level	
70	PINT5_BLOCK	PINT5 Pin Interrupt Block 5	Level	
71	PINT6_BLOCK	PINT6 Pin Interrupt Block 6	Level	
72	PINT7_BLOCK	PINT7 Pin Interrupt Block 7	Level	

ADSP-SC59x PINT Trigger List

Table 13-6: ADSP-SC59x PINT Trigger List Generators

Trigger ID	Name	Description	Sensitivity
68	PINT0_BLOCK	PINT0 Pin Interrupt Block	Level
69	PINT1_BLOCK	PINT1 Pin Interrupt Block	Level
70	PINT2_BLOCK	PINT2 Pin Interrupt Block	Level
71	PINT3_BLOCK	PINT3 Pin Interrupt Block	Level
72	PINT4_BLOCK	PINT4 Pin Interrupt Block	Level
73	PINT5_BLOCK	PINT5 Pin Interrupt Block	Level
74	PINT6_BLOCK	PINT6 Pin Interrupt Block	Level
75	PINT7_BLOCK	PINT7 Pin Interrupt Block	Level

Table 13-7: ADSP-SC59x PINT Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

ADSP-SC59x PADS Register List

The PADS controls signal hysteresis and other system interface signal features for a number of module interfaces.

Table 13-8: ADSP-SC59x PADS Register List

Name	Description
PADS_DAI0_0_DS	DAI0 1 to 10 pins DS control

Table 13-8: ADSP-SC59x PADS Register List (Continued)

Name	Description
PADS_DAI0_1_DS	DAI0 11 to 20 pins DS control
PADS_DAI0_IE	DAI0 Port Input Enable Control Register
PADS_DAI1_0_DS	DAI1 1 to 10 pins DS control
PADS_DAI1_1_DS	DAI1 11 to 20 pins DS control
PADS_DAI1_IE	DAI1 Port Input Enable Control Register
PADS_DAI[n]_PDE	DAI[n] Pull-Down Enable
PADS_DAI[n]_PUE	DAI[n] Pull-Up Enable
PADS_NONPORTS_DS	Non-GPIO Drive Strength Register
PADS_PCFG0	Peripheral PAD Configuration0 Register
PADS_PCFG1	Peripheral Configuration1 Register
PADS_PORTA0_DS	PORTA 0 to 7 pins DS control
PADS_PORTA1_DS	PORTA 8 - 15 pins DS control
PADS_PORTB0_DS	PORTB 0 to 7 pins DS control
PADS_PORTB1_DS	PORTB 8 - 15 pins DS control
PADS_PORTC0_DS	PORTC 0 to 7 pins DS control
PADS_PORTC1_DS	PORTC 8 - 15 pins DS control
PADS_PORTD0_DS	PORTD 0 to 7 pins DS control
PADS_PORTD1_DS	PORTD 8 to 15 pins DS control
PADS_PORTE0_DS	PORTE 0 to 7 pins DS control
PADS_PORTE1_DS	PORTE 8 to 15 pins DS control
PADS_PORTF0_DS	PORTF 0 to 7 pins DS control
PADS_PORTF1_DS	PORTF 8 to 15 pins DS control
PADS_PORTG0_DS	PORTG 0 to 7 pins DS control
PADS_PORTG1_DS	PORTG 8 to 15 pins DS control
PADS_PORTH0_DS	PORTH 0 to 7 pins DS control
PADS_PORTH1_DS	PORTH 8 to 15 pins DS control
PADS_PORTI0_DS	PORTI 0 to 7 pins DS control
PADS_PORT[n]_PDE	PORT[n] Pull-Down Enable
PADS_PORT[n]_PUE	PORT[n] Pull-Up Enable

PORT Architectural Concepts

These sections describe in more detail how the PORT module connects externally to pins and internally to the MMR bus. Ports are named alphabetically beginning with A.

- Internal Interfaces
- External Interfaces
- GPIO Pin Function
- Port Multiplexing Control

Internal Interfaces

All of the pin multiplexing, GPIO, and pin interrupt control block MMRs can be accessed through the MMR bus. There is no DMA support. Each of the pin interrupt (PINTx) modules has its own dedicated interrupt request output signal that connects directly to the system event controller (SEC).

External Interfaces

The pin multiplexing hardware can be seen as a layer between the on-chip peripherals and the silicon pads connecting to the physical pins/balls or the package, as controlled by the PORT unit.

GPIO Pin Function

By default, the PORT sets every GPIO pin to input mode. The input drivers are not enabled, which avoids the need for unnecessary current sinks and external termination resistors on unused pins.

Input Mode

The default mode of every GPIO pin after reset is input mode, but the input drivers are not enabled. To enable GPIO input drivers, set the bits corresponding to the PORT pins in the appropriate input enable register (`PORT_INEN`). When enabled, a read from the `PORT_DATA` register returns the logical state of the input pins. However, the input signal does not overwrite the state of the internal flip-flop used for providing output to the same pin. Only software can alter the state. If the input driver is enabled, a write to the `PORT_DATA` register can alter the state of the flip-flop, but the change cannot be read back.

Output Mode

Any GPIO pin can be configured for output mode. The GPIO output drivers are enabled by setting the bits corresponding to the PORT pins in the appropriate direction register. The PORT implements direction registers as a pair of write-1-to-set (W1S) and write-1-to-clear (W1C) MMRs called `PORT_DIR_SET` and `PORT_DIR_CLR`, respectively. As such, software can alter the direction of the signal flow on individual GPIO pins without impacting other GPIOs on the same port.

Both the `PORT_DIR_SET` and `PORT_DIR_CLR` registers return the same value when read, and a logical 1 indicates an enabled output. The `PORT_DATA` registers control the state of output pins. A logical 0 drives the output low while a logical 1 drives the output high.

While writes to the `PORT_DATA` register can alter all of the GPIOs on a specific port at once, there are also a pair of W1S and W1C MMRs called `PORT_DATA_SET` and `PORT_DATA_CLR`, respectively. These registers enable

the manipulation of individual GPIO outputs. The state of the outputs can be obtained by reading the PORT_DATA registers. Because the state of the GPIO output can be controlled before the output driver is enabled, set or clear the internal flip-flop first by programming this register to avoid volatile levels on the output pin.

Trigger Toggle Mode

Any GPIO pin that has been configured for output mode can be toggled using a trigger input from the Trigger Routing Unit (TRU). To enable this functionality for a particular GPIO, set the appropriate bit in the PORT_TRIG_TGL register. Any subsequent trigger for the designated port causes all GPIO outputs set in the PORT_TRIG_TGL register to toggle.

To avoid unpredictable behavior, do not set, clear, or toggle the PORT_DATA, PORT_DATA_SET, PORT_DATA_CLR, or PORT_DATA_TGL registers when the GPIO output has the corresponding PORT_TRIG_TGL bit set. To change the state of the GPIO output using one of these registers, first clear the corresponding bit in the PORT_TRIG_TGL register.

Open-Drain Mode

Every GPIO can also be used in open-drain mode. First, either clear the respective bit in the PORT_DATA register or set the respective bit in the PORT_DATA_CLR register. Then, set the appropriate bit in the PORT_INEN register. Read from the PORT_DATA register to return the status from the pin rather than the state of the internal flip-flop.

By toggling the output driver through the PORT_DIR_SET and PORT_DIR_CLR register pair, the output signal can be pulled low or three-stated, as required. The polarity of the driven signal can be inverted when the internal flip-flop is set. When using a GPIO port in open-drain mode, take care to not exceed the V_{IH} operating condition associated with the respective pins.

Port Multiplexing Control

To configure pins properly, consult the processor data sheet to determine which bits in the PORT_FER and PORT_MUX register map to the pin of interest, and then set these registers appropriately for the desired function.

After reset, all port pins default to GPIO input mode with their output and input drivers disabled. As a result, all unused port pins can be left unconnected. Disabled pins appear as a high impedance to external circuits.

Each port has two dedicated MMRs that control the port multiplexing, the 16-bit function enable (PORT_FER) registers and the 32-bit port multiplexing (PORT_MUX) registers.

The function enable register specifies whether the pin is used as a GPIO pin or allocated for use by a specific peripheral, but it does not specify what the peripheral function is. Each bit in the 16-bit PORT_FER register corresponds to an individual port pin. For example, if bit 1 (PA1) of the PORTA_FER register is cleared, the PA_01 pin is configured as a GPIO. When set, one of the available peripheral functions becomes active on the PA_01 pin instead.

Pairs of bits in the PORT_MUX register control the multiplexing between the peripheral functions available to an individual pin, as some PORT pins provide up to four possible peripheral functions.

Refer to the Signal Muxing table in the data sheet for the specific PORT_MUX settings.

PORT Event Control

The following sections describe event generation in the PORT module.

PORT Interrupt Signals

The pin interrupts are decoupled from GPIO functionality, providing the following advantages.

- Flexible mapping scheme enables pins from up to four different ports to be grouped into one common interrupt scheme.
- Interrupt requests work on input and output pins regardless of whether the pin is functioning as a GPIO or a peripheral.

The processor has a number of interrupt channels dedicated to pin interrupts, managed by a set of pin interrupt (PINTx) blocks. Each PINTx block can sense up to 32 GPIO pins, as described in the following list and figure.

- PINT0 can sense pin activity on PORTA and PORTB
- PINT1 can sense pin activity on PORTB and PORTC
- PINT2 can sense pin activity on PORTC and PORTD
- PINT3 can sense pin activity on PORTD and PORTE
- PINT4 can sense pin activity on PORTE and PORTF
- PINT5 can sense pin activity on PORTF and PORTG
- PINT6 can sense pin activity on PORTG and PORTH
- PINT7 can sense pin activity on PORTH and PORTI

The processor supports both 32-bit and 16-bit peripheral bus accesses to PINTx registers.

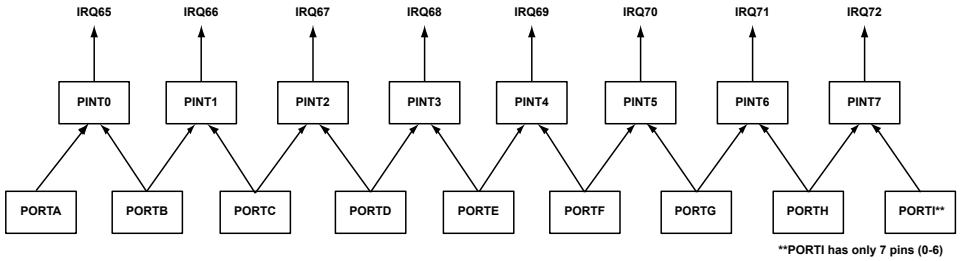


Figure 13-2: GPIO to PINTx Assignment

Pins connect to the PINTx module and then to the system event controller (SEC), as shown in the ***PINTx Block Diagram***.

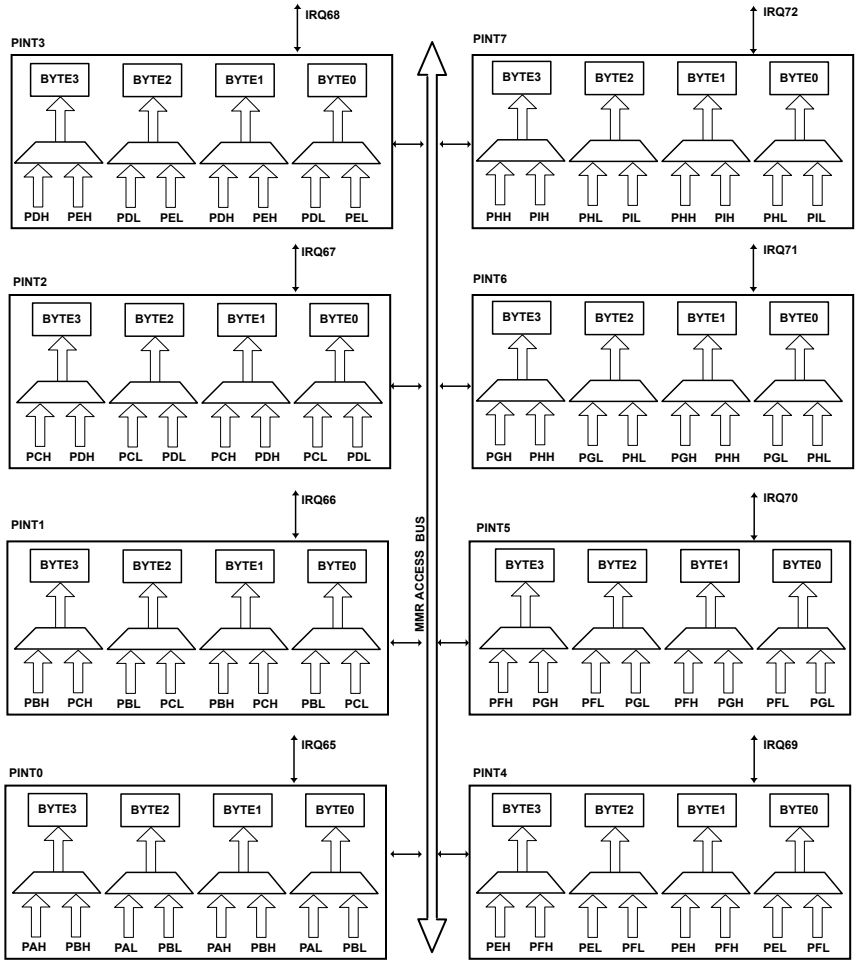


Figure 13-3: PINTx Block Diagram

As shown in the ***PINTx Block Diagram***, each port is subdivided into two 8-pin half ports, upper (`PxH`) and lower (`PxL`). The `PINT_ASSIGN` registers control the 8-bit multiplexers associated with these half ports, where the lower half units (eight pins) can be forwarded to either byte 0 or byte 2 of the PINTx blocks, and the upper half units (eight pins) can be forwarded to either byte 1 or byte 3 of the PINTx blocks.

When a half port is assigned to a byte in any PINTx block, the state of the eight pins appears in the `PINT_PINSTATE` register, regardless of whether the pin is enabled for GPIO or peripheral functions (input or output). When neither the input nor output drivers of the pin are enabled, the pin state is read as zero. The `PINT_PINSTATE` register reports the inverted state of the pin when the `PINT_INV_SET` register activates the signal inverter. The inverter can be enabled on an individual bit-by-bit basis. Each bit in the `PINT_INV_SET`/`PINT_INV_CLR` register pair represents a pin signal.

By default, PORT interrupt request generation is level-sensitive, and an interrupt request occurs when the enabled pin is sensed as active high. Use the `PINT_EDGE_SET` register to change the interrupt request genearion scheme to instead be edge-sensitive (rising edge generates the interrupt request). Use the `PINT_INV_SET` register to invert the polarity such that the PINTx block generates the interrupt request on active-low signals or falling edges.

The PINTx modules also assist when both signal edges must generate unique interrupt requests. If two different interrupt requests are required, the PINT_ASSIGN registers can route a single input signal to two different PINTx blocks, where one block inverts the signal in the PINT_INV_SET register and the other one does not. In this fashion, a unique software routine is associated with the hardware PINTx block that is generating the unique interrupt request for each signal edge. When both signal edges can be serviced by the same interrupt request, each half port can be routed to two separate bytes within a single PINTx block using the PINT_ASSIGN register, and then one of the half ports needs to have the inversion enabled in the PINT_INV_SET register. The servicing software routine can then detect from the PINT_LATCH register whether a falling, rising, or both edges have occurred.

Regardless of whether level-sensitive or edge-sensitive mode is used, the hardware always latches an interrupt request. Latched signals can be read from the PINT_LATCH registers. Only a software or hardware reset clears the latches. To clear the latch, a W1C operation must be performed to the PINT_REQ or PINT_LATCH register. When in level-sensitive mode, the interrupt request remains asserted if the pin state does not change by the time the interrupt service routine exits.

Because every PINTx block groups up to 32 pin signals, the PINT_MSK_SET/ PINT_MSK_CLR register pair can control which of the signals can request an interrupt at the system level. Software can interrogate the PINT_REQ register for signaling pins. The PINT_REQ bits represent a logical AND between the mask and the latch. When any of these bits is set, the block output interrupt request is asserted.

PORT Programming Model

The ***GPIO Programming Model Flow (Part 1)***, ***GPIO Programming Model Flow (Part 2)***, and ***GPIO Programming Model Flow (Part 3)*** figures show the programming model for the general-purpose ports. This programming includes the GPIO input and output operation, open-drain mode, and the pin interrupt PINTx modules.

NOTE: These process flow diagrams connect where call-out letters appear. For example, call-out A in the ***GPIO Programming Model Flow (Part 1)*** diagram connects to call-out A in the ***GPIO Programming Model Flow (Part 2)*** diagram.

The following flowcharts describe the processes for setting up pins for various functions. Begin the process from the start label in the ***GPIO Programming Model Flow (Part 1)*** figure. The first decision (GPIO or peripheral) determines how to program the PORT_FER register. Set the bit(s) corresponding to the pin(s) to 1 to enable peripheral functionality or to 0 to enable GPIO mode. For more information on setting up for peripheral functions, refer to Port Multiplexing Control.

If the pin is to be a GPIO pin, a subsequent series of decisions must be made that will impact how the PORT_DATA, PORT_POL, PORT_DIR, and PORT_INEN configuration registers must be programmed. Depending on the type of GPIO pin desired, some configurations do not apply and can have different meanings. The following paragraphs briefly describe the function of the different settings for each of the pin functions in the input, output, and open-drain GPIO modes. It is a best practice to use the SET or CLR versions of the PORT registers, where applicable, to effect changes on a pin-by-pin basis rather than on the full port.

For output mode, first clear the PORT_DATA register to set all the pins low. Then write the PORT_DIR register to define the direction of each pin (set the bits associated with the desired output pins to 1). In output mode, the other registers are not significant. The ***GPIO Programming Model Flow (Part 1)*** chart shows this flow starting at label 2.

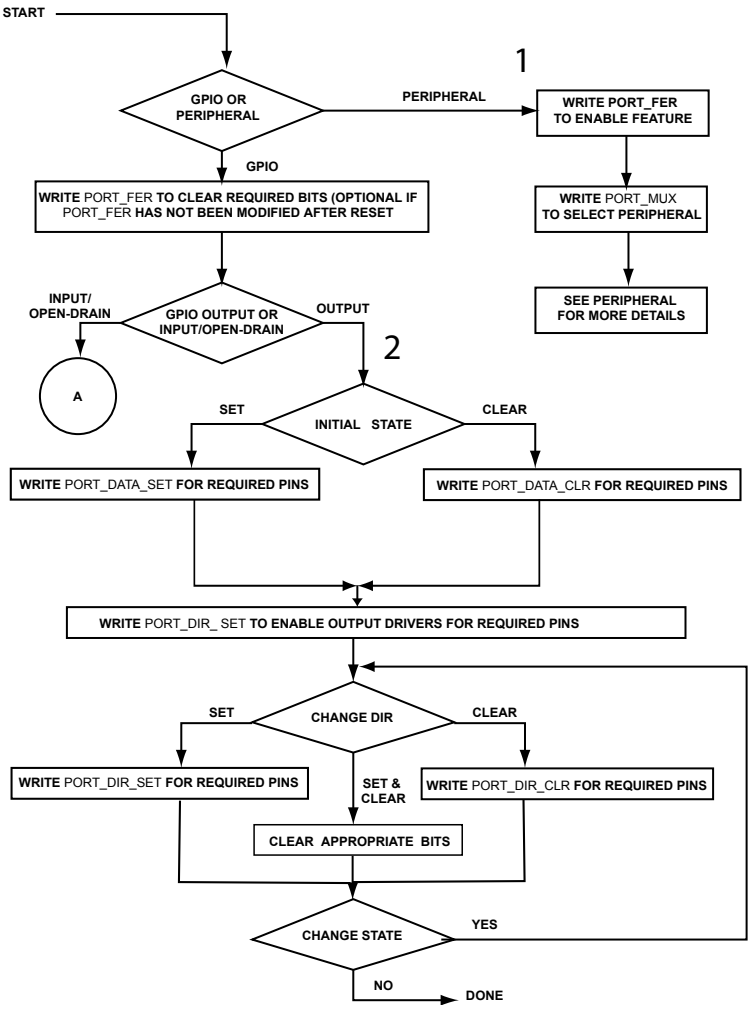


Figure 13-4: GPIO Programming Model Flow (Part 1)

For input mode, first decide the polarity for each pin using the PORT_POL register. Program the PORT_DIR register to define the appropriate pins as inputs (write a 0 to the bit location associated with the pin). If interrupt requests are desired, configure the PINT module as shown in the ***GPIO Programming Model Flow (Part 3)*** figure starting at label B. Finally, write the PORT_INEN register to enable the associated input drivers. The ***GPIO Programming Model Flow (Part 2)*** chart shows this entire flow starting at label 3.

For open-drain mode, set all pins low by clearing the PORT_DATA register. Then, use the PORT_INEN register to enable the appropriate input drivers. Set the PORT_DIR register in this mode to indicate whether the pin is in an active state or not (active being 0). The ***GPIO Programming Model Flow (Part 2)*** chart shows this flow starting at label 4.

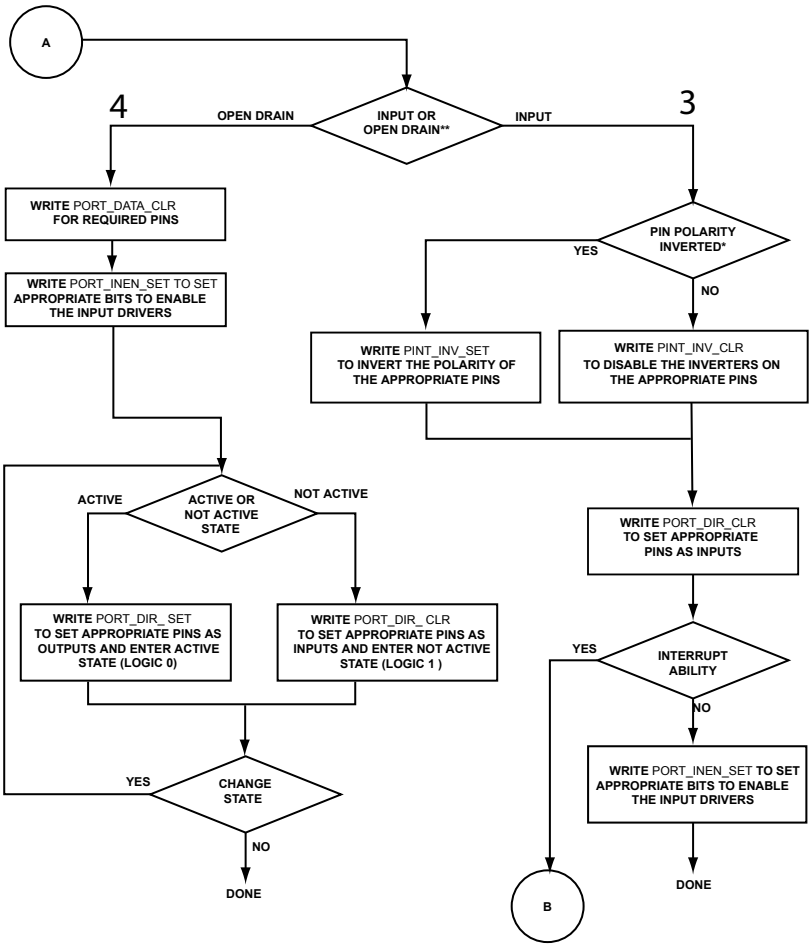


Figure 13-5: GPIO Programming Model Flow (Part 2)

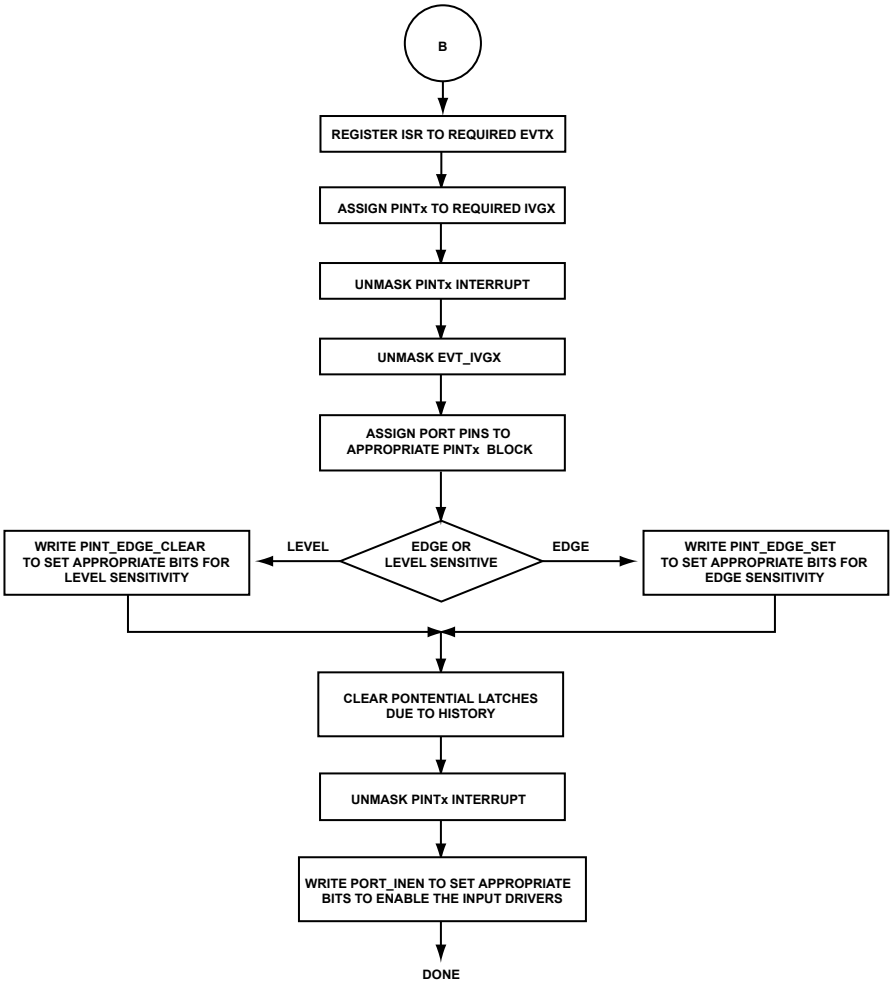


Figure 13-6: GPIO Programming Model Flow (Part 3)

Programmable Pull-up Resistors for PORT and DAI

There are programmable weak pull-up resistors for DAI and PORT pins. These resistors can be enabled or disabled by configuring the `PADS_DAI[n]_PUE`, `PADS_PORT[n]_PUE`, `PADS_DAI[n]_PDE`, and `PADS_PORT[n]_PDE` registers.

For DAI pins, if both the `PADS_DAI[n]_PDE` and `PADS_DAI[n]_PUE` registers are set (=1) for the same pin, then the `PADS_DAI[n]_PDE` register takes priority over `PADS_DAI[n]_PUE` register.

NOTE: For unused PORT/DAI pins, ensure that any unused pin does not go to floating state. Clear (=0) the PUD bit of that pin.

Refer to the ***DAI Pull-up Resistor States*** table. To enable the pull-up on a DAI pin, the `PADS_DAI[n]_PDE.PDE` bit must be cleared (=0) ***AND*** the `PADS_DAI[n]_PUE.PUE` bit must be set (=1). To disable the pull-up on a DAI pin, the `PADS_DAI[n]_PDE.PDE` bit must be set (=1) and `PADS_DAI[n]_PUE.PUE` bit can be any value.

Table 13-9: DAI Pull-up Resistor States

PADS_DAI[n]_PUE.PUE	PADS_DAI[n]_PDE.PDE	Pull-up Resistor State
0	0	Disabled
0	1	Disabled
1	0	Enabled
1	1	Disabled

Similarly, for PORT pins, if both PADS_PORT[n]_PDE.PDE and PADS_PORT[n]_PUE registers are set (=1) for the same pin, then the PADS_PORT[n]_PDE.PDE register takes priority over PADS_PORT[n]_PUE register.

Refer to the ***PORT Pull-up Resistor States*** table. To enable the pull-up on a PORT pin, the PADS_PORT[n]_PDE.PDE bit must be cleared (=0) ***AND*** the PADS_PORT[n]_PUE.PUE bit must be set (=1). To disable the pull-up on a PORT pin, PADS_DAI[n]_PDE.PDE bit must be set (=1) and PADS_PORT[n]_PUE.PUE bit can be any value.

Table 13-10: PORT Pull-up Resistor States

PADS_PORT[n]_PUE.PUE	PADS_PORT[n]_PDE.PDE	Pull-up Resistor State
0	0	Disabled
0	1	Disabled
1	0	Enabled
1	1	Disabled

NOTE: Refer to the ***Designer Quick Reference*** section of the data sheet for handling unused PORT and DAI pins.

NOTE: The internal pull-up on GPIO pins are very weak. These pins hold a defined logic level when pins are left floating. The internal pull-up resistance varies over a wide range. If an application needs a specified pull-up on the pin, use an external resistor.

NOTE: For unused PORT/DAI pins, ensure that any unused pin does not go to floating state. Clear (=0) the PUD bit of that pin.

ADSP-SC59x PORT Register Descriptions

The General-Purpose Input/Output Port (PORT) contains the following registers.

Table 13-11: ADSP-SC59x PORT Register List

Name	Description
PORT_DATA	Port x GPIO Data Register
PORT_DATA_CLR	Port x GPIO Data Clear Register

Table 13-11: ADSP-SC59x PORT Register List (Continued)

Name	Description
PORT_DATA_SET	Port x GPIO Data Set Register
PORT_DATA_TGL	Port x GPIO Output Toggle Register
PORT_DIR	Port x GPIO Direction Register
PORT_DIR_CLR	Port x GPIO Direction Clear Register
PORT_DIR_SET	Port x GPIO Direction Set Register
PORT_FER	Port x Function Enable Register
PORT_FER_CLR	Port x Function Enable Clear Register
PORT_FER_SET	Port x Function Enable Set Register
PORT_INEN	Port x GPIO Input Enable Register
PORT_INEN_CLR	Port x GPIO Input Enable Clear Register
PORT_INEN_SET	Port x GPIO Input Enable Set Register
PORT_LOCK	Port x GPIO Lock Register
PORT_MUX	Port x Multiplexer Control Register
PORT_POL	Port x GPIO Polarity Invert Register
PORT_POL_CLR	Port x GPIO Polarity Invert Clear Register
PORT_POL_SET	Port x GPIO Polarity Invert Set Register
PORT_TRIG_TGL	Port x GPIO Trigger Toggle Register

Port x GPIO Data Register

The operation of the PORT_DATA register depends on whether the bit/pin is in output mode or input mode. In both modes, a set bit in the PORT_DATA register corresponds to a signal high on a GPIO pin. A cleared bit in the PORT_DATA register corresponds to a signal low on a GPIO pin.

The PORT_DATA, PORT_DATA_SET, and PORT_DATA_CLR registers control the state of GPIO pins in output mode. To enable output mode (and output drivers), use the PORT_DIR_SET and PORT_DIR_CLR registers.

Writes to the PORT_DATA register affect the state of all pins of the port that are in output mode. To set or clear specific pins without impacting other pins of the port, use the PORT_DATA_SET and PORT_DATA_CLR registers.

When the GPIO pins are in input mode (input driver is enabled with the PORT_INEN register), reads from the PORT_DATA, PORT_DATA_SET, and PORT_DATA_CLR registers return the state of the respective GPIO pins.

Note that when the input driver is not enabled, reads from the PORT_DATA, PORT_DATA_SET, and PORT_DATA_CLR registers return the value previously written to the registers.

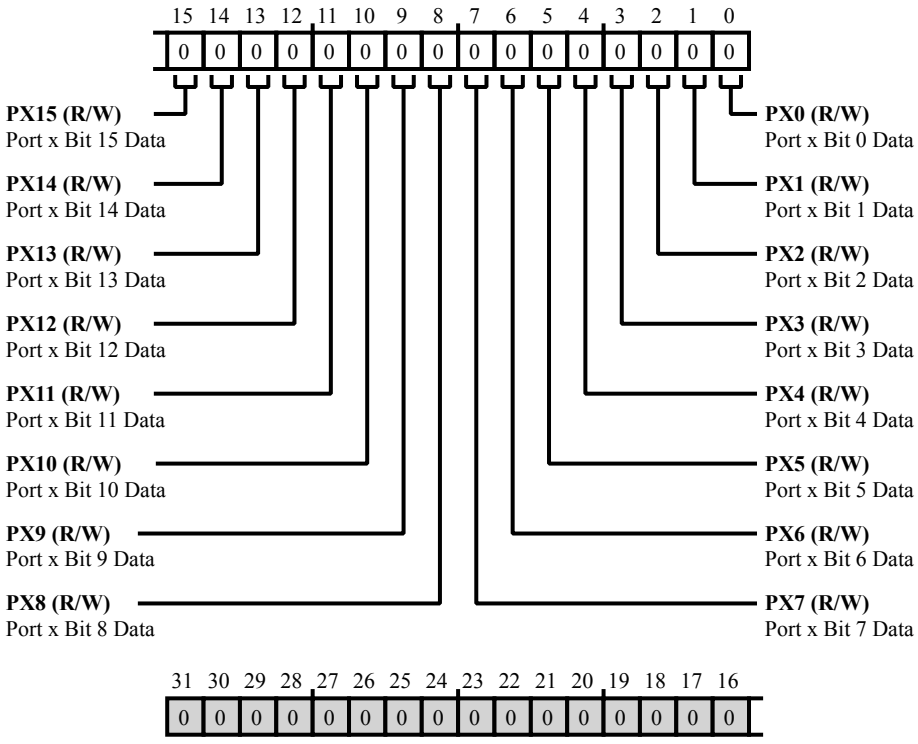


Figure 13-7: PORT_DATA Register Diagram

Table 13-12: PORT_DATA Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Data. The PORT_DATA.PX15 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
14 (R/W)	PX14	Port x Bit 14 Data. The PORT_DATA.PX14 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
13 (R/W)	PX13	Port x Bit 13 Data. The PORT_DATA.PX13 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
12 (R/W)	PX12	Port x Bit 12 Data. The PORT_DATA.PX12 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
11 (R/W)	PX11	Port x Bit 11 Data. The PORT_DATA.PX11 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
10 (R/W)	PX10	Port x Bit 10 Data. The PORT_DATA.PX10 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
9 (R/W)	PX9	Port x Bit 9 Data. The PORT_DATA.PX9 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High

Table 13-12: PORT_DATA Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	PX8	Port x Bit 8 Data. The PORT_DATA.PX8 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
7 (R/W)	PX7	Port x Bit 7 Data. The PORT_DATA.PX7 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
6 (R/W)	PX6	Port x Bit 6 Data. The PORT_DATA.PX6 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
5 (R/W)	PX5	Port x Bit 5 Data. The PORT_DATA.PX5 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
4 (R/W)	PX4	Port x Bit 4 Data. The PORT_DATA.PX4 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
3 (R/W)	PX3	Port x Bit 3 Data. The PORT_DATA.PX3 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
2 (R/W)	PX2	Port x Bit 2 Data. The PORT_DATA.PX2 bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High

Table 13-12: PORT_DATA Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	PX1	Port x Bit 1 Data. The `PORT_DATA.PX1` bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High
0 (R/W)	PX0	Port x Bit 0 Data. The `PORT_DATA.PX0` bit indicates a signal on a GPIO pin.	
		0	Signal Low
		1	Signal High

Port x GPIO Data Clear Register

The PORT_DATA_CLR register operates differently for port bits/pins, depending on whether the bit/pin is output mode or input mode. For more information, see the PORT_DATA register description.

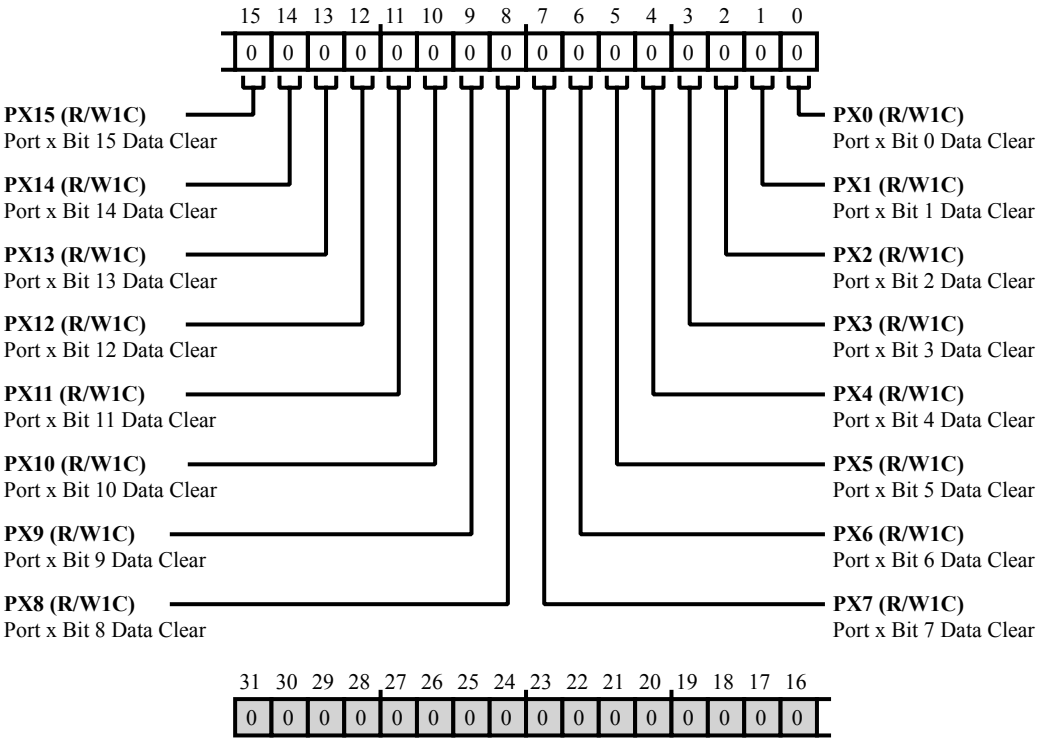


Figure 13-8: PORT_DATA_CLR Register Diagram

Table 13-13: PORT_DATA_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Data Clear. The PORT_DATA_CLR.PX15 bit clears the pin without impacting other pins of the port.	
		0	No Effect
		1	Clear Bit. Write 1 for signal low in output mode.
14 (R/W1C)	PX14	Port x Bit 14 Data Clear. The PORT_DATA_CLR.PX14 bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.

Table 13-13: PORT_DATA_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1C)	PX13	Port x Bit 13 Data Clear. The `PORT_DATA_CLR.PX13` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
12 (R/W1C)	PX12	Port x Bit 12 Data Clear. The `PORT_DATA_CLR.PX12` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
11 (R/W1C)	PX11	Port x Bit 11 Data Clear. The `PORT_DATA_CLR.PX11` bit clears the pin without impacting other pins of the port.	
		0	No Effect
		1	Clear Bit. Write 1 for signal low in output mode.
10 (R/W1C)	PX10	Port x Bit 10 Data Clear. The `PORT_DATA_CLR.PX10` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
9 (R/W1C)	PX9	Port x Bit 9 Data Clear. The `PORT_DATA_CLR.PX9` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
8 (R/W1C)	PX8	Port x Bit 8 Data Clear. The `PORT_DATA_CLR.PX8` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.

Table 13-13: PORT_DATA_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	PX7	Port x Bit 7 Data Clear. The `PORT_DATA_CLR.PX7` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
6 (R/W1C)	PX6	Port x Bit 6 Data Clear. The `PORT_DATA_CLR.PX6` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
5 (R/W1C)	PX5	Port x Bit 5 Data Clear. The `PORT_DATA_CLR.PX5` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
4 (R/W1C)	PX4	Port x Bit 4 Data Clear. The `PORT_DATA_CLR.PX4` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
3 (R/W1C)	PX3	Port x Bit 3 Data Clear. The `PORT_DATA_CLR.PX3` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
2 (R/W1C)	PX2	Port x Bit 2 Data Clear. The `PORT_DATA_CLR.PX2` bit clears the pin without impacting other pins of the port.	
		0	No Effect Write 0 has no effect in output mode.
		1	Clear Bit Write 1 for signal low in output mode.

Table 13-13: PORT_DATA_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	PX1	Port x Bit 1 Data Clear. The `PORT_DATA_CLR.PX1` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.
0 (R/W1C)	PX0	Port x Bit 0 Data Clear. The `PORT_DATA_CLR.PX0` bit clears the pin without impacting other pins of the port.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Clear Bit. Write 1 for signal low in output mode.

Port x GPIO Data Set Register

The PORT_DATA_SET register operates differently for port bits/pins, depending on whether the bit/pin is output mode or input mode. For more information, see the PORT_DATA register description.

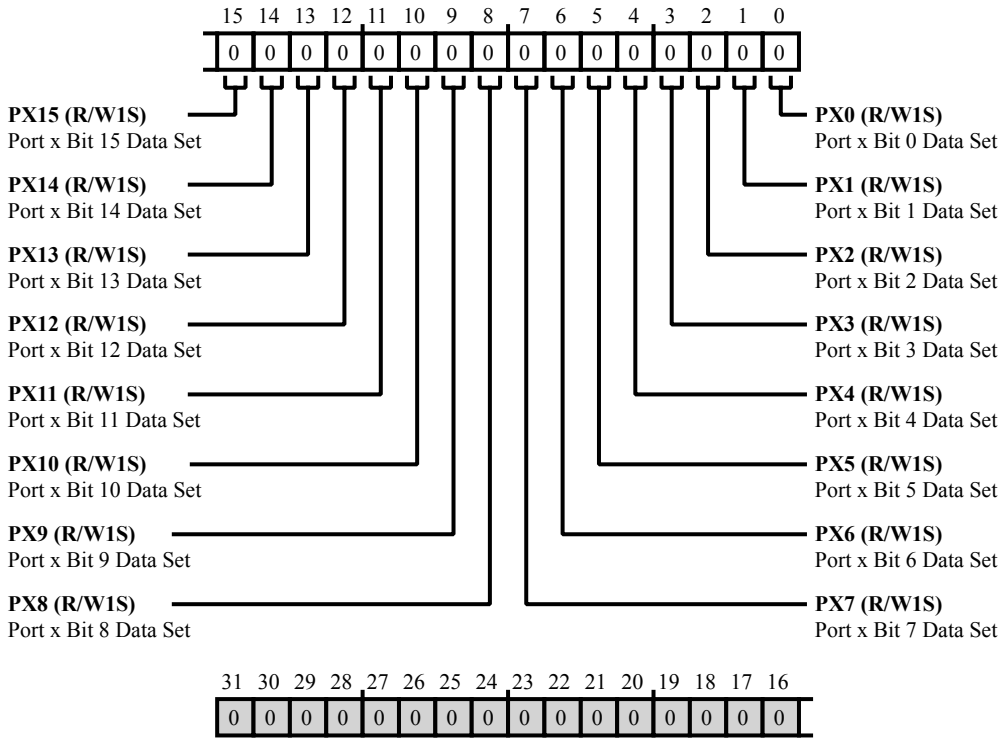


Figure 13-9: PORT_DATA_SET Register Diagram

Table 13-14: PORT_DATA_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
14 (R/W1S)	PX14	Port x Bit 14 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
13 (R/W1S)	PX13	Port x Bit 13 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.

Table 13-14: PORT_DATA_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W1S)	PX12	Port x Bit 12 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
11 (R/W1S)	PX11	Port x Bit 11 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit.
10 (R/W1S)	PX10	Port x Bit 10 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
9 (R/W1S)	PX9	Port x Bit 9 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
8 (R/W1S)	PX8	Port x Bit 8 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
7 (R/W1S)	PX7	Port x Bit 7 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
6 (R/W1S)	PX6	Port x Bit 6 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
5 (R/W1S)	PX5	Port x Bit 5 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
4 (R/W1S)	PX4	Port x Bit 4 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
3 (R/W1S)	PX3	Port x Bit 3 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.

Table 13-14: PORT_DATA_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1S)	PX2	Port x Bit 2 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
1 (R/W1S)	PX1	Port x Bit 1 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.
0 (R/W1S)	PX0	Port x Bit 0 Data Set.	
		0	No Effect. Write 0 has no effect in output mode.
		1	Set Bit. Write 1 for signal high in output mode.

Port x GPIO Output Toggle Register

The PORT_DATA_TGL register permits toggling the state of output GPIO pins. Setting bits in the PORT_DATA_TGL register affects the state of specific pins without impacting other pins of the port.

Reading the PORT_DATA_TGL returns the state of the PORT_DATA register output pin state, but does not return the input pin/signal state.

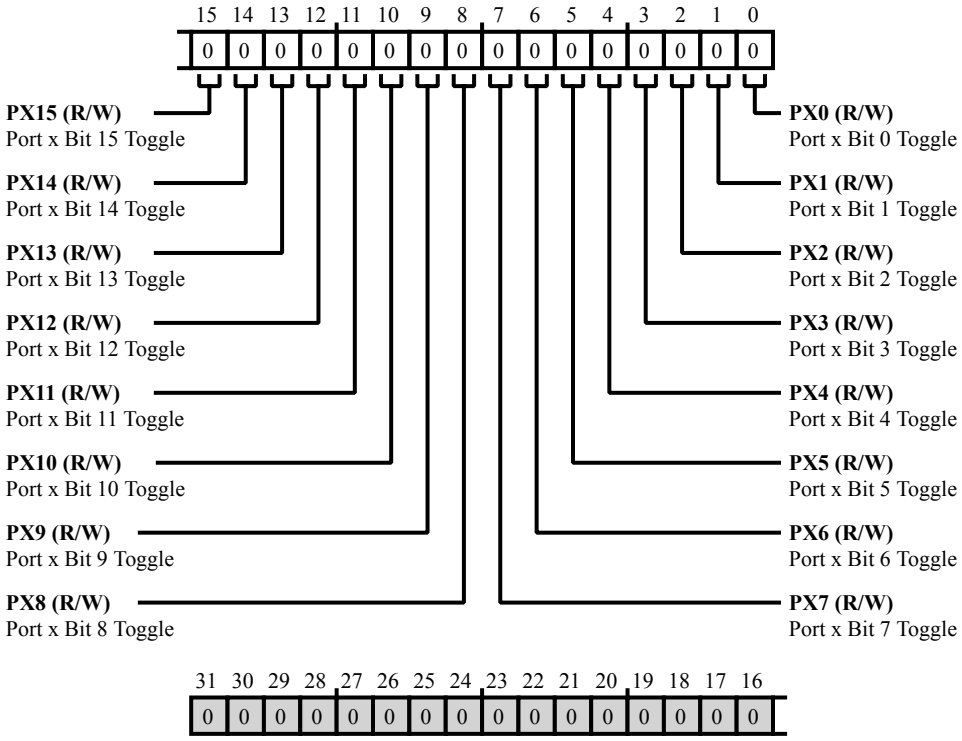


Figure 13-10: PORT_DATA_TGL Register Diagram

Table 13-15: PORT_DATA_TGL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Toggle. The PORT_DATA_TGL.PX15 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit.
14 (R/W)	PX14	Port x Bit 14 Toggle. The PORT_DATA_TGL.PX14 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit

Table 13-15: PORT_DATA_TGL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	PX13	Port x Bit 13 Toggle. The PORT_DATA_TGL.PX13 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
12 (R/W)	PX12	Port x Bit 12 Toggle. The PORT_DATA_TGL.PX12 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
11 (R/W)	PX11	Port x Bit 11 Toggle. The PORT_DATA_TGL.PX11 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
10 (R/W)	PX10	Port x Bit 10 Toggle. The PORT_DATA_TGL.PX10 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
9 (R/W)	PX9	Port x Bit 9 Toggle. The PORT_DATA_TGL.PX9 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
8 (R/W)	PX8	Port x Bit 8 Toggle. The PORT_DATA_TGL.PX8 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
7 (R/W)	PX7	Port x Bit 7 Toggle. The PORT_DATA_TGL.PX7 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit

Table 13-15: PORT_DATA_TGL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	PX6	Port x Bit 6 Toggle. The PORT_DATA_TGL.PX6 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
5 (R/W)	PX5	Port x Bit 5 Toggle. The PORT_DATA_TGL.PX5 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
4 (R/W)	PX4	Port x Bit 4 Toggle. The PORT_DATA_TGL.PX4 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
3 (R/W)	PX3	Port x Bit 3 Toggle. The PORT_DATA_TGL.PX3 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
2 (R/W)	PX2	Port x Bit 2 Toggle. The PORT_DATA_TGL.PX2 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
1 (R/W)	PX1	Port x Bit 1 Toggle. The PORT_DATA_TGL.PX1 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit
0 (R/W)	PX0	Port x Bit 0 Toggle. The PORT_DATA_TGL.PX0 bit toggles the output GPIO bit/pin state.	
		0	No Effect
		1	Toggle Bit

Port x GPIO Direction Register

The PORT_DIR, PORT_DIR_SET, and PORT_DIR_CLR registers select output or input mode for GPIO pins and enable output drivers. Use the PORT_INEN, PORT_INEN_SET, and PORT_INEN_CLR registers to enable or disable input drivers.

Writes to the PORT_DIR register affect the state of all pins of the port. To select a direction for specific pins without impacting other pins of the port, use the PORT_DIR_SET and PORT_DIR_CLR registers.

Setting a bit in the PORT_DIR register enables output mode on the corresponding a GPIO pin. Clearing a bit in the PORT_DIR register disables output mode on the corresponding GPIO pin.

Input Mode - The default mode of every GPIO pin after reset is the input mode, but the input drivers are not enabled. To enable GPIO input drivers, set the corresponding bits in the PORT_INEN register. When enabled, a read from the PORT_DATA register returns the logical state of the input pin. The input signal does not overwrite the state of the bit used for the output case. That state can only be altered by software. If the input driver is enabled, a write to the PORT_DATA register can alter the state of the bit, but the change cannot be read back.

Output Mode - Any GPIO pin can be configured for output mode. The GPIO output drivers are enabled by setting the corresponding bits in the PORT_DIR, PORT_DIR_SET, or PORT_DIR_CLR registers. By using the PORT_DIR_SET and PORT_DIR_CLR registers, the direction of the signal flow of individual GPIO pins can be altered by separate software threads without mutually impacting other GPIOs on the same port. Both registers return the same value when read. Because the state of the GPIO output can already be controlled before the output driver is enabled, it is recommended to first set or clear the bit (using the PORT_DATA, PORT_DATA_SET, or PORT_DATA_CLR registers) to avoid any volatile levels on the output.

Open-Drain Mode - Every GPIO can also be used in open-drain mode. To accomplish this, first, clear the respective bit in the PORT_DATA or PORT_DATA_CLR register. Then, set the one bit in the PORT_INEN register. Reads from the PORT_DATA register then return the status from the pin and do not return the state of the internal flip-flop. By toggling the output driver through the PORT_DIR_SET and PORT_DIR_CLR register pair, the output signal can be pulled low or three-stated as required. Note that the polarity of the driven signal can be inverted when the internal flip-flop is set instead. When a GPIO port is used in open-drain mode, take care to not exceed the V_{IH} operating condition associated with the respective pin.

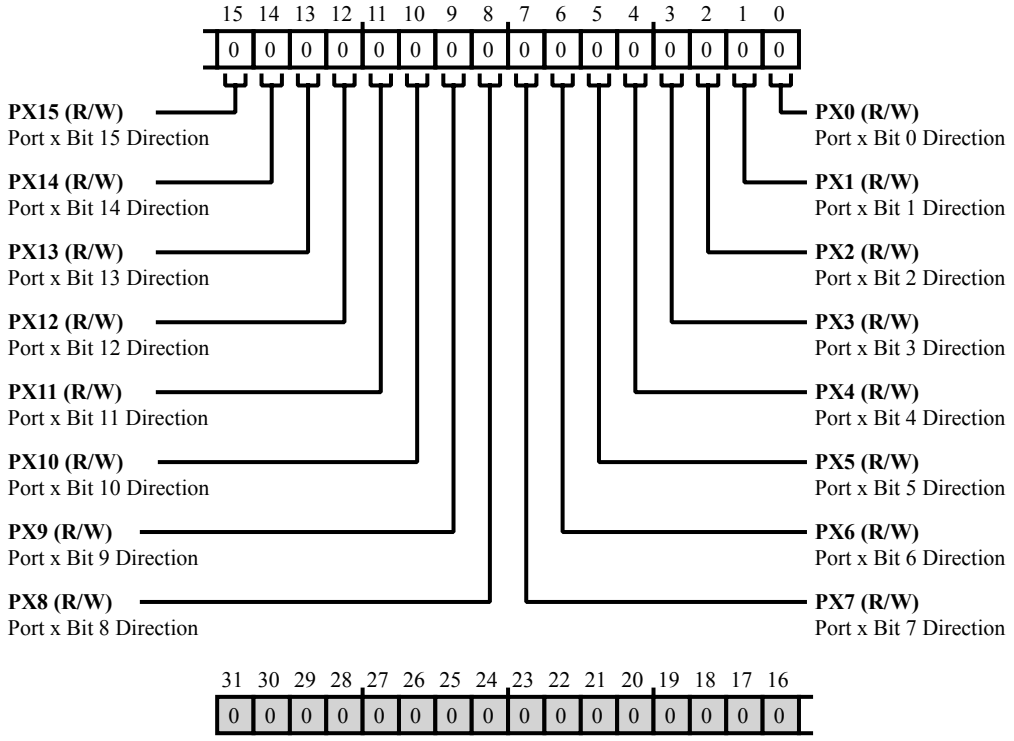


Figure 13-11: PORT_DIR Register Diagram

Table 13-16: PORT_DIR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
14 (R/W)	PX14	Port x Bit 14 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
13 (R/W)	PX13	Port x Bit 13 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
12 (R/W)	PX12	Port x Bit 12 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.

Table 13-16: PORT_DIR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	PX11	Port x Bit 11 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
10 (R/W)	PX10	Port x Bit 10 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
9 (R/W)	PX9	Port x Bit 9 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
8 (R/W)	PX8	Port x Bit 8 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
7 (R/W)	PX7	Port x Bit 7 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
6 (R/W)	PX6	Port x Bit 6 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
5 (R/W)	PX5	Port x Bit 5 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
4 (R/W)	PX4	Port x Bit 4 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
3 (R/W)	PX3	Port x Bit 3 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
2 (R/W)	PX2	Port x Bit 2 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.

Table 13-16: PORT_DIR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	PX1	Port x Bit 1 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.
0 (R/W)	PX0	Port x Bit 0 Direction.	
		0	Input mode. The output driver is disabled.
		1	Output mode. The output driver is enabled.

Port x GPIO Direction Clear Register

The PORT_DIR_CLR register disables output mode and disables the output drivers for GPIO pins. For more information, see the PORT_DIR register description.

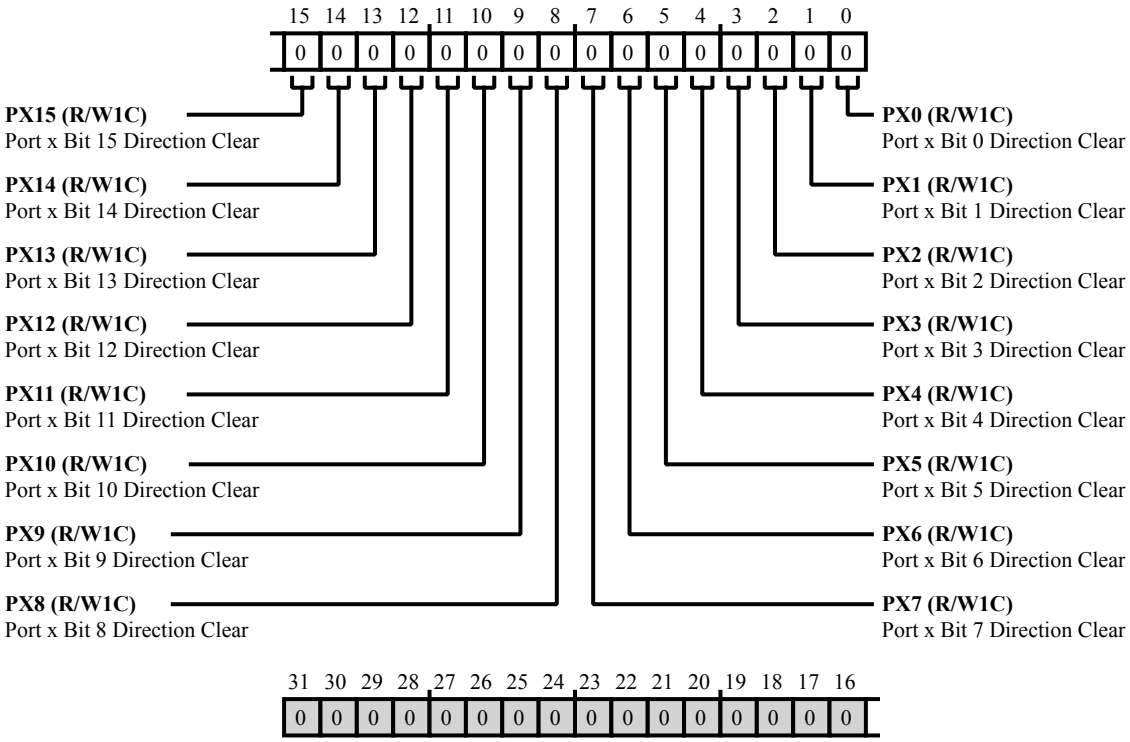


Figure 13-12: PORT_DIR_CLR Register Diagram

Table 13-17: PORT_DIR_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Direction Clear. The PORT_DIR_CLR.PX15 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
14 (R/W1C)	PX14	Port x Bit 14 Direction Clear. The PORT_DIR_CLR.PX14 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver

Table 13-17: PORT_DIR_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1C)	PX13	Port x Bit 13 Direction Clear. The PORT_DIR_CLR.PX13 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
12 (R/W1C)	PX12	Port x Bit 12 Direction Clear. The PORT_DIR_CLR.PX12 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
11 (R/W1C)	PX11	Port x Bit 11 Direction Clear. The PORT_DIR_CLR.PX11 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
10 (R/W1C)	PX10	Port x Bit 10 Direction Clear. The PORT_DIR_CLR.PX10 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
9 (R/W1C)	PX9	Port x Bit 9 Direction Clear. The PORT_DIR_CLR.PX9 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
8 (R/W1C)	PX8	Port x Bit 8 Direction Clear. The PORT_DIR_CLR.PX8 bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver

Table 13-17: PORT_DIR_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	PX7	Port x Bit 7 Direction Clear. The `PORT_DIR_CLR.PX7` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
6 (R/W1C)	PX6	Port x Bit 6 Direction Clear. The `PORT_DIR_CLR.PX6` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
5 (R/W1C)	PX5	Port x Bit 5 Direction Clear. The `PORT_DIR_CLR.PX5` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
4 (R/W1C)	PX4	Port x Bit 4 Direction Clear. The `PORT_DIR_CLR.PX4` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
3 (R/W1C)	PX3	Port x Bit 3 Direction Clear. The `PORT_DIR_CLR.PX3` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
2 (R/W1C)	PX2	Port x Bit 2 Direction Clear. The `PORT_DIR_CLR.PX2` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver

Table 13-17: PORT_DIR_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	PX1	Port x Bit 1 Direction Clear. The `PORT_DIR_CLR.PX1` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver
0 (R/W1C)	PX0	Port x Bit 0 Direction Clear. The `PORT_DIR_CLR.PX0` bit disables output mode and the output drivers for port x.	
		0	No Effect
		1	Disable output mode/driver

Port x GPIO Direction Set Register

The PORT_DIR_SET register enables output mode and output drivers for GPIO pins. For more information, see the PORT_DIR register description.

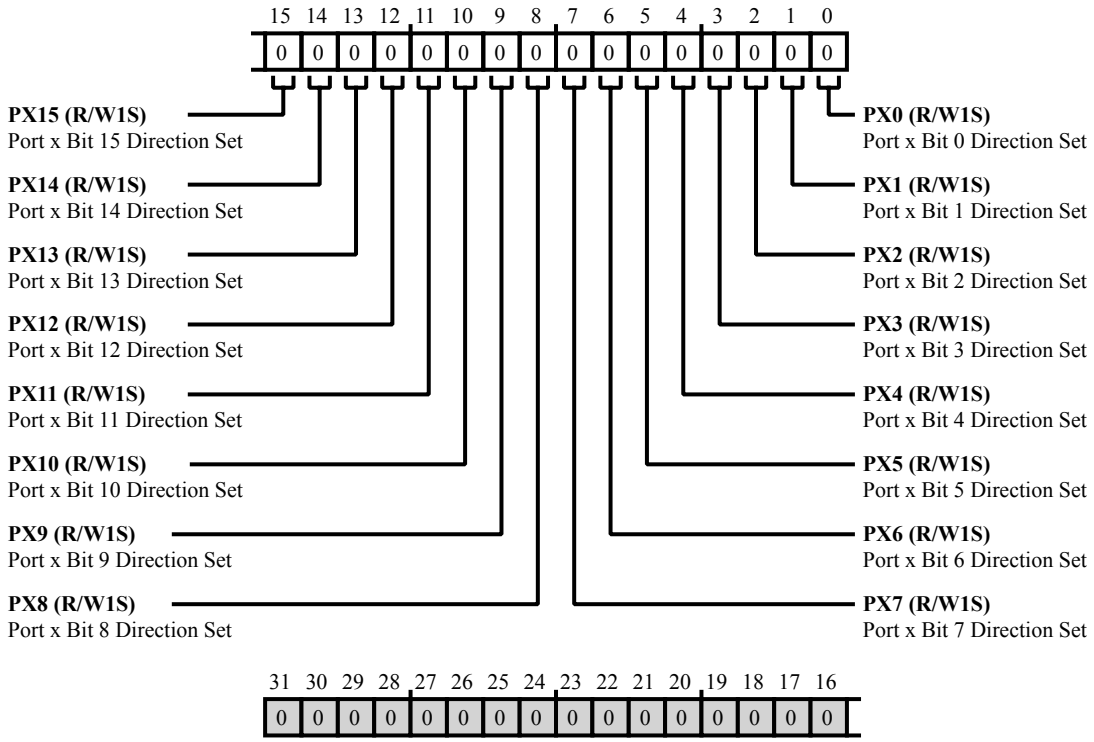


Figure 13-13: PORT_DIR_SET Register Diagram

Table 13-18: PORT_DIR_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Direction Set. The PORT_DIR_SET.PX15 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
14 (R/W1S)	PX14	Port x Bit 14 Direction Set. The PORT_DIR_SET.PX14 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver

Table 13-18: PORT_DIR_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1S)	PX13	Port x Bit 13 Direction Set. The PORT_DIR_SET.PX13 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
12 (R/W1S)	PX12	Port x Bit 12 Direction Set. The PORT_DIR_SET.PX12 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
11 (R/W1S)	PX11	Port x Bit 11 Direction Set. The PORT_DIR_SET.PX11 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
10 (R/W1S)	PX10	Port x Bit 10 Direction Set. The PORT_DIR_SET.PX10 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
9 (R/W1S)	PX9	Port x Bit 9 Direction Set. The PORT_DIR_SET.PX9 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
8 (R/W1S)	PX8	Port x Bit 8 Direction Set. The PORT_DIR_SET.PX8 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
7 (R/W1S)	PX7	Port x Bit 7 Direction Set. The PORT_DIR_SET.PX7 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver

Table 13-18: PORT_DIR_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1S)	PX6	Port x Bit 6 Direction Set. The PORT_DIR_SET.PX6 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
5 (R/W1S)	PX5	Port x Bit 5 Direction Set. The PORT_DIR_SET.PX5 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
4 (R/W1S)	PX4	Port x Bit 4 Direction Set. The PORT_DIR_SET.PX4 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
3 (R/W1S)	PX3	Port x Bit 3 Direction Set. The PORT_DIR_SET.PX3 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
2 (R/W1S)	PX2	Port x Bit 2 Direction Set. The PORT_DIR_SET.PX2 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
1 (R/W1S)	PX1	Port x Bit 1 Direction Set. The PORT_DIR_SET.PX1 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver
0 (R/W1S)	PX0	Port x Bit 0 Direction Set. The PORT_DIR_SET.PX0 bit enables the output mode/driver for port x.	
		0	No Effect
		1	Enable output mode/driver

Port x Function Enable Register

The PORT_FER register bits indicate each port bit's operating mode: general purpose I/O mode or peripheral mode. After reset, all pins default to GPIO mode. Setting a bit in the PORT_FER registers enables a peripheral module to take ownership of the pin. The function enable bits impact output control only. Regardless of the setting of the function enable bits, both GPIO and peripherals can still sense the pin input. After a function is enabled, it is up to the PORT_MUX registers as to which peripheral takes control.

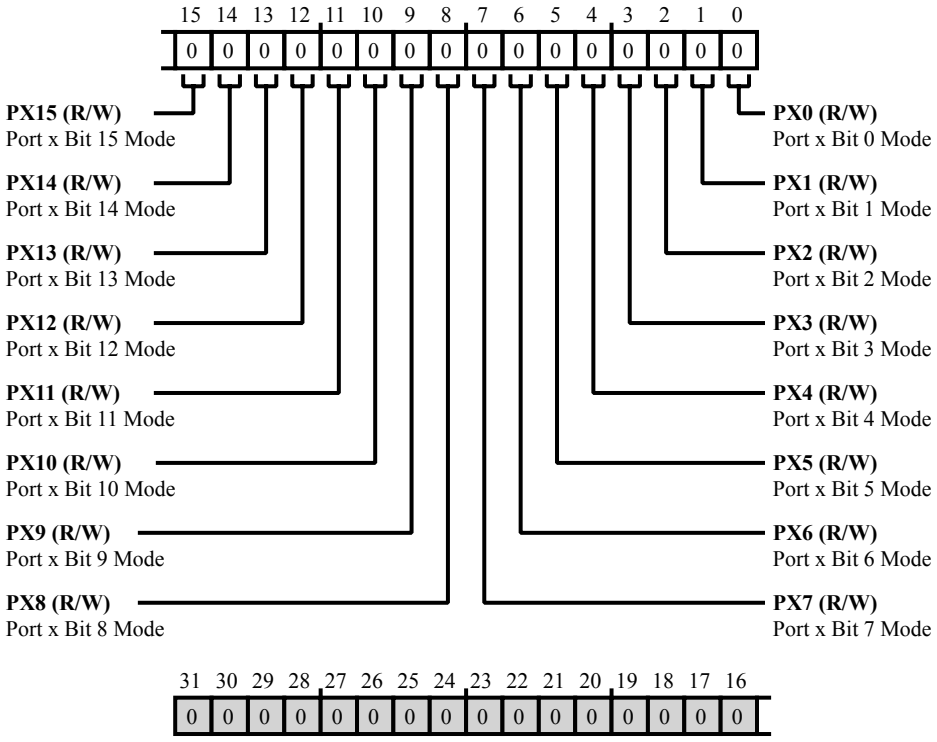


Figure 13-14: PORT_FER Register Diagram

Table 13-19: PORT_FER Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Mode. The PORT_FER.PX15 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
14 (R/W)	PX14	Port x Bit 14 Mode. The PORT_FER.PX14 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode

Table 13-19: PORT_FER Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	PX13	Port x Bit 13 Mode. The PORT_FER.PX13 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
12 (R/W)	PX12	Port x Bit 12 Mode. The PORT_FER.PX12 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
11 (R/W)	PX11	Port x Bit 11 Mode. The PORT_FER.PX11 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
10 (R/W)	PX10	Port x Bit 10 Mode. The PORT_FER.PX10 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
9 (R/W)	PX9	Port x Bit 9 Mode. The PORT_FER.PX9 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
8 (R/W)	PX8	Port x Bit 8 Mode. The PORT_FER.PX8 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
7 (R/W)	PX7	Port x Bit 7 Mode. The PORT_FER.PX7 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode

Table 13-19: PORT_FER Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	PX6	Port x Bit 6 Mode. The PORT_FER.PX6 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
5 (R/W)	PX5	Port x Bit 5 Mode. The PORT_FER.PX5 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
4 (R/W)	PX4	Port x Bit 4 Mode. The PORT_FER.PX4 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
3 (R/W)	PX3	Port x Bit 3 Mode. The PORT_FER.PX3 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
2 (R/W)	PX2	Port x Bit 2 Mode. The PORT_FER.PX2 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
1 (R/W)	PX1	Port x Bit 1 Mode. The PORT_FER.PX1 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode
0 (R/W)	PX0	Port x Bit 0 Mode. The PORT_FER.PX0 bit indicates the operating mode for port x.	
		0	GPIO Mode
		1	Peripheral Mode

Port x Function Enable Clear Register

The PORT_FER_CLR register permits enabling GPIO mode for each bit and corresponding GPIO pin. Writing 1 to a bit in PORT_FER_CLR enables GPIO mode for the corresponding pin.

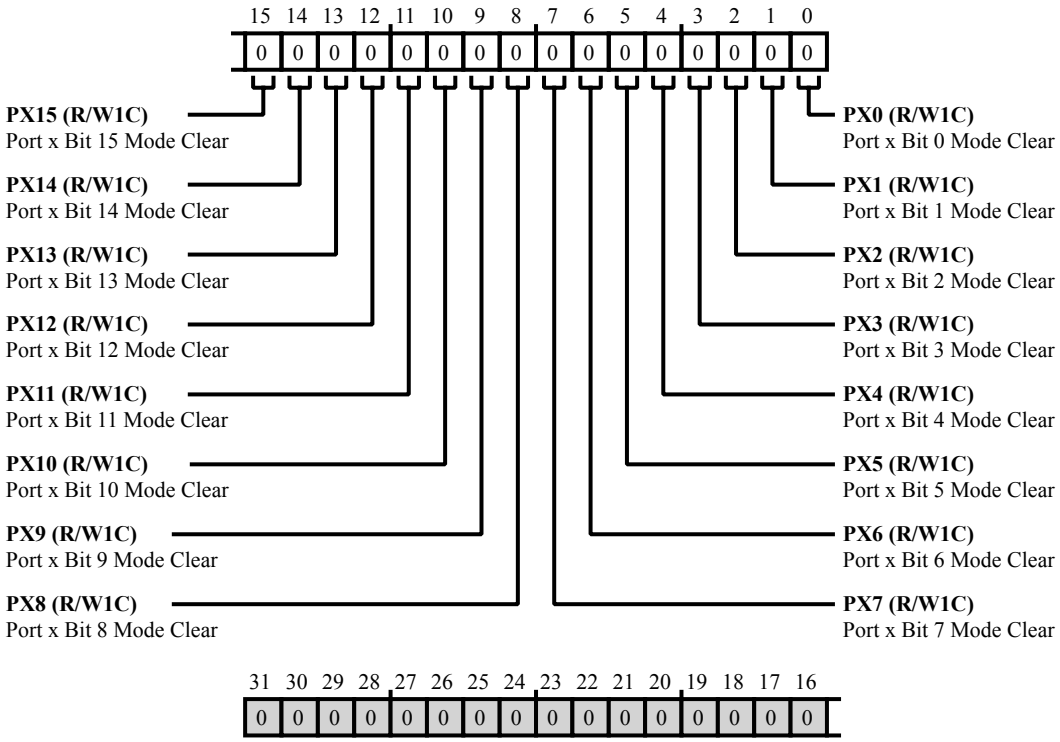


Figure 13-15: PORT_FER_CLR Register Diagram

Table 13-20: PORT_FER_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Mode Clear. The PORT_FER_CLR.PX15 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
14 (R/W1C)	PX14	Port x Bit 14 Mode Clear. The PORT_FER_CLR.PX14 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode

Table 13-20: PORT_FER_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1C)	PX13	Port x Bit 13 Mode Clear. The PORT_FER_CLR.PX13 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
12 (R/W1C)	PX12	Port x Bit 12 Mode Clear. The PORT_FER_CLR.PX12 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
11 (R/W1C)	PX11	Port x Bit 11 Mode Clear. The PORT_FER_CLR.PX11 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
10 (R/W1C)	PX10	Port x Bit 10 Mode Clear. The PORT_FER_CLR.PX10 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
9 (R/W1C)	PX9	Port x Bit 9 Mode Clear. The PORT_FER_CLR.PX9 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
8 (R/W1C)	PX8	Port x Bit 8 Mode Clear. The PORT_FER_CLR.PX8 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
7 (R/W1C)	PX7	Port x Bit 7 Mode Clear. The PORT_FER_CLR.PX7 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode

Table 13-20: PORT_FER_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1C)	PX6	Port x Bit 6 Mode Clear. The PORT_FER_CLR.PX6 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
5 (R/W1C)	PX5	Port x Bit 5 Mode Clear. The PORT_FER_CLR.PX5 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
4 (R/W1C)	PX4	Port x Bit 4 Mode Clear. The PORT_FER_CLR.PX4 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
3 (R/W1C)	PX3	Port x Bit 3 Mode Clear. The PORT_FER_CLR.PX3 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
2 (R/W1C)	PX2	Port x Bit 2 Mode Clear. The PORT_FER_CLR.PX2 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
1 (R/W1C)	PX1	Port x Bit 1 Mode Clear. The PORT_FER_CLR.PX1 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode
0 (R/W1C)	PX0	Port x Bit 0 Mode Clear. The PORT_FER_CLR.PX0 bit enables GPIO mode.	
		0	No Effect
		1	Set Bit for GPIO Mode

Port x Function Enable Set Register

The PORT_FER_SET register permits enabling peripheral mode for each bit and corresponding GPIO pin. Writing 1 to a bit in PORT_FER_SET enables peripheral mode for the corresponding pin.

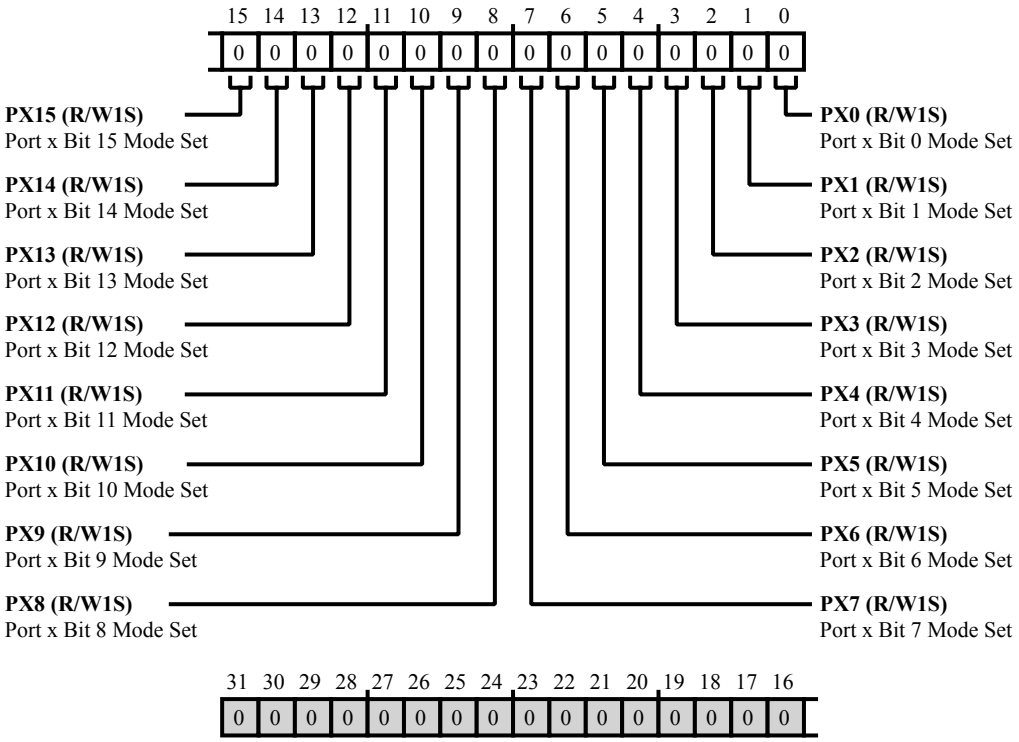


Figure 13-16: PORT_FER_SET Register Diagram

Table 13-21: PORT_FER_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Mode Set. The PORT_FER_SET.PX15 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
14 (R/W1S)	PX14	Port x Bit 14 Mode Set. The PORT_FER_SET.PX14 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode

Table 13-21: PORT_FER_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1S)	PX13	Port x Bit 13 Mode Set. The PORT_FER_SET.PX13 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
12 (R/W1S)	PX12	Port x Bit 12 Mode Set. The PORT_FER_SET.PX12 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
11 (R/W1S)	PX11	Port x Bit 11 Mode Set. The PORT_FER_SET.PX11 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
10 (R/W1S)	PX10	Port x Bit 10 Mode Set. The PORT_FER_SET.PX10 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
9 (R/W1S)	PX9	Port x Bit 9 Mode Set. The PORT_FER_SET.PX9 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
8 (R/W1S)	PX8	Port x Bit 8 Mode Set. The PORT_FER_SET.PX8 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
7 (R/W1S)	PX7	Port x Bit 7 Mode Set. The PORT_FER_SET.PX7 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode

Table 13-21: PORT_FER_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1S)	PX6	Port x Bit 6 Mode Set. The PORT_FER_SET.PX6 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
5 (R/W1S)	PX5	Port x Bit 5 Mode Set. The PORT_FER_SET.PX5 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
4 (R/W1S)	PX4	Port x Bit 4 Mode Set. The PORT_FER_SET.PX4 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
3 (R/W1S)	PX3	Port x Bit 3 Mode Set. The PORT_FER_SET.PX3 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
2 (R/W1S)	PX2	Port x Bit 2 Mode Set. The PORT_FER_SET.PX2 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
1 (R/W1S)	PX1	Port x Bit 1 Mode Set. The PORT_FER_SET.PX1 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode
0 (R/W1S)	PX0	Port x Bit 0 Mode Set. The PORT_FER_SET.PX0 bit enables peripheral mode.	
		0	No Effect
		1	Set Bit for Peripheral Mode

Port x GPIO Input Enable Register

The PORT_INEN, PORT_INEN_SET, and PORT_INEN_CLR registers enable or disable input drivers, which are required for using a GPIO pin in input mode.

Writes to the PORT_INEN register affect the input drivers for all pins of the port. To set or clear specific pin drivers without impacting other pin drivers of the port, use the PORT_INEN_SET and PORT_INEN_CLR registers.

If the input is enabled, reads from the PORT_DATA, PORT_DATA_SET, or PORT_DATA_CLR registers return the state of the pins. However, the state of the output is not overwritten by the input. It is altered by software writes only. Input and output drivers can be enabled at the same time. In this case, a read of the data register returns the true value of the data register and not the pin state.

For more information, see the PORT_DATA register description and the PORT_DIR register description.

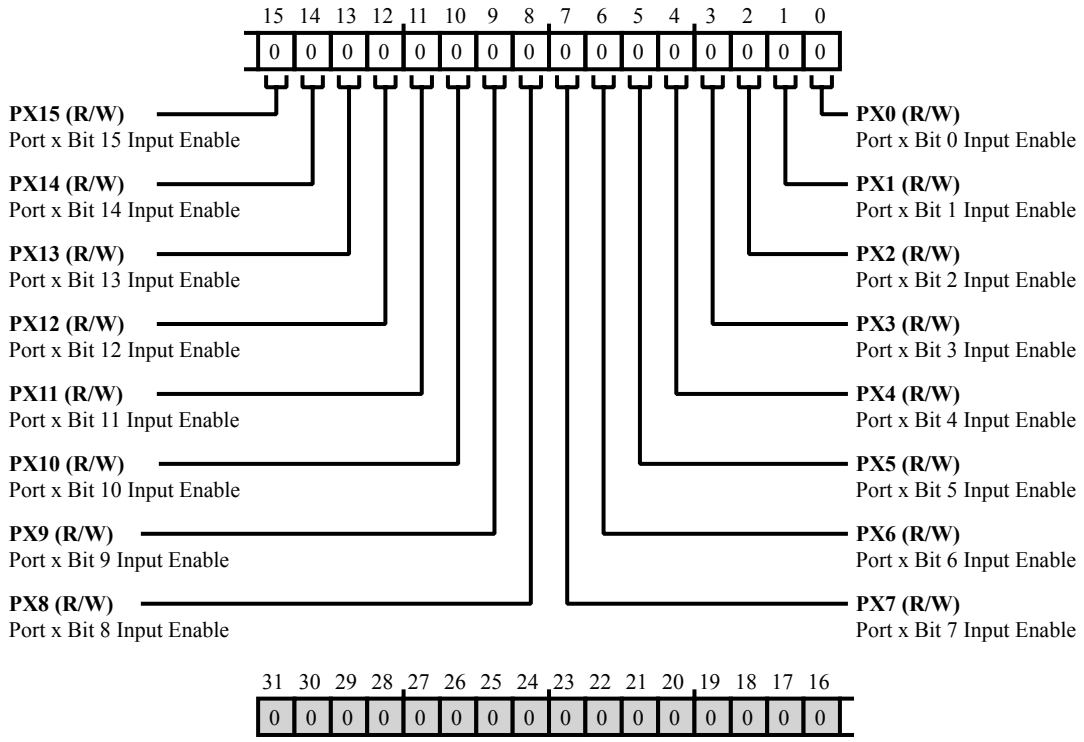


Figure 13-17: PORT_INEN Register Diagram

Table 13-22: PORT_INEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver

Table 13-22: PORT_INEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	PX14	Port x Bit 14 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
13 (R/W)	PX13	Port x Bit 13 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
12 (R/W)	PX12	Port x Bit 12 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
11 (R/W)	PX11	Port x Bit 11 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
10 (R/W)	PX10	Port x Bit 10 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
9 (R/W)	PX9	Port x Bit 9 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
8 (R/W)	PX8	Port x Bit 8 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
7 (R/W)	PX7	Port x Bit 7 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
6 (R/W)	PX6	Port x Bit 6 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
5 (R/W)	PX5	Port x Bit 5 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver

Table 13-22: PORT_INEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	PX4	Port x Bit 4 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
3 (R/W)	PX3	Port x Bit 3 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
2 (R/W)	PX2	Port x Bit 2 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
1 (R/W)	PX1	Port x Bit 1 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver
0 (R/W)	PX0	Port x Bit 0 Input Enable.	
		0	Disable Input Driver
		1	Enable Input Driver

Port x GPIO Input Enable Clear Register

The PORT_INEN_CLR register disables the input drivers for GPIO pins. For more information, see the PORT_INEN register description.

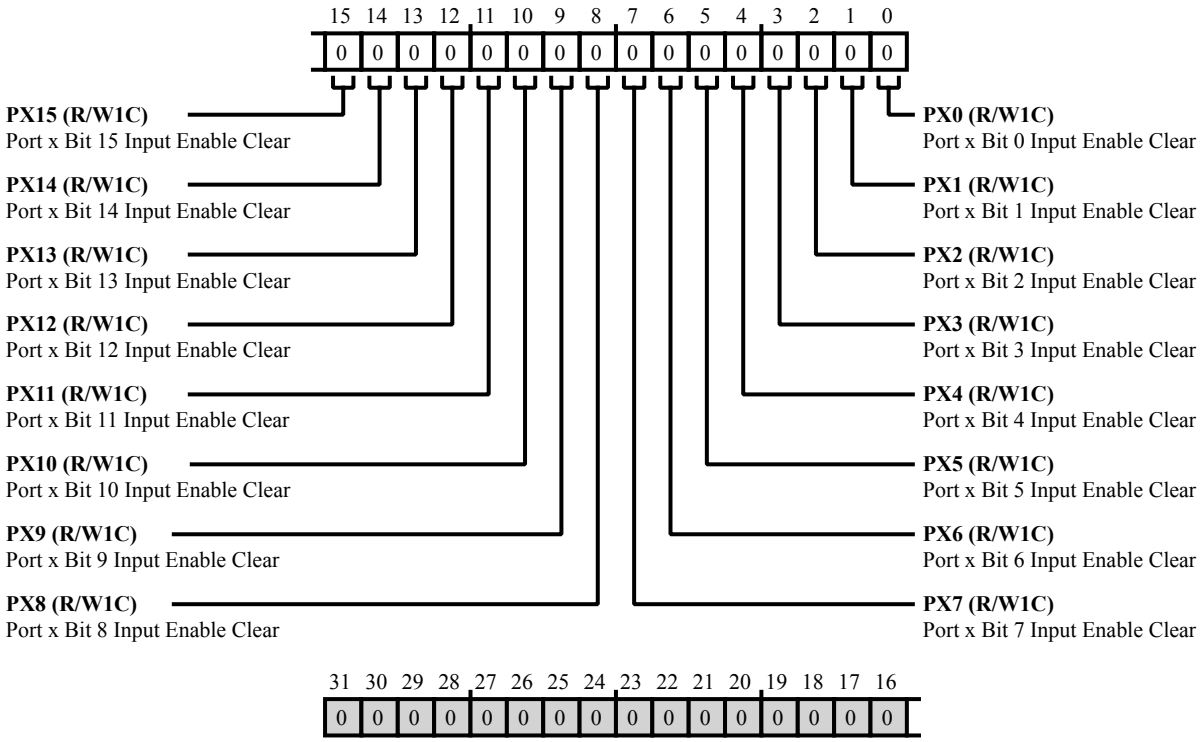


Figure 13-18: PORT_INEN_CLR Register Diagram

Table 13-23: PORT_INEN_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
14 (R/W1C)	PX14	Port x Bit 14 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
13 (R/W1C)	PX13	Port x Bit 13 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.

Table 13-23: PORT_INEN_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W1C)	PX12	Port x Bit 12 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
11 (R/W1C)	PX11	Port x Bit 11 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
10 (R/W1C)	PX10	Port x Bit 10 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
9 (R/W1C)	PX9	Port x Bit 9 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
8 (R/W1C)	PX8	Port x Bit 8 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
7 (R/W1C)	PX7	Port x Bit 7 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
6 (R/W1C)	PX6	Port x Bit 6 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
5 (R/W1C)	PX5	Port x Bit 5 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
4 (R/W1C)	PX4	Port x Bit 4 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
3 (R/W1C)	PX3	Port x Bit 3 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.

Table 13-23: PORT_INEN_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	PX2	Port x Bit 2 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
1 (R/W1C)	PX1	Port x Bit 1 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.
0 (R/W1C)	PX0	Port x Bit 0 Input Enable Clear.	
		0	No Effect
		1	Clear Bit. Set to disable the input driver.

Port x GPIO Input Enable Set Register

The PORT_INEN_SET register enables input drivers for GPIO pins. For more information, see the PORT_INEN register description.

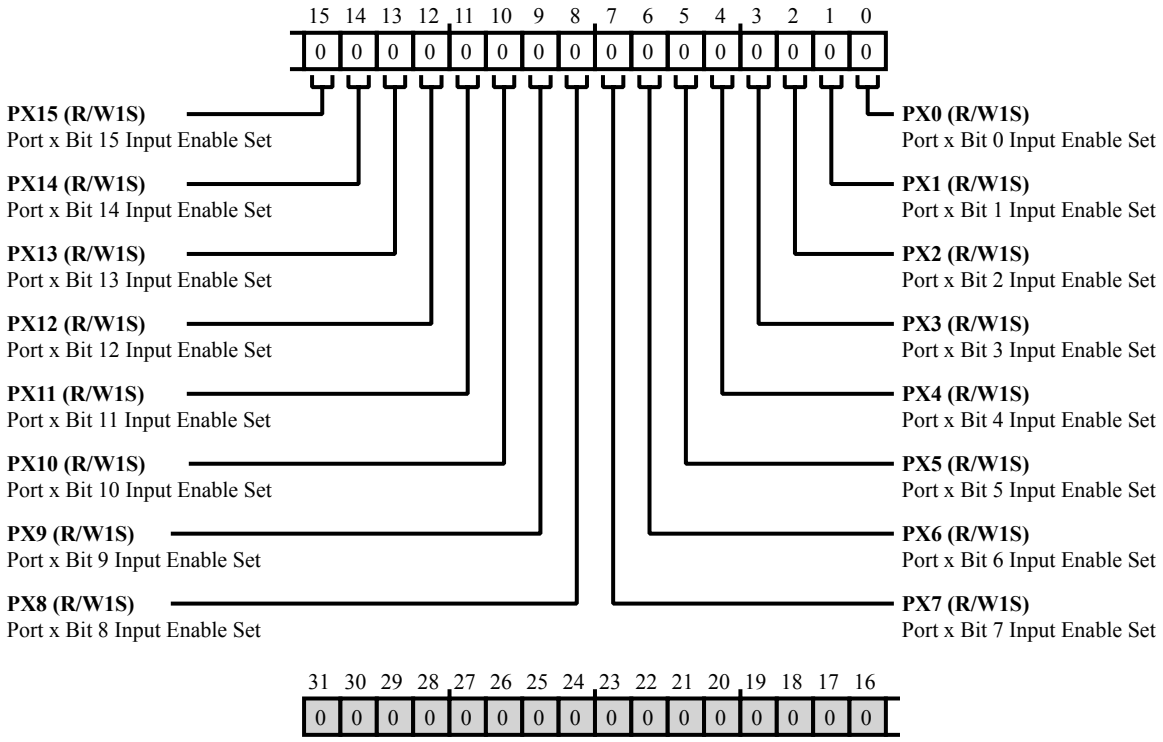


Figure 13-19: PORT_INEN_SET Register Diagram

Table 13-24: PORT_INEN_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
14 (R/W1S)	PX14	Port x Bit 14 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
13 (R/W1S)	PX13	Port x Bit 13 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.

Table 13-24: PORT_INEN_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W1S)	PX12	Port x Bit 12 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
11 (R/W1S)	PX11	Port x Bit 11 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
10 (R/W1S)	PX10	Port x Bit 10 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
9 (R/W1S)	PX9	Port x Bit 9 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
8 (R/W1S)	PX8	Port x Bit 8 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
7 (R/W1S)	PX7	Port x Bit 7 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
6 (R/W1S)	PX6	Port x Bit 6 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
5 (R/W1S)	PX5	Port x Bit 5 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
4 (R/W1S)	PX4	Port x Bit 4 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
3 (R/W1S)	PX3	Port x Bit 3 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.

Table 13-24: PORT_INEN_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1S)	PX2	Port x Bit 2 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
1 (R/W1S)	PX1	Port x Bit 1 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.
0 (R/W1S)	PX0	Port x Bit 0 Input Enable Set.	
		0	No Effect
		1	Set Bit. Set to enable the input driver.

Port x GPIO Lock Register

The PORT_LOCK register enables (unlocks) or disables (locks) write access selectively for the PORT control registers.

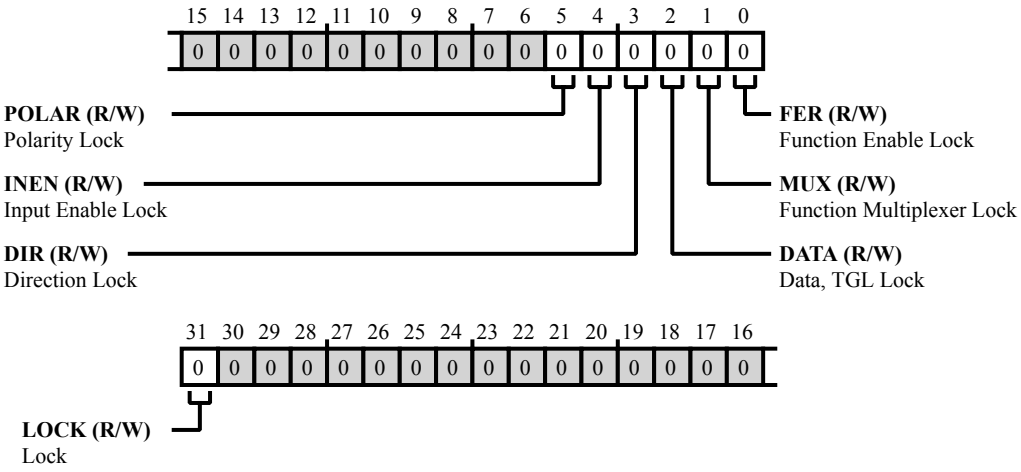


Figure 13-20: PORT_LOCK Register Diagram

Table 13-25: PORT_LOCK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock bit is set (SPU_CTL.GLCK bit =1) and the PORT_LOCK.LOCK bit is set, the PORT_LOCK register is read only (locked).	
		0	Unlock
		1	Lock
5 (R/W)	POLAR	Polarity Lock. The PORT_LOCK.POLAR disables write access to the PORT_POL, PORT_POL_SET, and PORT_POL_CLR registers.	
		0	Unlock POL
		1	Lock POL
4 (R/W)	INEN	Input Enable Lock. The PORT_LOCK.INEN disables write access to the PORT_INEN, PORT_INEN_SET, and PORT_INEN_CLR registers.	
		0	Unlock INEN
		1	Lock INEN

Table 13-25: PORT_LOCK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	DIR	Direction Lock. The PORT_LOCK.DIR disables write access to the PORT_DIR, PORT_DIR_SET, PORT_DIR_CLR registers.	
		0	Unlock DIR
		1	Lock DIR
2 (R/W)	DATA	Data, TGL Lock. The PORT_LOCK.DATA disables write access to the PORT_DATA, PORT_DATA_SET, PORT_DATA_CLR, and PORT_DATA_TGL registers.	
		0	Unlock DATA
		1	Lock DATA
1 (R/W)	MUX	Function Multiplexer Lock. The PORT_LOCK.MUX disables write accesses to the PORT_MUX register.	
		0	Unlock MUX
		1	Lock MUX
0 (R/W)	FER	Function Enable Lock. The PORT_LOCK.FER disables write access to the PORT_FER, PORT_FER_SET, and PORT_FER_CLR registers.	
		0	Unlock FER
		1	Lock FER

Port x Multiplexer Control Register

When a pin is in peripheral mode (not GPIO mode), the PORT_MUX register controls which peripheral takes ownership of a pin. Ports may have multiple, different peripheral functions. Two bits are required to describe every multiplexer on an individual pin-by-pin scheme. For example, bit 0 and bit 1 of the PORT_MUX register control the multiplexer of pin 0, bit 2 and bit 3 of PORT_MUX control the multiplexer of pin 1, and so on. The value of any PORT_MUX bit has no effect on the port pins when the associated bit in the PORT_FER register is 0 (selects GPIO mode). Even if a port has only one function, the PORT_MUX register is still present. For single function ports (no multiplexing is needed), leave the PORT_MUX bits at 0 (default). For all PORT_MUX bit fields: 00 = default/reset peripheral option, 01 = first alternate peripheral option, 10 = second alternate peripheral option, and 11 = third alternate peripheral option.

See the processor data sheet for details regarding the peripheral options associated with each port.

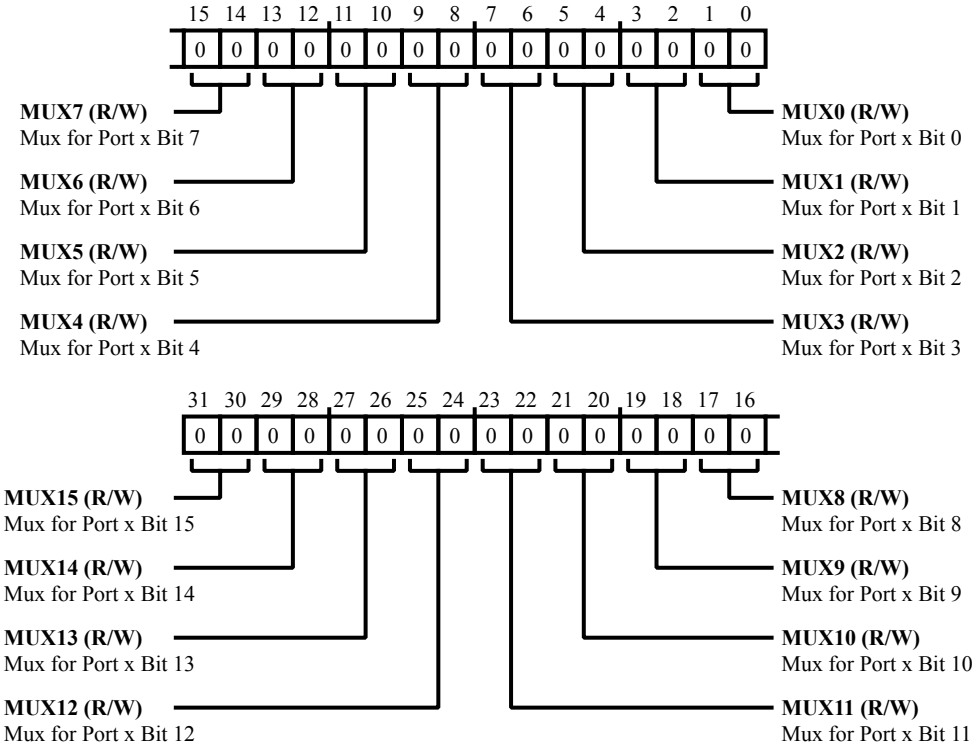


Figure 13-21: PORT_MUX Register Diagram

Table 13-26: PORT_MUX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:30 (R/W)	MUX15	Mux for Port x Bit 15. The PORT_MUX.MUX15 bit provides multiplexer control for port x bit 15.

Table 13-26: PORT_MUX Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29:28 (R/W)	MUX14	Mux for Port x Bit 14. The PORT_MUX.MUX14 bit provides multiplexer control for port x bit 14.
27:26 (R/W)	MUX13	Mux for Port x Bit 13. The PORT_MUX.MUX13 bit provides multiplexer control for port x bit 13.
25:24 (R/W)	MUX12	Mux for Port x Bit 12. The PORT_MUX.MUX12 bit provides multiplexer control for port x bit 12.
23:22 (R/W)	MUX11	Mux for Port x Bit 11. The PORT_MUX.MUX11 bit provides multiplexer control for port x bit 11.
21:20 (R/W)	MUX10	Mux for Port x Bit 10. The PORT_MUX.MUX10 bit provides multiplexer control for port x bit 10.
19:18 (R/W)	MUX9	Mux for Port x Bit 9. The PORT_MUX.MUX9 bit provides multiplexer control for port x bit 9.
17:16 (R/W)	MUX8	Mux for Port x Bit 8. The PORT_MUX.MUX8 bit provides multiplexer control for port x bit 8.
15:14 (R/W)	MUX7	Mux for Port x Bit 7. The PORT_MUX.MUX7 bit provides multiplexer control for port x bit 7.
13:12 (R/W)	MUX6	Mux for Port x Bit 6. The PORT_MUX.MUX6 bit provides multiplexer control for port x bit 6.
11:10 (R/W)	MUX5	Mux for Port x Bit 5. The PORT_MUX.MUX5 bit provides multiplexer control for port x bit 5.
9:8 (R/W)	MUX4	Mux for Port x Bit 4. The PORT_MUX.MUX4 bit provides multiplexer control for port x bit 4.
7:6 (R/W)	MUX3	Mux for Port x Bit 3. The PORT_MUX.MUX3 bit provides multiplexer control for port x bit 3.
5:4 (R/W)	MUX2	Mux for Port x Bit 2. The PORT_MUX.MUX2 bit provides multiplexer control for port x bit 2.
3:2 (R/W)	MUX1	Mux for Port x Bit 1. The PORT_MUX.MUX1 bit provides multiplexer control for port x bit 1.
1:0 (R/W)	MUX0	Mux for Port x Bit 0. The PORT_MUX.MUX0 bit provides multiplexer control for port x bit 0.

Port x GPIO Polarity Invert Register

The PORT_POL, PORT_POL_SET, and PORT_POL_CLR registers enable or disable inverting polarity of GPIO signals. To invert polarity of peripheral signals, use the inversion selection programming in the signal's corresponding module.

Writes to the PORT_POL register affect the polarity inversion selection of all pins of the port. To enable or disable polarity inversion for specific pins without impacting other pins of the port, use the PORT_POL_SET and PORT_POL_CLR registers.

Setting a bit in the PORT_POL register enables polarity inversion on the corresponding inversion GPIO pin, making the pin active-low or falling-edge sensitive. Clearing a bit in the PORT_POL register disables polarity (default state) on the corresponding GPIO pin, making it active-high or rising-edge sensitive.

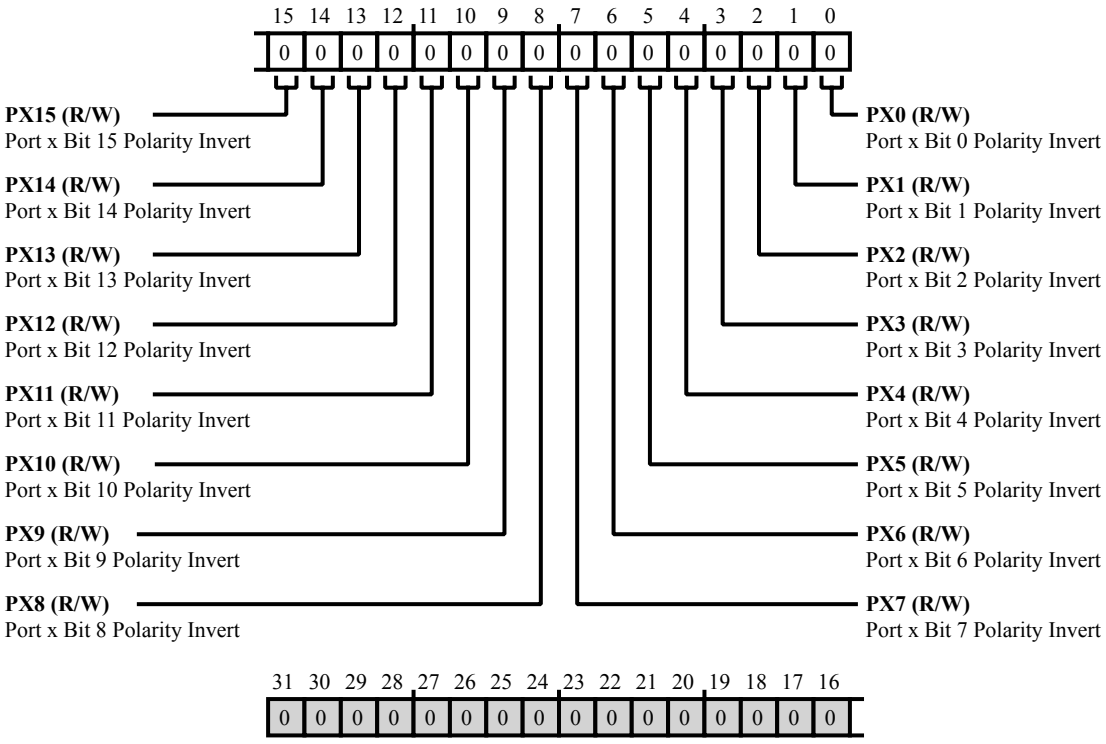


Figure 13-22: PORT_POL Register Diagram

Table 13-27: PORT_POL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PX15	Port x Bit 15 Polarity Invert. The PORT_POL.PX15 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.

Table 13-27: PORT_POL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	PX14	Port x Bit 14 Polarity Invert. The PORT_POL.PX14 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
13 (R/W)	PX13	Port x Bit 13 Polarity Invert. The PORT_POL.PX13 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
12 (R/W)	PX12	Port x Bit 12 Polarity Invert. The PORT_POL.PX12 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
11 (R/W)	PX11	Port x Bit 11 Polarity Invert. The PORT_POL.PX11 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
10 (R/W)	PX10	Port x Bit 10 Polarity Invert. The PORT_POL.PX10 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
9 (R/W)	PX9	Port x Bit 9 Polarity Invert. The PORT_POL.PX9 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
8 (R/W)	PX8	Port x Bit 8 Polarity Invert. The PORT_POL.PX8 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.

Table 13-27: PORT_POL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	PX7	Port x Bit 7 Polarity Invert. The PORT_POL.PX7 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
6 (R/W)	PX6	Port x Bit 6 Polarity Invert. The PORT_POL.PX6 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
5 (R/W)	PX5	Port x Bit 5 Polarity Invert. The PORT_POL.PX5 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
4 (R/W)	PX4	Port x Bit 4 Polarity Invert. The PORT_POL.PX4 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
3 (R/W)	PX3	Port x Bit 3 Polarity Invert. The PORT_POL.PX3 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
2 (R/W)	PX2	Port x Bit 2 Polarity Invert. The PORT_POL.PX2 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.
1 (R/W)	PX1	Port x Bit 1 Polarity Invert. The PORT_POL.PX1 bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.

Table 13-27: PORT_POL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	PX0	Port x Bit 0 Polarity Invert. The `PORT_POL.PX0` bit enables polarity inversion.	
		0	No Invert. GPIO is active high or rising edge sensitive.
		1	Invert. GPIO is active low or falling edge sensitive.

Port x GPIO Polarity Invert Clear Register

The PORT_POL_CLR register disables polarity inversion for GPIO pins. For more information, see the PORT_POL register description.

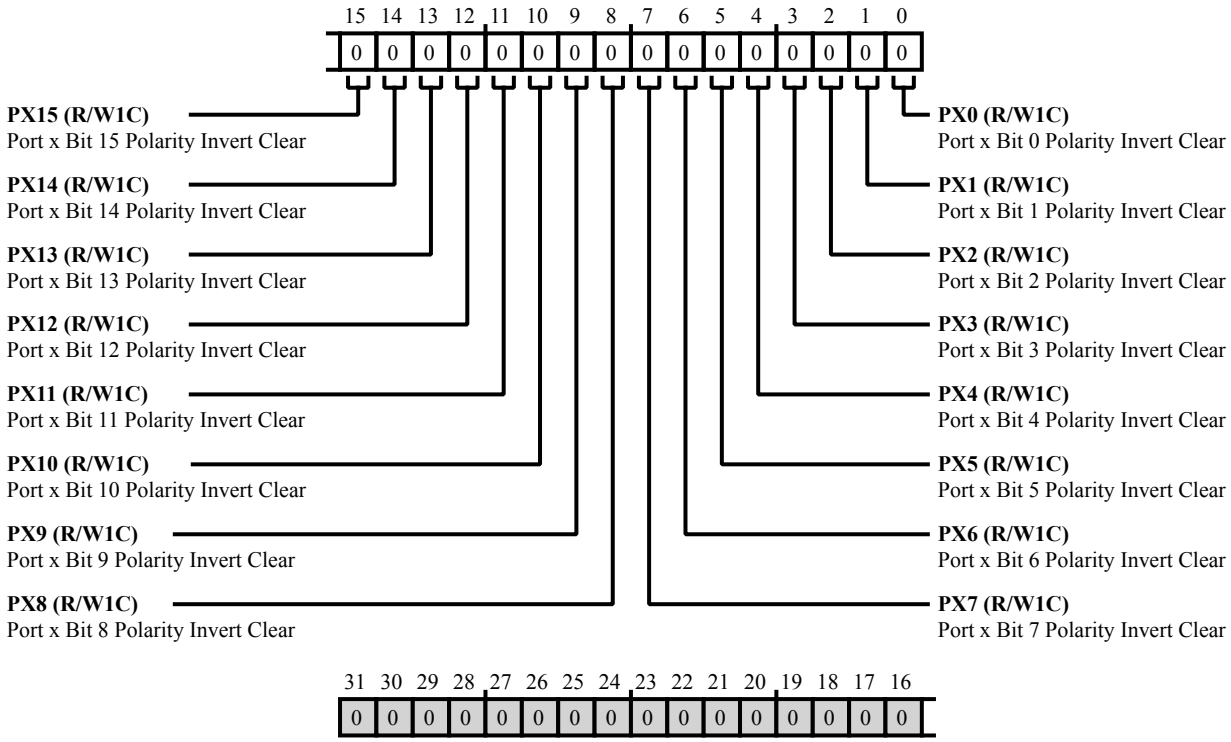


Figure 13-23: PORT_POL_CLR Register Diagram

Table 13-28: PORT_POL_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	PX15	Port x Bit 15 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
14 (R/W1C)	PX14	Port x Bit 14 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
13 (R/W1C)	PX13	Port x Bit 13 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.

Table 13-28: PORT_POL_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W1C)	PX12	Port x Bit 12 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
11 (R/W1C)	PX11	Port x Bit 11 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
10 (R/W1C)	PX10	Port x Bit 10 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
9 (R/W1C)	PX9	Port x Bit 9 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
8 (R/W1C)	PX8	Port x Bit 8 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
7 (R/W1C)	PX7	Port x Bit 7 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
6 (R/W1C)	PX6	Port x Bit 6 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
5 (R/W1C)	PX5	Port x Bit 5 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
4 (R/W1C)	PX4	Port x Bit 4 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
3 (R/W1C)	PX3	Port x Bit 3 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.

Table 13-28: PORT_POL_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	PX2	Port x Bit 2 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
1 (R/W1C)	PX1	Port x Bit 1 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.
0 (R/W1C)	PX0	Port x Bit 0 Polarity Invert Clear.	
		0	No Effect
		1	Clear Bit. Set to disable GPIO pin polarity invert.

Port x GPIO Polarity Invert Set Register

The PORT_POL_SET register enables polarity inversion for GPIO pins. For more information, see the PORT_POL register description.

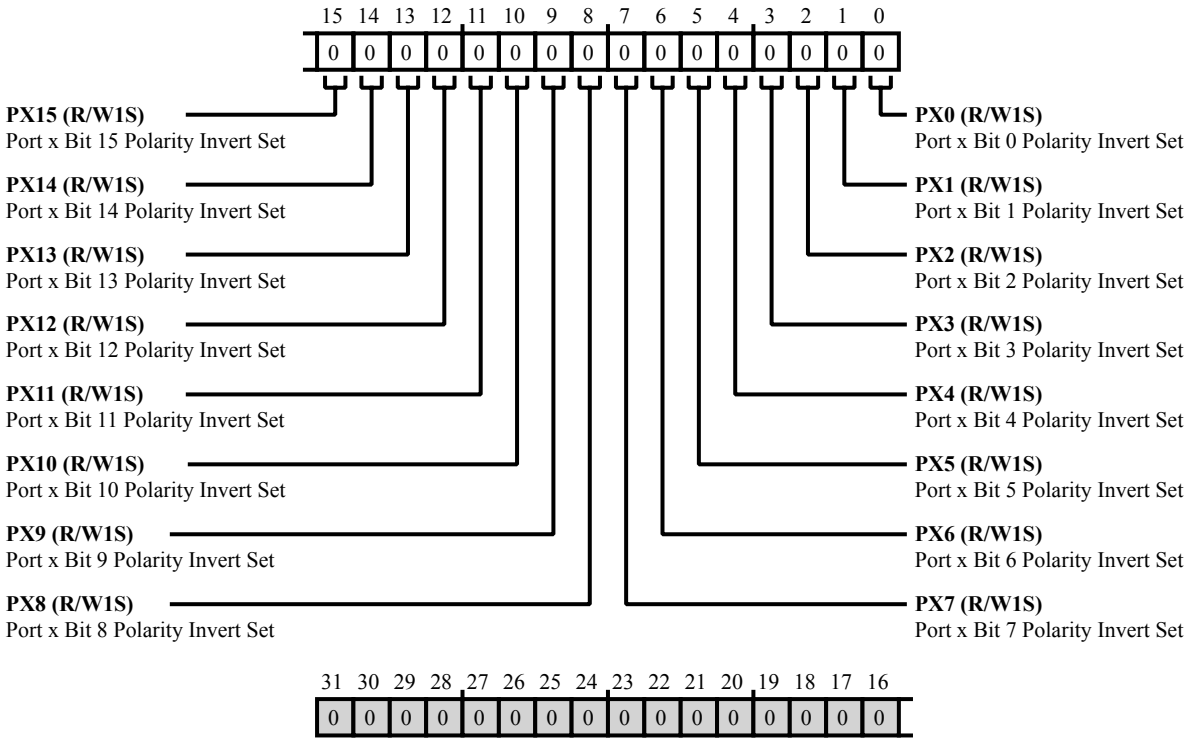


Figure 13-24: PORT_POL_SET Register Diagram

Table 13-29: PORT_POL_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1S)	PX15	Port x Bit 15 Polarity Invert Set. The PORT_POL_SET.PX15 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
14 (R/W1S)	PX14	Port x Bit 14 Polarity Invert Set. The PORT_POL_SET.PX14 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.

Table 13-29: PORT_POL_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W1S)	PX13	Port x Bit 13 Polarity Invert Set. The PORT_POL_SET.PX13 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
12 (R/W1S)	PX12	Port x Bit 12 Polarity Invert Set. The PORT_POL_SET.PX12 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
11 (R/W1S)	PX11	Port x Bit 11 Polarity Invert Set. The PORT_POL_SET.PX11 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
10 (R/W1S)	PX10	Port x Bit 10 Polarity Invert Set. The PORT_POL_SET.PX10 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
9 (R/W1S)	PX9	Port x Bit 9 Polarity Invert Set. The PORT_POL_SET.PX9 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
8 (R/W1S)	PX8	Port x Bit 8 Polarity Invert Set. The PORT_POL_SET.PX8 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
7 (R/W1S)	PX7	Port x Bit 7 Polarity Invert Set. The PORT_POL_SET.PX7 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.

Table 13-29: PORT_POL_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1S)	PX6	Port x Bit 6 Polarity Invert Set. The PORT_POL_SET.PX6 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
5 (R/W1S)	PX5	Port x Bit 5 Polarity Invert Set. The PORT_POL_SET.PX5 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
4 (R/W1S)	PX4	Port x Bit 4 Polarity Invert Set. The PORT_POL_SET.PX4 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
3 (R/W1S)	PX3	Port x Bit 3 Polarity Invert Set. The PORT_POL_SET.PX3 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
2 (R/W1S)	PX2	Port x Bit 2 Polarity Invert Set. The PORT_POL_SET.PX2 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
1 (R/W1S)	PX1	Port x Bit 1 Polarity Invert Set. The PORT_POL_SET.PX1 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.
0 (R/W1S)	PX0	Port x Bit 0 Polarity Invert Set. The PORT_POL_SET.PX0 bit enables pin polarity inversion.	
		0	No Effect
		1	Set Bit. Set to enable GPIO pin polarity invert.

Port x GPIO Trigger Toggle Register

The PORT_TRIG_TGL register permits toggling the state of output GPIO pins in response to a trigger from the TRU for the corresponding port. Setting bits in the PORT_TRIG_TGL register enables triggers to toggle the state of those specific pins without impacting other pins of the port.

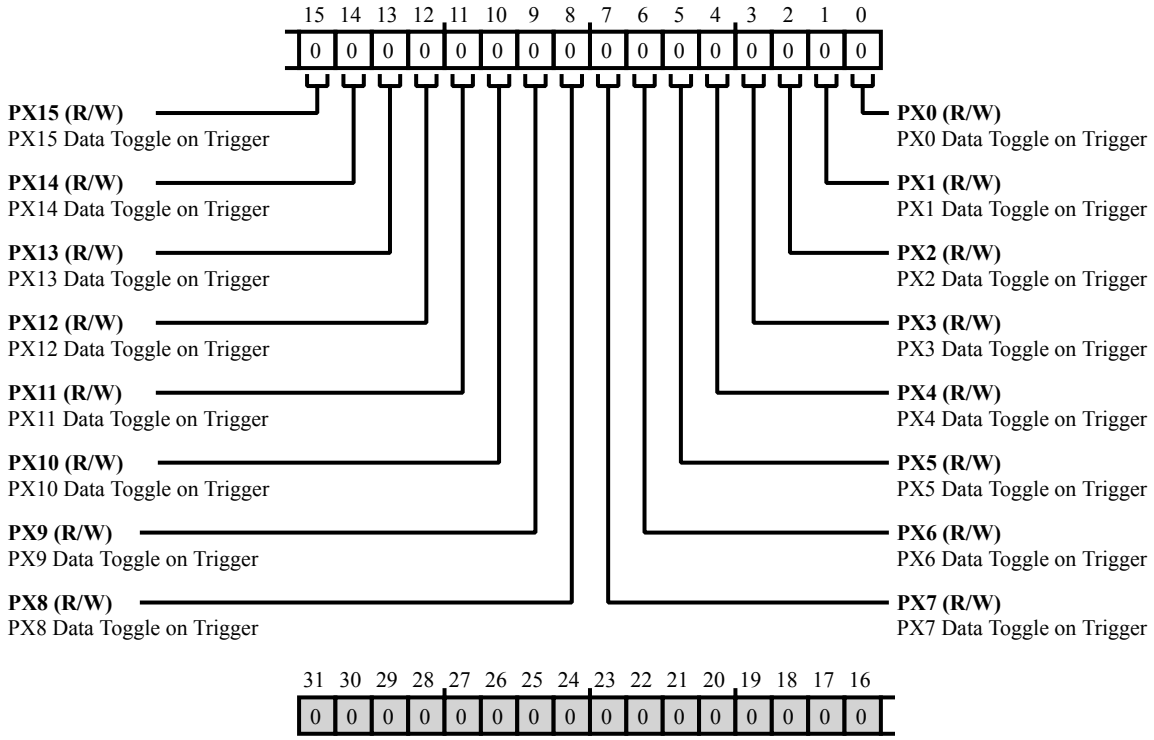


Figure 13-25: PORT_TRIG_TGL Register Diagram

Table 13-30: PORT_TRIG_TGL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	PX15	PX15 Data Toggle on Trigger. The PORT_TRIG_TGL.PX15 bit enables triggers to toggle the state of the pin.
14 (R/W)	PX14	PX14 Data Toggle on Trigger. The PORT_TRIG_TGL.PX14 bit enables triggers to toggle the state of the pin.
13 (R/W)	PX13	PX13 Data Toggle on Trigger. The PORT_TRIG_TGL.PX13 bit enables triggers to toggle the state of the pin.
12 (R/W)	PX12	PX12 Data Toggle on Trigger. The PORT_TRIG_TGL.PX12 bit enables triggers to toggle the state of the pin.
11 (R/W)	PX11	PX11 Data Toggle on Trigger. The PORT_TRIG_TGL.PX11 bit enables triggers to toggle the state of the pin.

Table 13-30: PORT_TRIG_TGL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (R/W)	PX10	PX10 Data Toggle on Trigger. The PORT_TRIG_TGL.PX10 bit enables triggers to toggle the state of the pin.
9 (R/W)	PX9	PX9 Data Toggle on Trigger. The PORT_TRIG_TGL.PX9 bit enables triggers to toggle the state of the pin.
8 (R/W)	PX8	PX8 Data Toggle on Trigger. The PORT_TRIG_TGL.PX8 bit enables triggers to toggle the state of the pin.
7 (R/W)	PX7	PX7 Data Toggle on Trigger. The PORT_TRIG_TGL.PX7 bit enables triggers to toggle the state of the pin.
6 (R/W)	PX6	PX6 Data Toggle on Trigger. The PORT_TRIG_TGL.PX6 bit enables triggers to toggle the state of the pin.
5 (R/W)	PX5	PX5 Data Toggle on Trigger. The PORT_TRIG_TGL.PX5 bit enables triggers to toggle the state of the pin.
4 (R/W)	PX4	PX4 Data Toggle on Trigger. The PORT_TRIG_TGL.PX4 bit enables triggers to toggle the state of the pin.
3 (R/W)	PX3	PX3 Data Toggle on Trigger. The PORT_TRIG_TGL.PX3 bit enables triggers to toggle the state of the pin.
2 (R/W)	PX2	PX2 Data Toggle on Trigger. The PORT_TRIG_TGL.PX2 bit enables triggers to toggle the state of the pin.
1 (R/W)	PX1	PX1 Data Toggle on Trigger. The PORT_TRIG_TGL.PX1 bit enables triggers to toggle the state of the pin.
0 (R/W)	PX0	PX0 Data Toggle on Trigger. The PORT_TRIG_TGL.PX0 bit enables triggers to toggle the state of the pin.

ADSP-SC59x PINT Register Descriptions

The Pin Interrupt module (PINT) contains the following registers.

Table 13-31: ADSP-SC59x PINT Register List

Name	Description
PINT_ASSIGN	PINT Assign Register
PINT_EDGE_CLR	PINT Edge Clear Register
PINT_EDGE_SET	PINT Edge Set Register
PINT_INV_CLR	PINT Invert Clear Register

Table 13-31: ADSP-SC59x PINT Register List (Continued)

Name	Description
PINT_INV_SET	PINT Invert Set Register
PINT_LATCH	PINT Latch Register
PINT_MSK_CLR	PINT Mask Clear Register
PINT_MSK_SET	PINT Mask Set Register
PINT_PINSTATE	PINT Pin State Register
PINT_REQ	PINT Request Register

PINT Assign Register

The PINT_ASSIGN register controls the pin-to-interrupt request assignment in a byte-wide manner. This register consists of four control bytes that each function as a multiplexer control.

The PINT ports are subdivided into 8-bit half ports, resulting in lower and upper half 8-bit units. Using the multiplexers controlled by the PINT_ASSIGN register, the lower half units of eight pins can be forwarded to either byte 0 or byte 2 of either associated PINT block. The upper half units can be forwarded to either byte 1 or byte 3 of the PINT block, without further restrictions.

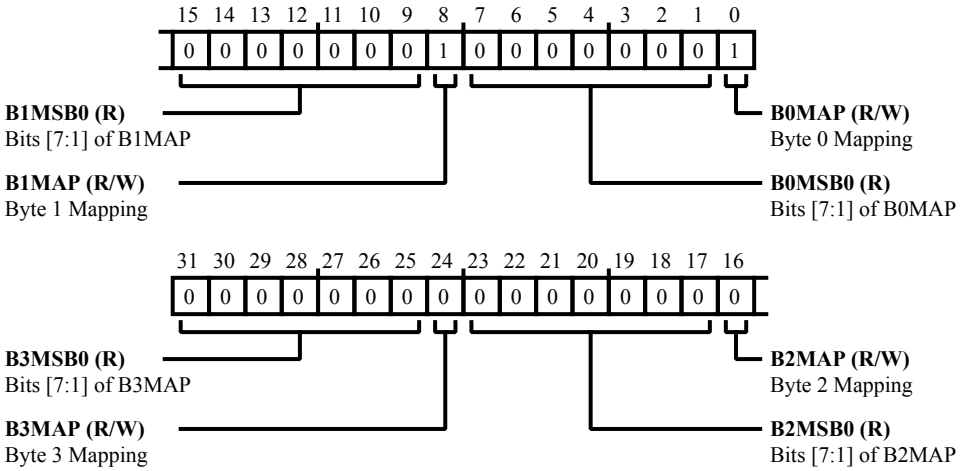


Figure 13-26: PINT_ASSIGN Register Diagram

Table 13-32: PINT_ASSIGN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:25 (R/NW)	B3MSB0	Bits [7:1] of B3MAP.	
24 (R/W)	B3MAP	Byte 3 Mapping.	
		0	B3MAP_PAH. Byte 3 = PA.H
		1	B3MAP_PBH. Byte 3 = PB.H
23:17 (R/NW)	B2MSB0	Bits [7:1] of B2MAP.	
16 (R/W)	B2MAP	Byte 2 Mapping.	
		0	B2MAP_PAL. Byte 2 = PA.L
		1	B2MAP_PBL. Byte 2 = PB.L
15:9 (R/NW)	B1MSB0	Bits [7:1] of B1MAP.	

Table 13-32: PINT_ASSIGN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	B1MAP	Byte 1 Mapping.	
		0	B1MAP_PAH. Byte 1 = PA.H
		1	B1MAP_PBH. Byte 1 = PB.H
7:1 (R/NW)	B0MSB0	Bits [7:1] of B0MAP.	
0 (R/W)	B0MAP	Byte 0 Mapping.	
		0	B0MAP_PAL. Byte 0 = PA.L
		1	B0MAP_PBL. Byte 0 = PB.L

PINT Edge Clear Register

The PINT_EDGE_CLR register permits selecting level-sensitive interrupts. Writing 1 to a bit in PINT_EDGE_CLR enables level sensitivity for the corresponding pin interrupt.

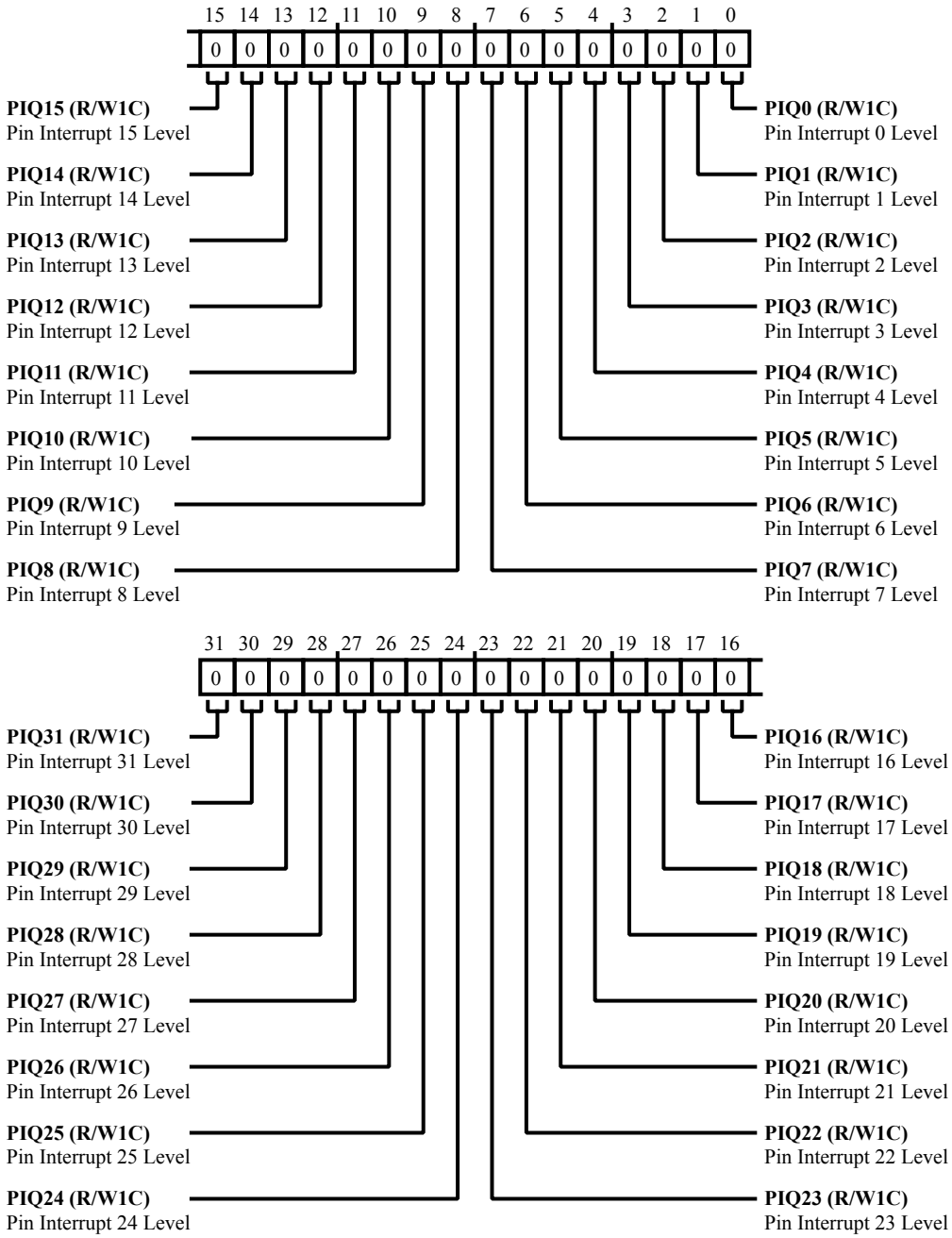


Figure 13-27: PINT_EDGE_CLR Register Diagram

Table 13-33: PINT_EDGE_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 Level. Set the PINT_EDGE_CLR.PIQ31 bit to enable level sensitivity.
30 (R/W1C)	PIQ30	Pin Interrupt 30 Level. Set the PINT_EDGE_CLR.PIQ30 bit to enable level sensitivity.
29 (R/W1C)	PIQ29	Pin Interrupt 29 Level. Set the PINT_EDGE_CLR.PIQ29 bit to enable level sensitivity.
28 (R/W1C)	PIQ28	Pin Interrupt 28 Level. Set the PINT_EDGE_CLR.PIQ28 bit to enable level sensitivity.
27 (R/W1C)	PIQ27	Pin Interrupt 27 Level. Set the PINT_EDGE_CLR.PIQ27 bit to enable level sensitivity.
26 (R/W1C)	PIQ26	Pin Interrupt 26 Level. Set the PINT_EDGE_CLR.PIQ26 bit to enable level sensitivity.
25 (R/W1C)	PIQ25	Pin Interrupt 25 Level. Set the PINT_EDGE_CLR.PIQ25 bit to enable level sensitivity.
24 (R/W1C)	PIQ24	Pin Interrupt 24 Level. Set the PINT_EDGE_CLR.PIQ24 bit to enable level sensitivity.
23 (R/W1C)	PIQ23	Pin Interrupt 23 Level. Set the PINT_EDGE_CLR.PIQ23 bit to enable level sensitivity.
22 (R/W1C)	PIQ22	Pin Interrupt 22 Level. Set the PINT_EDGE_CLR.PIQ22 bit to enable level sensitivity.
21 (R/W1C)	PIQ21	Pin Interrupt 21 Level. Set the PINT_EDGE_CLR.PIQ21 bit to enable level sensitivity.
20 (R/W1C)	PIQ20	Pin Interrupt 20 Level. Set the PINT_EDGE_CLR.PIQ20 bit to enable level sensitivity.
19 (R/W1C)	PIQ19	Pin Interrupt 19 Level. Set the PINT_EDGE_CLR.PIQ19 bit to enable level sensitivity.
18 (R/W1C)	PIQ18	Pin Interrupt 18 Level. Set the PINT_EDGE_CLR.PIQ18 bit to enable level sensitivity.
17 (R/W1C)	PIQ17	Pin Interrupt 17 Level. Set the PINT_EDGE_CLR.PIQ17 bit to enable level sensitivity.
16 (R/W1C)	PIQ16	Pin Interrupt 16 Level. Set the PINT_EDGE_CLR.PIQ16 bit to enable level sensitivity.

Table 13-33: PINT_EDGE_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1C)	PIQ15	Pin Interrupt 15 Level. Set the PINT_EDGE_CLR.PIQ15 bit to enable level sensitivity.
14 (R/W1C)	PIQ14	Pin Interrupt 14 Level. Set the PINT_EDGE_CLR.PIQ14 bit to enable level sensitivity.
13 (R/W1C)	PIQ13	Pin Interrupt 13 Level. Set the PINT_EDGE_CLR.PIQ13 bit to enable level sensitivity.
12 (R/W1C)	PIQ12	Pin Interrupt 12 Level. Set the PINT_EDGE_CLR.PIQ12 bit to enable level sensitivity.
11 (R/W1C)	PIQ11	Pin Interrupt 11 Level. Set the PINT_EDGE_CLR.PIQ11 bit to enable level sensitivity.
10 (R/W1C)	PIQ10	Pin Interrupt 10 Level. Set the PINT_EDGE_CLR.PIQ10 bit to enable level sensitivity.
9 (R/W1C)	PIQ9	Pin Interrupt 9 Level. Set the PINT_EDGE_CLR.PIQ9 bit to enable level sensitivity.
8 (R/W1C)	PIQ8	Pin Interrupt 8 Level. Set the PINT_EDGE_CLR.PIQ8 bit to enable level sensitivity.
7 (R/W1C)	PIQ7	Pin Interrupt 7 Level. Set the PINT_EDGE_CLR.PIQ7 bit to enable level sensitivity.
6 (R/W1C)	PIQ6	Pin Interrupt 6 Level. Set the PINT_EDGE_CLR.PIQ6 bit to enable level sensitivity.
5 (R/W1C)	PIQ5	Pin Interrupt 5 Level. Set the PINT_EDGE_CLR.PIQ5 bit to enable level sensitivity.
4 (R/W1C)	PIQ4	Pin Interrupt 4 Level. Set the PINT_EDGE_CLR.PIQ4 bit to enable level sensitivity.
3 (R/W1C)	PIQ3	Pin Interrupt 3 Level. Set the PINT_EDGE_CLR.PIQ3 bit to enable level sensitivity.
2 (R/W1C)	PIQ2	Pin Interrupt 2 Level. Set the PINT_EDGE_CLR.PIQ2 bit to enable level sensitivity.
1 (R/W1C)	PIQ1	Pin Interrupt 1 Level. Set the PINT_EDGE_CLR.PIQ1 bit to enable level sensitivity.
0 (R/W1C)	PIQ0	Pin Interrupt 0 Level. Set the PINT_EDGE_CLR.PIQ0 bit to enable level sensitivity.

PINT Edge Set Register

The PINT_EDGE_SET register permits selecting edge-sensitive interrupts. Writing 1 to a bit in PINT_EDGE_SET enables edge sensitivity for the corresponding pin interrupt.

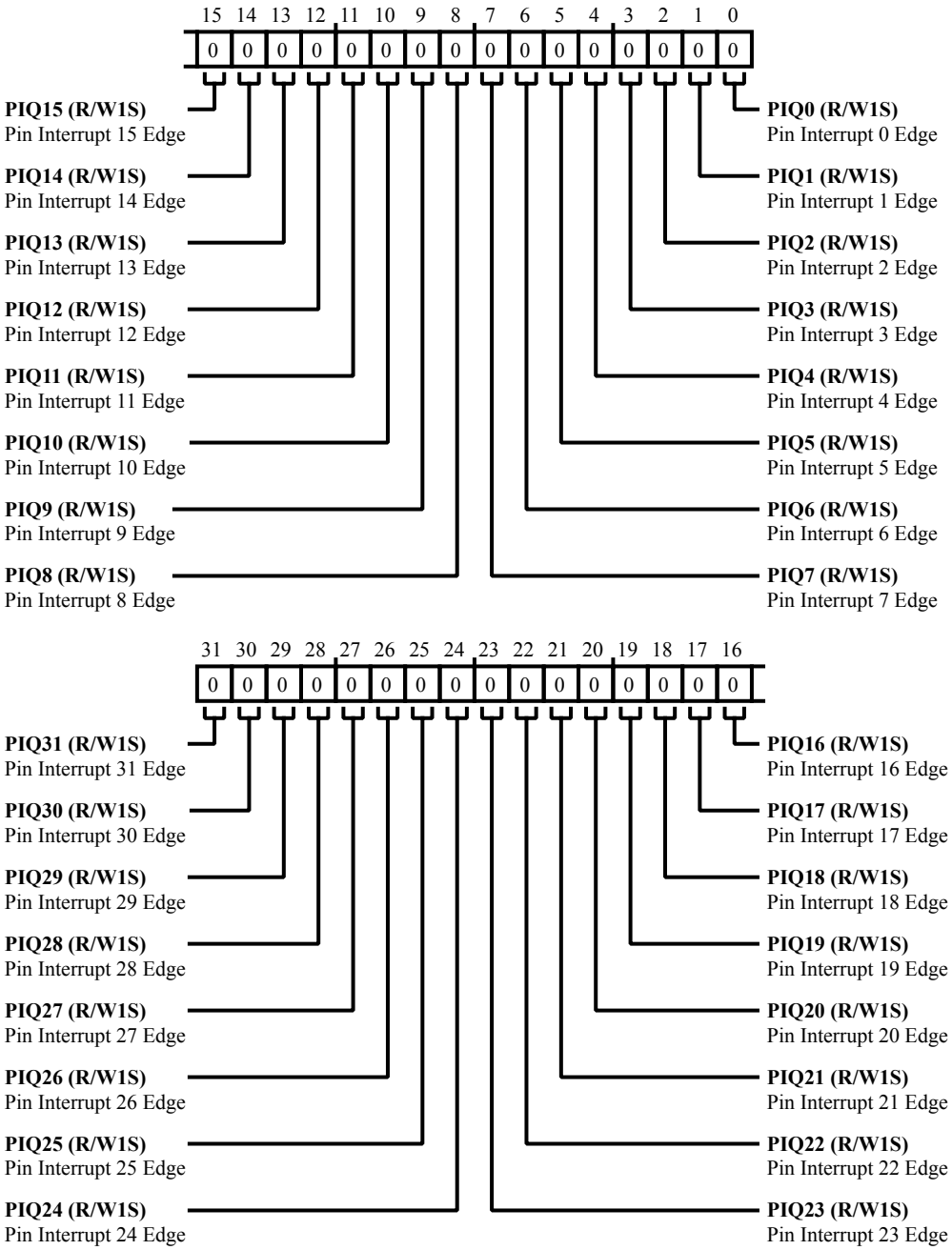


Figure 13-28: PINT_EDGE_SET Register Diagram

Table 13-34: PINT_EDGE_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	PIQ31	Pin Interrupt 31 Edge. Set the PINT_EDGE_SET.PIQ31 bit to enable edge sensitivity.
30 (R/W1S)	PIQ30	Pin Interrupt 30 Edge. Set the PINT_EDGE_SET.PIQ30 bit to enable edge sensitivity.
29 (R/W1S)	PIQ29	Pin Interrupt 29 Edge. Set the PINT_EDGE_SET.PIQ29 bit to enable edge sensitivity.
28 (R/W1S)	PIQ28	Pin Interrupt 28 Edge. Set the PINT_EDGE_SET.PIQ28 bit to enable edge sensitivity.
27 (R/W1S)	PIQ27	Pin Interrupt 27 Edge. Set the PINT_EDGE_SET.PIQ27 bit to enable edge sensitivity.
26 (R/W1S)	PIQ26	Pin Interrupt 26 Edge. Set the PINT_EDGE_SET.PIQ26 bit to enable edge sensitivity.
25 (R/W1S)	PIQ25	Pin Interrupt 25 Edge. Set the PINT_EDGE_SET.PIQ25 bit to enable edge sensitivity.
24 (R/W1S)	PIQ24	Pin Interrupt 24 Edge. Set the PINT_EDGE_SET.PIQ24 bit to enable edge sensitivity.
23 (R/W1S)	PIQ23	Pin Interrupt 23 Edge. Set the PINT_EDGE_SET.PIQ23 bit to enable edge sensitivity.
22 (R/W1S)	PIQ22	Pin Interrupt 22 Edge. Set the PINT_EDGE_SET.PIQ22 bit to enable edge sensitivity.
21 (R/W1S)	PIQ21	Pin Interrupt 21 Edge. Set the PINT_EDGE_SET.PIQ21 bit to enable edge sensitivity.
20 (R/W1S)	PIQ20	Pin Interrupt 20 Edge. Set the PINT_EDGE_SET.PIQ20 bit to enable edge sensitivity.
19 (R/W1S)	PIQ19	Pin Interrupt 19 Edge. Set the PINT_EDGE_SET.PIQ19 bit to enable edge sensitivity.
18 (R/W1S)	PIQ18	Pin Interrupt 18 Edge. Set the PINT_EDGE_SET.PIQ18 bit to enable edge sensitivity.
17 (R/W1S)	PIQ17	Pin Interrupt 17 Edge. Set the PINT_EDGE_SET.PIQ17 bit to enable edge sensitivity.
16 (R/W1S)	PIQ16	Pin Interrupt 16 Edge. Set the PINT_EDGE_SET.PIQ16 bit to enable edge sensitivity.

Table 13-34: PINT_EDGE_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1S)	PIQ15	Pin Interrupt 15 Edge. Set the PINT_EDGE_SET.PIQ15 bit to enable edge sensitivity.
14 (R/W1S)	PIQ14	Pin Interrupt 14 Edge. Set the PINT_EDGE_SET.PIQ14 bit to enable edge sensitivity.
13 (R/W1S)	PIQ13	Pin Interrupt 13 Edge. Set the PINT_EDGE_SET.PIQ13 bit to enable edge sensitivity.
12 (R/W1S)	PIQ12	Pin Interrupt 12 Edge. Set the PINT_EDGE_SET.PIQ12 bit to enable edge sensitivity.
11 (R/W1S)	PIQ11	Pin Interrupt 11 Edge. Set the PINT_EDGE_SET.PIQ11 bit to enable edge sensitivity.
10 (R/W1S)	PIQ10	Pin Interrupt 10 Edge. Set the PINT_EDGE_SET.PIQ10 bit to enable edge sensitivity.
9 (R/W1S)	PIQ9	Pin Interrupt 9 Edge. Set the PINT_EDGE_SET.PIQ9 bit to enable edge sensitivity.
8 (R/W1S)	PIQ8	Pin Interrupt 8 Edge. Set the PINT_EDGE_SET.PIQ8 bit to enable edge sensitivity.
7 (R/W1S)	PIQ7	Pin Interrupt 7 Edge. Set the PINT_EDGE_SET.PIQ7 bit to enable edge sensitivity.
6 (R/W1S)	PIQ6	Pin Interrupt 6 Edge. Set the PINT_EDGE_SET.PIQ6 bit to enable edge sensitivity.
5 (R/W1S)	PIQ5	Pin Interrupt 5 Edge. Set the PINT_EDGE_SET.PIQ5 bit to enable edge sensitivity.
4 (R/W1S)	PIQ4	Pin Interrupt 4 Edge. Set the PINT_EDGE_SET.PIQ4 bit to enable edge sensitivity.
3 (R/W1S)	PIQ3	Pin Interrupt 3 Edge. Set the PINT_EDGE_SET.PIQ3 bit to enable edge sensitivity.
2 (R/W1S)	PIQ2	Pin Interrupt 2 Edge. Set the PINT_EDGE_SET.PIQ2 bit to enable edge sensitivity.
1 (R/W1S)	PIQ1	Pin Interrupt 1 Edge. Set the PINT_EDGE_SET.PIQ1 bit to enable edge sensitivity.
0 (R/W1S)	PIQ0	Pin Interrupt 0 Edge. Set the PINT_EDGE_SET.PIQ0 bit to enable edge sensitivity.

PINT Invert Clear Register

The PINT_INV_CLR register disables inverting input polarity. Writing 1 to a bit in PINT_INV_CLR disables an inverter for input on the corresponding pin.

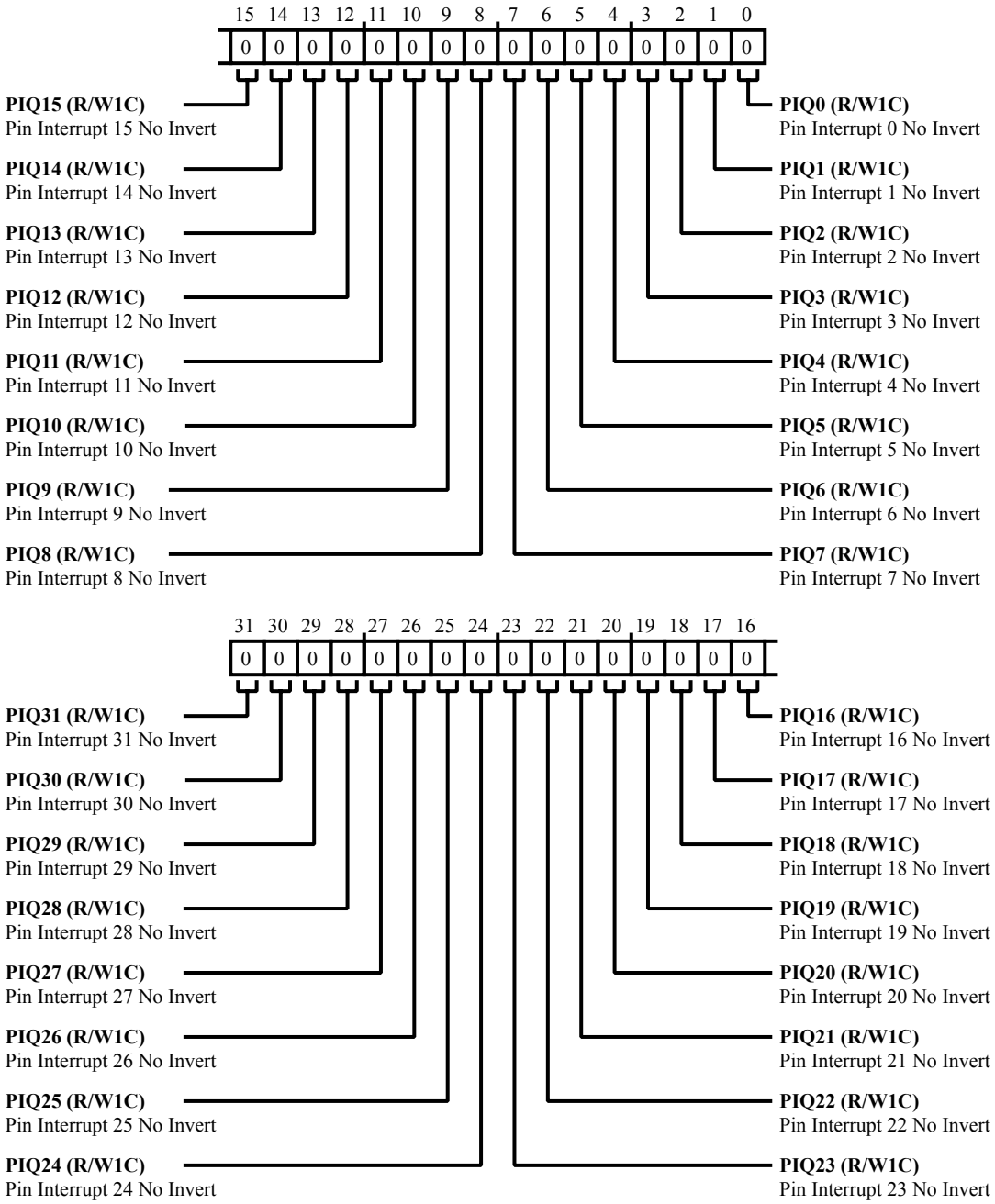


Figure 13-29: PINT_INV_CLR Register Diagram

Table 13-35: PINT_INV_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 No Invert. Set the PINT_INV_CLR.PIQ31 bit to disable inverted input.
30 (R/W1C)	PIQ30	Pin Interrupt 30 No Invert. Set the PINT_INV_CLR.PIQ30 bit to disable inverted input.
29 (R/W1C)	PIQ29	Pin Interrupt 29 No Invert. Set the PINT_INV_CLR.PIQ29 bit to disable inverted input.
28 (R/W1C)	PIQ28	Pin Interrupt 28 No Invert. Set the PINT_INV_CLR.PIQ28 bit to disable inverted input.
27 (R/W1C)	PIQ27	Pin Interrupt 27 No Invert. Set the PINT_INV_CLR.PIQ27 bit to disable inverted input.
26 (R/W1C)	PIQ26	Pin Interrupt 26 No Invert. Set the PINT_INV_CLR.PIQ26 bit to disable inverted input.
25 (R/W1C)	PIQ25	Pin Interrupt 25 No Invert. Set the PINT_INV_CLR.PIQ25 bit to disable inverted input.
24 (R/W1C)	PIQ24	Pin Interrupt 24 No Invert. Set the PINT_INV_CLR.PIQ24 bit to disable inverted input.
23 (R/W1C)	PIQ23	Pin Interrupt 23 No Invert. Set the PINT_INV_CLR.PIQ23 bit to disable inverted input.
22 (R/W1C)	PIQ22	Pin Interrupt 22 No Invert. Set the PINT_INV_CLR.PIQ22 bit to disable inverted input.
21 (R/W1C)	PIQ21	Pin Interrupt 21 No Invert. Set the PINT_INV_CLR.PIQ21 bit to disable inverted input.
20 (R/W1C)	PIQ20	Pin Interrupt 20 No Invert. Set the PINT_INV_CLR.PIQ20 bit to disable inverted input.
19 (R/W1C)	PIQ19	Pin Interrupt 19 No Invert. Set the PINT_INV_CLR.PIQ19 bit to disable inverted input.
18 (R/W1C)	PIQ18	Pin Interrupt 18 No Invert. Set the PINT_INV_CLR.PIQ18 bit to disable inverted input.
17 (R/W1C)	PIQ17	Pin Interrupt 17 No Invert. Set the PINT_INV_CLR.PIQ17 bit to disable inverted input.
16 (R/W1C)	PIQ16	Pin Interrupt 16 No Invert. Set the PINT_INV_CLR.PIQ16 bit to disable inverted input.

Table 13-35: PINT_INV_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1C)	PIQ15	Pin Interrupt 15 No Invert. Set the PINT_INV_CLR.PIQ15 bit to disable inverted input.
14 (R/W1C)	PIQ14	Pin Interrupt 14 No Invert. Set the PINT_INV_CLR.PIQ14 bit to disable inverted input.
13 (R/W1C)	PIQ13	Pin Interrupt 13 No Invert. Set the PINT_INV_CLR.PIQ13 bit to disable inverted input.
12 (R/W1C)	PIQ12	Pin Interrupt 12 No Invert. Set the PINT_INV_CLR.PIQ12 bit to disable inverted input.
11 (R/W1C)	PIQ11	Pin Interrupt 11 No Invert. Set the PINT_INV_CLR.PIQ11 bit to disable inverted input.
10 (R/W1C)	PIQ10	Pin Interrupt 10 No Invert. Set the PINT_INV_CLR.PIQ10 bit to disable inverted input.
9 (R/W1C)	PIQ9	Pin Interrupt 9 No Invert. Set the PINT_INV_CLR.PIQ9 bit to disable inverted input.
8 (R/W1C)	PIQ8	Pin Interrupt 8 No Invert. Set the PINT_INV_CLR.PIQ8 bit to disable inverted input.
7 (R/W1C)	PIQ7	Pin Interrupt 7 No Invert. Set the PINT_INV_CLR.PIQ7 bit to disable inverted input.
6 (R/W1C)	PIQ6	Pin Interrupt 6 No Invert. Set the PINT_INV_CLR.PIQ6 bit to disable inverted input.
5 (R/W1C)	PIQ5	Pin Interrupt 5 No Invert. Set the PINT_INV_CLR.PIQ5 bit to disable inverted input.
4 (R/W1C)	PIQ4	Pin Interrupt 4 No Invert. Set the PINT_INV_CLR.PIQ4 bit to disable inverted input.
3 (R/W1C)	PIQ3	Pin Interrupt 3 No Invert. Set the PINT_INV_CLR.PIQ3 bit to disable inverted input.
2 (R/W1C)	PIQ2	Pin Interrupt 2 No Invert. Set the PINT_INV_CLR.PIQ2 bit to disable inverted input.
1 (R/W1C)	PIQ1	Pin Interrupt 1 No Invert. Set the PINT_INV_CLR.PIQ1 bit to disable inverted input.
0 (R/W1C)	PIQ0	Pin Interrupt 0 No Invert. Set the PINT_INV_CLR.PIQ0 bit to disable inverted input.

PINT Invert Set Register

The PINT_INV_SET register enables inverting input polarity. Writing 1 to a bit in PINT_INV_SET enables an inverter for input on the corresponding pin.

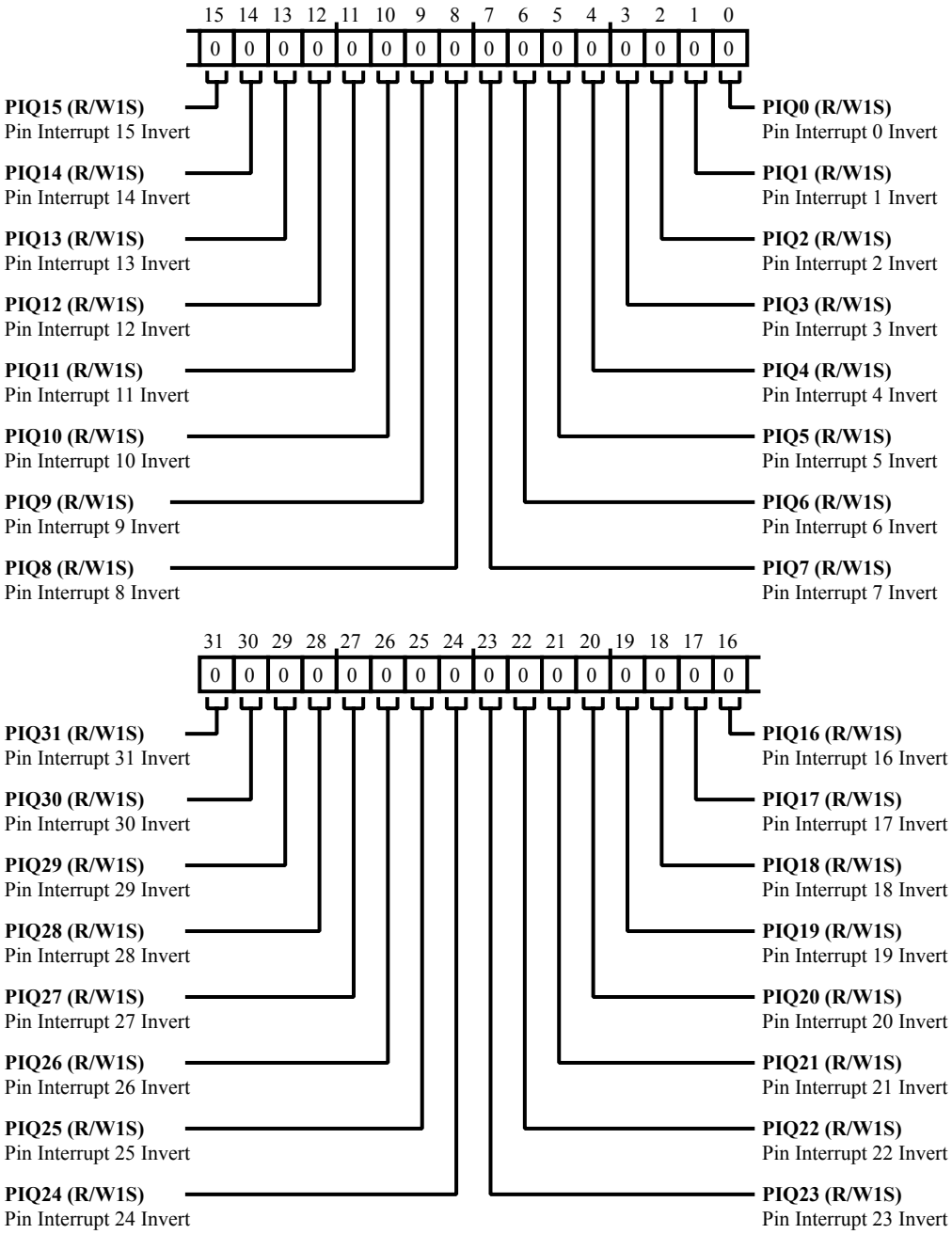


Figure 13-30: PINT_INV_SET Register Diagram

Table 13-36: PINT_INV_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	PIQ31	Pin Interrupt 31 Invert. Set the PINT_INV_SET.PIQ31 bit to enable inverted input.
30 (R/W1S)	PIQ30	Pin Interrupt 30 Invert. Set the PINT_INV_SET.PIQ30 bit to enable inverted input.
29 (R/W1S)	PIQ29	Pin Interrupt 29 Invert. Set the PINT_INV_SET.PIQ29 bit to enable inverted input.
28 (R/W1S)	PIQ28	Pin Interrupt 28 Invert. Set the PINT_INV_SET.PIQ28 bit to enable inverted input.
27 (R/W1S)	PIQ27	Pin Interrupt 27 Invert. Set the PINT_INV_SET.PIQ27 bit to enable inverted input.
26 (R/W1S)	PIQ26	Pin Interrupt 26 Invert. Set the PINT_INV_SET.PIQ26 bit to enable inverted input.
25 (R/W1S)	PIQ25	Pin Interrupt 25 Invert. Set the PINT_INV_SET.PIQ25 bit to enable inverted input.
24 (R/W1S)	PIQ24	Pin Interrupt 24 Invert. Set the PINT_INV_SET.PIQ24 bit to enable inverted input.
23 (R/W1S)	PIQ23	Pin Interrupt 23 Invert. Set the PINT_INV_SET.PIQ23 bit to enable inverted input.
22 (R/W1S)	PIQ22	Pin Interrupt 22 Invert. Set the PINT_INV_SET.PIQ22 bit to enable inverted input.
21 (R/W1S)	PIQ21	Pin Interrupt 21 Invert. Set the PINT_INV_SET.PIQ21 bit to enable inverted input.
20 (R/W1S)	PIQ20	Pin Interrupt 20 Invert. Set the PINT_INV_SET.PIQ20 bit to enable inverted input.
19 (R/W1S)	PIQ19	Pin Interrupt 19 Invert. Set the PINT_INV_SET.PIQ19 bit to enable inverted input.
18 (R/W1S)	PIQ18	Pin Interrupt 18 Invert. Set the PINT_INV_SET.PIQ18 bit to enable inverted input.
17 (R/W1S)	PIQ17	Pin Interrupt 17 Invert. Set the PINT_INV_SET.PIQ17 bit to enable inverted input.
16 (R/W1S)	PIQ16	Pin Interrupt 16 Invert. Set the PINT_INV_SET.PIQ16 bit to enable inverted input.

Table 13-36: PINT_INV_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1S)	PIQ15	Pin Interrupt 15 Invert. Set the PINT_INV_SET.PIQ15 bit to enable inverted input.
14 (R/W1S)	PIQ14	Pin Interrupt 14 Invert. Set the PINT_INV_SET.PIQ14 bit to enable inverted input.
13 (R/W1S)	PIQ13	Pin Interrupt 13 Invert. Set the PINT_INV_SET.PIQ13 bit to enable inverted input.
12 (R/W1S)	PIQ12	Pin Interrupt 12 Invert. Set the PINT_INV_SET.PIQ12 bit to enable inverted input.
11 (R/W1S)	PIQ11	Pin Interrupt 11 Invert. Set the PINT_INV_SET.PIQ11 bit to enable inverted input.
10 (R/W1S)	PIQ10	Pin Interrupt 10 Invert. Set the PINT_INV_SET.PIQ10 bit to enable inverted input.
9 (R/W1S)	PIQ9	Pin Interrupt 9 Invert. Set the PINT_INV_SET.PIQ9 bit to enable inverted input.
8 (R/W1S)	PIQ8	Pin Interrupt 8 Invert. Set the PINT_INV_SET.PIQ8 bit to enable inverted input.
7 (R/W1S)	PIQ7	Pin Interrupt 7 Invert. Set the PINT_INV_SET.PIQ7 bit to enable inverted input.
6 (R/W1S)	PIQ6	Pin Interrupt 6 Invert. Set the PINT_INV_SET.PIQ6 bit to enable inverted input.
5 (R/W1S)	PIQ5	Pin Interrupt 5 Invert. Set the PINT_INV_SET.PIQ5 bit to enable inverted input.
4 (R/W1S)	PIQ4	Pin Interrupt 4 Invert. Set the PINT_INV_SET.PIQ4 bit to enable inverted input.
3 (R/W1S)	PIQ3	Pin Interrupt 3 Invert. Set the PINT_INV_SET.PIQ3 bit to enable inverted input.
2 (R/W1S)	PIQ2	Pin Interrupt 2 Invert. Set the PINT_INV_SET.PIQ2 bit to enable inverted input.
1 (R/W1S)	PIQ1	Pin Interrupt 1 Invert. Set the PINT_INV_SET.PIQ1 bit to enable inverted input.
0 (R/W1S)	PIQ0	Pin Interrupt 0 Invert. Set the PINT_INV_SET.PIQ0 bit to enable inverted input.

PINT Latch Register

The PINT_LATCH register indicates the interrupt latch status for pin interrupts. When set, an interrupt request is latched. When cleared, there is no interrupt request latched.

Both the PINT_REQ and PINT_LATCH registers indicate whether an interrupt request is latched on the respective pin. The PINT_LATCH register is a latch that operates regardless of the interrupt masks. Bits of the PINT_REQ register depend on the mask register. The PINT_REQ register is a logical AND of the PINT_LATCH register and the interrupt mask.

Having two separate registers here enables the user to interrogate certain pins in polling mode while others work in interrupt mode. The PINT_LATCH registers can be used for edge detection or pin activity detection.

Both registers have W1C behavior. Writing a 1 to either register clears the respective bits in both registers. For interrupt operation, the user may prefer to W1C the PINT_REQ register (address still loaded in Px pointer). In polling mode, it might be cleaner to W1C the PINT_LATCH register.

Whether in edge-sensitive mode or level-sensitive mode, PINT_LATCH bits are never cleared by hardware except at system reset. Even in level-sensitive mode, the PINT_LATCH register functions as latch.

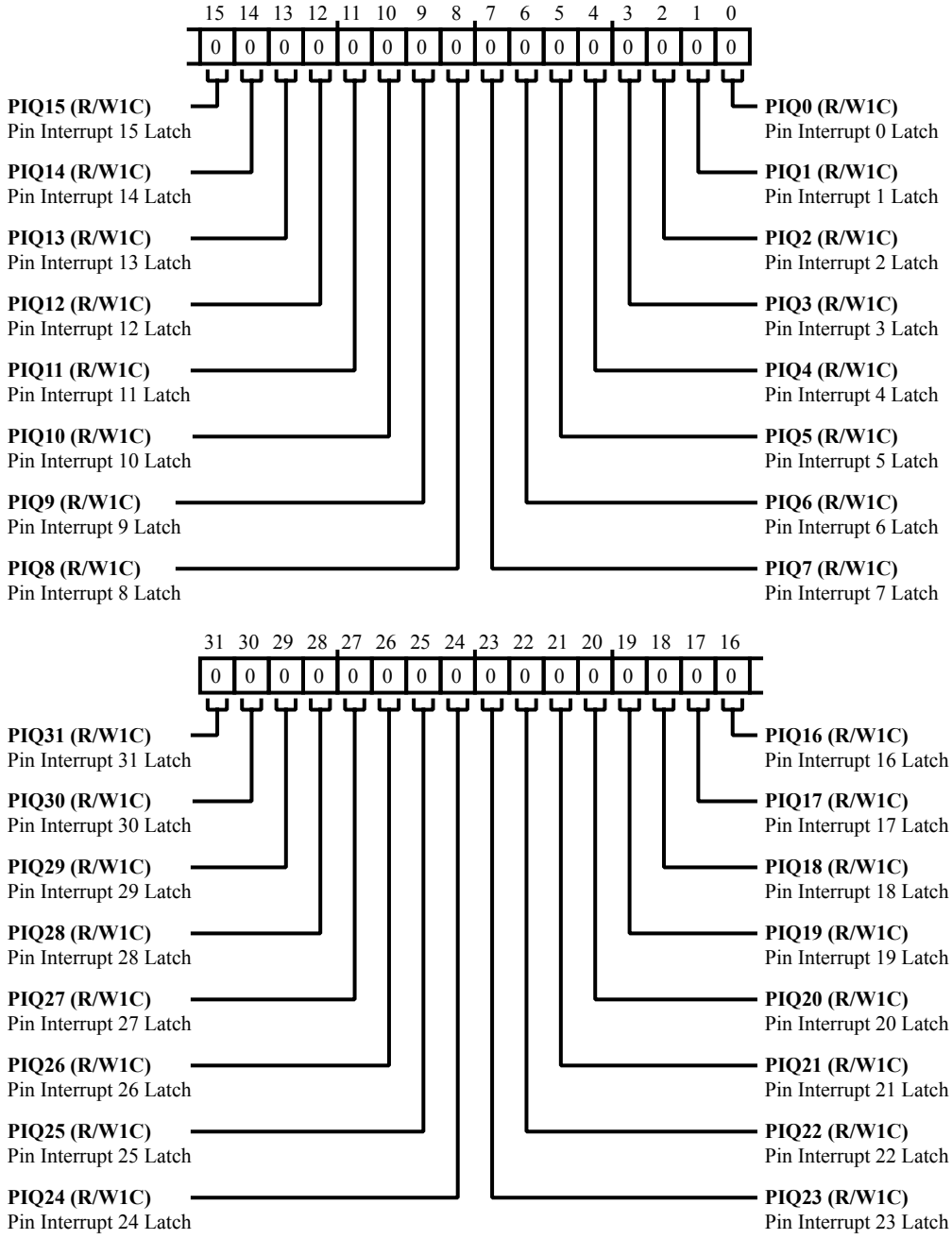


Figure 13-31: PINT_LATCH Register Diagram

Table 13-37: PINT_LATCH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 Latch. If the `PINT_LATCH.PIQ31` bit is set, the request is latched.

Table 13-37: PINT_LATCH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W1C)	PIQ30	Pin Interrupt 30 Latch. If the `PINT_LATCH.PIQ30` bit is set, the request is latched.
29 (R/W1C)	PIQ29	Pin Interrupt 29 Latch. If the `PINT_LATCH.PIQ29` bit is set, the request is latched.
28 (R/W1C)	PIQ28	Pin Interrupt 28 Latch. If the `PINT_LATCH.PIQ28` bit is set, the request is latched.
27 (R/W1C)	PIQ27	Pin Interrupt 27 Latch. If the `PINT_LATCH.PIQ27` bit is set, the request is latched.
26 (R/W1C)	PIQ26	Pin Interrupt 26 Latch. If the `PINT_LATCH.PIQ26` bit is set, the request is latched.
25 (R/W1C)	PIQ25	Pin Interrupt 25 Latch. If the `PINT_LATCH.PIQ25` bit is set, the request is latched.
24 (R/W1C)	PIQ24	Pin Interrupt 24 Latch. If the `PINT_LATCH.PIQ24` bit is set, the request is latched.
23 (R/W1C)	PIQ23	Pin Interrupt 23 Latch. If the `PINT_LATCH.PIQ23` bit is set, the request is latched.
22 (R/W1C)	PIQ22	Pin Interrupt 22 Latch. If the `PINT_LATCH.PIQ22` bit is set, the request is latched.
21 (R/W1C)	PIQ21	Pin Interrupt 21 Latch. If the `PINT_LATCH.PIQ21` bit is set, the request is latched.
20 (R/W1C)	PIQ20	Pin Interrupt 20 Latch. If the `PINT_LATCH.PIQ20` bit is set, the request is latched.
19 (R/W1C)	PIQ19	Pin Interrupt 19 Latch. If the `PINT_LATCH.PIQ19` bit is set, the request is latched.
18 (R/W1C)	PIQ18	Pin Interrupt 18 Latch. If the `PINT_LATCH.PIQ18` bit is set, the request is latched.
17 (R/W1C)	PIQ17	Pin Interrupt 17 Latch. If the `PINT_LATCH.PIQ17` bit is set, the request is latched.
16 (R/W1C)	PIQ16	Pin Interrupt 16 Latch. If the `PINT_LATCH.PIQ16` bit is set, the request is latched.
15 (R/W1C)	PIQ15	Pin Interrupt 15 Latch. If the `PINT_LATCH.PIQ15` bit is set, the request is latched.

Table 13-37: PINT_LATCH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W1C)	PIQ14	Pin Interrupt 14 Latch. If the `PINT_LATCH.PIQ14` bit is set, the request is latched.
13 (R/W1C)	PIQ13	Pin Interrupt 13 Latch. If the `PINT_LATCH.PIQ13` bit is set, the request is latched.
12 (R/W1C)	PIQ12	Pin Interrupt 12 Latch. If the `PINT_LATCH.PIQ12` bit is set, the request is latched.
11 (R/W1C)	PIQ11	Pin Interrupt 11 Latch. If the `PINT_LATCH.PIQ11` bit is set, the request is latched.
10 (R/W1C)	PIQ10	Pin Interrupt 10 Latch. If the `PINT_LATCH.PIQ10` bit is set, the request is latched.
9 (R/W1C)	PIQ9	Pin Interrupt 9 Latch. If the `PINT_LATCH.PIQ9` bit is set, the request is latched.
8 (R/W1C)	PIQ8	Pin Interrupt 8 Latch. If the `PINT_LATCH.PIQ8` bit is set, the request is latched.
7 (R/W1C)	PIQ7	Pin Interrupt 7 Latch. If the `PINT_LATCH.PIQ7` bit is set, the request is latched.
6 (R/W1C)	PIQ6	Pin Interrupt 6 Latch. If the `PINT_LATCH.PIQ6` bit is set, the request is latched.
5 (R/W1C)	PIQ5	Pin Interrupt 5 Latch. If the `PINT_LATCH.PIQ5` bit is set, the request is latched.
4 (R/W1C)	PIQ4	Pin Interrupt 4 Latch. If the `PINT_LATCH.PIQ4` bit is set, the request is latched.
3 (R/W1C)	PIQ3	Pin Interrupt 3 Latch. If the `PINT_LATCH.PIQ3` bit is set, the request is latched.
2 (R/W1C)	PIQ2	Pin Interrupt 2 Latch. If the `PINT_LATCH.PIQ2` bit is set, the request is latched.
1 (R/W1C)	PIQ1	Pin Interrupt 1 Latch. If the `PINT_LATCH.PIQ1` bit is set, the request is latched.
0 (R/W1C)	PIQ0	Pin Interrupt 0 Latch. If the `PINT_LATCH.PIQ0` bit is set, the request is latched.

PINT Mask Clear Register

The PINT_MSK_CLR register permits masking (disabling) of interrupt requests. Writing 1 to a bit in PINT_MSK_CLR masks the corresponding pin interrupt.

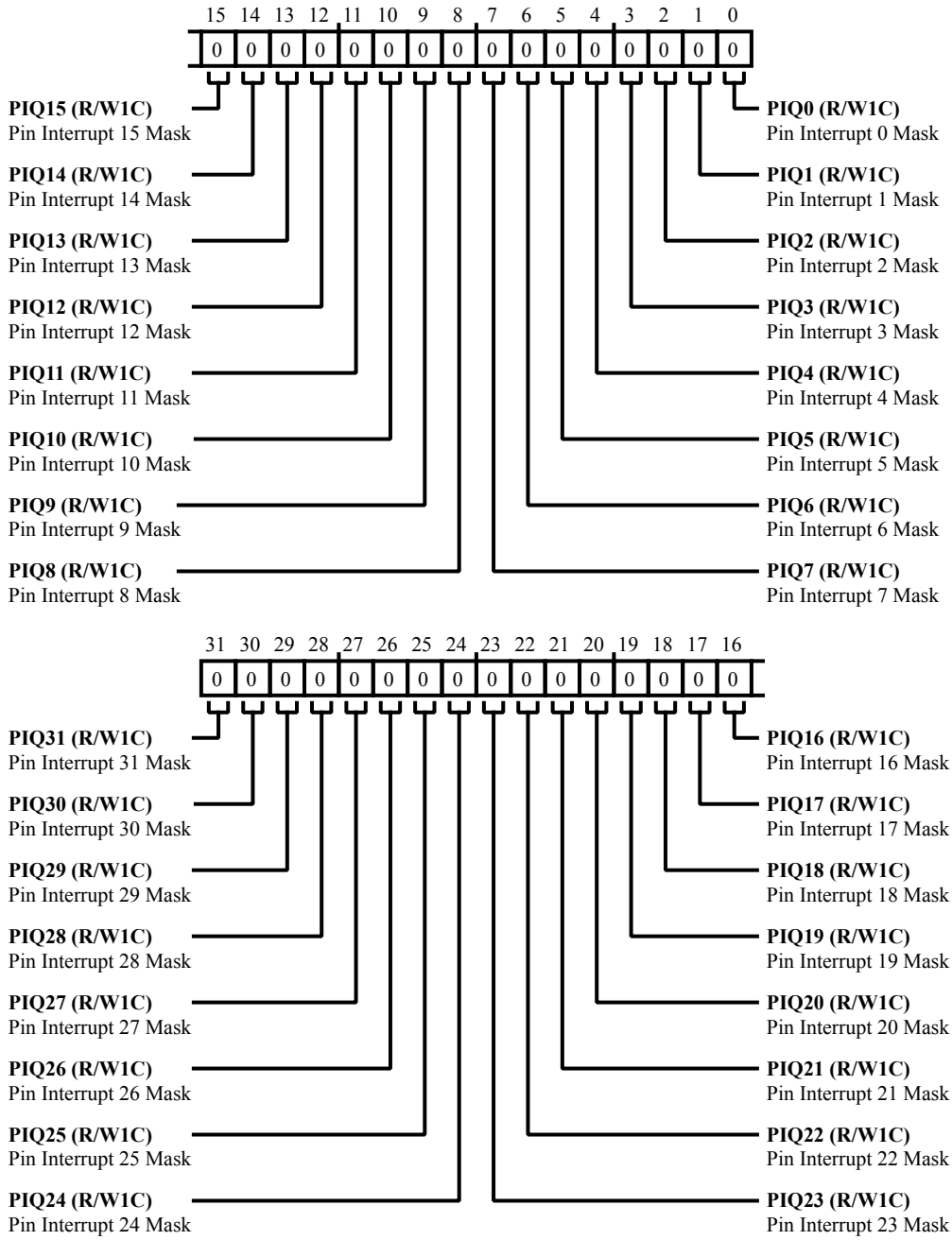


Figure 13-32: PINT_MSK_CLR Register Diagram

Table 13-38: PINT_MSK_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 Mask. Set the PINT_MSK_CLR.PIQ31 bit to disable the interrupt.
30 (R/W1C)	PIQ30	Pin Interrupt 30 Mask. Set the PINT_MSK_CLR.PIQ30 bit to disable the interrupt.
29 (R/W1C)	PIQ29	Pin Interrupt 29 Mask. Set the PINT_MSK_CLR.PIQ29 bit to disable the interrupt.
28 (R/W1C)	PIQ28	Pin Interrupt 28 Mask. Set the PINT_MSK_CLR.PIQ28 bit to disable the interrupt.
27 (R/W1C)	PIQ27	Pin Interrupt 27 Mask. Set the PINT_MSK_CLR.PIQ27 bit to disable the interrupt.
26 (R/W1C)	PIQ26	Pin Interrupt 26 Mask. Set the PINT_MSK_CLR.PIQ26 bit to disable the interrupt.
25 (R/W1C)	PIQ25	Pin Interrupt 25 Mask. Set the PINT_MSK_CLR.PIQ25 bit to disable the interrupt.
24 (R/W1C)	PIQ24	Pin Interrupt 24 Mask. Set the PINT_MSK_CLR.PIQ24 bit to disable the interrupt.
23 (R/W1C)	PIQ23	Pin Interrupt 23 Mask. Set the PINT_MSK_CLR.PIQ23 bit to disable the interrupt.
22 (R/W1C)	PIQ22	Pin Interrupt 22 Mask. Set the PINT_MSK_CLR.PIQ22 bit to disable the interrupt.
21 (R/W1C)	PIQ21	Pin Interrupt 21 Mask. Set the PINT_MSK_CLR.PIQ21 bit to disable the interrupt.
20 (R/W1C)	PIQ20	Pin Interrupt 20 Mask. Set the PINT_MSK_CLR.PIQ20 bit to disable the interrupt.
19 (R/W1C)	PIQ19	Pin Interrupt 19 Mask. Set the PINT_MSK_CLR.PIQ19 bit to disable the interrupt.
18 (R/W1C)	PIQ18	Pin Interrupt 18 Mask. Set the PINT_MSK_CLR.PIQ18 bit to disable the interrupt.
17 (R/W1C)	PIQ17	Pin Interrupt 17 Mask. Set the PINT_MSK_CLR.PIQ17 bit to disable the interrupt.
16 (R/W1C)	PIQ16	Pin Interrupt 16 Mask. Set the PINT_MSK_CLR.PIQ16 bit to disable the interrupt.

Table 13-38: PINT_MSK_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1C)	PIQ15	Pin Interrupt 15 Mask. Set the PINT_MSK_CLR.PIQ15 bit to disable the interrupt.
14 (R/W1C)	PIQ14	Pin Interrupt 14 Mask. Set the PINT_MSK_CLR.PIQ14 bit to disable the interrupt.
13 (R/W1C)	PIQ13	Pin Interrupt 13 Mask. Set the PINT_MSK_CLR.PIQ13 bit to disable the interrupt.
12 (R/W1C)	PIQ12	Pin Interrupt 12 Mask. Set the PINT_MSK_CLR.PIQ12 bit to disable the interrupt.
11 (R/W1C)	PIQ11	Pin Interrupt 11 Mask. Set the PINT_MSK_CLR.PIQ11 bit to disable the interrupt.
10 (R/W1C)	PIQ10	Pin Interrupt 10 Mask. Set the PINT_MSK_CLR.PIQ10 bit to disable the interrupt.
9 (R/W1C)	PIQ9	Pin Interrupt 9 Mask. Set the PINT_MSK_CLR.PIQ9 bit to disable the interrupt.
8 (R/W1C)	PIQ8	Pin Interrupt 8 Mask. Set the PINT_MSK_CLR.PIQ8 bit to disable the interrupt.
7 (R/W1C)	PIQ7	Pin Interrupt 7 Mask. Set the PINT_MSK_CLR.PIQ7 bit to disable the interrupt.
6 (R/W1C)	PIQ6	Pin Interrupt 6 Mask. Set the PINT_MSK_CLR.PIQ6 bit to disable the interrupt.
5 (R/W1C)	PIQ5	Pin Interrupt 5 Mask. Set the PINT_MSK_CLR.PIQ5 bit to disable the interrupt.
4 (R/W1C)	PIQ4	Pin Interrupt 4 Mask. Set the PINT_MSK_CLR.PIQ4 bit to disable the interrupt.
3 (R/W1C)	PIQ3	Pin Interrupt 3 Mask. Set the PINT_MSK_CLR.PIQ3 bit to disable the interrupt.
2 (R/W1C)	PIQ2	Pin Interrupt 2 Mask. Set the PINT_MSK_CLR.PIQ2 bit to disable the interrupt.
1 (R/W1C)	PIQ1	Pin Interrupt 1 Mask. Set the PINT_MSK_CLR.PIQ1 bit to disable the interrupt.
0 (R/W1C)	PIQ0	Pin Interrupt 0 Mask. Set the PINT_MSK_CLR.PIQ0 bit to disable the interrupt.

PINT Mask Set Register

The PINT_MSK_SET register permits unmasking (enabling) of interrupt requests. Writing 1 to a bit in PINT_MSK_SET unmasks the corresponding pin interrupt.

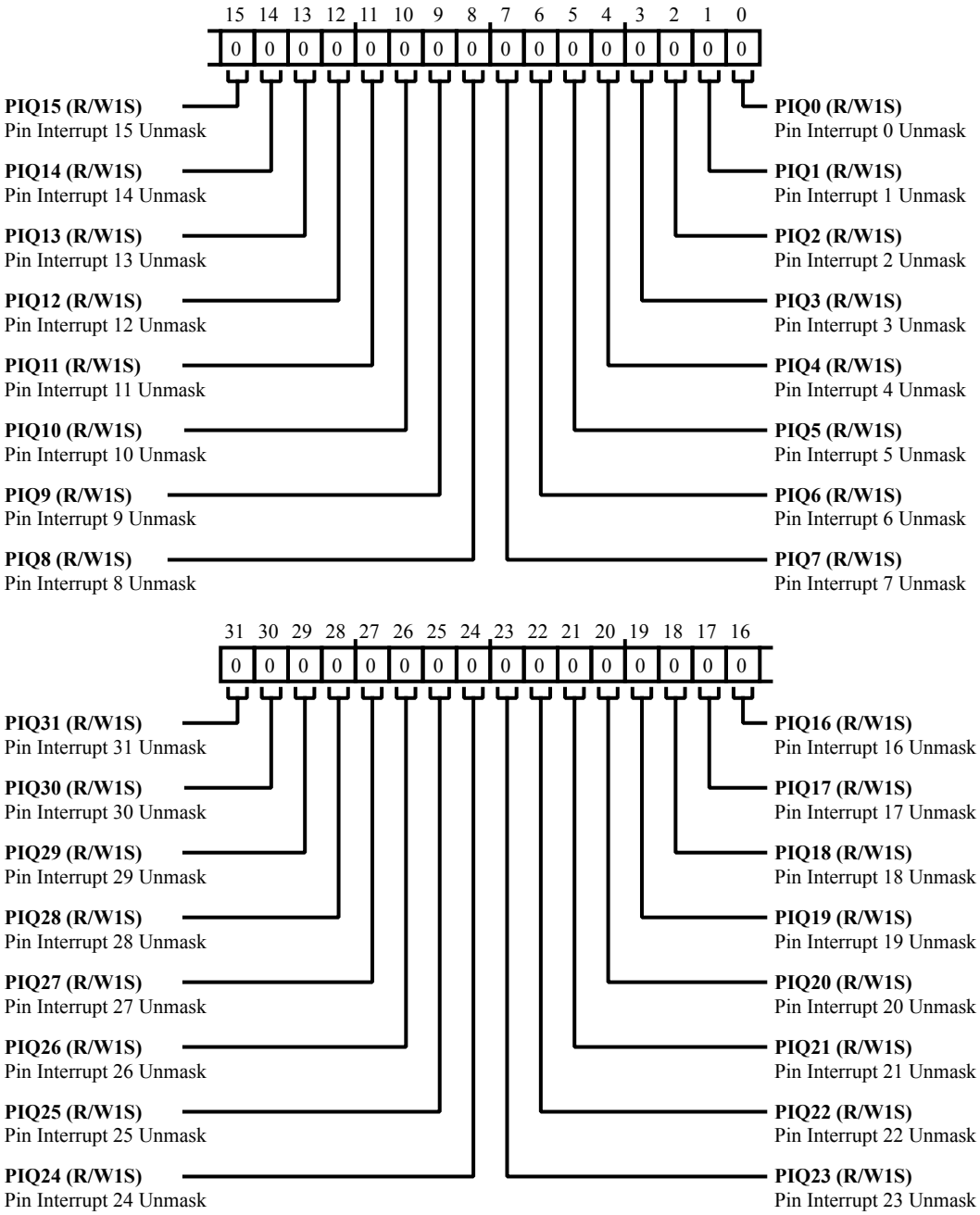


Figure 13-33: PINT_MSK_SET Register Diagram

Table 13-39: PINT_MSK_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	PIQ31	Pin Interrupt 31 Unmask. Set the PINT_MSK_SET.PIQ31 bit to enable the interrupt.
30 (R/W1S)	PIQ30	Pin Interrupt 30 Unmask. Set the PINT_MSK_SET.PIQ30 bit to enable the interrupt.
29 (R/W1S)	PIQ29	Pin Interrupt 29 Unmask. Set the PINT_MSK_SET.PIQ29 bit to enable the interrupt.
28 (R/W1S)	PIQ28	Pin Interrupt 28 Unmask. Set the PINT_MSK_SET.PIQ28 bit to enable the interrupt.
27 (R/W1S)	PIQ27	Pin Interrupt 27 Unmask. Set the PINT_MSK_SET.PIQ27 bit to enable the interrupt.
26 (R/W1S)	PIQ26	Pin Interrupt 26 Unmask. Set the PINT_MSK_SET.PIQ26 bit to enable the interrupt.
25 (R/W1S)	PIQ25	Pin Interrupt 25 Unmask. Set the PINT_MSK_SET.PIQ25 bit to enable the interrupt.
24 (R/W1S)	PIQ24	Pin Interrupt 24 Unmask. Set the PINT_MSK_SET.PIQ24 bit to enable the interrupt.
23 (R/W1S)	PIQ23	Pin Interrupt 23 Unmask. Set the PINT_MSK_SET.PIQ23 bit to enable the interrupt.
22 (R/W1S)	PIQ22	Pin Interrupt 22 Unmask. Set the PINT_MSK_SET.PIQ22 bit to enable the interrupt.
21 (R/W1S)	PIQ21	Pin Interrupt 21 Unmask. Set the PINT_MSK_SET.PIQ21 bit to enable the interrupt.
20 (R/W1S)	PIQ20	Pin Interrupt 20 Unmask. Set the PINT_MSK_SET.PIQ20 bit to enable the interrupt.
19 (R/W1S)	PIQ19	Pin Interrupt 19 Unmask. Set the PINT_MSK_SET.PIQ19 bit to enable the interrupt.
18 (R/W1S)	PIQ18	Pin Interrupt 18 Unmask. Set the PINT_MSK_SET.PIQ18 bit to enable the interrupt.
17 (R/W1S)	PIQ17	Pin Interrupt 17 Unmask. Set the PINT_MSK_SET.PIQ17 bit to enable the interrupt.
16 (R/W1S)	PIQ16	Pin Interrupt 16 Unmask. Set the PINT_MSK_SET.PIQ16 bit to enable the interrupt.

Table 13-39: PINT_MSK_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W1S)	PIQ15	Pin Interrupt 15 Unmask. Set the PINT_MSK_SET.PIQ15 bit to enable the interrupt.
14 (R/W1S)	PIQ14	Pin Interrupt 14 Unmask. Set the PINT_MSK_SET.PIQ14 bit to enable the interrupt.
13 (R/W1S)	PIQ13	Pin Interrupt 13 Unmask. Set the PINT_MSK_SET.PIQ13 bit to enable the interrupt.
12 (R/W1S)	PIQ12	Pin Interrupt 12 Unmask. Set the PINT_MSK_SET.PIQ12 bit to enable the interrupt.
11 (R/W1S)	PIQ11	Pin Interrupt 11 Unmask. Set the PINT_MSK_SET.PIQ11 bit to enable the interrupt.
10 (R/W1S)	PIQ10	Pin Interrupt 10 Unmask. Set the PINT_MSK_SET.PIQ10 bit to enable the interrupt.
9 (R/W1S)	PIQ9	Pin Interrupt 9 Unmask. Set the PINT_MSK_SET.PIQ9 bit to enable the interrupt.
8 (R/W1S)	PIQ8	Pin Interrupt 8 Unmask. Set the PINT_MSK_SET.PIQ8 bit to enable the interrupt.
7 (R/W1S)	PIQ7	Pin Interrupt 7 Unmask. Set the PINT_MSK_SET.PIQ7 bit to enable the interrupt.
6 (R/W1S)	PIQ6	Pin Interrupt 6 Unmask. Set the PINT_MSK_SET.PIQ6 bit to enable the interrupt.
5 (R/W1S)	PIQ5	Pin Interrupt 5 Unmask. Set the PINT_MSK_SET.PIQ5 bit to enable the interrupt.
4 (R/W1S)	PIQ4	Pin Interrupt 4 Unmask. Set the PINT_MSK_SET.PIQ4 bit to enable the interrupt.
3 (R/W1S)	PIQ3	Pin Interrupt 3 Unmask. Set the PINT_MSK_SET.PIQ3 bit to enable the interrupt.
2 (R/W1S)	PIQ2	Pin Interrupt 2 Unmask. Set the PINT_MSK_SET.PIQ2 bit to enable the interrupt.
1 (R/W1S)	PIQ1	Pin Interrupt 1 Unmask. Set the PINT_MSK_SET.PIQ1 bit to enable the interrupt.
0 (R/W1S)	PIQ0	Pin Interrupt 0 Unmask. Set the PINT_MSK_SET.PIQ0 bit to enable the interrupt.

PINT Pin State Register

When a half port is assigned to a byte in any PINT block, the state of the eight pins (regardless of GPIO or function, input or output) can be seen in the PINT_PINSTATE register. While neither input nor output drivers of the pin are enabled, reads of the pin state in PINT_PINSTATE return zero. The PINT_PINSTATE register reports the inverted state of the pin if the signal inverter is activated by the PINT_INV_SET register. The inverter can be enabled on an individual bit-by-bit basis. Every bit in the PINT_INV_SET and PINT_INV_CLR register pair represents a pin signal.

The pin interrupt pin state registers enable the service routine to read the current state of the pin without reading from GPIO space. If there was an edge-sensitive interrupt, the service routine can check whether the state of the pin is still high or turned low.

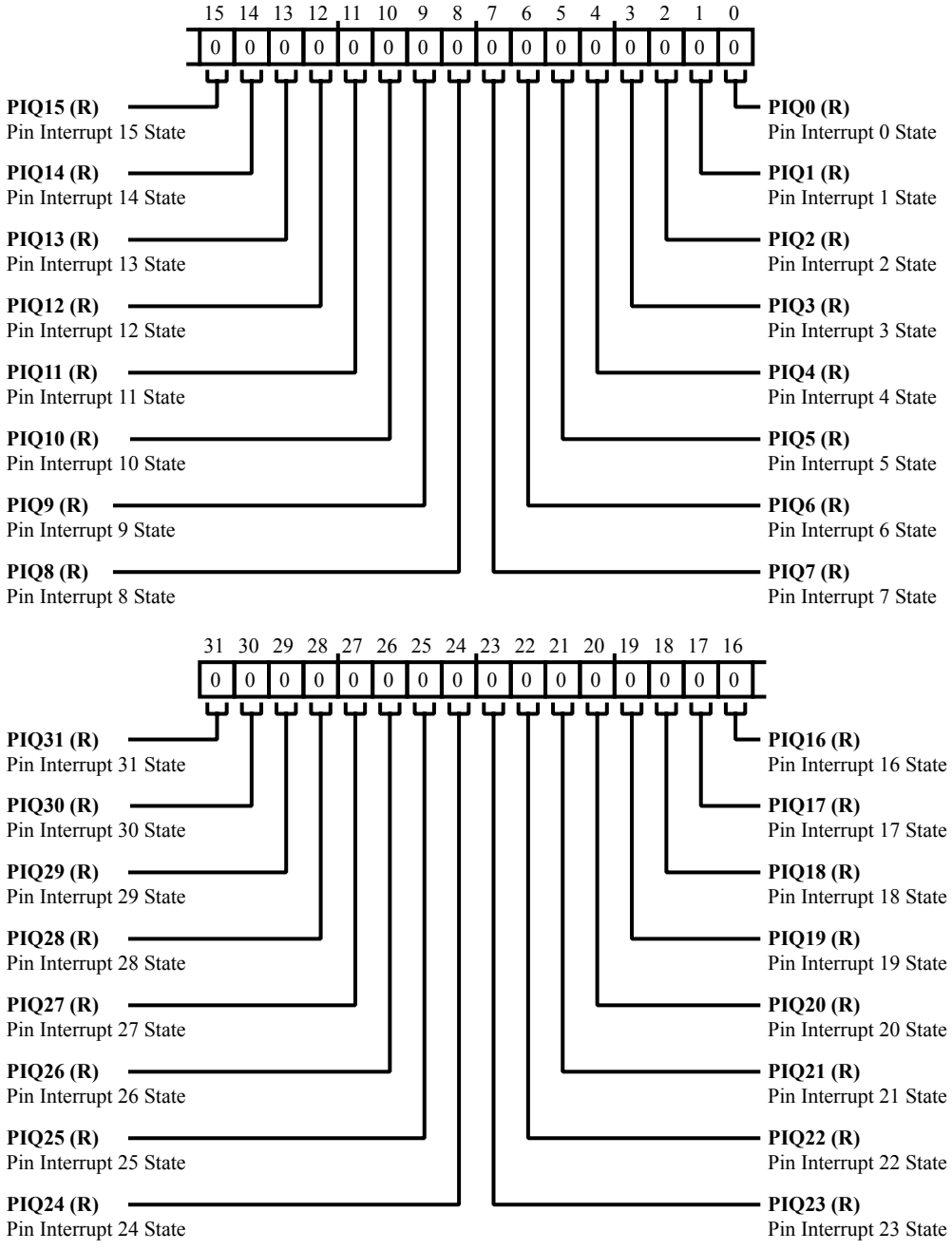


Figure 13-34: PINT_PINSTATE Register Diagram

Table 13-40: PINT_PINSTATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	PIQ31	Pin Interrupt 31 State. A read of the `PINT_PINSTATE.PIQ31` bit returns the pin state.

Table 13-40: PINT_PINSTATE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/NW)	PIQ30	Pin Interrupt 30 State. A read of the PINT_PINSTATE.PIQ30 bit returns the pin state.
29 (R/NW)	PIQ29	Pin Interrupt 29 State. A read of the PINT_PINSTATE.PIQ29 bit returns the pin state.
28 (R/NW)	PIQ28	Pin Interrupt 28 State. A read of the PINT_PINSTATE.PIQ28 bit returns the pin state.
27 (R/NW)	PIQ27	Pin Interrupt 27 State. A read of the PINT_PINSTATE.PIQ27 bit returns the pin state.
26 (R/NW)	PIQ26	Pin Interrupt 26 State. A read of the PINT_PINSTATE.PIQ26 bit returns the pin state.
25 (R/NW)	PIQ25	Pin Interrupt 25 State. A read of the PINT_PINSTATE.PIQ25 bit returns the pin state.
24 (R/NW)	PIQ24	Pin Interrupt 24 State. A read of the PINT_PINSTATE.PIQ24 bit returns the pin state.
23 (R/NW)	PIQ23	Pin Interrupt 23 State. A read of the PINT_PINSTATE.PIQ23 bit returns the pin state.
22 (R/NW)	PIQ22	Pin Interrupt 22 State. A read of the PINT_PINSTATE.PIQ22 bit returns the pin state.
21 (R/NW)	PIQ21	Pin Interrupt 21 State. A read of the PINT_PINSTATE.PIQ21 bit returns the pin state.
20 (R/NW)	PIQ20	Pin Interrupt 20 State. A read of the PINT_PINSTATE.PIQ20 bit returns the pin state.
19 (R/NW)	PIQ19	Pin Interrupt 19 State. A read of the PINT_PINSTATE.PIQ19 bit returns the pin state.
18 (R/NW)	PIQ18	Pin Interrupt 18 State. A read of the PINT_PINSTATE.PIQ18 bit returns the pin state.
17 (R/NW)	PIQ17	Pin Interrupt 17 State. A read of the PINT_PINSTATE.PIQ17 bit returns the pin state.
16 (R/NW)	PIQ16	Pin Interrupt 16 State. A read of the PINT_PINSTATE.PIQ16 bit returns the pin state.
15 (R/NW)	PIQ15	Pin Interrupt 15 State. A read of the PINT_PINSTATE.PIQ15 bit returns the pin state.

Table 13-40: PINT_PINSTATE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/NW)	PIQ14	Pin Interrupt 14 State. A read of the PINT_PINSTATE.PIQ14 bit returns the pin state.
13 (R/NW)	PIQ13	Pin Interrupt 13 State. A read of the PINT_PINSTATE.PIQ13 bit returns the pin state.
12 (R/NW)	PIQ12	Pin Interrupt 12 State. A read of the PINT_PINSTATE.PIQ12 bit returns the pin state.
11 (R/NW)	PIQ11	Pin Interrupt 11 State. A read of the PINT_PINSTATE.PIQ11 bit returns the pin state.
10 (R/NW)	PIQ10	Pin Interrupt 10 State. A read of the PINT_PINSTATE.PIQ10 bit returns the pin state.
9 (R/NW)	PIQ9	Pin Interrupt 9 State. A read of the PINT_PINSTATE.PIQ9 bit returns the pin state.
8 (R/NW)	PIQ8	Pin Interrupt 8 State. A read of the PINT_PINSTATE.PIQ8 bit returns the pin state.
7 (R/NW)	PIQ7	Pin Interrupt 7 State. A read of the PINT_PINSTATE.PIQ7 bit returns the pin state.
6 (R/NW)	PIQ6	Pin Interrupt 6 State. A read of the PINT_PINSTATE.PIQ6 bit returns the pin state.
5 (R/NW)	PIQ5	Pin Interrupt 5 State. A read of the PINT_PINSTATE.PIQ5 bit returns the pin state.
4 (R/NW)	PIQ4	Pin Interrupt 4 State. A read of the PINT_PINSTATE.PIQ4 bit returns the pin state.
3 (R/NW)	PIQ3	Pin Interrupt 3 State. A read of the PINT_PINSTATE.PIQ3 bit returns the pin state.
2 (R/NW)	PIQ2	Pin Interrupt 2 State. A read of the PINT_PINSTATE.PIQ2 bit returns the pin state.
1 (R/NW)	PIQ1	Pin Interrupt 1 State. A read of the PINT_PINSTATE.PIQ1 bit returns the pin state.
0 (R/NW)	PIQ0	Pin Interrupt 0 State. A read of the PINT_PINSTATE.PIQ0 bit returns the pin state.

PINT Request Register

The PINT_REQ register indicates the interrupt request status for pin interrupts. When set, an interrupt request is pending. When cleared, there is no interrupt request pending.

Both the PINT_REQ and PINT_LATCH registers indicate whether an interrupt request is latched on the respective pin. The PINT_LATCH register is a latch that operates regardless of the interrupt masks. Bits of the PINT_REQ register depend on the mask register. The PINT_REQ register is a logical AND of the PINT_LATCH register and the interrupt mask.

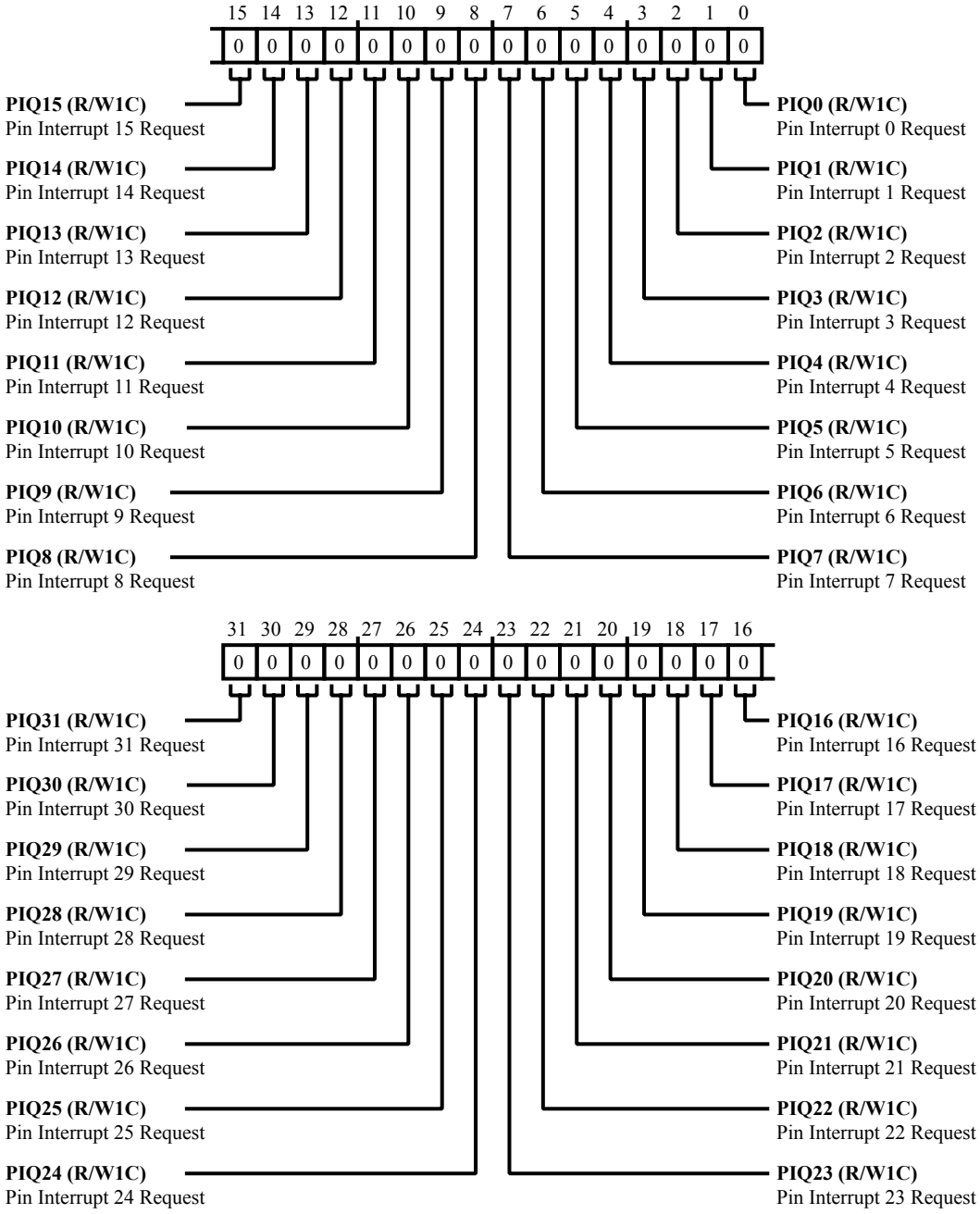


Figure 13-35: PINT_REQ Register Diagram

Table 13-41: PINT_REQ Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	PIQ31	Pin Interrupt 31 Request. If the `PINT_REQ.PIQ31` bit is set, a request is pending.

Table 13-41: PINT_REQ Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W1C)	PIQ30	Pin Interrupt 30 Request. If the `PINT_REQ.PIQ30` bit is set, a request is pending.
29 (R/W1C)	PIQ29	Pin Interrupt 29 Request. If the `PINT_REQ.PIQ29` bit is set, a request is pending.
28 (R/W1C)	PIQ28	Pin Interrupt 28 Request. If the `PINT_REQ.PIQ28` bit is set, a request is pending.
27 (R/W1C)	PIQ27	Pin Interrupt 27 Request. If the `PINT_REQ.PIQ27` bit is set, a request is pending.
26 (R/W1C)	PIQ26	Pin Interrupt 26 Request. If the `PINT_REQ.PIQ26` bit is set, a request is pending.
25 (R/W1C)	PIQ25	Pin Interrupt 25 Request. If the `PINT_REQ.PIQ25` bit is set, a request is pending.
24 (R/W1C)	PIQ24	Pin Interrupt 24 Request. If the `PINT_REQ.PIQ24` bit is set, a request is pending.
23 (R/W1C)	PIQ23	Pin Interrupt 23 Request. If the `PINT_REQ.PIQ23` bit is set, a request is pending.
22 (R/W1C)	PIQ22	Pin Interrupt 22 Request. If the `PINT_REQ.PIQ22` bit is set, a request is pending.
21 (R/W1C)	PIQ21	Pin Interrupt 21 Request. If the `PINT_REQ.PIQ21` bit is set, a request is pending.
20 (R/W1C)	PIQ20	Pin Interrupt 20 Request. If the `PINT_REQ.PIQ20` bit is set, a request is pending.
19 (R/W1C)	PIQ19	Pin Interrupt 19 Request. If the `PINT_REQ.PIQ19` bit is set, a request is pending.
18 (R/W1C)	PIQ18	Pin Interrupt 18 Request. If the `PINT_REQ.PIQ18` bit is set, a request is pending.
17 (R/W1C)	PIQ17	Pin Interrupt 17 Request. If the `PINT_REQ.PIQ17` bit is set, a request is pending.
16 (R/W1C)	PIQ16	Pin Interrupt 16 Request. If the `PINT_REQ.PIQ16` bit is set, a request is pending.
15 (R/W1C)	PIQ15	Pin Interrupt 15 Request. If the `PINT_REQ.PIQ15` bit is set, a request is pending.

Table 13-41: PINT_REQ Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W1C)	PIQ14	Pin Interrupt 14 Request. If the `PINT_REQ.PIQ14` bit is set, a request is pending.
13 (R/W1C)	PIQ13	Pin Interrupt 13 Request. If the `PINT_REQ.PIQ13` bit is set, a request is pending.
12 (R/W1C)	PIQ12	Pin Interrupt 12 Request. If the `PINT_REQ.PIQ12` bit is set, a request is pending.
11 (R/W1C)	PIQ11	Pin Interrupt 11 Request. If the `PINT_REQ.PIQ11` bit is set, a request is pending.
10 (R/W1C)	PIQ10	Pin Interrupt 10 Request. If the `PINT_REQ.PIQ10` bit is set, a request is pending.
9 (R/W1C)	PIQ9	Pin Interrupt 9 Request. If the `PINT_REQ.PIQ9` bit is set, a request is pending.
8 (R/W1C)	PIQ8	Pin Interrupt 8 Request. If the `PINT_REQ.PIQ8` bit is set, a request is pending.
7 (R/W1C)	PIQ7	Pin Interrupt 7 Request. If the `PINT_REQ.PIQ7` bit is set, a request is pending.
6 (R/W1C)	PIQ6	Pin Interrupt 6 Request. If the `PINT_REQ.PIQ6` bit is set, a request is pending.
5 (R/W1C)	PIQ5	Pin Interrupt 5 Request. If the `PINT_REQ.PIQ5` bit is set, a request is pending.
4 (R/W1C)	PIQ4	Pin Interrupt 4 Request. If the `PINT_REQ.PIQ4` bit is set, a request is pending.
3 (R/W1C)	PIQ3	Pin Interrupt 3 Request. If the `PINT_REQ.PIQ3` bit is set, a request is pending.
2 (R/W1C)	PIQ2	Pin Interrupt 2 Request. If the `PINT_REQ.PIQ2` bit is set, a request is pending.
1 (R/W1C)	PIQ1	Pin Interrupt 1 Request. If the `PINT_REQ.PIQ1` bit is set, a request is pending.
0 (R/W1C)	PIQ0	Pin Interrupt 0 Request. If the `PINT_REQ.PIQ0` bit is set, a request is pending.

ADSP-SC59x PADS Register Descriptions

Pads Controller (PADS) contains the following registers.

Table 13-42: ADSP-SC59x PADS Register List

Name	Description
PADS_DAI0_0_DS	DAI0 1 to 10 pins DS control
PADS_DAI0_1_DS	DAI0 11 to 20 pins DS control
PADS_DAI0_IE	DAI0 Port Input Enable Control Register
PADS_DAI1_0_DS	DAI1 1 to 10 pins DS control
PADS_DAI1_1_DS	DAI1 11 to 20 pins DS control
PADS_DAI1_IE	DAI1 Port Input Enable Control Register
PADS_DAI[n]_PDE	DAI[n] Pull-Down Enable
PADS_DAI[n]_PUE	DAI[n] Pull-Up Enable
PADS_NONPORTS_DS	Non-GPIO Drive Strength Register
PADS_PCFG0	Peripheral PAD Configuration0 Register
PADS_PCFG1	Peripheral Configuration1 Register
PADS_PORTA0_DS	PORTA 0 to 7 pins DS control
PADS_PORTA1_DS	PORTA 8 - 15 pins DS control
PADS_PORTB0_DS	PORTB 0 to 7 pins DS control
PADS_PORTB1_DS	PORTB 8 - 15 pins DS control
PADS_PORTC0_DS	PORTC 0 to 7 pins DS control
PADS_PORTC1_DS	PORTC 8 - 15 pins DS control
PADS_PORTD0_DS	PORTD 0 to 7 pins DS control
PADS_PORTD1_DS	PORTD 8 to 15 pins DS control
PADS_PORTE0_DS	PORTE 0 to 7 pins DS control
PADS_PORTE1_DS	PORTE 8 to 15 pins DS control
PADS_PORTF0_DS	PORTF 0 to 7 pins DS control
PADS_PORTF1_DS	PORTF 8 to 15 pins DS control
PADS_PORTG0_DS	PORTG 0 to 7 pins DS control
PADS_PORTG1_DS	PORTG 8 to 15 pins DS control
PADS_PORTH0_DS	PORTH 0 to 7 pins DS control
PADS_PORTH1_DS	PORTH 8 to 15 pins DS control
PADS_PORTI0_DS	PORTI 0 to 7 pins DS control
PADS_PORT[n]_PDE	PORT[n] Pull-Down Enable

Table 13-42: ADSP-SC59x PADS Register List (Continued)

Name	Description
PADS_PORT[n]_PUE	PORT[n] Pull-Up Enable

DAI0 1 to 10 pins DS control

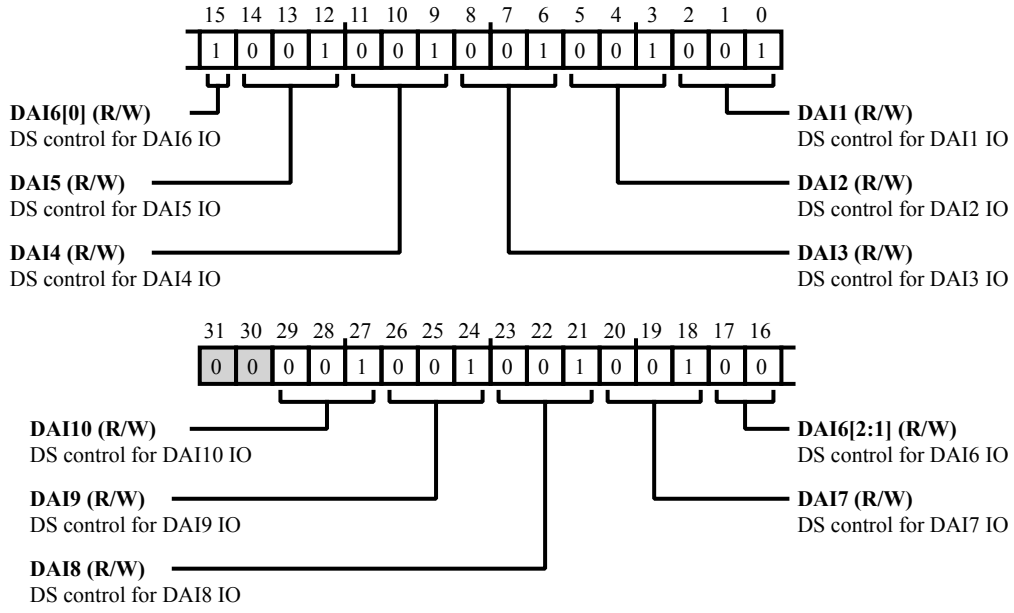


Figure 13-36: PADS_DAI0_0_DS Register Diagram

Table 13-43: PADS_DAI0_0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:27 (R/W)	DAI10	DS control for DAI10 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010. To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
26:24 (R/W)	DAI9	DS control for DAI9 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-43: PADS_DAI0_0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	DAI8	DS control for DAI8 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
20:18 (R/W)	DAI7	DS control for DAI7 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
17:15 (R/W)	DAI6	DS control for DAI6 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
14:12 (R/W)	DAI5	DS control for DAI5 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	DAI4	DS control for DAI4 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-43: PADS_DAI0_0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:6 (R/W)	DAI3	DS control for DAI3 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	DAI2	DS control for DAI2 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
2:0 (R/W)	DAI1	DS control for DAI1 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

DAI0 11 to 20 pins DS control

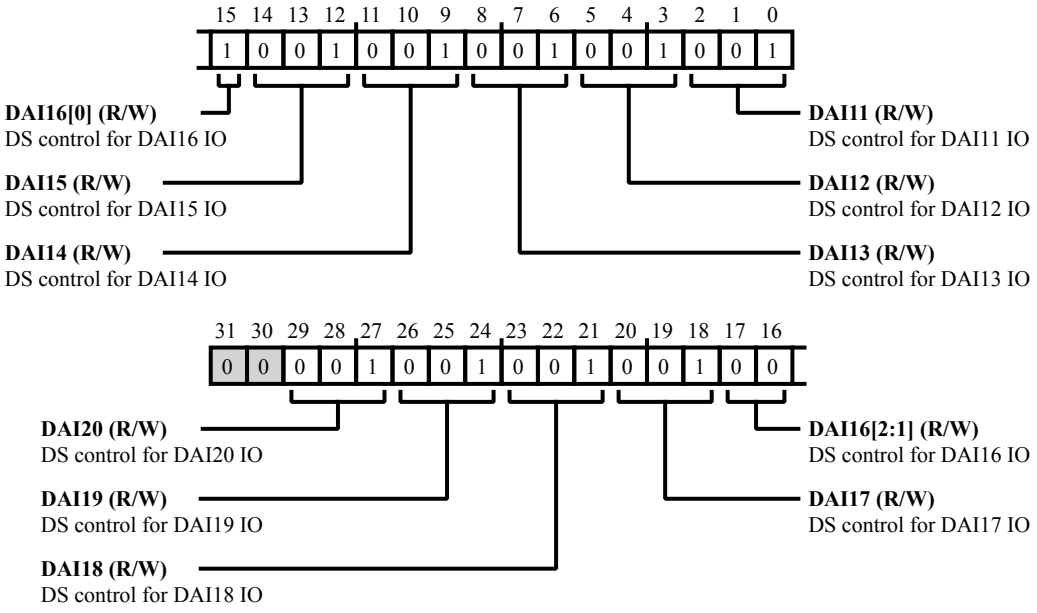


Figure 13-37: PADS_DAI0_1_DS Register Diagram

Table 13-44: PADS_DAI0_1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:27 (R/W)	DAI20	DS control for DAI20 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
26:24 (R/W)	DAI19	DS control for DAI19 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-44: PADS_DAI0_1_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	DAI18	DS control for DAI18 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
20:18 (R/W)	DAI17	DS control for DAI17 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
17:15 (R/W)	DAI16	DS control for DAI16 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
14:12 (R/W)	DAI15	DS control for DAI15 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	DAI14	DS control for DAI14 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-44: PADS_DAI0_1_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:6 (R/W)	DAI13	DS control for DAI13 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	DAI12	DS control for DAI12 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
2:0 (R/W)	DAI11	DS control for DAI11 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

DAI0 Port Input Enable Control Register

The PADS_DAI0_IE register configures input enable control of the DAI0 (20 pins) pads. If =0 implies input buffer disable and if =1 implies enable.

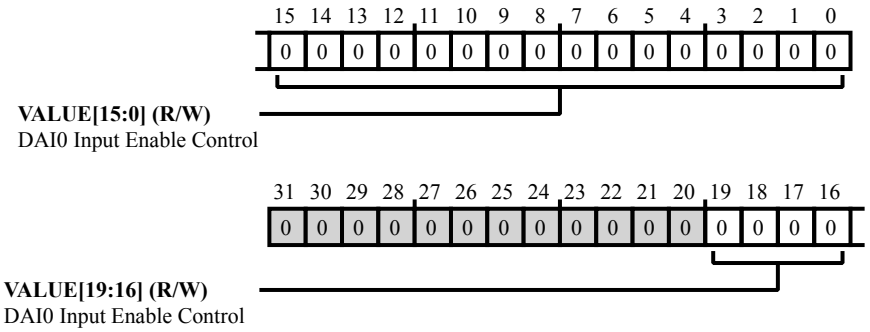


Figure 13-38: PADS_DAI0_IE Register Diagram

Table 13-45: PADS_DAI0_IE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:0 (R/W)	VALUE	DAI0 Input Enable Control.

DAI1 1 to 10 pins DS control

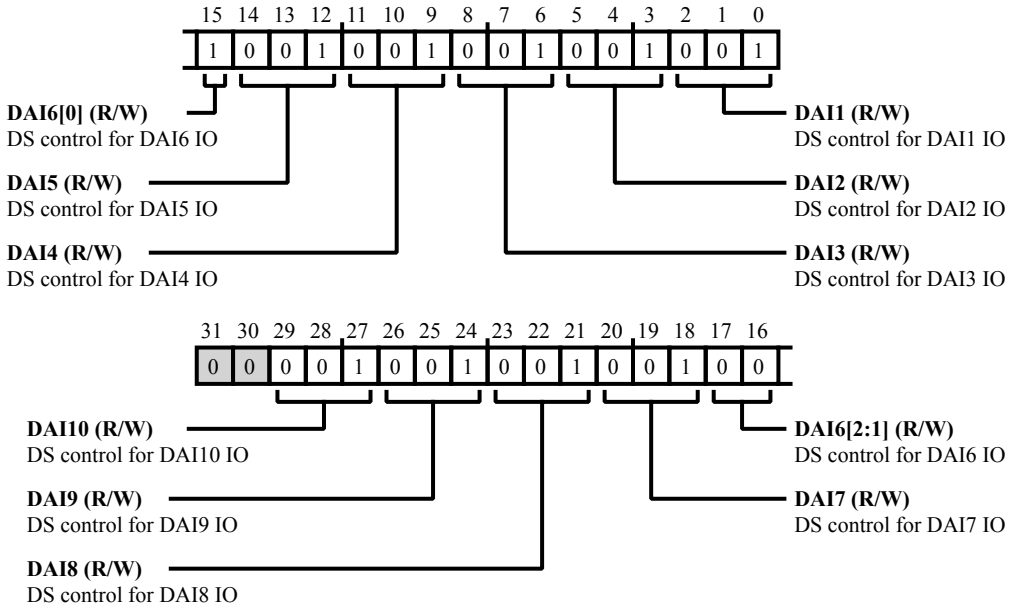


Figure 13-39: PADS_DAI1_0_DS Register Diagram

Table 13-46: PADS_DAI1_0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:27 (R/W)	DAI10	DS control for DAI10 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
26:24 (R/W)	DAI9	DS control for DAI9 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-46: PADS_DAI1_0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	DAI8	DS control for DAI8 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
20:18 (R/W)	DAI7	DS control for DAI7 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
17:15 (R/W)	DAI6	DS control for DAI6 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
14:12 (R/W)	DAI5	DS control for DAI5 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	DAI4	DS control for DAI4 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-46: PADS_DAI1_0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:6 (R/W)	DAI3	DS control for DAI3 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	DAI2	DS control for DAI2 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
2:0 (R/W)	DAI1	DS control for DAI1 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

DAI1 11 to 20 pins DS control

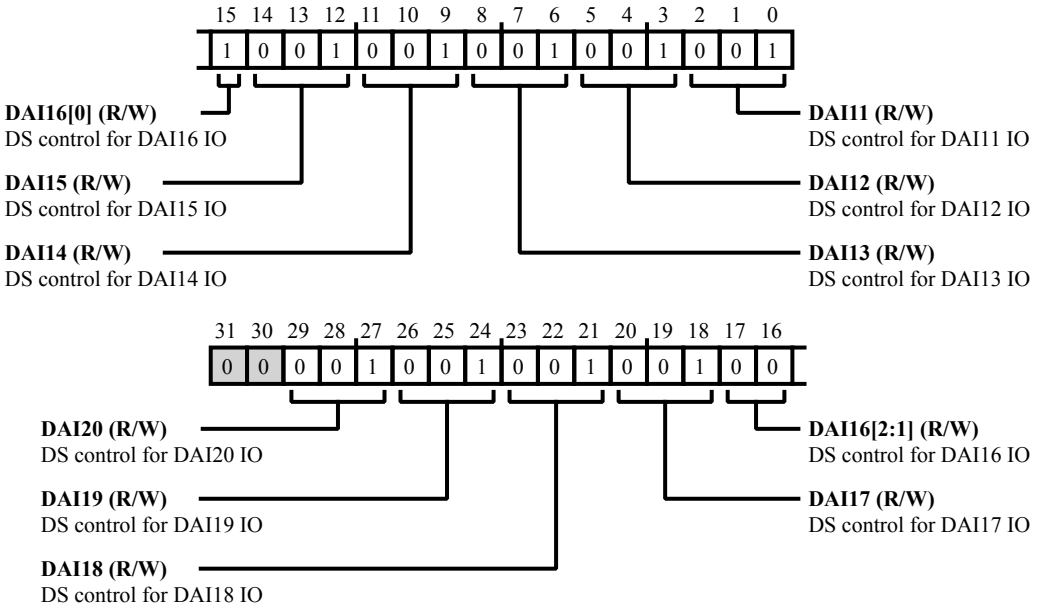


Figure 13-40: PADS_DAI1_1_DS Register Diagram

Table 13-47: PADS_DAI1_1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:27 (R/W)	DAI20	DS control for DAI20 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
26:24 (R/W)	DAI19	DS control for DAI19 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-47: PADS_DAI1_1_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	DAI18	DS control for DAI18 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
20:18 (R/W)	DAI17	DS control for DAI17 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
17:15 (R/W)	DAI16	DS control for DAI16 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
14:12 (R/W)	DAI15	DS control for DAI15 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	DAI14	DS control for DAI14 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-47: PADS_DAI1_1_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:6 (R/W)	DAI13	DS control for DAI13 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	DAI12	DS control for DAI12 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
2:0 (R/W)	DAI11	DS control for DAI11 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

DAI1 Port Input Enable Control Register

The PADS_DAI1_IE register configures input enable control of the DAI1 (20 pins) pads. If =0 implies input buffer disable and if =1 implies enable.

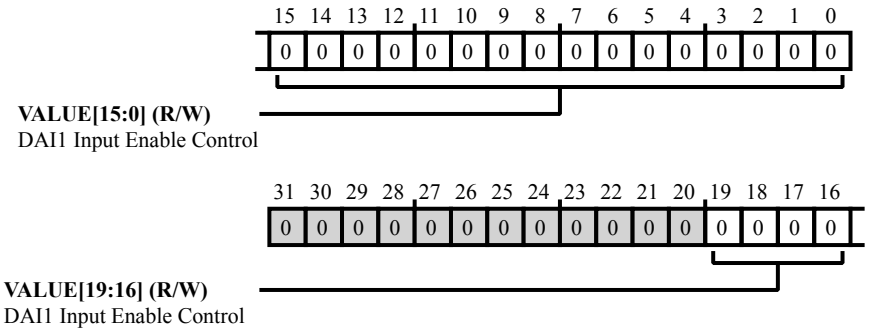


Figure 13-41: PADS_DAI1_IE Register Diagram

Table 13-48: PADS_DAI1_IE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:0 (R/W)	VALUE	DAI1 Input Enable Control.

DAI[n] Pull-Down Enable

Each bit of the PADS_DAI[n]_PDE register is mapped to a corresponding DAI pin pull-down enable.

Do not enable both the pull-up and pull-down at the same time for any pad. Check the Pull-Up Enable register to make sure there is no conflict.

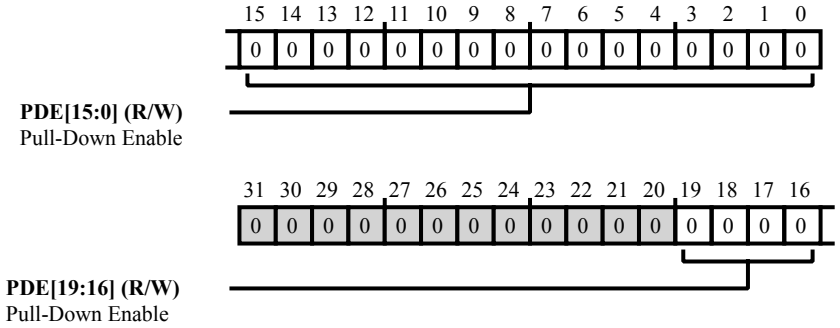


Figure 13-42: PADS_DAI[n]_PDE Register Diagram

Table 13-49: PADS_DAI[n]_PDE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:0 (R/W)	PDE	Pull-Down Enable. Set (=1) the PADS_DAI[n]_PDE.PDE bit to enable the pull-down. Clear (=0) the PADS_DAI[n]_PDE.PDE bit to disable the pull-down.

DAI[n] Pull-Up Enable

The PADS_DAI[n]_PUE register enables the pull-up resistor for DAI pins. Each bit of the PADS_DAI[n]_PUE register is mapped to the corresponding DAI pin pull-up enable.

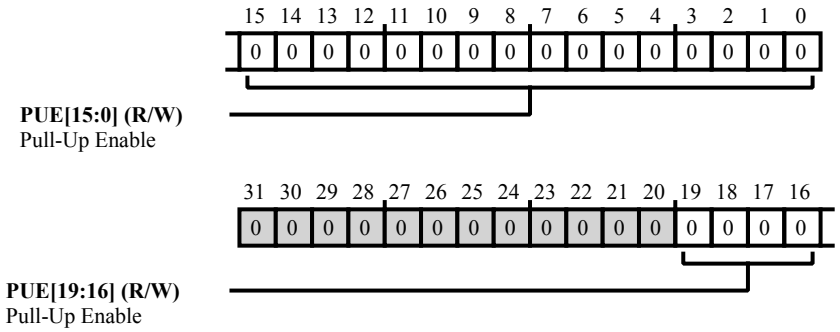


Figure 13-43: PADS_DAI[n]_PUE Register Diagram

Table 13-50: PADS_DAI[n]_PUE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:0 (R/W)	PUE	Pull-Up Enable. Set (=1) the PADS_DAI[n]_PUE.PUE bit to enable the pull-up. Clear (=0) the PADS_DAI[n]_PUE.PUE bit to disable the pull-up.

Non-GPIO Drive Strength Register

The PADS_NONPORTS_DS register sets the drive strength and tolerance for the non-GPIO pins. The drive strength is only controlled for pin groups classified by the interface type.

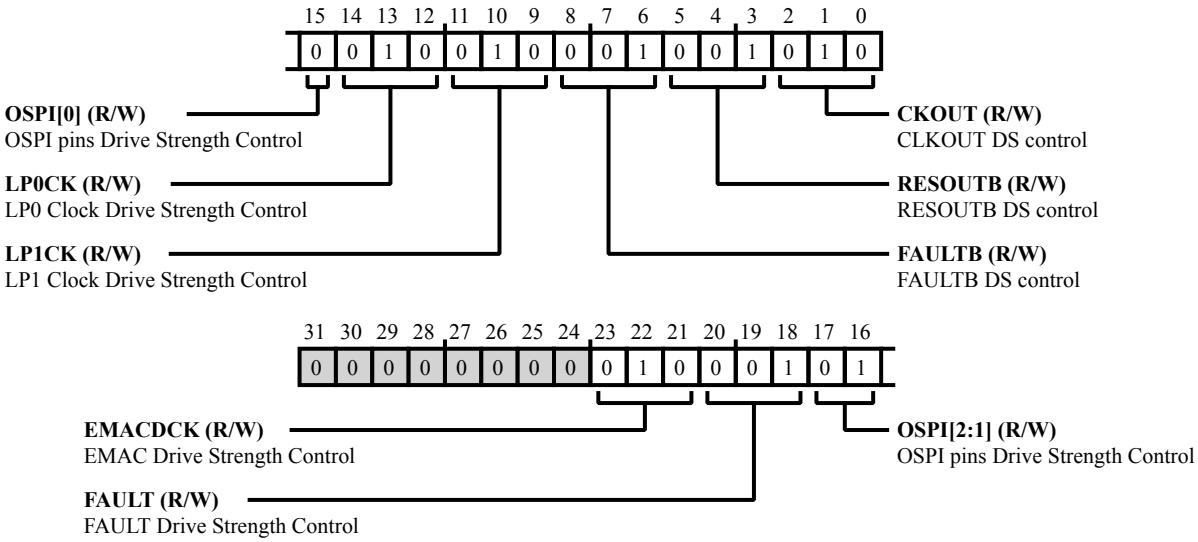


Figure 13-44: PADS_NONPORTS_DS Register Diagram

Table 13-51: PADS_NONPORTS_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	EMACDCK	EMAC Drive Strength Control.	
		1	Drive Strength 001 For an operating frequency less than or equal to 62.5MHz. A drive strength of 000/1xx is invalid.
		2	Drive Strength 010 For an operating frequency more than 62.5MHz. A drive strength of 000/1xx is invalid.
20:18 (R/W)	FAULT	FAULT Drive Strength Control.	
		1	Drive Strength 001 For an operating frequency less than or equal to 62.5MHz. A drive strength of 000/1xx is invalid.
		2	Drive Strength 010 For an operating frequency more than 62.5MHz. A drive strength of 000/1xx is invalid.

Table 13-51: PADS_NONPORTS_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:15 (R/W)	OSPI	OSPI pins Drive Strength Control.	
		1	Drive Strength 001 For an operating frequency less than or equal to 62.5MHz. A drive strength of 000/1xx is invalid.
		2	Drive Strength 010 For an operating frequency more than 62.5MHz. A drive strength of 000/1xx is invalid.
14:12 (R/W)	LP0CK	LP0 Clock Drive Strength Control.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	LP1CK	LP1 Clock Drive Strength Control.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
8:6 (R/W)	FAULTB	FAULTB DS control.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	RESOUTB	RESOUTB DS control.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-51: PADS_NONPORTS_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	CKOUT	CLKOUT DS control.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Peripheral PAD Configuration0 Register

The PADS_PCFG0 register provides several configuration options for the pads and multiplexing for peripherals.

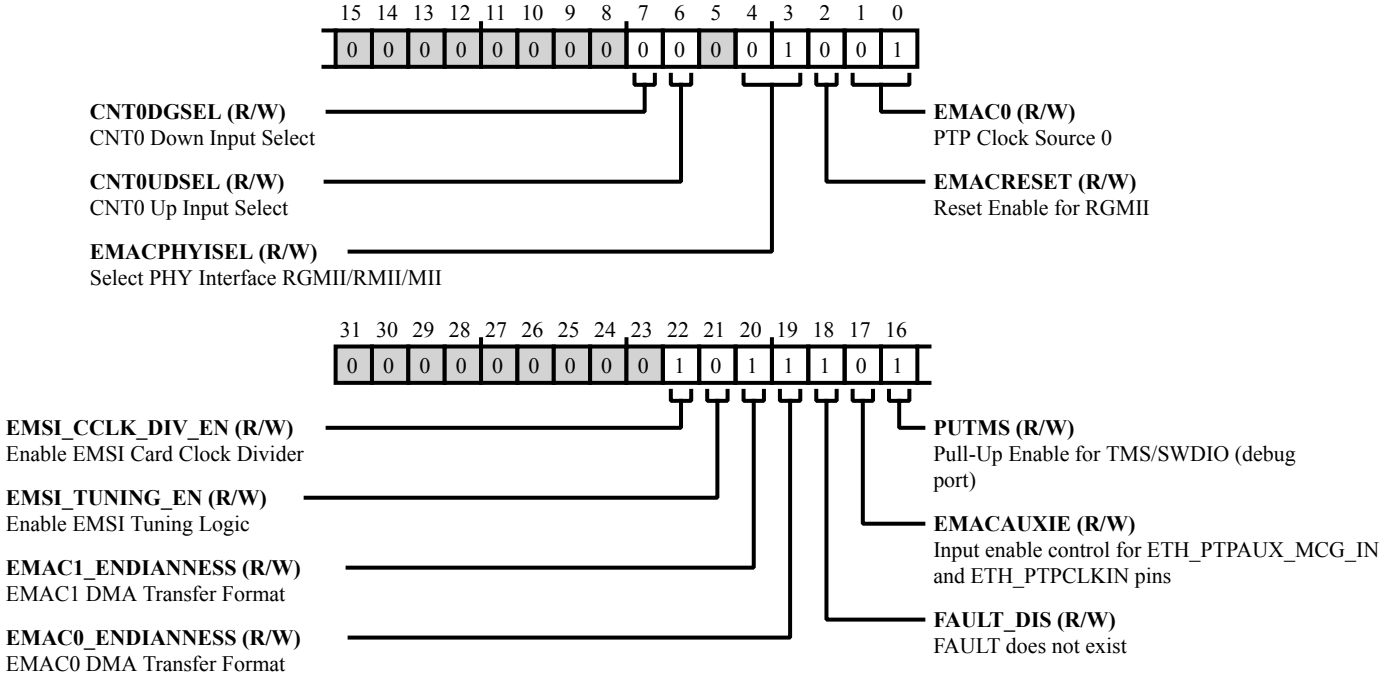


Figure 13-45: PADS_PCFG0 Register Diagram

Table 13-52: PADS_PCFG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	EMSI_CCLK_DIV_EN	Enable EMSI Card Clock Divider.	
21 (R/W)	EMSI_TUNING_EN	Enable EMSI Tuning Logic.	
20 (R/W)	EMAC1_ENDIANNESS	EMAC1 DMA Transfer Format. Configure the PADS_PCFG0.EMAC1_ENDIANNESS bit field to set the EMAC1 DMA transfers to use big-endian or little-endian format.	
		0	Little-Endian
		1	Big-Endian

Table 13-52: PADS_PCFG0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	EMAC0_ENDIANNESS	EMAC0 DMA Transfer Format. Configure the `PADS_PCFG0.EMAC0_ENDIANNESS` bit field to set the EMAC0 DMA transfers to use big-endian or little-endian format.	
		0	Little-Endian
		1	Big-Endian
18 (R/W)	FAULT_DIS	FAULT does not exist.	
		0	Disable input
		1	Enable input
17 (R/W)	EMACAUXIE	Input enable control for ETH_PTPAUX_MCG_IN and ETH_PTPCLKIN pins.	
		0	Disable input
		1	Enable input
16 (R/W)	PUTMS	Pull-Up Enable for TMS/SWDIO (debug port).	
		0	Disable pull-up
		1	Enable pull-up
7 (R/W)	CNT0DGSEL	CNT0 Down Input Select. The `PADS_PCFG0` bit selects the input for the CNT0 counter down and gate.	
		0	GPIO port pins
		1	Trigger connection
6 (R/W)	CNT0UDSEL	CNT0 Up Input Select. The `PADS_PCFG0` bit selects the input for the CNT0 counter up and direction.	
		0	GPIO port pins
		1	Trigger connection
4:3 (R/W)	EMACPHYISEL	Select PHY Interface RGMII/RMII/MII.	
		0	MII Interface Selects PHY Interface MII
		1	RGMII Interface Selects PHY Interface RGMII
		2	RMII Interface Selects PHY Interface RMII
		3	Reserved

Table 13-52: PADS_PCFG0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	EMACRESET	Reset Enable for RGMII. The `PADS_PCFG0.EMACRESET` bit asserts the reset on the RGMII interface. To select the PHY interface (RGMII or RMII), set the EMACPHYISEL bit as required and then set `PADS_PCFG0.EMACRESET`.	
		0	RGMII reset is asserted
		1	RGMII reset reset is deasserted
1:0 (R/W)	EMAC0	PTP Clock Source 0. The `PADS_PCFG0.EMAC0` selects the clock source for the PTP Block in EMAC0.	
		0	REFCLK (MII or RMII) or CLKO7 (RGMII)
		1	SCLK0
		2	External PTP clock
		3	SCLK0

Peripheral Configuration1 Register

The PADS_PCFG1 register provides bits to configure PWM secondary inputs to timers and enable inputs from the TRU.

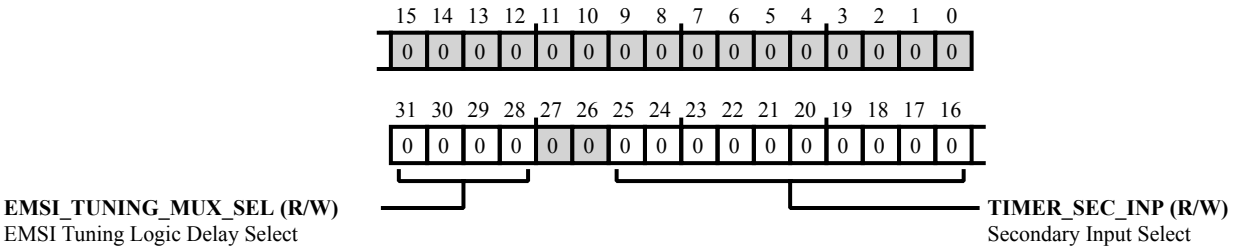


Figure 13-46: PADS_PCFG1 Register Diagram

Table 13-53: PADS_PCFG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/W)	EMSI_TUNING_MUX_SEL	EMSI Tuning Logic Delay Select. Delay Select for EMSI Tuning Logic to Generate Rx Clock.	
		0-3	Reserved
		4-7	Invert cclk_rx
		8	Reserved
		9	Reserved
		10	Reserved
		11	Reserved
		12	Reserved
		13	Reserved
		14	Reserved
		15	Reserved
25:16 (R/W)	TIMER_SEC_INP	Secondary Input Select. The PADS_PCFG1.TIMER_SEC_INP field selects secondary inputs to timers. The enumerated bits below define the source for the timer.	
		1	TIMER0_CLK The TIMER0_CLK source is PB_01 when bit[0] is cleared (=0) The TIMER0_CLK source is PF_05 when this bit is set (=1)

Table 13-53: PADS_PCFG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	TIMER0_ACI0 The TIMER0_ACI0 source is PA_07 when bit[1] is cleared (=0) The TIMER0_ACI0 source is PD_08 when this bit is set (=1)
		4	TIMER0_ACI1 The TIMER0_ACI1 source is PA_14 when bit[2] is cleared (=0) The TIMER0_ACI1 source is PD_04 when this bit is set (=1)
		8	TIMER0_ACI2 The TIMER0_ACI2 source is PB_11 when bit[3] is cleared (=0) The TIMER0_ACI2 source is PD_10 when this bit is set (=1)
		16	TIMER0_ACI3 The TIMER0_ACI3 source is PB_00 when bit[4] is cleared (=0) The TIMER0_ACI3 source is PG_04 when this bit is set (=1)
		32	TIMER0_ACI4 The TIMER0_ACI4 source is PA_11 when bit[5] is cleared (=0) The TIMER0_ACI4 source is PF_15 when this bit is set (=1)
		64	TIMER0_ACLK1 The TIMER0_ACLK1 source is PA_06 when bit[6] is cleared (=0) The TIMER0_ACLK1 source is PG_09 when this bit is set (=1)
		128	TIMER0_ACLK2 The TIMER0_ACLK2 source is PA_08 when bit[7] is cleared (=0) The TIMER0_ACLK2 source is PI_05 when this bit is set (=1)

Table 13-53: PADS_PCFG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		256	TIMER0_ACLK3 The TIMER0_ACLK3 source is PA_02 when bit[8] is cleared (=0) The TIMER0_ACLK3 source is PG_10 when this bit is set (=1)
		512	TIMER0_ACLK4 The TIMER0_ACLK4 source is PB_02 when bit[9] is cleared (=0) The TIMER0_ACLK4 source is PE_03 when this bit is set (=1)

PORTA 0 to 7 pins DS control

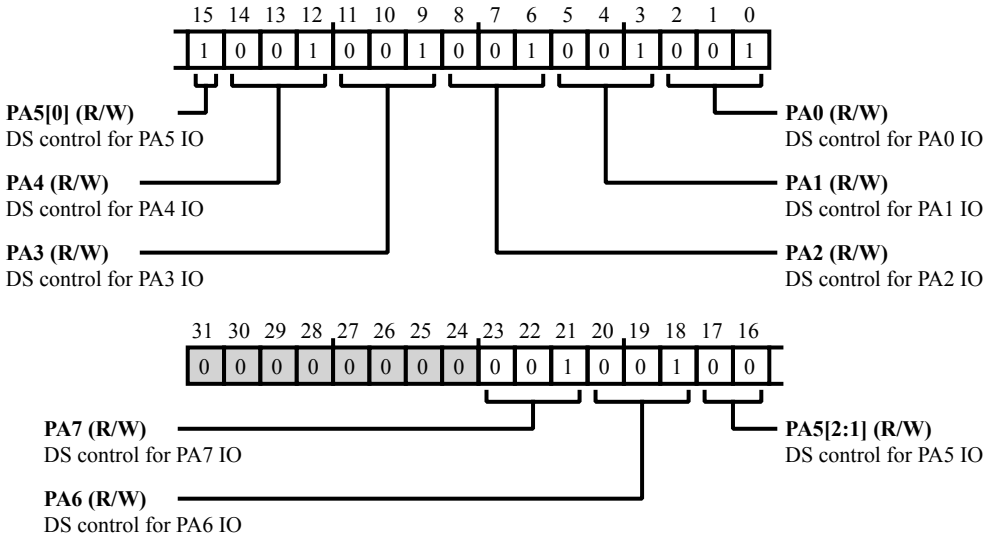


Figure 13-47: PADS_PORTA0_DS Register Diagram

Table 13-54: PADS_PORTA0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	PA7	DS control for PA7 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
20:18 (R/W)	PA6	DS control for PA6 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-54: PADS_PORTA0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:15 (R/W)	PA5	DS control for PA5 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
14:12 (R/W)	PA4	DS control for PA4 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	PA3	DS control for PA3 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
8:6 (R/W)	PA2	DS control for PA2 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	PA1	DS control for PA1 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-54: PADS_PORTA0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	PA0	DS control for PA0 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

PORTA 8 - 15 pins DS control

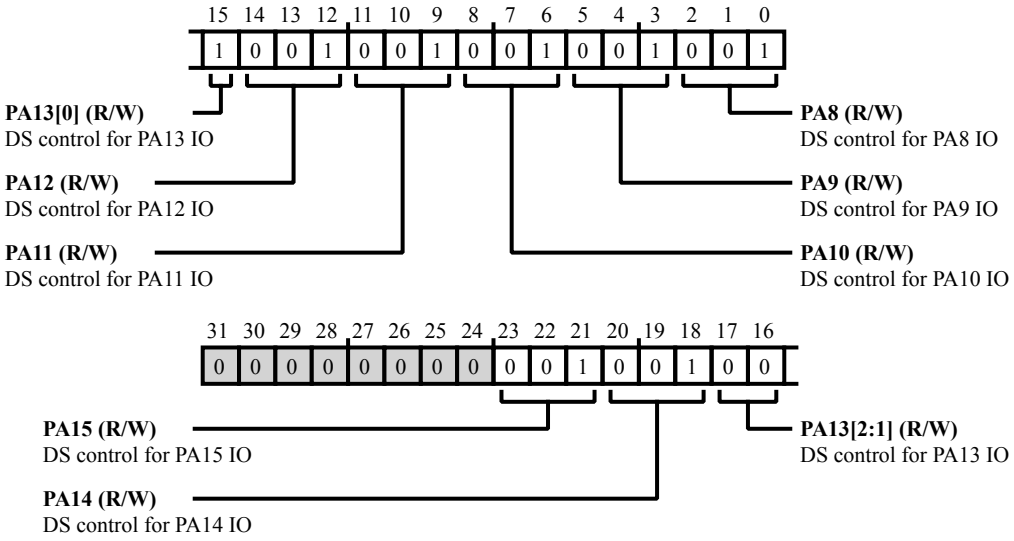


Figure 13-48: PADS_PORTA1_DS Register Diagram

Table 13-55: PADS_PORTA1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	PA15	DS control for PA15 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
20:18 (R/W)	PA14	DS control for PA14 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-55: PADS_PORTA1_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:15 (R/W)	PA13	DS control for PA13 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
14:12 (R/W)	PA12	DS control for PA12 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	PA11	DS control for PA11 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
8:6 (R/W)	PA10	DS control for PA10 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	PA9	DS control for PA9 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-55: PADS_PORTA1_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	PA8	DS control for PA8 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

PORTB 0 to 7 pins DS control

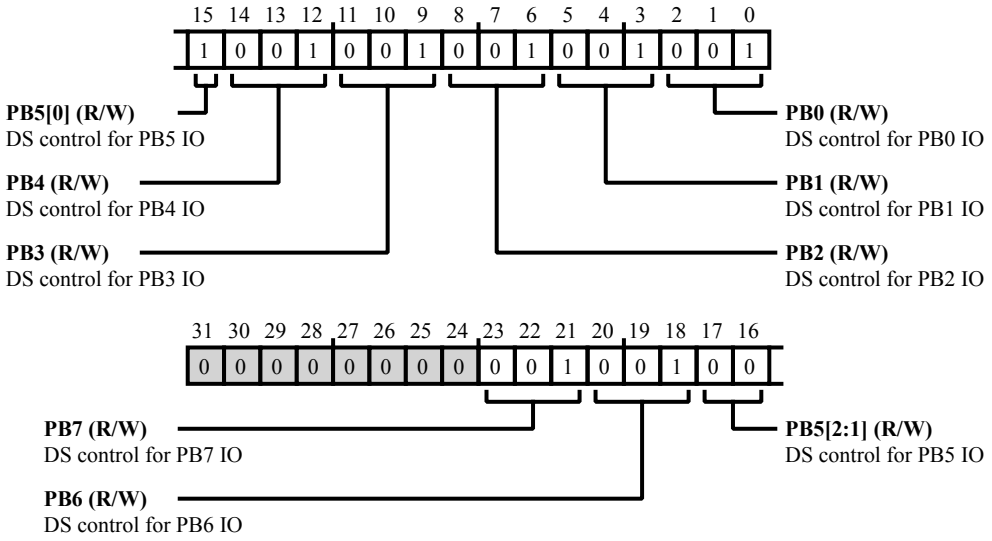


Figure 13-49: PADS_PORTB0_DS Register Diagram

Table 13-56: PADS_PORTB0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	PB7	DS control for PB7 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
20:18 (R/W)	PB6	DS control for PB6 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-56: PADS_PORTB0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:15 (R/W)	PB5	DS control for PB5 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
14:12 (R/W)	PB4	DS control for PB4 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	PB3	DS control for PB3 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
8:6 (R/W)	PB2	DS control for PB2 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	PB1	DS control for PB1 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-56: PADS_PORTB0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	PB0	DS control for PB0 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

PORTB 8 - 15 pins DS control

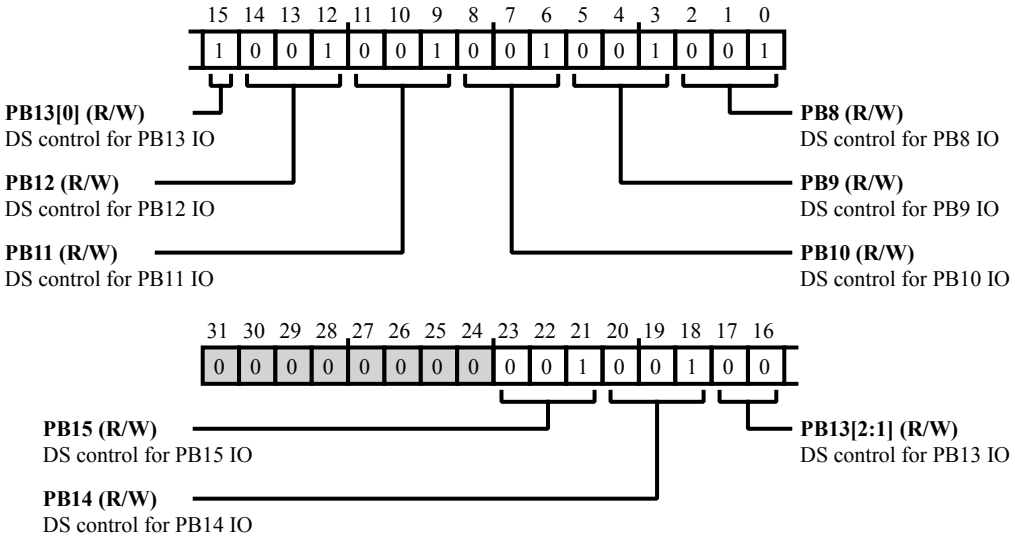


Figure 13-50: PADS_PORTB1_DS Register Diagram

Table 13-57: PADS_PORTB1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	PB15	DS control for PB15 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
20:18 (R/W)	PB14	DS control for PB14 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-57: PADS_PORTB1_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:15 (R/W)	PB13	DS control for PB13 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
14:12 (R/W)	PB12	DS control for PB12 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	PB11	DS control for PB11 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
8:6 (R/W)	PB10	DS control for PB10 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	PB9	DS control for PB9 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-57: PADS_PORTB1_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	PB8	DS control for PB8 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

PORTC 0 to 7 pins DS control

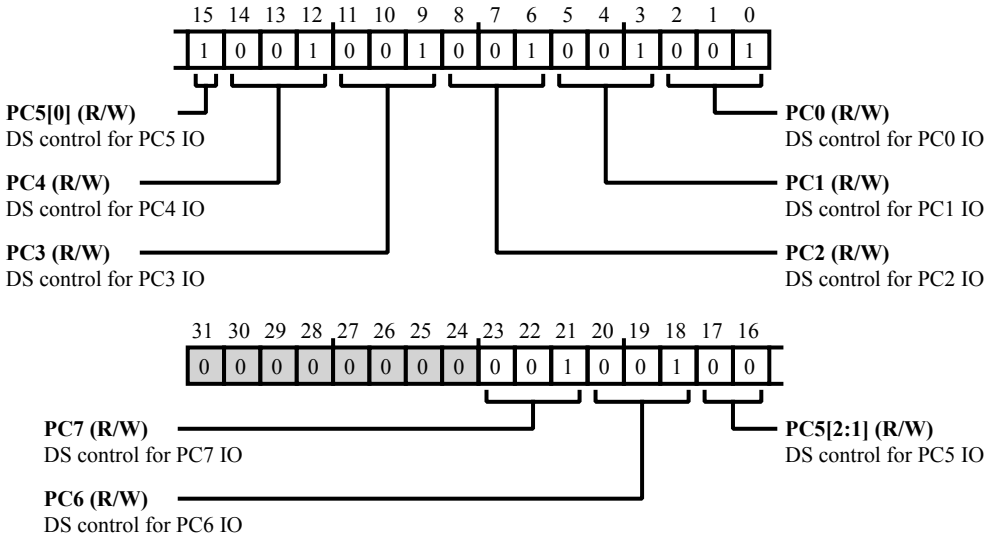


Figure 13-51: PADS_PORTC0_DS Register Diagram

Table 13-58: PADS_PORTC0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:21 (R/W)	PC7	DS control for PC7 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
20:18 (R/W)	PC6	DS control for PC6 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-58: PADS_PORTC0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:15 (R/W)	PC5	DS control for PC5 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
14:12 (R/W)	PC4	DS control for PC4 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
11:9 (R/W)	PC3	DS control for PC3 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
8:6 (R/W)	PC2	DS control for PC2 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid
5:3 (R/W)	PC1	DS control for PC1 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

Table 13-58: PADS_PORTC0_DS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	PC0	DS control for PC0 IO.	
		1	Drive strength 001. To be used for operating frequency less than or equal to 62.5MHz. Drive Strength 000/1xx are invalid
		2	Drive strength 010.To be used for operating frequency more than 62.5MHz. Drive Strength 000/1xx are invalid

PORTC 8 - 15 pins DS control

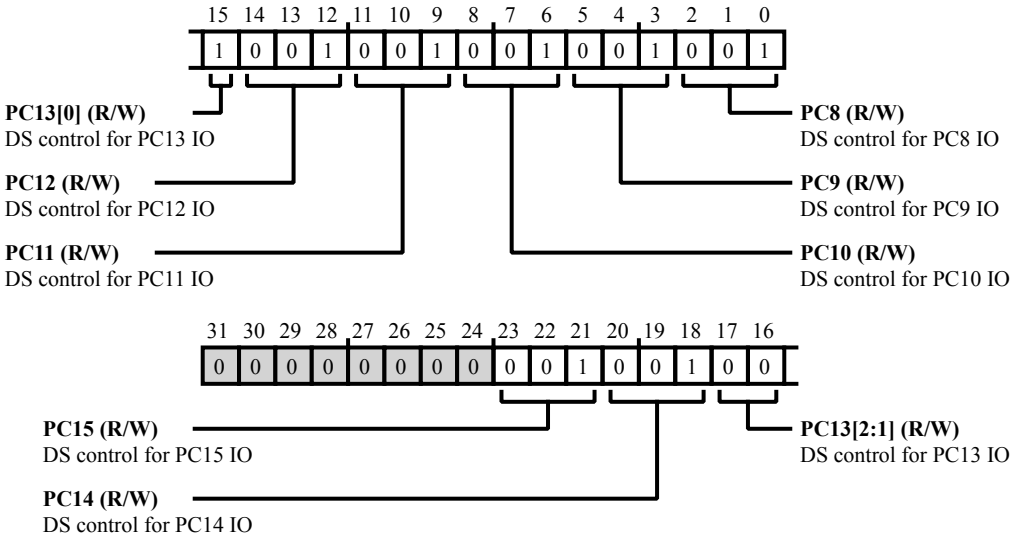


Figure 13-52: PADS_PORTC1_DS Register Diagram

Table 13-59: PADS_PORTC1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PC15	DS control for PC15 IO.
20:18 (R/W)	PC14	DS control for PC14 IO.
17:15 (R/W)	PC13	DS control for PC13 IO.
14:12 (R/W)	PC12	DS control for PC12 IO.
11:9 (R/W)	PC11	DS control for PC11 IO.
8:6 (R/W)	PC10	DS control for PC10 IO.
5:3 (R/W)	PC9	DS control for PC9 IO.
2:0 (R/W)	PC8	DS control for PC8 IO.

PORTD 0 to 7 pins DS control

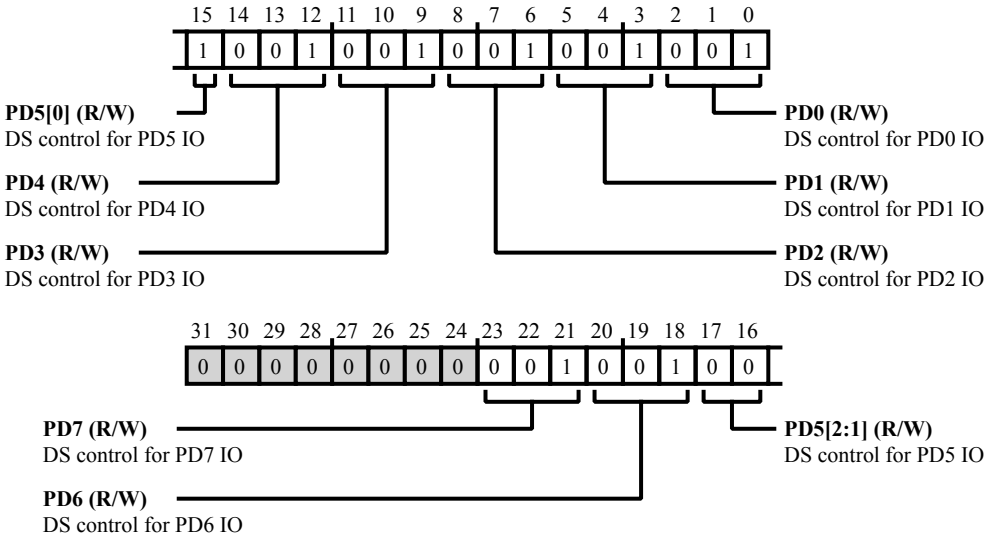


Figure 13-53: PADS_PORTD0_DS Register Diagram

Table 13-60: PADS_PORTD0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PD7	DS control for PD7 IO.
20:18 (R/W)	PD6	DS control for PD6 IO.
17:15 (R/W)	PD5	DS control for PD5 IO.
14:12 (R/W)	PD4	DS control for PD4 IO.
11:9 (R/W)	PD3	DS control for PD3 IO.
8:6 (R/W)	PD2	DS control for PD2 IO.
5:3 (R/W)	PD1	DS control for PD1 IO.
2:0 (R/W)	PD0	DS control for PD0 IO.

PORTD 8 to 15 pins DS control

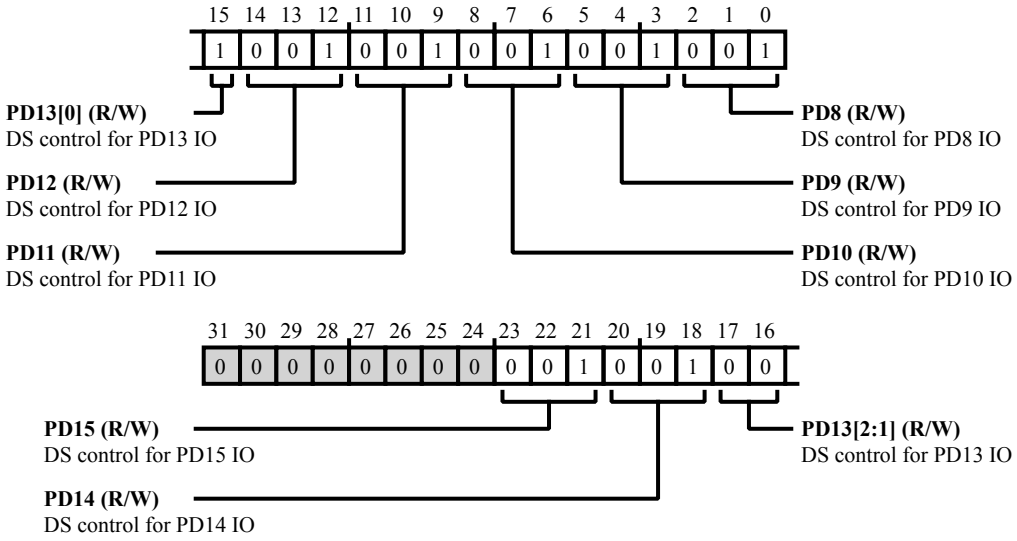


Figure 13-54: PADS_PORTD1_DS Register Diagram

Table 13-61: PADS_PORTD1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PD15	DS control for PD15 IO.
20:18 (R/W)	PD14	DS control for PD14 IO.
17:15 (R/W)	PD13	DS control for PD13 IO.
14:12 (R/W)	PD12	DS control for PD12 IO.
11:9 (R/W)	PD11	DS control for PD11 IO.
8:6 (R/W)	PD10	DS control for PD10 IO.
5:3 (R/W)	PD9	DS control for PD9 IO.
2:0 (R/W)	PD8	DS control for PD8 IO.

PORTE 0 to 7 pins DS control

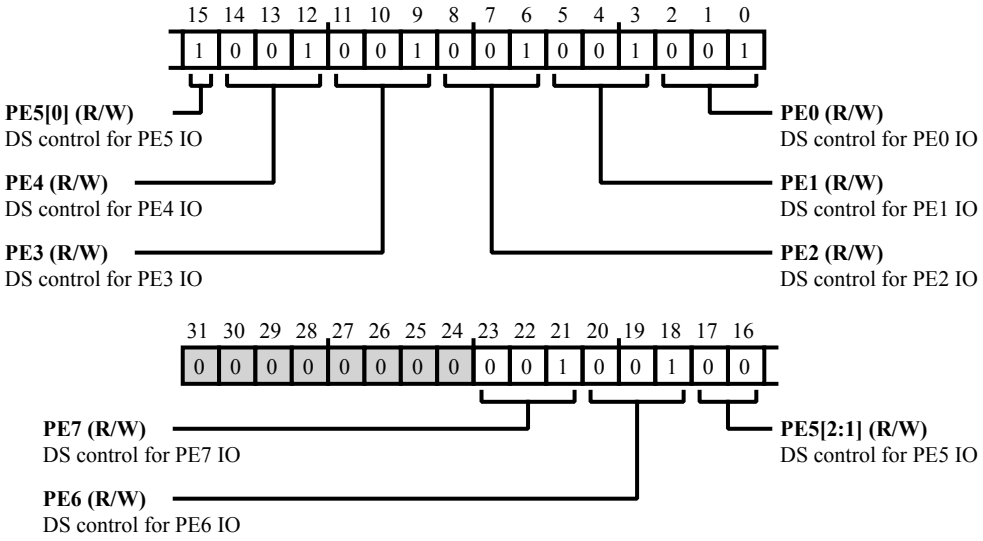


Figure 13-55: PADS_PORTE0_DS Register Diagram

Table 13-62: PADS_PORTE0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PE7	DS control for PE7 IO.
20:18 (R/W)	PE6	DS control for PE6 IO.
17:15 (R/W)	PE5	DS control for PE5 IO.
14:12 (R/W)	PE4	DS control for PE4 IO.
11:9 (R/W)	PE3	DS control for PE3 IO.
8:6 (R/W)	PE2	DS control for PE2 IO.
5:3 (R/W)	PE1	DS control for PE1 IO.
2:0 (R/W)	PE0	DS control for PE0 IO.

PORTE 8 to 15 pins DS control

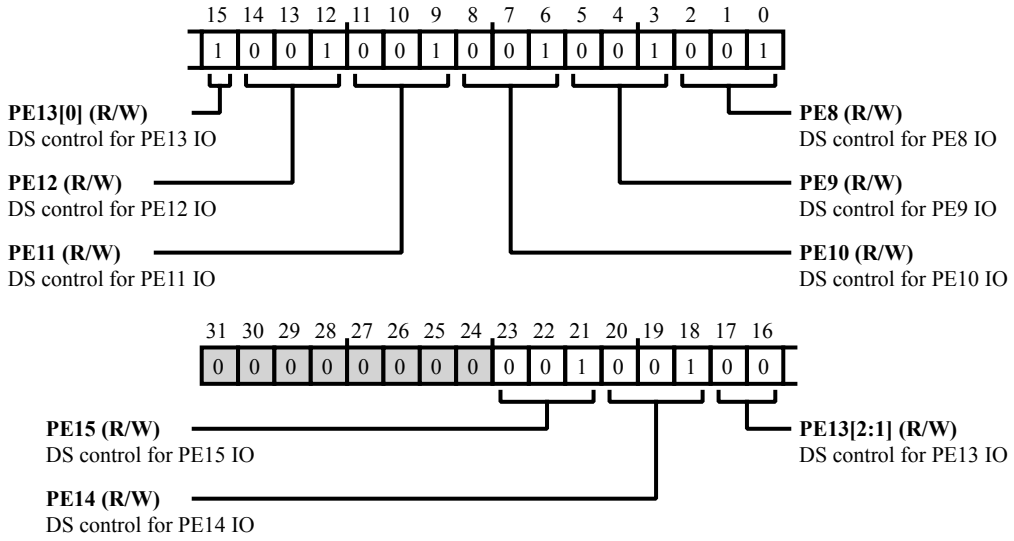


Figure 13-56: PADS_PORTE1_DS Register Diagram

Table 13-63: PADS_PORTE1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PE15	DS control for PE15 IO.
20:18 (R/W)	PE14	DS control for PE14 IO.
17:15 (R/W)	PE13	DS control for PE13 IO.
14:12 (R/W)	PE12	DS control for PE12 IO.
11:9 (R/W)	PE11	DS control for PE11 IO.
8:6 (R/W)	PE10	DS control for PE10 IO.
5:3 (R/W)	PE9	DS control for PE9 IO.
2:0 (R/W)	PE8	DS control for PE8 IO.

PORTF 0 to 7 pins DS control

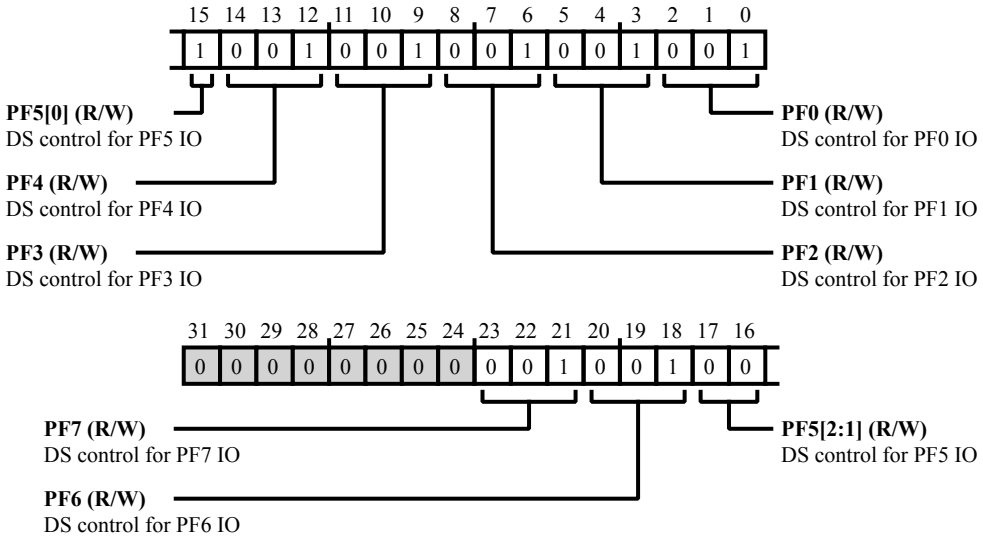


Figure 13-57: PADS_PORTF0_DS Register Diagram

Table 13-64: PADS_PORTF0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PF7	DS control for PF7 IO.
20:18 (R/W)	PF6	DS control for PF6 IO.
17:15 (R/W)	PF5	DS control for PF5 IO.
14:12 (R/W)	PF4	DS control for PF4 IO.
11:9 (R/W)	PF3	DS control for PF3 IO.
8:6 (R/W)	PF2	DS control for PF2 IO.
5:3 (R/W)	PF1	DS control for PF1 IO.
2:0 (R/W)	PF0	DS control for PF0 IO.

PORTF 8 to 15 pins DS control

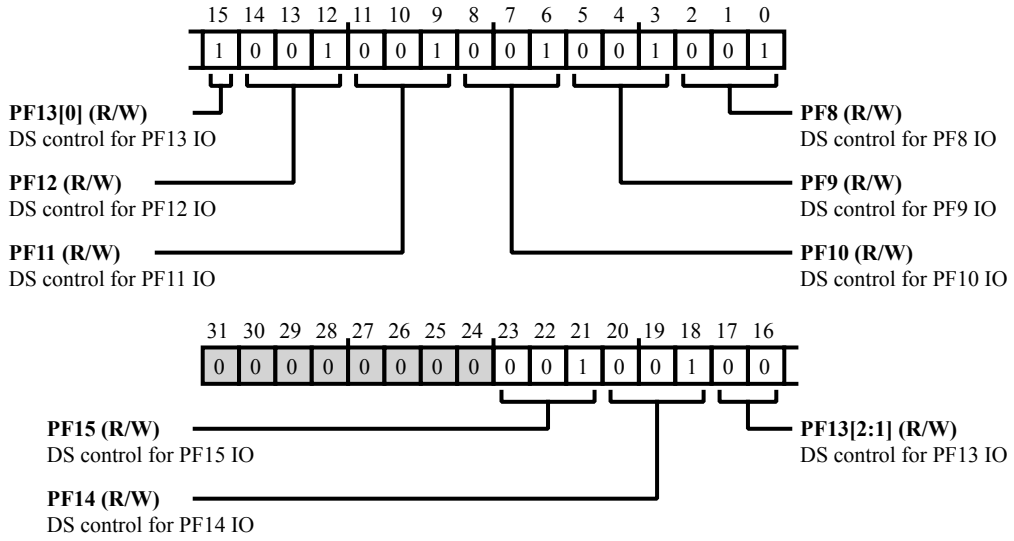


Figure 13-58: PADS_PORTF1_DS Register Diagram

Table 13-65: PADS_PORTF1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PF15	DS control for PF15 IO.
20:18 (R/W)	PF14	DS control for PF14 IO.
17:15 (R/W)	PF13	DS control for PF13 IO.
14:12 (R/W)	PF12	DS control for PF12 IO.
11:9 (R/W)	PF11	DS control for PF11 IO.
8:6 (R/W)	PF10	DS control for PF10 IO.
5:3 (R/W)	PF9	DS control for PF9 IO.
2:0 (R/W)	PF8	DS control for PF8 IO.

PORTG 0 to 7 pins DS control

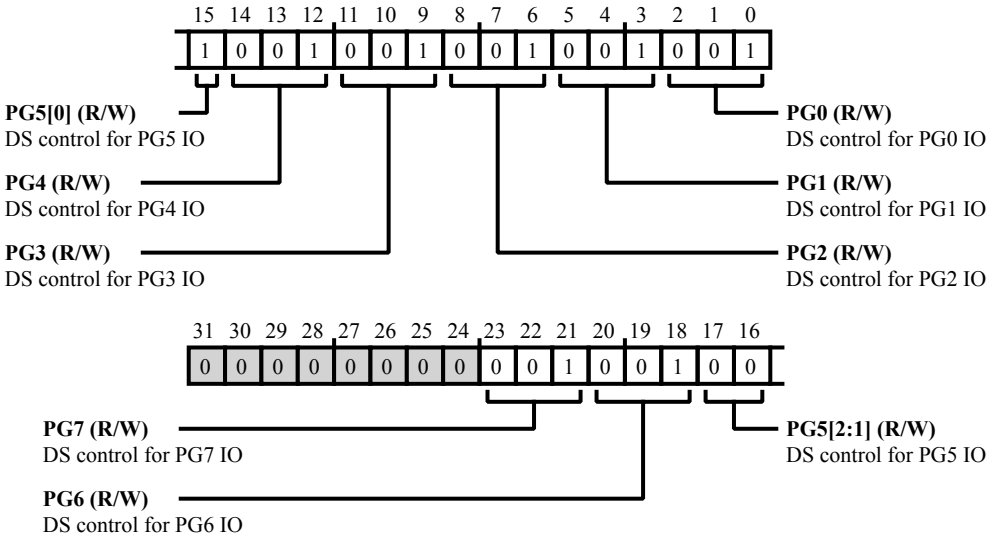


Figure 13-59: PADS_PORTG0_DS Register Diagram

Table 13-66: PADS_PORTG0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PG7	DS control for PG7 IO.
20:18 (R/W)	PG6	DS control for PG6 IO.
17:15 (R/W)	PG5	DS control for PG5 IO.
14:12 (R/W)	PG4	DS control for PG4 IO.
11:9 (R/W)	PG3	DS control for PG3 IO.
8:6 (R/W)	PG2	DS control for PG2 IO.
5:3 (R/W)	PG1	DS control for PG1 IO.
2:0 (R/W)	PG0	DS control for PG0 IO.

PORTG 8 to 15 pins DS control

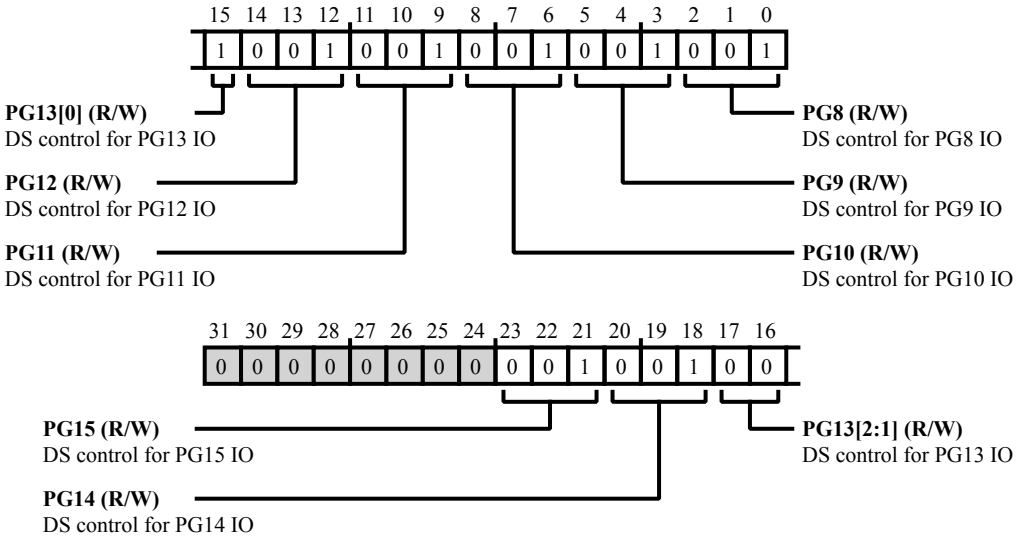


Figure 13-60: PADS_PORTG1_DS Register Diagram

Table 13-67: PADS_PORTG1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PG15	DS control for PG15 IO.
20:18 (R/W)	PG14	DS control for PG14 IO.
17:15 (R/W)	PG13	DS control for PG13 IO.
14:12 (R/W)	PG12	DS control for PG12 IO.
11:9 (R/W)	PG11	DS control for PG11 IO.
8:6 (R/W)	PG10	DS control for PG10 IO.
5:3 (R/W)	PG9	DS control for PG9 IO.
2:0 (R/W)	PG8	DS control for PG8 IO.

PORTH 0 to 7 pins DS control

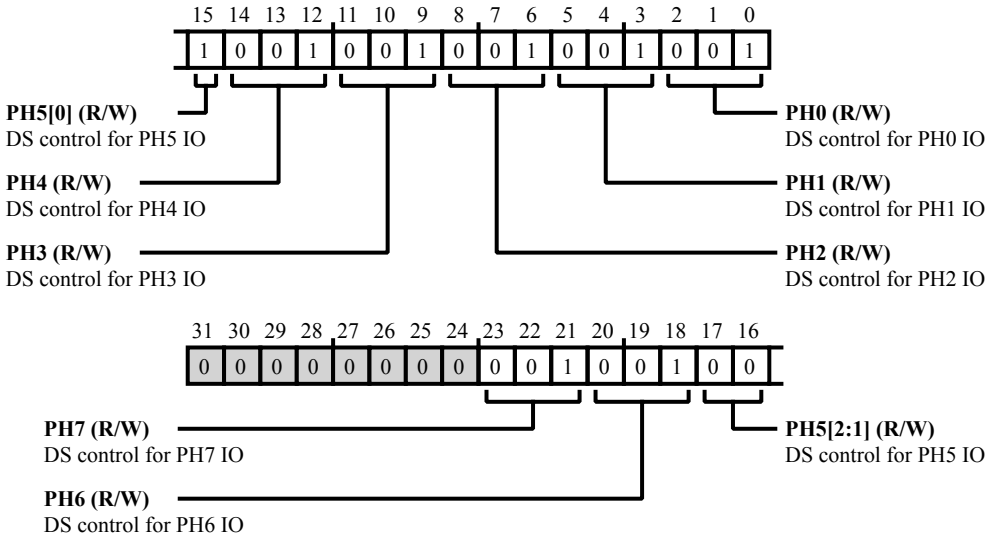


Figure 13-61: PADS_PORTH0_DS Register Diagram

Table 13-68: PADS_PORTH0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PH7	DS control for PH7 IO.
20:18 (R/W)	PH6	DS control for PH6 IO.
17:15 (R/W)	PH5	DS control for PH5 IO.
14:12 (R/W)	PH4	DS control for PH4 IO.
11:9 (R/W)	PH3	DS control for PH3 IO.
8:6 (R/W)	PH2	DS control for PH2 IO.
5:3 (R/W)	PH1	DS control for PH1 IO.
2:0 (R/W)	PH0	DS control for PH0 IO.

PORTH 8 to 15 pins DS control

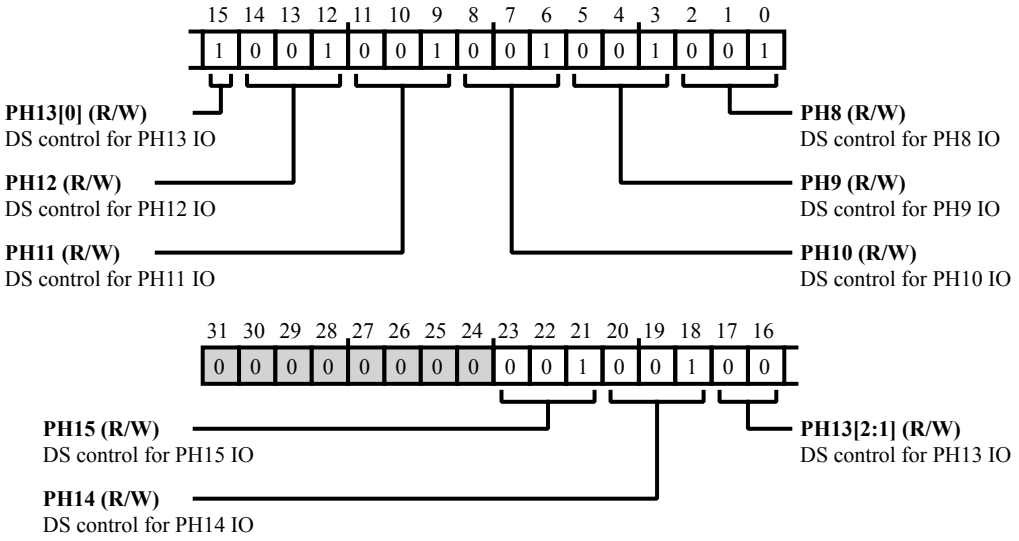


Figure 13-62: PADS_PORTH1_DS Register Diagram

Table 13-69: PADS_PORTH1_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PH15	DS control for PH15 IO.
20:18 (R/W)	PH14	DS control for PH14 IO.
17:15 (R/W)	PH13	DS control for PH13 IO.
14:12 (R/W)	PH12	DS control for PH12 IO.
11:9 (R/W)	PH11	DS control for PH11 IO.
8:6 (R/W)	PH10	DS control for PH10 IO.
5:3 (R/W)	PH9	DS control for PH9 IO.
2:0 (R/W)	PH8	DS control for PH8 IO.

PORTI 0 to 7 pins DS control

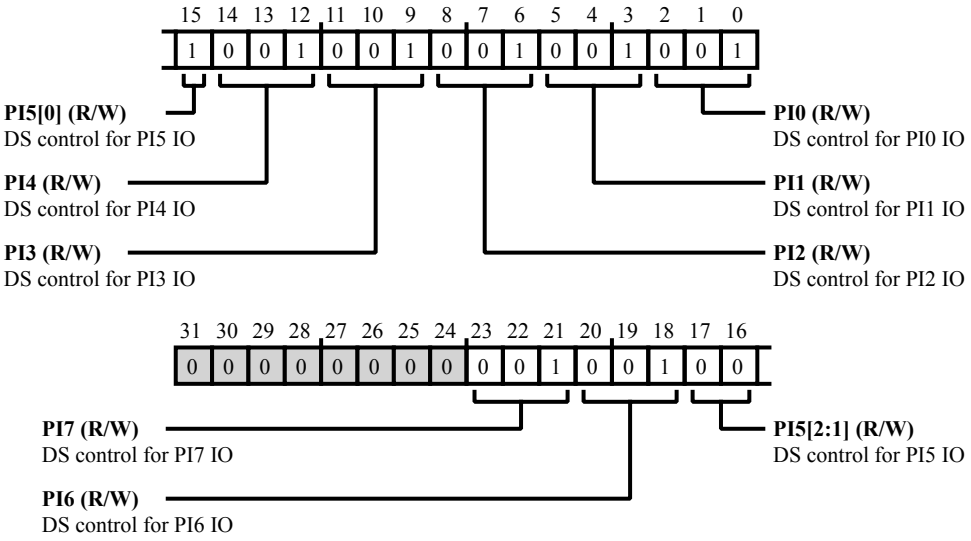


Figure 13-63: PADS_PORTI0_DS Register Diagram

Table 13-70: PADS_PORTI0_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:21 (R/W)	PI7	DS control for PI7 IO.
20:18 (R/W)	PI6	DS control for PI6 IO.
17:15 (R/W)	PI5	DS control for PI5 IO.
14:12 (R/W)	PI4	DS control for PI4 IO.
11:9 (R/W)	PI3	DS control for PI3 IO.
8:6 (R/W)	PI2	DS control for PI2 IO.
5:3 (R/W)	PI1	DS control for PI1 IO.
2:0 (R/W)	PI0	DS control for PI0 IO.

PORT[n] Pull-Down Enable

Each bit of the PADS_PORT[n]_PDE register is mapped to a corresponding pull-down enable.

Do not enable both the pull-up and pull-down at the same time for any pad. Check the Pull-Up Enable register to make sure there is no conflict.

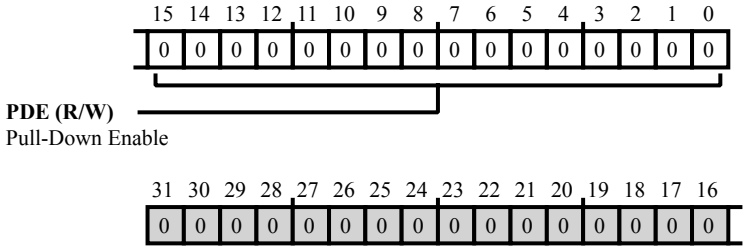


Figure 13-64: PADS_PORT[n]_PDE Register Diagram

Table 13-71: PADS_PORT[n]_PDE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	PDE	Pull-Down Enable. Set (=1) the PADS_PORT[n]_PDE.PDE bit to enable the pull-down. Clear (=0) the PADS_PORT[n]_PDE.PDE bit to disable the pull-down.

PORT[n] Pull-Up Enable

Each bit of the PADS_PORT[n]_PUE register is mapped to a corresponding pull-up enable.

Do not enable both the pull-up and pull-down at the same time for any pad. Check the Pull-Down Enable register to make sure there is no conflict.

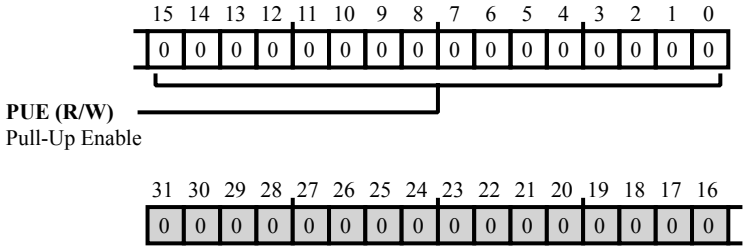


Figure 13-65: PADS_PORT[n]_PUE Register Diagram

Table 13-72: PADS_PORT[n]_PUE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	PUE	Pull-Up Enable. Set (=1) the PADS_PORT[n]_PUE.PUE bit to enable the pull-up. Clear (=0) the PADS_PORT[n]_PUE.PUE bit to disable the pull-up.

14 Thermal Monitoring Unit (TMU)

The TMU provides on-chip temperature measurement which is important in applications that have substantial power consumption. The TMU is integrated into the processor die and digital infrastructure using an MMR-based system access to measure the die temperature variations in real-time.

TMU Features

The TMU supports the following features:

- On-chip temperature sensing
- Programmable over-temperature and under-temperature limits
- Programmable conversion rate
- Averaging feature available
- Temperature gain and offset correction options
- Uses dedicated channel of HADC for conversion
- Programmable blanking and refresh period for TMU conversion

TMU Functional Description

Following sections provide the functional description of TMU.

ADSP-SC59x TMU Register List

Thermal monitoring unit

Table 14-1: ADSP-SC59x TMU Register List

Name	Description
TMU_ALRT_LIM_HI	Alert High Limit Register
TMU_ALRT_LIM_LO	Alert Low Limit Register
TMU_AVG	Averaging Register

Table 14-1: ADSP-SC59x TMU Register List (Continued)

Name	Description
TMU_CNV_BLANK	Temperature Conversion Blank Register
TMU_CTL	TMU Control Register
TMU_FLT_LIM_HI	Fault High Limit Register
TMU_FLT_LIM_LO	Fault Low Limit Register
TMU_GAIN	Gain Value Register
TMU_IMSK	Interrupt Mask Register
TMU_OFFSET	Offset Register
TMU_REFR_CNTR	Temperature Refresh Counter
TMU_STAT	Status Register
TMU_TEMP	Temperature Value Register

ADSP-SC59x TMU Interrupt List

Table 14-2: ADSP-SC59x TMU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
7	TMU0_FAULT	TMU0 Fault		
8	TMU0_ALERT	TMU0 Fault		

ADSP-SC59x TMU Trigger List

Table 14-3: ADSP-SC59x TMU Trigger List Generators

Trigger ID	Name	Description	Sensitivity
153	TMU0_FAULT	TMU0 TM0 Fault Event	
154	TMU0_ALERT	TMU0 TM0 Alert Event	

Table 14-4: ADSP-SC59x TMU Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

TMU Definitions

The following definitions are helpful when using the TMU module.

Thermal Diode

A special type of diode whose electrical properties change with the temperature in a defined way.

DTM

Dynamic thermal management is a set of techniques that adapt the run-time behavior of a processor to achieve the highest performance under thermal constraints.

TMU Block Diagram

The ***Thermal Monitoring Unit Block Diagram*** shows the main block inside TMU.

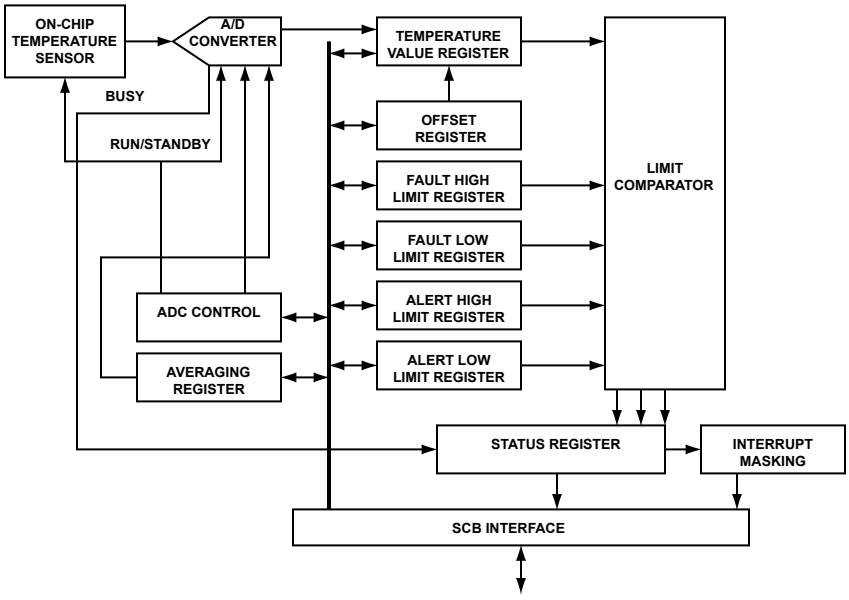


Figure 14-1: Thermal Monitoring Unit Block Diagram

TMU Architectural Concepts

The following sections provide information on the architecture and system integration of the TMU.

System Integration

Dynamic thermal management (DTM) techniques adapt the run-time behavior of a processor to achieve the highest performance under thermal constraints. One of the most important aspects of DTM is to capture the run-time variations in the temperature caused by power consumption variations due to workload changes.

The accuracy of thermal measurements directly affects the efficiency of thermal management as well as the performance of the processor. Temperature estimations lower or higher than the actual temperature may cause late or early activation of DTM techniques.

- Late activation of DTM can result in degraded reliability because the temperature may exceed the designated thresholds.
- Early activation of DTM can have significant impact on performance.

The TMU is an analog thermal sensor that consists of a temperature-sensing diode, a calibrated reference current source, and a current comparator. The sensor placement error is one of the most important sources of inaccuracy in values obtained from thermal sensors.

The TMU module provides the temperature monitoring capability to the chip and implements thermal management in the end system. It provides many features which ensure the minimum load on the software and also minimal or no external components for a flexible temperature monitoring system.

Digital Thermometer

The TMU functions as a digital thermometer with an MMR-based system access. The on-die temperature value is measured and digitized through an A/D converter. The temperature value is stored and updated periodically in an MMR register. The TMU can be configured to generate an interrupt as it crosses the upper temperature limit (`TMU_FLT_LIM_HI` register). A thermal event is also routed to the interrupt port of the sensor. The event is then routed to the SEC, to ensure that core intervention is not required in the event of overheating.

Temperature Sensor Averaging

The temperature sensor averaging feature enhances the accuracy of the temperature measurements. To enable this feature, the `TMU_AVG` bits must be enabled in the control register after power-up. In this mode, the averaging reduces the effect of noise on the temperature result. The temperature is measured each time a conversion is performed. A moving average method is used to determine the result in the temperature value register. The total time to measure a temperature channel is typically 1 ms.

TMU and HADC

The TMU modules uses one of the ADC channels to digitize the temperature reading. Therefore, the HADC module is shared by the TMU. If the TMU is not enabled, the HADC operates at its full throughput. TMU can be enabled periodically or permanently. In periodic enable mode, it is enabled once in a specific refresh period. After enabling, the TMU measures the temperature for a specific blanking period. During the blanking period, the TMU uses the ADC for temperature measurement, so that the ADC is not available for data conversion.

The typical value of blanking period is 21×10^4 system clock (SCLK0) cycles

The typical value of refresh period is 21×10^6 SCLK0 cycles

So, for every 21×10^6 cycles, 21×10^4 cycles are used for temperature measurement. Program these values for blanking period and refresh period through the TMU registers.

When TMU is enabled, the HADC modules is not available for external input conversion. This state is indicated by the `HADC_STAT.TMUHADC_BUSY` status bit. All the requests for HADC conversion are ignored when `HADC_STAT.TMUHADC_BUSY` bit is high.

In the periodic enable mode, the ongoing HADC conversion is completed activating the TMU to measure temperature as shown. If HADC is in autoscan mode, the TMU is activated after completing the ongoing sequence. If HADC is in fixed-conversion mode, all the fixed conversions are completed and followed by TMU activation.

TMU Event Control

The TMU generates different events depending on the state of the TMU temperature measurement and the different thresholds set in thresh hold registers. These events are reported in the TMU_STAT register as shown below. It can generate an event for each of following conditions.

- The fault high limit is configured in the TMU_FLT_LIM_HI register. The interrupt is generated when the temperature value is greater than or equal to this value.
- The alert high limit is configured in the TMU_ALRT_LIM_HI register. The interrupt is generated when the temperature value is greater than or equal to this value.
- The fault low limit is configured in the TMU_FLT_LIM_LO register. The TMU_STAT.FLTLO status bit is set when the temperature value is less than or equal to this value.
- The alert low limit is configured in the TMU_ALRT_LIM_LO register. The TMU_STAT.FLTLO status bit is set when the temperature value is less than or equal to this value.

Interrupts and status conditions can be masked (disabled) or unmasked (enabled) by setting and clearing bits in the TMU_IMSK register.

NOTE: Only TMU_FLT_LIM_HI and TMU_ALRT_LIM_HI generate an interrupt to core. Other events only result in status change in the TMU_STAT register.

Status and Error Signals

When the measured temperature value exceeds the high or low limits that are configured in the TMU_FLT_LIM_HI/TMU_FLT_LIM_LO and the TMU_ALRT_LIM_HI/TMU_ALRT_LIM_LO registers, corresponding thermal events are triggered.

The fault and alert events are sent to the core through the status register (TMU_STAT). Both the TMU_STAT.FLTHI and TMU_STAT.ALRTHI bits are sticky bits and are cleared by a W1C operation by the core.

The alert and fault registers can be programmed through the MMR bus interface as shown in the TMU block diagram. A write into the TMU_FLT_LIM_HI/TMU_FLT_LIM_LO and TMU_ALRT_LIM_HI/TMU_ALRT_LIM_LO registers or the TMU_TEMP register is not allowed when the events are being triggered.

TMU Programming Guidelines

The following section provides basic programming information for the TMU module.

To get the best performance and accuracy from the TMU, initialize the TMU_GAIN, TMU_OFFSET and TMU_AVG registers before enabling the module.

After these registers are programmed, the rest of the TMU initialization can take place. This includes setting up the fault and alert limits and then enabling the TMU.

TMU Calibration

Calibration settings are comprised of gain and offset settings. These settings must be programmed into the corresponding device registers before the temperature is read from the TMU temperature value register. The calibration setting ensures that the reported temperature approximates the actual die temperature (within the error specified in the data sheet). The specified accuracy of the reported temperature cannot be met if gain and offset settings are not used.

The TMU configuration software programs gain and offset values as part of the TMU initialization process. The gain value is programmed in the TMU_GAIN.VALUE field. The offset value is programmed in the TMU_OFFSET.VALUE bit field. The *TMU Calibration Settings* table provides the calibration setting for the processor.

Table 14-5: TMU Calibration Settings

Bit Field	Programmed Value
TMU_GAIN.VALUE	0x004
TMU_OFFSET.VALUE	0x7D40

NOTE: Refer to the product data sheet for the expected TMU accuracy achieved using the settings.

NOTE: The calibration settings provide expected TMU performance at temperatures above room temperature. The accuracy is not guaranteed for temperatures below room temperature.

ADSP-SC59x TMU Register Descriptions

Thermal monitoring unit (TMU) contains the following registers.

Table 14-6: ADSP-SC59x TMU Register List

Name	Description
TMU_ALRT_LIM_HI	Alert High Limit Register
TMU_ALRT_LIM_LO	Alert Low Limit Register
TMU_AVG	Averaging Register
TMU_CNV_BLANK	Temperature Conversion Blank Register
TMU_CTL	TMU Control Register
TMU_FLT_LIM_HI	Fault High Limit Register

Table 14-6: ADSP-SC59x TMU Register List (Continued)

Name	Description
TMU_FLT_LIM_LO	Fault Low Limit Register
TMU_GAIN	Gain Value Register
TMU_IMSK	Interrupt Mask Register
TMU_OFFSET	Offset Register
TMU_REFR_CNTR	Temperature Refresh Counter
TMU_STAT	Status Register
TMU_TEMP	Temperature Value Register

Alert High Limit Register

The TMU_ALRT_LIM_HI register sets the temperature alert high limit as an integer value. The value is stored in two's complement format. Asserts TMU_STAT.ALRTHI if the TMU_TEMP value is greater than or equal to TMU_ALRT_LIM_HI. The TMU_ALRT_LIM_HI value should be programmed for value greater than 8'h3C (60 degC).

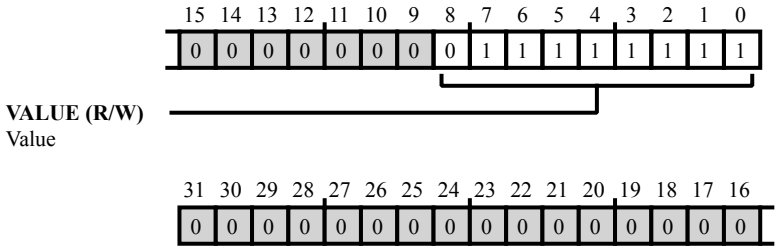


Figure 14-2: TMU_ALRT_LIM_HI Register Diagram

Table 14-7: TMU_ALRT_LIM_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Value. The TMU_ALRT_LIM_HI.VALUE bit field configures the alert temperature high limit as an integer value stored in two's complement format. If the temperature value is greater than or equal to the high-limit the TMU_STAT.ALRTHI bit is set.

Alert Low Limit Register

The TMU_ALRT_LIM_LO register configures the alert temperature low limit.

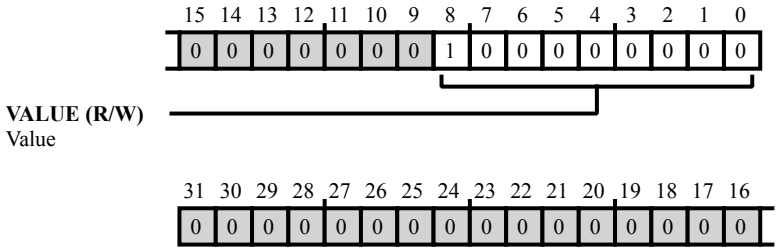


Figure 14-3: TMU_ALRT_LIM_LO Register Diagram

Table 14-8: TMU_ALRT_LIM_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Value. The TMU_ALRT_LIM_LO.VALUE bit field configures the alert temperature low limit as an integer value stored in two's complement format. If the temperature value is less than or equal to the low-limit the TMU_STAT.ALRTLO bit is set.

Averaging Register

The TMU_AVG register enables averaging on the TMU.

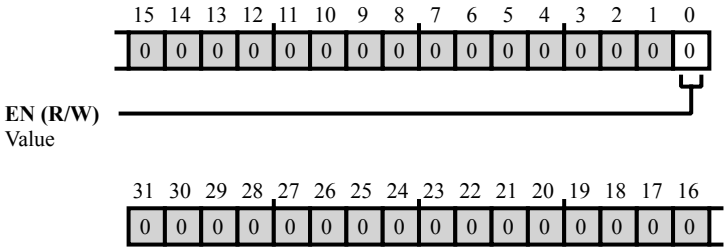


Figure 14-4: TMU_AVG Register Diagram

Table 14-9: TMU_AVG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Value. The TMU_AVG.EN bit enables averaging on the TMU. Averaging is done using the formula (7 x previous_avg_value + current_value)/8. Initially the current_value is taken as previous_avg_value.	
		0	No averaging
		1	Enable averaging

Temperature Conversion Blank Register

The TMU_CNV_BLANK register specifies blanking period in which TMU utilizes HADC for temperature conversion.

The conversion time is calculated as: (VALUE+1)*50k SCLK cycles. So, the minimum conversion time is 50k SCLK cycles. Default conversion time is 200k SCLK cycles. For this duration, the system cannot use HADC for data conversion and HADC does take input for data-conversion. The status is indicated by TMUHADC_BUSY bit of the HADC block.

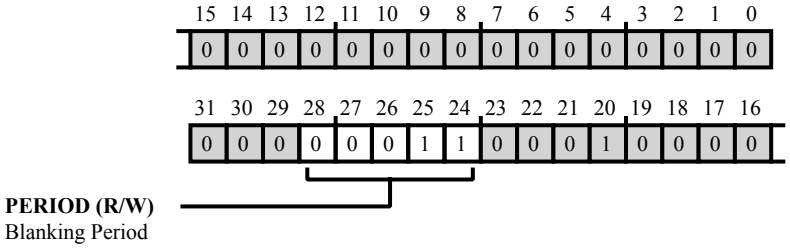


Figure 14-5: TMU_CNV_BLANK Register Diagram

Table 14-10: TMU_CNV_BLANK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:24 (R/W)	PERIOD	Blanking Period.

TMU Control Register

The TMU_CTL register contains bits that allow programs to configure and enable the TMU.

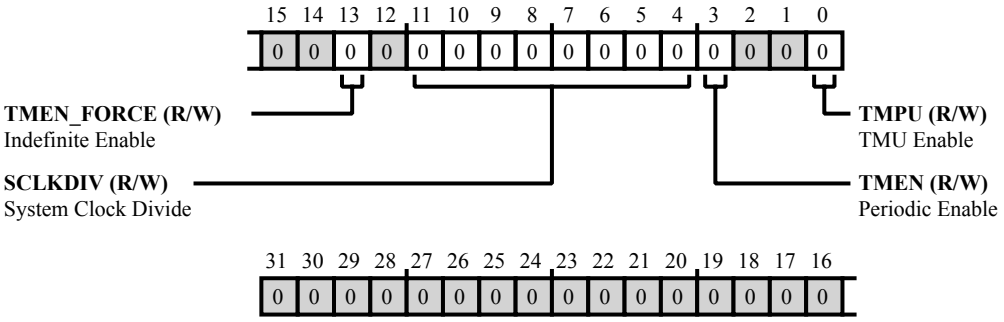


Figure 14-6: TMU_CTL Register Diagram

Table 14-11: TMU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	TMEN_FORCE	Indefinite Enable. Asserting the TMU_CTL.TMEN_FORCE bit enables the TMU indefinitely. HADC is always used by the TMU, so the HADC cannot be used to convert channel inputs.	
11:4 (R/W)	SCLKDIV	System Clock Divide. The TMU_CTL.SCLKDIV bit selects the division ratio for the system clock (SCLK). SCLK is divided by (21+4*SCLKDIV). Thus, by default the SCLK is divided by 21.	
3 (R/W)	TMEN	Periodic Enable. Asserting the TMU_CTL.TMEN bit enables the TMU periodically. TMU is enabled once in every refresh period defined by the TMU_REFR_CNTR register. And in each refresh period, TMU measures temperature for the blanking period (Tblank). It is defined by CNV_BLANK bit.	
		0	Disable TMU
		1	Enable TMU periodically
0 (R/W)	TMPU	TMU Enable. The TMU_CTL.TMPU bit enables the TMU. By default, the module is in power-down mode. The peripheral interface is active even in power-down mode (the TMU registers can be read/written).	
		0	Power Down the analog circuitry of TMU
		1	Power Up the analog circuitry of TMU

Fault High Limit Register

The TMU_FLT_LIM_HI register sets the temperature fault high limit as an integer value. The value is stored in two's complement format. Asserts TMU_STAT.FLTHI if the TMU_TEMP value is greater than or equal to TMU_FLT_LIM_HI. The TMU_FLT_LIM_HI value should be programmed for value greater than 8'h3C (60 degC).

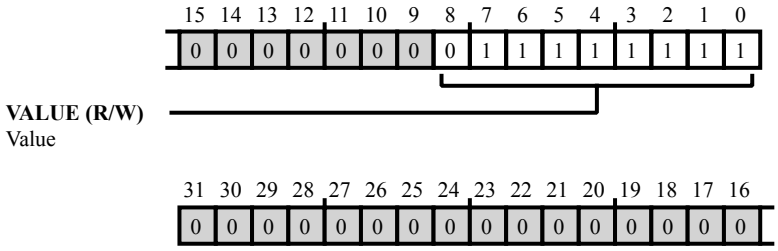


Figure 14-7: TMU_FLT_LIM_HI Register Diagram

Table 14-12: TMU_FLT_LIM_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Value. The TMU_FLT_LIM_HI.VALUE bit field sets the temperature high limit as an integer in two's complement format. If the limit value is greater than or equal to the high-limit the TMU_STAT.FLTHI bit is set.

Fault Low Limit Register

The TMU_FLT_LIM_LO register configures the fault temperature low limit.

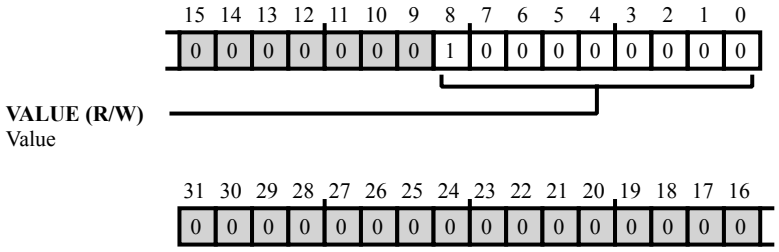


Figure 14-8: TMU_FLT_LIM_LO Register Diagram

Table 14-13: TMU_FLT_LIM_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Value. The TMU_FLT_LIM_LO.VALUE bit field configures the fault temperature low limit as an integer which is stored in two's complement format. If the temperature value is less than or equal to the low-limit, the TMU_STAT.FLTLO bit is set.

Gain Value Register

The TMU_GAIN register is used to configure the gain value in two's complement format. This value is used to correct the gain error in the TMU_TEMP register.

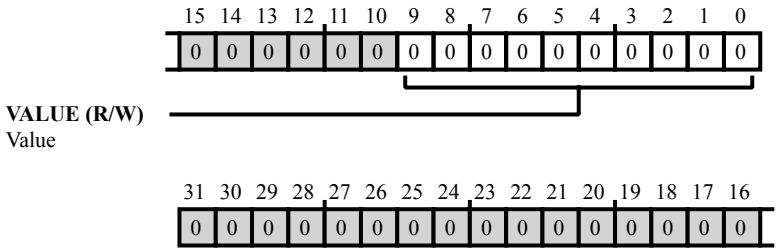


Figure 14-9: TMU_GAIN Register Diagram

Table 14-14: TMU_GAIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/W)	VALUE	Value. The TMU_GAIN.VALUE bit field configures the gain value in two's complement format. This value is used to correct the gain error in the TMU_TEMP register.

Interrupt Mask Register

The TMU_IMSK register provides bits that are used to mask and unmask the interrupts associated with the TMU module.

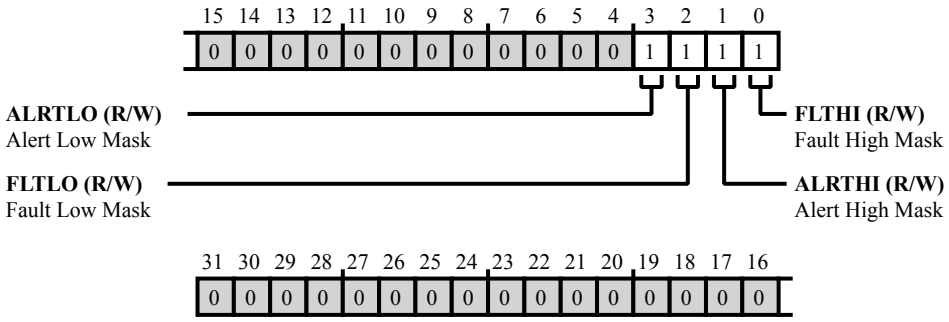


Figure 14-10: TMU_IMSK Register Diagram

Table 14-15: TMU_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	ALRTLO	Alert Low Mask. The TMU_IMSK.ALRTLO bit masks or unmasks the alert low status change (TMU_STAT.ALRTLO).	
		0	Unmask alert low status change
		1	Mask alert low status change
2 (R/W)	FLTLO	Fault Low Mask. The TMU_IMSK.FLTLO bit masks or unmasks the fault low status change (TMU_STAT.FLTLO).	
		0	Unmask fault low status change
		1	Mask fault low status change
1 (R/W)	ALRTHI	Alert High Mask. The TMU_IMSK.ALRTHI bit masks or unmasks the alert high status change (TMU_STAT.ALRTHI).	
		0	Unmask alert high status change
		1	Mask alert high status change
0 (R/W)	FLTHI	Fault High Mask. The TMU_IMSK.FLTHI bit masks or unmasks the fault high interrupt (TMU_STAT.FLTHI).	
		0	Unmask fault high interrupt
		1	Mask fault high interrupt

Offset Register

The value programmed in the TMU_OFFSET register is used to correct the offset error in TMU_TEMP register.

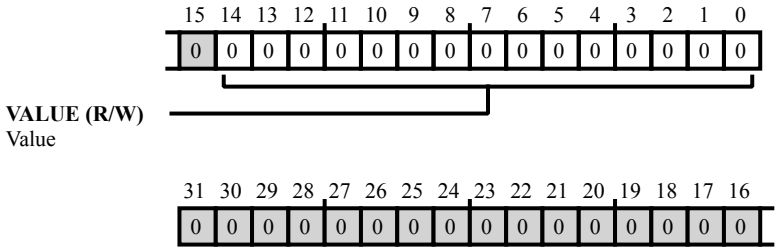


Figure 14-11: TMU_OFFSET Register Diagram

Table 14-16: TMU_OFFSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14:0 (R/W)	VALUE	Value. The TMU_OFFSET.VALUE bit field provides the offset value which is used to correct the offset error in the TMU_TEMP register. This value is in a two's complement fixed point Q7.7 format i.e 8 MSB's represent integer part in 2's complement format. and remaining 7 LSB's represent decimal part. Offset will be applied after multiplying gain.

Temperature Refresh Counter

The TMU_REFR_CNTR register defines the refresh-period to refresh temperature value.

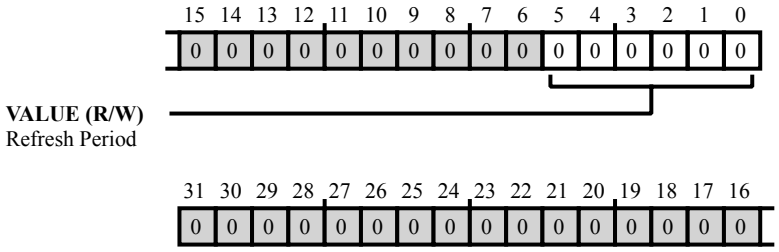


Figure 14-12: TMU_REFR_CNTR Register Diagram

Table 14-17: TMU_REFR_CNTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	VALUE	Refresh Period. The TMU_REFR_CNTR.VALUE bit field defines the period to refresh the temperature value. Temperature is refreshed for every (VALUE+1)*(21+4*SCLKDIV)*1M system-clock cycles. SCLKDIV is defined in TMU_CTL register. So, by default, the period is 21 million system-clock cycles.

Status Register

The TMU_STAT register bits indicate when an error or fault is detected.

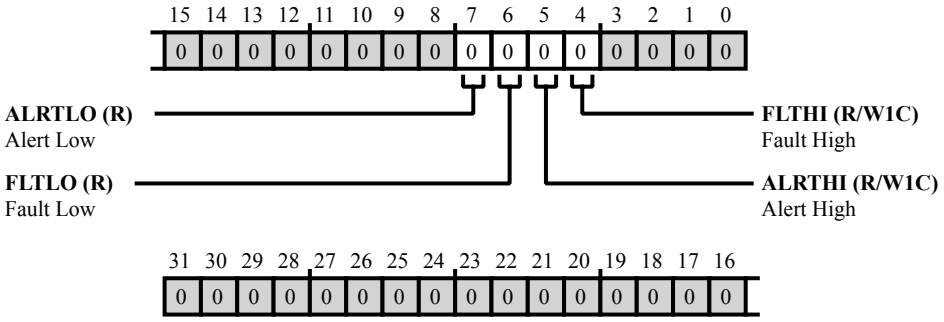


Figure 14-13: TMU_STAT Register Diagram

Table 14-18: TMU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	ALRTLO	Alert Low. The TMU_STAT.ALRTLO bit is set when the temperature value is less than or equal to the setting in the TMU_ALRT_LIM_LO register. The bit assertion is masked if TMU_IMSK.ALRTLO is HIGH.
6 (R/NW)	FLTLO	Fault Low. The TMU_STAT.FLTLO bit is set when the temperature value is less than or equal to the setting in the TMU_FLT_LIM_LO register. The bit assertion is masked if TMU_IMSK.FLTLO is HIGH.
5 (R/W1C)	ALRTHI	Alert High. The TMU_STAT.ALRTHI bit is set when the temperature value is greater than or equal to the setting in the TMU_ALRT_LIM_HI register. The temperature value register is set to 16'h0D80 (effective temperature value of 27degrees C) when a W1C operation is done on this bit. The bit assertion is masked if TMU_IMSK.ALRTHI is HIGH.
4 (R/W1C)	FLTHI	Fault High. The TMU_STAT.FLTHI bit is set when the temperature value is greater than or equal to the setting in the TMU_FLT_LIM_HI register. The temperature value register is set to 16'h0D80 (effective temperature value of 27 degrees C) when a W1C is done on this bit. The bit assertion is masked if TMU_IMSK.FLTHI is HIGH.

Temperature Value Register

The TMU_TEMP register provides the temperature value from the A/D converter and the status.

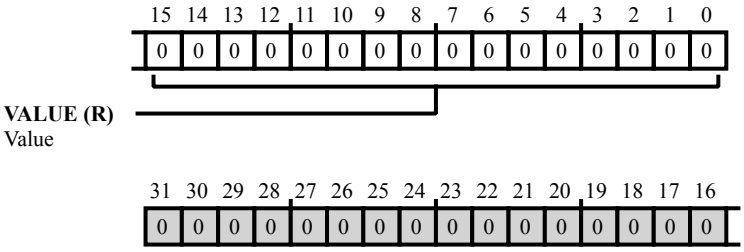


Figure 14-14: TMU_TEMP Register Diagram

Table 14-19: TMU_TEMP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	VALUE	Value. The TMU_TEMP.VALUE bit field is the temperature value from A/D converter. This value is stored in two's complement fixed point Q8.7 format where the 9 MSB's represent the integer part in 2's complement, and the remaining 7 LSB's represent the decimal part.

15 Housekeeping ADC (HADC)

The Housekeeping ADC is a 12-bit (with 10-bit accuracy), successive approximation ADC. It operates from single supply and features throughput rates up to 1 MSPS. The HADC can be used for the collection of housekeeping parameters like voltages, temperatures in the system or for any general-purpose use as well.

NOTE: HADC is used by the TMU for temperature monitor, so it may be unavailable during temperature conversion. For details, refer to the TMU and HADC section.

HADC Features

The HADC supports following features:

- 12-bit ADC core with built-in sample and hold
- ENOB = 10 bit
- 8 input channels which can be extended to 15 channels by putting an external channel multiplexer
- Throughput rates up to 1 MSPS
- Single ended operation
- External reference nominal at 1.8 V
- Analog input 0 V to 1.8 V.
- Selectable ADC clock frequency through a pre-scaler
- Conversion type adaptable to each application–allows single or continuous conversion with option of auto-scan
- Auto sequencing capability provides up to 8 *auto-conversions* in a single session. Each conversion can be programmed to select any of the available input channels.

HADC Functional Description

The HADC provides the analog to digital conversion capability for general-purpose housekeeping tasks, such as voltage and temperature monitoring. The core of HADC is a 12-bit SAR ADC, providing multiple analog input channels.

The HADC has the following functionality:

Fixed and continuous conversion modes

ADC converts the input channel sequence for a fixed number of times or continuously converts an input channel sequence.

Auto scanning

All the input channels can be sampled in a sequential manner.

Channel sequence programming

The sequence of a channel can be selected by programming the channel mask register. If the bit corresponding to the channel is programmed to zero, that channel is included in the auto-scan chain.

ADSP-SC59x HADC Register List

The Housekeeping ADC (HADC) provides a general purpose, multi-channel successive approximation A-to-D converter. A set of registers governs HADC operations. For more information on HADC functionality, see the HADC register descriptions.

Table 15-1: ADSP-SC59x HADC Register List

Name	Description
HADC_CHAN_MSK	Channel Mask Register
HADC_CTL	Control Register
HADC_DATA[nn]	Channel Data Registers
HADC_IMSK	Interrupt Mask Register
HADC_STAT	Status Register

ADSP-SC59x HADC Interrupt List

Table 15-2: ADSP-SC59x HADC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
186	HADC0_EVT	HADC0 Event	Edge	

ADSP-SC59x HADC Trigger List

Table 15-3: ADSP-SC59x HADC Trigger List Generators

Trigger ID	Name	Description	Sensitivity
33	HADC0_EOC	HADC0 HADC0 End of Conversion	Edge

Table 15-4: ADSP-SC59x HADC Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

HADC Definitions

The following definitions are helpful when using the HADC module.

Auto-scan

Auto-scan is a feature which allows the multiple channels to be scanned and converted in sequence one after the other.

HADC Wakeup Time

It is the time required by the module after coming out of power down before it can start converting.

Signal-to-Noise and Distortion Ratio (SINAD)

The measured ratio of signal-to-noise and distortion at the output of the ADC. The signal is the rms amplitude of the fundamental. Noise is the sum of all non-fundamental signals up to half the sampling frequency ($f_S/2$), excluding dc. The ratio depends on the number of quantization levels in the digitization process; the more levels, the smaller the quantization noise. The theoretical signal-to-noise and distortion ratio for an ideal N-bit converter with a sine wave input is given by:

Signal-to-(Noise + Distortion) = (6.02 N + 1.76) dB

Total Harmonic Distortion (THD)

The ratio of the rms sum to the harmonics to the fundamental.

Peak Harmonic or Spurious Noise

The ratio of the rms value of the next largest component in the ADC output spectrum (up to $f_S/2$ and excluding dc) to the rms value of the fundamental. Typically, the value of this specification is determined by the largest harmonic in the spectrum, but for ADCs where the harmonics are buried in the noise floor, it is a noise peak.

Integral Nonlinearity

The maximum deviation from a straight line passing through the endpoints of the ADC transfer function. The endpoints are zero scale, a point 1 LSB below the first code transition, and full scale, a point 1 LSB above the last code transition.

Differential Nonlinearity

The difference between the measured and the ideal 1 LSB change between any two adjacent codes in the ADC.

Offset Error

The deviation of the first code transition (00...000) to (00...001) from the ideal—that is, GND1 + 1 LSB.

Offset Error Match

The difference in offset error between any two channels.

Gain Error

The deviation of the last code transition (111...110) to (111...111) from the ideal (that is, REFIN – 1 LSB) after the offset error has been adjusted out.

Gain Error Matching

The difference in gain error between any two channels.

Power Supply Rejection Ratio (PSRR)

PSRR is defined as the ratio of the power in the ADC output at full-scale frequency, f, to the power of a 100 mV p-p sine wave applied to the ADC VDD supply of frequency, f_S. The frequency of the input varies from 5 kHz to 25 MHz. PSRR (dB) = 10 log(Pf/Pf_S) where: Pf is the power at frequency, f, in the ADC output. Pf_S is the power at frequency, f_S, in the ADC output.

HADC Block Diagram

The ***HADC Block Diagram*** figure shows the functional blocks within the HADC and the interface to the processor core and the peripherals. The HADC has an internal channel multiplexer that is controlled by a programmable sequencer which selects the desired channel or sequence of channels.

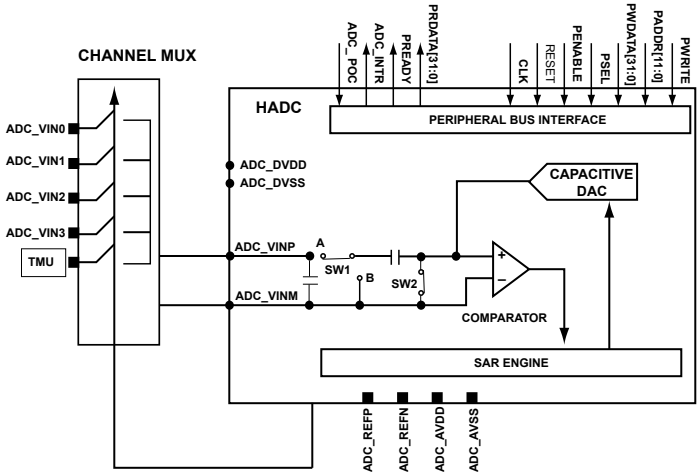


Figure 15-1: HADC Block Diagram

HADC Signal Descriptions

The ***HADC Signal Descriptions*** table provides descriptions of the signals used by the HADC.

Table 15-5: HADC Signal Descriptions

Signal Name	Signal Description
AVDD	ADC I/O supply
AVSS	I/O ground for analog blocks - shorted with other ground pins within the package
VREFP	External reference for ADC
VREFN	Ground reference for ADC
VINn	Analog input at channel n

HADC Architectural Concepts

The HADC is based on a 12-bit SAR ADC that provides a simple register-based access model to obtain the results of conversion. The digital front end of the HADC provides a set of registers to configure the mode of operation, sampling frequency, and input channel selection control. The ADC supports multiple input analog channels which can be individually selected or deselected for conversion. The results of each analog channel are stored in a register. The core can access the register to read the conversion results once the conversion is complete. The HADC also provides the interrupts on completion of each channel conversion to avoid polling by the core. The following sections provide more details about the architecture of the HADC.

Converter Operation

The housekeeping ADC is a 12-bit successive approximation ADC based around a capacitive DAC. The ***ADC Acquisition Phase*** figure and the ***ADC Conversion Phase*** figure show simplified schematics of the ADC. The ADC is comprised of control logic, SAR, and a capacitive DAC. The components are used to add and subtract fixed amounts of charge from the sampling capacitor to bring the comparator back into a balanced condition. The ***ADC Conversion Phase*** figure shows the ADC during its acquisition phase. SW2 is closed and SW1 is in Position A. The

comparator is held in a balanced condition and the sampling capacitor acquires the signal on the selected V_{IN} channel.

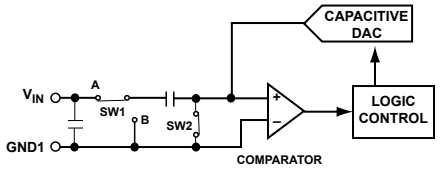


Figure 15-2: ADC Acquisition Phase

When the ADC starts a conversion (see the ***ADC Conversion Phase*** figure), SW2 opens and SW1 moves to Position B, causing the comparator to become unbalanced. The control logic and the capacitive DAC are used to add and subtract fixed amounts of charge to bring the comparator back into a balanced condition. When the comparator is rebalanced, the conversion is complete. The control logic generates the ADC output code.

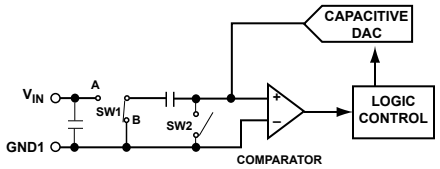


Figure 15-3: ADC Conversion Phase

Auto-Scan

The HADC features auto-scan mode where all the input channels can be sampled in a sequential manner. The number of channels enabled in auto-scan mode can be selected by programming the HADC_CHAN_MSK register. If the bit corresponding to the channel is set high, that particular channel is masked, and is not included in the auto-scan chain. In this way programs can sample all, none, or a selected set of channels by writing a high or a low for the individual channel. Auto sequencing allows the system to convert the same channel multiple times, allowing programs to perform oversampling algorithms.

For example, if the HADC_CHAN_MSK register bits [3:0] are set to 1101, then channel 0, channel 2 and channel 3 are not included in the auto-scan chain. Whether the conversion is a single or fixed number or continuous depends on the status of HADC_CTL.CONT bit. If this bit is low, the HADC_CTL.FIXEDCNV bits determine the number of sequence conversions.

The maximum number of fixed sequence conversions is 15. By default, the first eight channels of the HADC are enabled and extended channels are masked. An extended channel is the increased number of channels when a single channel of ADC is externally multiplexed, effectively increasing the total number of available channels. The total number of channels supported is 15. The program must configure the HADC_CHAN_MSK register to enable any desired channel.

Channel Sequence Programming

The sequence of a channel can be selected by programming the HADC_CHAN_MSK register. If the bit corresponding to the channel is programmed to zero, that channel is included in the auto-scan chain. If the program must get the conversion results from a particular channel, then the bit corresponding to that channel should be zero.

Channels 8-15 are virtual channels; in that they can connect an external multiplexer and increase the effective number of channels. As the output of the external multiplexer is routed to channel 7, this channel is not available (or converted) separately when the virtual channels are not masked in the HADC_CHAN_MSK register. The auto-scan section has more details.

ADC Transfer Function

The output coding is straight binary for the analog input channel conversion. The designed code transitions occur at successive LSB values (that is, 1 LSB, 2 LSBs, and so forth). The LSB size is $V_{REF}/4096$ for the HADC. The *ADC Transfer Function* figure shows the ideal transfer characteristic for straight binary coding.

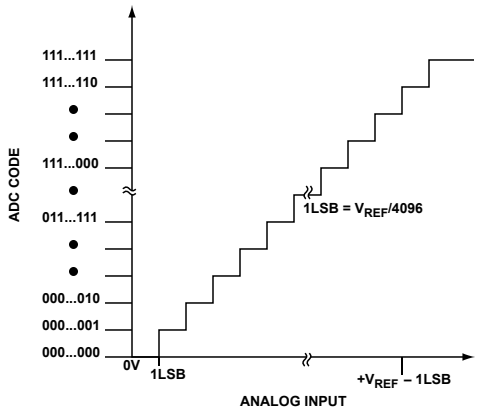


Figure 15-4: ADC Transfer Function

Results

The HADC takes 20 cycles of f_{SAMPLE} for one channel conversion. (The value of the HADC_CTL.FDIV bit field determines f_{SAMPLE}). The time taken to complete one sequence depends on the number of channels in the auto-scan chain. There is a latency of 1 cycle from the time the channel is selected internally to sample to the time the data is ready. After the end of each channel conversion, the data is written into the corresponding data register. An interrupt is generated (if the interrupt mask is not enabled) to signal that the data is ready for that particular channel.

HADC Operating Modes

The HADC has two modes of operation described in the following sections.

Fixed Conversion Mode

In this mode, the ADC converts the input channel sequence for a fixed number of times. The frequency is configured in the HADC_CTL.FIXEDCNV bit field. To use this mode, clear the HADC_CTL.CONT bit.

Continuous Conversion Mode

In this mode ADC continuously converts an input channel sequence, when the HADC_CTL.STARTCNV bit is held high. To use this mode, set the HADC_CTL.CONT bit.

HADC Event Control

The HADC generates different events depending on the state of the ADC and the status of channel conversions. It can generate an event for each of the following conditions:

- When ADC is ready for conversion
- At the end of sequence conversion
- At the end of each individual channel conversion

Each of these events can generate an interrupt. To generate an interrupt on any desired event, clear the respective bit in the HADC_IMSK register.

HADC Programming Model

Following sections provide some guidelines for HADC programming.

Powering Up the HADC

To power-up the HADC, program the following bits in the HADC_CTL register.

- Deassert the HADC_CTL.PD bit (HADC power down)
- Set the HADC_CTL.NRST bit (Reset)
- Set the HADC_CTL.ENLS bit (Enable level shifters)

After deasserting HADC_CTL.PD, the HADC requires a finite wake-up time (t_{WAKEUP}) before it can start converting. The HADC requires only two f_{SAMPLE} clocks from the assertion of the HADC_CTL.NRST bit before the module is ready to convert. (HADC_CTL.PD is low). Poll the HADC_STAT.RDY bit. A 1 on this bit indicates that the HADC is ready to convert data.

Enabling the HADC

Setting the HADC_CTL.STARTCNV bit enables the HADC. When this bit is kept high, the HADC can work in either continuous or fixed conversion mode. After the HADC_CTL.STARTCNV bit is set =1, the HADC_CHAN_MSK can still be re-programmed, but the new sequence only comes into effect after the current sequence conversion is complete.

ADSP-SC59x HADC Register Descriptions

Housekeeping ADC (HADC) contains the following registers.

Table 15-6: ADSP-SC59x HADC Register List

Name	Description
HADC_CHAN_MSK	Channel Mask Register

Table 15-6: ADSP-SC59x HADC Register List (Continued)

Name	Description
HADC_CTL	Control Register
HADC_DATA[nn]	Channel Data Registers
HADC_IMSK	Interrupt Mask Register
HADC_STAT	Status Register

Channel Mask Register

The HADC_CHAN_MSK register provides bits that mask each channel. The LSB corresponds to channel 0 , the second LSB to channel 1 and so on. If a mask bit is set, the corresponding channel is not converted. By default, channels 0-7 are not masked and the extended channels 8-15 are masked.

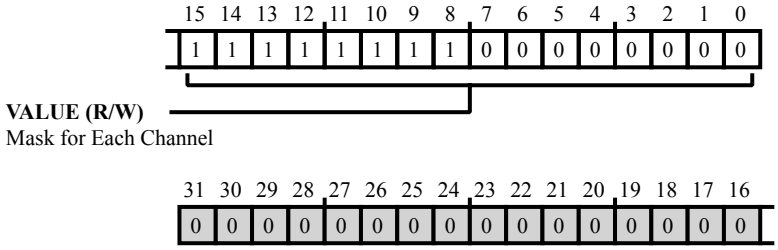


Figure 15-5: HADC_CHAN_MSK Register Diagram

Table 15-7: HADC_CHAN_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Mask for Each Channel. The HADC_CHAN_MSK.VALUE bit field is the mask bit for each channel. The first MSB corresponds to channel 15, the second MSB to channel 14 and so on. If the mask is set for a particular channel, that channel is not converted. By default, channels 0-7 are not masked and the extended channels 8-15 are masked. The first MSB corresponds to channel 7, the second MSB to channel 6 and so on. If the mask is set for a particular channel, that channel is not converted. By default, channels 0-7 are not masked.

Control Register

The HADC_CTL register contains control bits that configure various module settings start or reset the module.

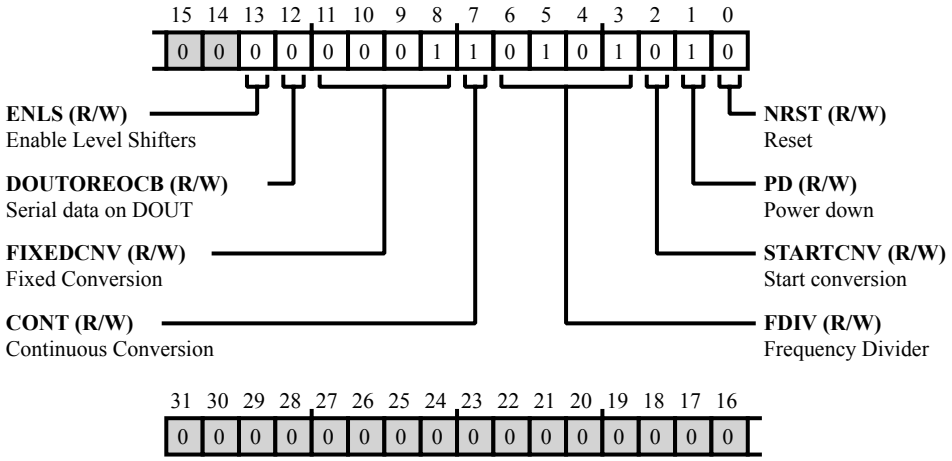


Figure 15-6: HADC_CTL Register Diagram

Table 15-8: HADC_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	ENLS	Enable Level Shifters. Setting the HADC_CTL.ENLS bit enables the level shifters, allowing the HADC analog side (which works in the VDD_EXT domain) to work with the digital core and interface is (in the VDD_INT domain).	
		0	Disable down level shifters
		1	Enable down level shifters
12 (R/W)	DOUTOREOCB	Serial data on DOUT. If the HADC_CTL.DOUTOREOCB bit =1, serial data arrives on the EOC_DOUT pin. If this bit =0 (default) it acts as an EOC only if the external multiplexer is connected.	
		0	Reserved
		1	Reserved
11:8 (R/W)	FIXEDCNV	Fixed Conversion. The HADC_CTL.FIXEDCNV bit configures the number of conversions = FIXEDCNV. This value determines how many times a sequence is converted when the HADC is in fixed conversion mode. This only applies when the HADC_CTL.CONT bit =0. Before changing the HADC_CTL.FIXEDCNV bit, clear the HADC_CTL.NRST bit (=0).	

Table 15-8: HADC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	CONT	Continuous Conversion. When the HADC_CTL.CONT bit =0, the ADC converts a sequence for a fixed number of times. This number is configured using the HADC_CTL.FIXEDCNV bit field. When the HADC_CTL.CONT bit =1, the ADC continuously converts a given sequence, provided the HADC_CTL.STARTCNV is held high.	
		0	ADC converts sequence for fixed number of times
		1	ADC continuously converts given sequence
6:3 (R/W)	FDIV	Frequency Divider. The HADC_CTL.FDIV bit field configures the f_SAMPLE=f_CLK/(FDIV+1). Select f_CLK and HADC_CTL.FDIV values so that f_SAMPLE is in range of 50 kHz to 22.5 MHz. The minimum value for HADC_CTL.FDIV is 1. Before changing the HADC_CTL.FDIV bits, clear the HADC_CTL.NRST bit.	
2 (R/W)	STARTCNV	Start conversion. The HADC_CTL.STARTCNV bit needs to be set for the ADC to start converting data. If the ADC is running in non continuous mode, it is reset by hardware after the desired number of conversions is completed.	
		0	No action
		1	Start converting
1 (R/W)	PD	Power down. The HADC_CTL.PD bit powers down the analog circuitry of the ADC. After this bit returns to 0 a finite power-up time is required before the ADC can start converting data.	
		0	No action
		1	Power down the analog circuitry of the ADC
0 (R/W)	NRST	Reset. The HADC_CTL.NRST bit resets the ADC.	
		0	Reset the ADC
		1	No action

Channel Data Registers

The `HADC_DATA[nn]` registers NN ranges from 0-14. Each corresponding to an ADC channel. ADC_DATA_0 corresponds to channel 0, ADC_DATA_1 to channel 1 and so on.

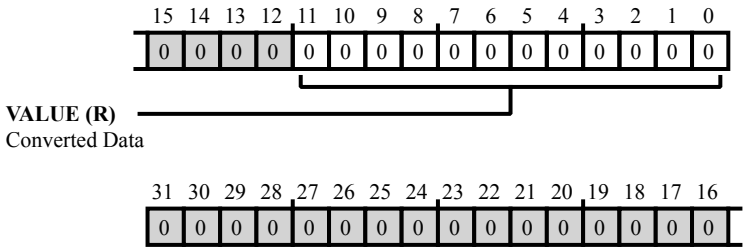


Figure 15-7: HADC_DATA[nn] Register Diagram

Table 15-9: HADC_DATA[nn] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/NW)	VALUE	Converted Data. The `HADC_DATA[nn].VALUE` bit field contains the digital code for the sampled analog value. Each channel has its own data register.

Interrupt Mask Register

The HADC_IMSK register masks (disables) or unmasks (enables) the interrupts as programmed.

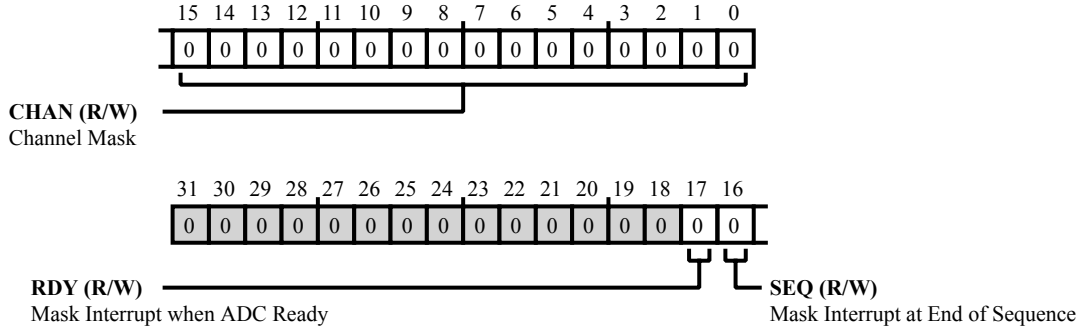


Figure 15-8: HADC_IMSK Register Diagram

Table 15-10: HADC_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	RDY	Mask Interrupt when ADC Ready. The HADC_IMSK.RDY bit masks the interrupt generated when ADC is ready to convert.	
16 (R/W)	SEQ	Mask Interrupt at End of Sequence. The HADC_IMSK.SEQ bit masks the interrupt which is generated at the end of sequence completion.	
		0	Interrupt is unmasked
		1	Interrupt is masked
15:0 (R/W)	CHAN	Channel Mask. The HADC_IMSK.CHAN bit field provides the interrupt mask bit for each channel. N ranges from 0-15. The MSB corresponds to channel 15, the second MSB to channel 14 and so on. If the bit is SET, interrupt is masked for corresponding channel.	

Status Register

The HADC_STAT register contains bits that provide status information on the HADC module.

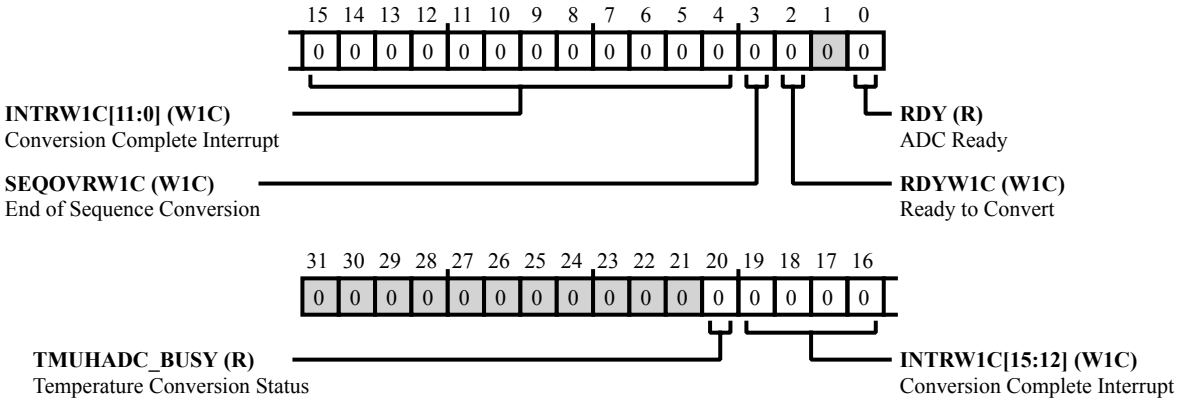


Figure 15-9: HADC_STAT Register Diagram

Table 15-11: HADC_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20 (R/NW)	TMUHADC_BUSY	Temperature Conversion Status. The HADC_STAT.TMUHADC_BUSY bit, if high, indicates to the system that temperature conversion is ongoing. All ADC conversion requests are ignored.
19:4 (RX/W1C)	INTRW1C	Conversion Complete Interrupt. The HADC_STAT.INTRW1C bit field indicates when the corresponding ADC channel completes conversion. N ranges from 0-15 and the MSB corresponds to channel 15, the second MSB to channel 14 and so on. N ranges from 0-7 and the MSB corresponds to channel 7, the second MSB to channel 6 and so on. These bits are sticky and are W1C.
3 (RX/W1C)	SEQOVRW1C	End of Sequence Conversion. The HADC_STAT.SEQOVRW1C bit indicates the end of a sequence conversion and is a sticky status bit which is W1C.
2 (RX/W1C)	RDYW1C	Ready to Convert. The HADC_STAT.RDYW1C bit is the stick version of the HADC_STAT.RDY bit.
0 (R/NW)	RDY	ADC Ready. The HADC_STAT.RDY bit is set (=1) when the ADC is ready to convert data.

16 Controller Area Network Flexible Data Rate (CANFD)

The controller area network (CAN) protocol is primarily for use as a vehicle serial data bus. It meets the specific requirements of this field, including real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

The CANFD module is a full implementation of the CAN protocol specification, the CAN with flexible data rate (CAN FD) protocol, and the CAN 2.0 Part B protocol. It supports both standard and extended message frames and long payloads up to 64 bytes, transferred at rates up to 8 Mbps. The message buffers are stored in an embedded RAM dedicated to the CANFD module.

NOTE: This document assumes familiarity with the CAN standard. For more information, refer to Version 2.0 of the CAN specification from Robert Bosch GmbH.

CANFD Features

Key features of the CANFD module include:

- Full implementation of the CAN FD protocol and the CAN specification 2.0 (Part B) including:
  - Standard data frames
  - Extended data frames
  - Zero to sixty-four bytes data length
  - Programmable bit rate
  - Content-related addressing
- ISO 11898-1 standard compliance
- Flexible mailboxes
  - Configurable data lengths from 0 to 64 bytes
  - Configurable as receive or transmit, supporting standard and extended messages
  - Individual Rx mask registers per mailbox

- Full-featured Rx FIFO
  - Storage capacity for up to six frames
  - Automatic internal pointer handling with DMA support
- Transmission abort capability
- 28 message buffers of 8 bytes data length each, configurable as Rx or Tx
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Independence from the transmission medium (an external transceiver is required)
- Low-power modes, with programmable wakeup on bus activity or matching with received frames (Pretended Networking)
- Transceiver Delay Compensation feature when transmitting CAN FD messages at faster data rates
- Remote request frames may be handled automatically or by software
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- Status of synchronization with CAN bus
- CRC status for transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process
- Rx FIFO ID filtering, matching incoming IDs against 128 extended, 256 standard, or 512 partial (8 bit) IDs
  - Up to 32 ID filter table elements
- Detection and correction of memory read access errors, with five parity bits for each byte of CANFD memory
- Pretended networking functionality in low-power doze mode

CANFD Functional Description

The CANFD module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system consists of a set of message buffers (MBs) that store configuration and control data, time stamp, message ID, and data. The memory corresponding to the first 38 MBs is configurable to support a FIFO reception scheme with a powerful ID filtering mechanism. The ID filtering mechanism is capable of checking

incoming frames against a table of IDs (up to 128 extended IDs, 256 standard IDs, or 512 8-bit ID slices), with an individual mask register for up to 32 ID filter table elements.

Classical CAN frames support simultaneous reception through a FIFO and mailboxes. CAN FD frames support reception only through mailboxes. For mailbox reception, there is a matching algorithm to store received frames into the MBs that have the same ID programmed in the ID field. There is a masking scheme to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

The CANFD module can also receive and transmit messages in CAN FD format. The MBs are sized to store the quantity of data bytes selected in the MBDSRn fields of the CANFD_FD_CTL register.

The following sections provides listings of the CANFD registers, interrupts, and triggers. It section also provides information on the architectural concepts and functional operation of the CANFD module.

ADSP-SC59x CANFD Register List

CANFD module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification. Features : Flexible DataRate, PretendedNetwork, DMA support, ECC for RAM.

Table 16-1: ADSP-SC59x CANFD Register List

Name	Description
CANFD_TIMING	Can Bit Timing Register
CANFD_CRC	CRC Register
CANFD_CTL1	Control 1 Register
CANFD_PN_CTL1	Pretended Networking Control1 Register
CANFD_CTL2	Control 2 Register
CANFD_PN_CTL2	Pretended Networking Control2 Register
CANFD_ECR	Error Counter Register
CANFD_ERR_IADDR	Error Injection Address Register
CANFD_ERR_IDP	Error Injection Data Pattern Register
CANFD_ERR_IPP	Error Injection Parity Pattern Register
CANFD_ERR_STAT	Error Status Register
CANFD_ESR1	Error and Status 1 Register
CANFD_ESR2	Error and Status 2 Register
CANFD_FD_TIMING	CANFD Bit Timing Register
CANFD_FD_CRC	CANFD CRC Register
CANFD_FD_CTL	CANFD Control Register

Table 16-1: ADSP-SC59x CANFD Register List (Continued)

Name	Description
CANFD_FLTR_DLC	Pretended Networking DLC Filter Register
CANFD_FLTR_ID1	Pretended Networking ID Filter1 Register
CANFD_FLTR_ID2_IDMSK	Pretended Networking ID Filter2 / IDMask Register
CANFD_IFLG1	Mailbox Interrupt Flag 1 Register
CANFD_IFLG2	Mailbox Interrupt Flag 2 Register
CANFD_IMSK1	Mailbox Interrupt Mask 1 Register
CANFD_IMSK2	Mailbox Interrupt Mask 2 Register
CANFD_CFG	Module Configuration Register
CANFD_MEC	Memory Error Control Register
CANFD_FLTR_DATA1_HI	Pretended Networking Payload Low Filter2 Register
CANFD_FLTR_DATA1_LO	Pretended Networking Payload Low Filter1 Register
CANFD_FLTR_DATA2_DMSK_HI	Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register
CANFD_FLTR_DATA2_DMSK_LO	Pretended Networking Payload Low Filter2 / Payload Low Mask Register
CANFD_ERR_RADDR	Error Report Address Register
CANFD_ERR_RDAT	Error Report Data Register
CANFD_ERR_RSYN	Error Report Syndrome Register
CANFD_RX_14_MSK	Receive Mailbox14 Mask Register
CANFD_RX_15_MSK	Receive Mailbox15 Mask Register
CANFD_RX_FIFO_GMSK	Receive FIFO Global Mask Register
CANFD_RX_FIFO	Receive FIFO Information Register
CANFD_RX_IMSK[n]	Receive Individual Mask Register
CANFD_RX_MB_GMSK	Receive Mailbox Global Mask Register
CANFD_TMR	Free Running Timer Register
CANFD_WMB[n]_DATA_HI	Wakeup Message Buffer Data 4-7 Register
CANFD_WMB[n]_DATA_LO	Wakeup Message Buffer Data 0-3 Register
CANFD_WMB[n]_ID	Wakeup Message ID Buffer Register
CANFD_WMB[n]_STAT	Wakeup Message Buffer Control/Status Register
CANFD_WUM	Pretended Networking Wakeup Match Register

ADSP-SC59x CANFD Interrupt List

Table 16-2: ADSP-SC59x CANFD Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
231	CANFD0_WU	CANFD0 CAN0 Wakeup Interrupt	None	
233	CANFD0_IRQ	CANFD0 CAN0 Interrupt	None	
235	CANFD0_MSG	CANFD0 CAN0 Message Receive/Transmit Interrupt	None	
236	CANFD1_WU	CANFD1 CAN1 Wakeup Interrupt	None	
238	CANFD1_IRQ	CANFD1 CAN1 Interrupt	None	
240	CANFD1_MSG	CANFD1 CAN1 Message Receive/Transmit Interrupt	None	

ADSP-SC59x CANFD Trigger List

Table 16-3: ADSP-SC59x CANFD Trigger List Generators

Trigger ID	Name	Description	Sensitivity
170	`CANFD0_IPD_REQ`	CANFD0 CAN0 DMA request interrupt	None
171	`CANFD1_IPD_REQ`	CANFD1 CAN1 DMA request interrupt	None

Table 16-4: ADSP-SC59x CANFD Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

CANFD Architectural Concepts

The following sections provide information about the CANFD architecture.

Block Diagram

The *CANFD Block Diagram* figure shows a block diagram of the CANFD module.

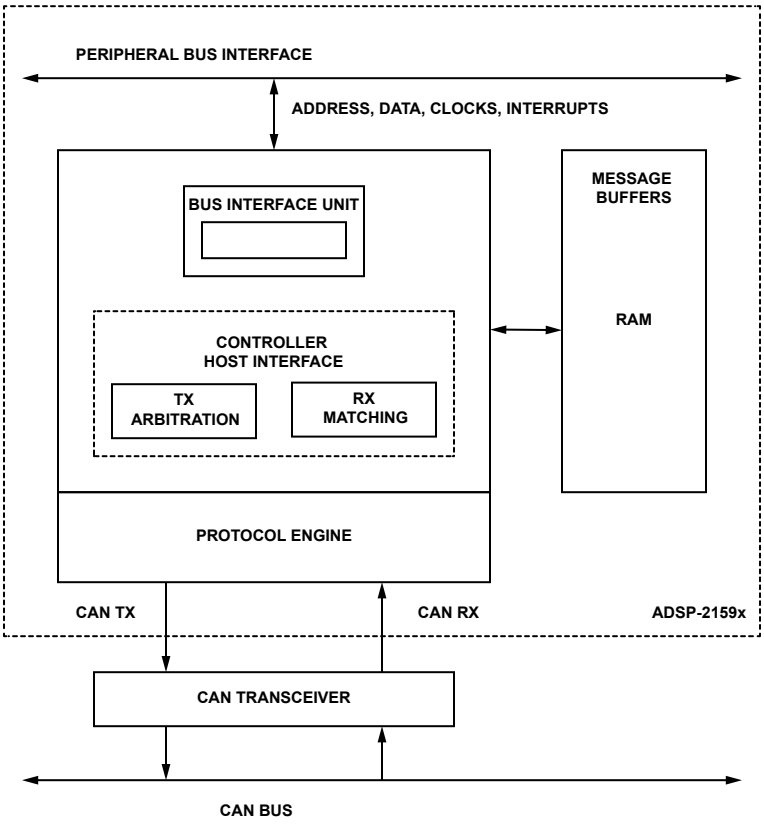


Figure 16-1: CANFD Controller Block Diagram

The CANFD module has the following three submodules and message buffers in an embedded RAM:

Protocol Engine (PE)

The PE submodule manages the serial communication on the CAN bus. It requests RAM access for receiving and transmitting message frames, validates received messages, preforms error handling, and detects CAN FD messages.

Controller Host Interface (CHI)

The CHI submodule handles message buffer selection for reception and transmission, including arbitration, and ID matching algorithms for both flexible data rate (FD) and non-FD message formats.

Bus Interface Unit (BIU)

The BIU submodule controls the access to and from the internal interface bus to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs, DMA, and test signals are accessed through the BIU.

Message Buffer (MB)

The CANFD module uses a message buffer structure that represents both extended (29-bit identifier) and standard (11-bit identifier) frames from the CAN specification (Version 2.0, Part B).

Each individual message buffer is 16, 24, 40, or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32, or 64 data bytes, respectively.

The memory area for mailboxes is from address offset 0x80 to 0x47F. When CAN FD is enabled, the exact address for each memory byte depends on the size of the payload.

The *Message Buffer Structure 64-Byte Payload* figure shows an example of the message buffer structure with a 64-byte payload.

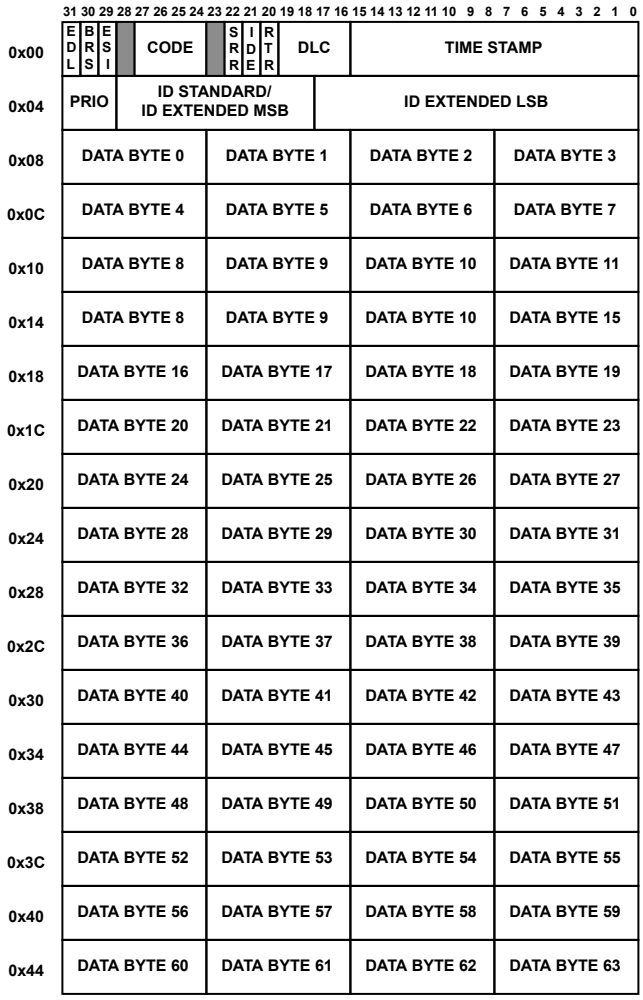


Figure 16-2: Message Buffer Structure 64-Byte Payload

In the example, the first eight bytes of the message buffer is control information associated with the data, followed by the bytes of data in the payload. The control information is defined in the following sections:

Extended Data Length (EDL)

The EDL bit distinguishes between CAN and CAN FD format frames. Do not set EDL when message buffers are configured to RANSWER with code field 0b1010.

Bit Rate Switch (BRS)

The BRS bit defines whether the bit rate is switched inside a CAN FD format frame.

Error State Indicator (ESI)

The ESI bit indicates if the transmitting node is error active or error passive.

Message Buffer Code (CODE)

The CODE field is accessible (read or write) to the processor and the CANFD module, as part of the message buffer matching and arbitration process. The *Message Buffer Code – Rx Buffer* and *Message Buffer Code – Tx Buffer* tables show the encoding.

Table 16-5: Message Buffer Code – Rx Buffer

CODE Description	Rx Code Before New Frame	SRV[*1]	Rx Code After Successful Reception [*2]	RRS [*3]	Comment
0000: INACTIVE – MB is not active.	INACTIVE	N/A	N/A	N/A	MB does not participate in the matching process.
0100: EMPTY – MB is active and empty	EMPTY	N/A	FULL	N/A	When a frame is received successfully, the CODE field is automatically updated to FULL.
0010: FULL – MB is full.	FULL	Yes	FULL	N/A	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching Process.
		No	OVERRUN	N/A	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching Process.
0110: OVERRUN – MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	N/A	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB, the code returns to FULL.
		No	OVERRUN	N/A	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching Process.

Table 16-5: Message Buffer Code – Rx Buffer (Continued)

CODE Description	Rx Code Before New Frame	SRV*1	Rx Code After Successful Reception *2	RRS *3	Comment
1010: RANSWER*4 – A frame was configured to recognize a remote request frame and transmit a response frame in return.	RANSWER	N/A	TANSWER(1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that, an MB is set to transmit a response frame. The code is automatically changed to TANSWER (1110). See Matching Process. If the CANFD_CTL2.RRS bit is negated, transmit a response frame whenever a remote request frame with the same ID is received.
		N/A	N/A	1	This code is ignored during matching and arbitration process.
CODE[0]=1; BUSY – CANFD is updating the contents of the MB. The CPU must not access the MB.	BUSY*5	N/A	FULL	N/A	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		N/A	OVERRUN	N/A	

*1 Serviced MB. The MB has been read and unlocked by reading the CANFD_TMR register or other MB.

*2 A frame has a successful reception after it is moved to an MB. See Move Process.

*3 Remote Request Stored bit, see the CANFD_CTL2 register.

*4 Code 1010 is not considered Tx and an MB with this code should not be aborted.

*5 6. For Tx MBs, the BUSY bit is ignored upon read, except when the CANFD_CFG.ABORTEN bit is set. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 16-6: Buffer Message Code – Tx Buffer

CODE Description	Tx Code Before Tx Frame	MB RTR	Tx Code After Successful Transmission	Comment
1000: INACTIVE – MB is not active.	INACTIVE	N/A	N/A	MB does not participate in arbitration process.
1001: ABORT – MB is aborted.	ABORT	N/A	N/A	MB does not participate in arbitration process.
1100: DATA – MB is a Tx Data Frame (MB RTR must be 0).	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.

Table 16-6: Buffer Message Code – Tx Buffer (Continued)

CODE Description	Tx Code Before Tx Frame	MB RTR	Tx Code After Successful Transmission	Comment
1100: REMOTE – MB is a Tx Remote Request Frame (MB RTR must be 1).	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
1110: TANSWER – MB is a Tx Response Frame from an incoming Remote Request Frame.	TANSWER	N/A	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching Process and Arbitration Process.

Substitute Remote Request (SRR)

SRR is a fixed recessive bit, only for the extended format. The SRR bit is set to one for transmission (Tx buffers) and is stored with the value received on the CAN bus for Rx receiving buffers. It is received as either recessive or dominant. If the CANFD module receives this bit as dominant, then it is interpreted as an arbitration loss.

When the bit is set, the recessive value is compulsory for transmission in extended format frames. When it is clear, dominant is not a valid value for transmission in extended format frames.

ID Extended Bit (IDE)

IDE identifies whether the frame format is standard or extended.

When the bit is set, the frame format is extended. When the bit is clear, the frame format is standard.

Remote Transmission Request (RTR)

The RTR bit affects the behavior of remote frames and is part of the reception filter:

When the bit is set, the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx, then incoming remote request frames may be stored. When the RTR bit is clear, the current MB has a data frame to be transmitted. In an Rx MB, it may be considered in matching processes.

See the *Message Buffer Code – Rx Buffer* table, the *Message Buffer Code – Tx Buffer* table, and the description of the `CANFD_CTL2.RRS` bit for additional details.

If the CANFD module transmits RTR as 1 (recessive) and receives it as 0 (dominant), it is interpreted as an arbitration loss. If RTR is transmitted as 0 (dominant) and received as 1 (recessive), the CANFD treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

NOTE: When configuring CAN FD frames, the RTR bit must be negated.

Length of Data in Bytes (DLC)

The 4-bit DLC field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space. In reception, DLC is written by the CANFD module, copied from the DLC (Data Length Code) field of the received frame. In transmission, DLC is written by the processor and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field. See the *Data Byte Validity* table.

Free-Running Counter Time Stamp (TIME STAMP)

The 16-bit TIME STAMP field is a copy of the free-running timer, captured for Tx and Rx frames when the beginning of the identifier field appears on the CAN bus.

Local Priority (PRIO)

The 3-bit PRIO field is used only when the CANFD_CFG.LPRIO_EN bit is set and only makes sense for Tx mailboxes. The PRIO field is not transmitted. It is appended to the regular ID to define the transmission priority. See the Arbitration Process section.

Frame Identifier (ID)

In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification for both receive and transmit, and the 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification for both receive and transmit.

Data Field (DATA BYTE 0 to 63)

Up to 64 bytes can be used for a data frame, depending on the size of payload selected for the Message Buffers.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE(n) is valid only if n is less than DLC as shown in the *Data Byte Validity* table.

Table 16-7: Data Byte Validity

DLC	Valid Data Bytes
0	None
1	Data Byte 0

Table 16-7: Data Byte Validity (Continued)

DLC	Valid Data Bytes
2	Data Byte 0 to 1
3	Data Byte 0 to 2
4	Data Byte 0 to 3
5	Data Byte 0 to 4
6	Data Byte 0 to 5
7	Data Byte 0 to 6
8	Data Byte 0 to 7
9	Data Byte 0 to 11
10	Data Byte 0 to 15
11	Data Byte 0 to 19
12	Data Byte 0 to 23
13	Data Byte 0 to 31
14	Data Byte 0 to 47
15	Data Byte 0 to 63

Message Buffer Memory Map

The CANFD memory buffers are allocated in memory according to the *8-Byte Message Buffers*, *16-Byte Message Buffers*, *32-Byte Message Buffers*, and *64-Byte Message Buffers* tables.

Table 16-8: 8-Byte Message Buffers

Address Offset	MBDSR = 00 (8-Byte Payload)
0x0080	MB0
0x0090	MB1
0x00A0	MB2
0x00B0	MB3
0x00C0	MB4
0x00D0	MB5
0x00E0	MB6
0x00F0	MB7
0x0100	MB8
0x0110	MB9
0x0120	MB10

Table 16-8: 8-Byte Message Buffers (Continued)

Address Offset	MBDSR = 00 (8-Byte Payload)
0x0130	MB11
0x0140	MB12
0x0150	MB13
0x0160	MB14
0x0170	MB15
0x0180	MB16
0x0190	MB17
0x01A0	MB18
0x01B0	MB19
0x10C0	MB20
0x10D0	MB21
0x10E0	MB22
0x10F0	MB23
0x0200	MB24
0x0210	MB25
0x0220	MB26
0x0230	MB27
0x0240	MB28
0x0250	MB29
0x0260	MB30
0x0270	MB31
0x0280	MB32
0x0290	MB33
0x02A0	MB34
0x02B0	MB35
0x02C0	MB36
0x02D0	MB37
0x02E0	MB38
0x02F0	MB39
0x0300	MB40
0x0310	MB41

Table 16-8: 8-Byte Message Buffers (Continued)

Address Offset	MBDSR = 00 (8-Byte Payload)
0x0320	MB42
0x0330	MB43
0x0340	MB44
0x0350	MB45
0x0360	MB46
0x0370	MB47
0x0380	MB48
0x0390	MB49
0x03A0	MB50
0x03B0	MB51
0x03C0	MB52
0x03D0	MB53
0x03E0	MB54
0x03F0	MB55
0x0400	MB56
0x0410	MB57
0x0420	MB58
0x0430	MB59
0x0440	MB60
0x0450	MB61
0x0460	MB62
0x0470	MB63

Table 16-9: 16-Byte Message Buffers

Address Offset	MBSR = 01 (16-Byte Payload)
0x0080	MB0
0x0098	MB1
0x00B0	MB2
0x00C8	MB3
0x00E0	MB4
0x00F8	MB5
0x0100	MB6

Table 16-9: 16-Byte Message Buffers (Continued)

Address Offset	MBSR = 01 (16-Byte Payload)
0x0128	MB7
0x0140	MB8
0x0158	MB9
0x0170	MB10
0x0188	MB11
0x01A0	MB12
0x01B8	MB13
0x01D0	MB14
0x01E8	MB15
0x0200	MB16
0x0218	MB17
0x0230	MB18
0x0248	MB19
0x0260	MB20
0x0280	MB21
0x0298	MB22
0x02B0	MB23
0x02C8	MB24
0x02E0	MB25
0x02F8	MB26
0x0310	MB27
0x0328	MB28
0x0340	MB29
0x0358	MB30
0x0370	MB31
0x0388	MB32
0x03A0	MB33
0x03B8	MB34
0x03D0	MB35
0x03E8	MB36
0x0400	MB37

Table 16-9: 16-Byte Message Buffers (Continued)

Address Offset	MBSR = 01 (16-Byte Payload)
0x0418	MB38
0x0430	MB39
0x0448	MB40
0x0460	MB41

Table 16-10: 32-Byte Message Buffers

Address Offset	MBDSR = 10 (32-Byte Payload)
0x0080	MB0
0x00A8	MB1
0x00D0	MB2
0x00F8	MB3
0x0120	MB4
0x0248	MB5
0x0170	MB6
0x0198	MB7
0x01C0	MB8
0x01E8	MB9
0x0210	MB10
0x0238	MB11
0x0280	MB12
0x02A8	MB13
0x02D0	MB14
0x02F8	MB15
0x0320	MB16
0x0348	MB17
0x0270	MB18
0x0298	MB19
0x03C0	MB20
0x03E8	MB21
0x0410	MB22
0x0438	MB23

Table 16-11: 64-Byte Message Buffers

Address Offset	MBSR = 11 (64-Byte Payload)
0x0080	MB0
0x0C8	MB1
0x0110	MB2
0x0158	MB3
0x01A0	MB4
0x01E8	MB5
0x0230	MB6
0x0280	MB7
0x02C8	MB8
0x0310	MB9
0x0258	MB10
0x03A0	MB11
0x03E8	MB12
0x0430	MB13

Memory Partition

When CAN FD is enabled, the CANFD RAM can be partitioned in two blocks of 512 bytes. Each block can accommodate a number of message buffers. This depends on the configuration of the size control bit fields in the `CANFD_FD_CTL` register as shown in the *RAM Partition* table.

Table 16-12: RAM Partition

RAM Block	Number of MBs with 8 Bytes*1	Size Control Bit Field	Number of MBs of Different Sizes, per Block
0	0 to 31	`CANFD_FD_CTL.MBDSIZR0`	MBDSIZR0=00, 32 MBs with 8 bytes payload MBDSIZR0=01, 21 MBs with 16 bytes payload MBDSIZR0=10, 12 MBs with 32 bytes payload, MBDSIZR0=11, 7 MBs with 64 bytes payload
1	32 to 63	`CANFD_FD_CTL.MBDSIZR1`	MBDSIZR1=00, 32 MBs with 8 bytes payload MBDSIZR1=01, 21 MBs with 16 bytes payload MBDSIZR1=10, 12 MBs with 32 bytes payload MBDSIZR1=11, 7 MBs with 64 bytes payload

*1 default range

When payload sizes of 16, 32, or 64 bytes are configured in some or all RAM blocks, the total number of MBs, and the respective number order may differ from the default configuration of 8 bytes. For example, if Block 0 is configured to 8 bytes payload, Block 1 to 16 bytes, the following table indicates how the message buffers are arranged into RAM.

Table 16-13: RAM Partition Example

RAM Block	Payload Size	Number of MBs in the RAM Block	MB Range
0	CANFD_FD_CTL[MBDSIZR0]=00, 8 bytes payload	32	0 to 31
1	CANFD_FD_CTL[MBDSIZR1]=01, 16 bytes payload	21	32 to 52

Rx FIFO Structure

When the `CANFD_CFG.RFEN` bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0-5) is used by the Rx FIFO engine.

The region 0x80 to 0x8C contains the output of the FIFO, which must be read by the processor as a message buffer. The FIFO output contains the oldest message that has been received but not yet read. The region 0x90–0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6 to 37) depending on the `CANFD_CTL2.RFFNUM` bit field setting, contains the ID filter table (configurable from 8 to 128 table elements), which specifies filtering criteria for accepting frames into the FIFO. Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for the `CANFD_CTL2.RFFNUM` bit field setting equal to zero.

The *Rx FIFO Structure* figure shows the Rx FIFO data structure.

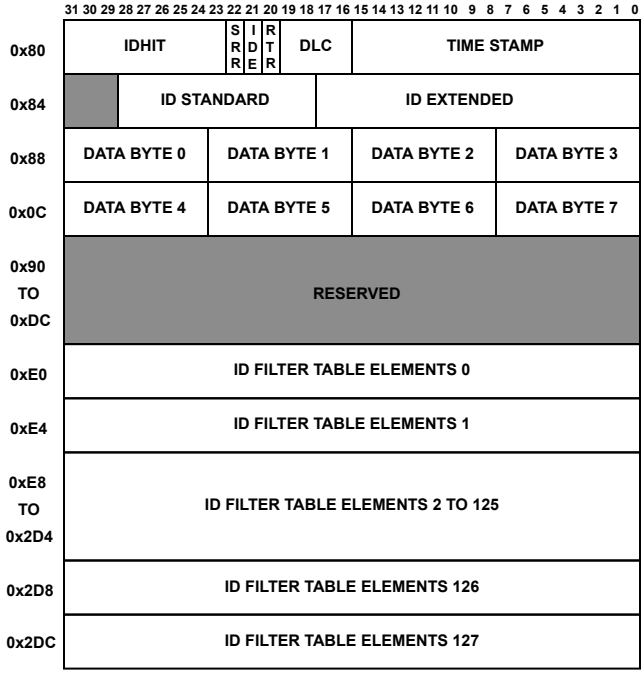


Figure 16-3: Rx FIFO Structure

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the `CANFD_CFG.IDAM` bit field setting.

The *ID Filter Table Structure* figure shows the three different formats of the ID table elements. All elements of the table must have the same format, as selected by the `CANFD_CFG.IDAM` bit field.

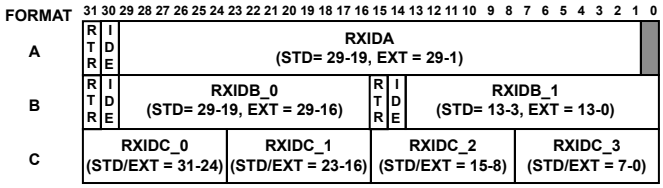


Figure 16-4: ID Filter Table Structure

The *Rx FIFO Filters* table shows the number of Rx FIFO filters configured with the `CANFD_CTL2.RFFNUM` bit field setting.

Table 16-14: Rx FIFO Filters

`CANFD_CTL2.RFFNUM`	**Number of Rx FIFO Filter Elements**	**Message Buffers Occupied by Rx FIFO and ID filter table**	**Remaining Mailboxes**	**Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks**	**Rx FIFO ID filter table elements affected by Rx FIFO global mask**
0x0	8	MB 0 to 7	MB 8 to 63	Elements 0 to 7	None
0x1	16	MB 0 to 9	MB 10 to 63	Elements 0 to 9	Elements 10 to 15
0x2	24	MB 0 to 11	MB 12 to 63	Elements 0 to 11	Elements 12 to 23

Table 16-14: Rx FIFO Filters (Continued)

CANFD_CTL2.RFFNUM	Number of Rx FIFO Filter Elements	Message Buffers Occupied by Rx FIFO and ID filter table	Remaining Mailboxes	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks	Rx FIFO ID filter table elements affected by Rx FIFO global mask
0x3	32	MB 0 to 13	MB 14 to 63	Elements 0 to 13	Elements 14 to 31
0x4	40	MB 0 to 15	MB 16 to 63	Elements 0 to 15	Elements 16 to 39
0x5	48	MB 0 to 17	MB 18 to 63	Elements 0 to 17	Elements 18 to 47
0x6	56	MB 0 to 19	MB 20 to 63	Elements 0 to 19	Elements 20 to 55
0x7	64	MB 0 to 21	MB 22 to 63	Elements 0 to 21	Elements 22 to 63
0x8	72	MB 0 to 23	MB 24 to 63	Elements 0 to 23	Elements 24 to 71
0x9	80	MB 0 to 25	MB 26 to 63	Elements 0 to 25	Elements 26 to 79
0xA	88	MB 0 to 27	MB 28 to 63	Elements 0 to 27	Elements 28 to 87
0xB	96	MB 0 to 29	MB 30 to 63	Elements 0 to 29	Elements 30 to 95
0xC	104	MB 0 to 31	MB 32 to 63	Elements 0 to 31	Elements 32 to 103
0xD	112	MB 0 to 33	MB 34 to 63	Elements 0 to 31	Elements 32 to 111
0xE	120	MB 0 to 35	MB 36 to 63	Elements 0 to 31	Elements 32 to 119
0xF	128	MB 0 to 37	MB 38 to 63	Elements 0 to 31	Elements 32 to 127

Remote Frame (RTR)

The RTR bit specifies if remote frames are accepted into the FIFO if they match the target ID. If the bit is set, remote frames are accepted and data frames are rejected. If the bit is clear, remote frames are rejected and data frames are accepted.

Extended Frame (IDE)

The IDE bit specifies whether extended or standard frames are accepted into the FIFO if they match the target ID. If the bit is set, extended frames are accepted and standard frames are rejected. If the bit is clear, extended frames are rejected and standard frames are accepted.

Rx Frame Identifier Format A (RXIDA)

Specifies the ID for as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (29 to 19) are used for frame identification. In the extended frame format, all bits are used.

Rx Frame Identifier Format B (RXIDB_0, RXIDB_1)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (29 to 19 and 13 to 3) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

Rx Frame Identifier Format C (RXIDC_1, RXIDC_2, RXIDC_3)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

Identifier Acceptance Filter Hit Indicator (IDHIT)

The 9-bit IDHIT field indicates which identifier acceptance filter is hit by the received message that is in the output of the Rx FIFO. See the Rx FIFO section for more information.

CANFD Processes

The following sections describe the how to transmit and receive CAN and CAN FD frames using the mailbox system.

Transmit Process

To transmit a CAN frame, the processor must prepare a message buffer for transmission by executing the following procedure:

1. Check if the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (1001) to the CODE field of the control and status (C/S) word to request an abortion of the transmission. Wait for the corresponding IFLAG bit to be asserted by polling the `CANFD_IFLG1` or `CANFD_IFLG2` register or by the interrupt request if enabled in the respective IMASK bit in the `CANFD_IMSK1` or `CANFD_IMSK2` register. Then, read back the CODE field to check if the transmission was aborted or transmitted (see the Transmission Abort Mechanism section).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control, and CODE fields of the C/S word to activate the MB. When `CANFD_CFG.FDEN` is set, also write the EDL, BRS and ESI bits.

When the MB is activated, it participates in the arbitration process and is eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission is larger than the respective MB payload size, the CANFD adds the necessary number of bytes with constant 0xCC pattern to complete the expected DLC.

At the end of the successful transmission, the value of the `CANFD_TMR` register is written into the TIME STAMP field, the CODE field in the C/S word is updated, both `CANFD_CRC` and `CANFD_FD_CRC` registers are updated, a status flag is set in the `CANFD_IFLG1` or `CANFD_IFLG2` register, and an interrupt is generated if allowed by the corresponding CANFD_IFLAGn bit. The new CODE field after transmission depends on the code that was used to activate the MB. See the *Message Buffer Code – Rx Buffer* table.

When the Abort feature is enabled (`CANFD_CFG.ABORTEN` is asserted), after the interrupt flag is asserted for a mailbox configured as transmit buffer, the mailbox is blocked. Therefore, the processor is not able to update it until the interrupt flag is disabled by the processor. This means that the processor must clear the corresponding CANFD_IFLAGn bit before starting to prepare this MB for a new transmission or reception.

Arbitration Process

The arbitration process scans the mailboxes, searching for the Tx mailbox that holds the message to be sent in the next opportunity. This mailbox is called the arbitration winner.

The scan starts from the lowest number mailbox and runs toward the higher ones. The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the `CANFD_CTL2.TXASDLY` field value.
- During the error delimiter field of a CAN frame.
- During the overload delimiter field of a CAN frame.
- When the winner is inactivated, and the CAN bus has still not reached the first bit of the intermission field.
- When there is a processor write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the intermission field.
- When CHI is in idle state and the processor writes to the C/S word of any MB.
- When CANFD module exits the bus off state.
- Upon leaving freeze mode or low power mode.

If the arbitration process does not manage to evaluate all mailboxes before the CAN bus has reached the first bit of the intermission field, the temporary arbitration winner is invalidated and the CANFD module does not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx mailboxes at the end of the scan according to the `CANFD_CTL1.LBUF` and `CANFD_CFG.LPRIOEN` bit settings.

Lowest Number Mailbox First

If the `CANFD_CTL1.LBUF` bit is enabled, the first (lowest number) active Tx mailbox found is the arbitration winner. The `CANFD_CFG.LPRIOEN` bit has no effect when the `CANFD_CTL1.LBUF` bit is enabled.

Highest-Priority Mailbox First

If the `CANFD_CTL1.LBUF` bit is disabled, then the arbitration process searches the active Tx mailbox with the highest priority. This mailbox frame with the highest priority has a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the arbitration value of the mailbox. The highest-priority Tx mailbox is the one that has the lowest arbitration value among all Tx mailboxes.

If two or more mailboxes have equivalent arbitration values, the mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on the `CANFD_CFG.LPRIOEN` bit.

Local Priority Disabled

When the CANFD_CFG.LPRIOEN is disabled, local priority is disabled, and the arbitration value is built in the exact sequence of bits transmitted in a CAN frame as shown in the *Arbitration Value Priority Disabled* table.

Table 16-15: Arbitration Value Priority Disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1bit)	IDE (1 bit)	Extended ID[17:0] (11 bits)	RTR (1bit)

Local Priority Enabled

If the CANFD_CFG.LPRIOEN is enabled, local priority is enabled. In this case, the mailbox PRIO field is included at the very left of the arbitration value as shown in the *Arbitration Value Priority Enabled table.*

Table 16-16: Arbitration Value Priority Enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1bit)	IDE (1 bit)	Extended ID[17:0] (11 bits)	RTR (1bit)

As the PRIO field is the most significant part of the arbitration value, mailboxes with low PRIO values have higher priority than mailboxes with high PRIO values, regardless of the rest of their arbitration values.

NOTE: The PRIO field is not part of the frame on the CAN bus and only affects the internal arbitration process.

Arbitration Completion

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and, after it is done, write access to the C/S word of the corresponding MB is blocked (if CANFD_CFG.ABORTEN is asserted). Write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the processor.
- The CANFD module enters freeze mode or bus off.
- The CANFD module loses the bus arbitration or there is an error during the transmission.

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules.

Arbitration Start and Stop Conditions

The arbitration process is triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. The `CANFD_CTL2.TXASDLY` value may be changed to optimize the arbitration start point.
- During CAN bus off state from TX_ERR_CNT = 124 to 128. The `CANFD_CTL2.TXASDLY` value may be changed to optimize the arbitration start point.
- During C/S write by the processor in bus idle mode. The first C/S write starts the arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, the Tx arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and the bus idle state starts, an arbitration process is triggered. In this case, the first and second C/S write in the bus idle state do not restart the arbitration process. It is possible that there is not enough time to finish arbitration in the wait for bus idle state and the next state is idle. In this case, the scan is not interrupted, and it is completed during the bus idle state. During this arbitration, a C/S write does not cause an arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon exiting freeze mode (first bit of the wait for bus idle state). If there is a re- synchronization during wait for bus idle state, the arbitration process restarts.

Arbitration process stops in the following situations:

- All mailboxes were scanned.
- A Tx active mailbox is found when the lowest buffer feature is enabled.
- Arbitration winner inactivation or abort during any arbitration process.
- There was not enough time to finish the Tx arbitration process (for example, when a deactivation was performed near the end of frame). In this case, the arbitration process is pending.
- There is an error or overload flag in the bus.
- There is a low power or freeze mode request in the idle state.

Arbitration is considered pending as described below:

- It was not possible to finish the arbitration process in time.
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer.
- Any C/S write if there is no Tx arbitration process in progress.
- Rx Match has just updated an Rx code to Tx code
- Entering the bus off state.

C/S write during arbitration has the following effect:

- If the C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If the C/S write is performed in an MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

Receive Process

To receive CAN frames into a mailbox, the processor must prepare the mailbox for reception by executing the following steps:

1. If the mailbox is active (either Tx or Rx), inactivate the mailbox (See Mailbox Inactivation), preferably with a safe inactivation (See Transmission Abort Mechanism).
2. Write the ID word.
3. Write the EMPTY code (0b0100) to the CODE field of the C/S word to activate the mailbox. No setup is required for the EDL, BRS, and ESI bits; they are overwritten by the respective bit fields in the received message.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the mailbox is updated by the move-in process (See Move-In) as follows:

1. The received data field (8 bytes at most for classical CAN message format and up to 64 bytes for the CAN FD message format) is stored.
2. The received ID field is stored.
3. The value of the `CANFD_TMR` register at the time of the second bit of the frame identifier field is written into the mailbox TIME STAMP field.
4. The received SRR, IDE, RTR, EDL, BRS, ESI, and DLC fields are stored.
5. The CODE field in the C/S word is updated according to the *Message Buffer Code – Rx Buffer* and *Message Buffer Code – Tx Buffer* tables.
6. A status flag is set in the `CANFD_IFLG1` or `CANFD_IFLG2` register and an interrupt is generated if allowed by the corresponding `CANFD_IMSK1` or `CANFD_IMSK2` register bit.

The recommended procedure for the processor to service (read) the frame received in a mailbox is:

The processor polls for frame reception by checking the status flag bit for the specific mailbox in the respective CANFD_IFLAGn register and not by polling the CODE field of that mailbox. Polling the CODE field does not work because, once a frame is received and the processor services the mailbox (by reading the C/S word followed by unlocking the mailbox), the CODE field does not return to EMPTY. It remains FULL, as explained in the *Message Buffer Code – Rx Buffer* table. If the processor tries to work around this behavior by writing to the C/S word to force an EMPTY code after reading the mailbox without a prior safe inactivation, a newly received frame matching the filter of that mailbox may be lost.

NOTE: Never poll by reading the C/S word directly from the mailboxes. Always read the `CANFD_IFLG1` or `CANFD_IFLG2` register.

The identifier field of the receive frame is always stored in the matching mailbox. Therefore, the contents of the ID field in a mailbox may change if the match was due to masking.

When the CANFD_CFG.SRXDIS bit is enabled, the CANFD module does not store frames transmitted by itself in any MB, even if it contains a matching Rx mailbox, and no interrupt flag or interrupt signal is generated. Otherwise, when the CANFD_CFG.SRXDIS bit is disabled, the CANFD module can receive frames transmitted by itself if there is a matching Rx mailbox.

To be able to receive CAN frames through the Rx FIFO, the processor must enable and configure the Rx FIFO during freeze mode (See Rx FIFO). Upon receiving the frames available in Rx FIFO interrupt (see the CANFD_IFLG1.MB05 bit description), the processor services the received frame using the following procedure:

1. Read the C/S word (optional: needed only if a mask was used for the IDE and RTR bits).
2. Read the ID field (optional: needed only if a mask was used).
3. Read the Data field.
4. Read the CANFD_RX_FIFO register (optional).
5. Clear the frames available in Rx FIFO interrupt by writing a one to the CANFD_IFLG1.MB05 bit to release the MB and allow the processor to read the next Rx FIFO entry.

When the CANFD_CFG.DMAEN bit is enabled, upon receiving a frame in the Rx FIFO, the CANFD_IFLG1.MB05 bit generates a DMA request and does not generate a processor interrupt (See Rx FIFO Under DMA Operation). The CANFD_IMSK1 bits in the Rx FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1. Read the C/S word (read 0x80 address, optional).
2. Read the ID field (read 0x84 address, optional).
3. Read all data bytes (start read at 0x88 address, optional).
4. Read the last data bytes (read 0x8C address is mandatory).

Matching Process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between mailboxes and FIFO filters. The matching starts from the lowest number MB toward the higher ones. When no match is found within the first structure, then the other is scanned subsequently. When the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- When the received frame is a remote frame, the start point is the CRC field of the frame.

- When the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame.
- When the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame.

When a matching ID is found in the FIFO table or in one of the mailboxes, the contents of the Rx SMB are transferred to the FIFO or to the matched mailbox by the move-in process. When any CAN protocol error is detected, then no match results are transferred to the FIFO or to the matched mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and the active Rx mailboxes (CODE is EMPTY, FULL, OVERRUN, or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The Rx SMB has the same structure as a mailbox. The reception structures (Rx FIFO or mailboxes) associated with the matching elements that had a successful comparison are the matched structures. The matching winner is selected at the end of the scan among those matched structures and depends on conditions described below.

Table 16-17: Matching Architecture

Structure	SMB.RTR	CTRL2.RSS	CTRL2.EACEN	MB.IDE	MB.RTR	MB.ID*1	MB.CODE
Mailbox	0	N/A	0	cmp*2	no_cmp*3	cmp_msk*4	
Mailbox	0	N/A	1	cmp_msk	cmp_msk	cmp_msk	
Mailbox	1	0	N/A	cmp	no_cmp	cmp	
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	
FIFO*5	N/A	N/A	N/A	cmp_msk	cmp_msk	cmp_msk	

*1 1. For mailbox structure, If SMB.IDE is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB.IDE is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on CANFD_CFG.IDAM.

*2 cmp: Compares the Rx SMB contents with the MB contents regardless the masks.

*3 no_cmp: The Rx SMB contents are not compared with the MB contents.

*4 cmp_msk: Compares the Rx SMB contents with MB contents taking into account the masks.

*5 SMB.IDE and SMB.RTR. are not considered when CANFD_CFG.IDAM is format C.

A reception structure is free-to-receive when any of the following conditions is satisfied:

- The CODE field of the mailbox is EMPTY.
- The CODE field of the mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the processor and unlocked as described in the Mailbox Lock Mechanism section).
- The CODE field of the mailbox is either FULL or OVERRUN and an inactivation is performed. See the Mailbox Inactivation section.
- The Rx FIFO is not full.

The scan order for mailboxes and the Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for mailboxes is affected by the CANFD_CFG.IRMQEN bit. If the CANFD_CFG.IRMQEN bit is disabled, the matching winner is the first matched mailbox regardless if it is free-to-receive or not. If the CANFD_CFG.IRMQEN bit is enabled, the matching winner is selected according to the priority below:

1. The first free-to-receive matched mailbox.
2. The last non free-to-receive matched mailbox.

It is possible to select the priority of scan between mailboxes and the Rx FIFO by using theCANFD_CTL2.MRPRIO bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive, then the Rx FIFO is the matching winner regardless of the scan for mailboxes.
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched for among mailboxes as described above.

If the selected priority is mailboxes first:

- If a free-to-receive matched mailbox is found, it is the matching winner regardless of the scan for the Rx FIFO.
- If no matched mailbox is found, then the matching winner is searched for in the scan for the Rx FIFO.

If both conditions above are not satisfied and a non free-to-receive matched mailbox is found, then the matching winner determination is conditioned by the CANFD_CFG.IRMQEN bit:

- If the CANFD_CFG.IRMQEN bit is disabled, the matching winner is the first matched mailbox.
- If the CANFD_CFG.IRMQEN bit is enabled, the matching winner is the Rx FIFO if it is a free-to- receive matched structure; otherwise, the matching winner is the last non free-to- receive matched mailbox.

See the *Matching Possibilities and Resulting Reception Structures* table for a summary of matching possibilities.

Table 16-18: Matching Possibilities and Resulting Reception Structures

RFEN	IRMQEN	MRPRIO	Matched in MB[*1]	Matched in FIFO[*2]	Reception Structure	Description
No FIFO, only MB, match is always MB first:						
0	0	X[*3]	None	_[*4]	None	Frame lost by no match
0	0	X	Free	–	FirstMB	
0	1	X	None	–	None	Frame lost by no match

Table 16-18: Matching Possibilities and Resulting Reception Structures (Continued)

RFEN	IRMQEN	MRPRIO	Matched in MB*1	Matched in FIFO*2	Reception Structure	Description
0	1	X	Free	–	FirstMB	
0	1	X	Not Free	–	LastMB	Overrun
FIFO enabled, no match in FIFO (as if FIFO does not exist):						
1	0	X	None	None	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMB	
1	1	X	Not Free	None	LastMB	Overrun
FIFO enabled, queue disabled:						
1	0	0	X	NotFull	FIFO	
1	0	0	None	Full	None	Frame lost by FIFO full (FIFO overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	Not Free	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO overflow)
1	0	1	Free	X	FirstMB	
1	0	1	Not Free	X	FirstMB	Overrun
FIFO enabled, queue enabled:						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	Not Free	Full	LastMB	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	Not Free	NotFull	FIFO	

Table 16-18: Matching Possibilities and Resulting Reception Structures (Continued)

RFEN	IRMQEN	MRPRIO	Matched in MB*1	Matched in FIFO*2	Reception Structure	Description
1	1	1	Not Free	Full	LastMB	Overrun

*1 In the Matched in MB column, the term "None" means the frame has not matched an MB (free-to-receive or non-free-to receive). The term "Free" means the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.

*2 In the Matched in FIFO column, the term "None" means the frame has not matched any filter for the Rx FIFO. It is as if the Rx FIFO does not exist (CANFD_CTL2.RFEN = 0). The term "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it. The term "Full" means that the frame has matched a FIFO filter, but the FIFO couldn't store it because the FIFO has no empty slots to receive it.

*3 The X denotes a don't-care condition.

*4 The – denotes a forbidden condition.

When a non-safe mailbox inactivation (See Mailbox Inactivation) occurs during the matching process and the mailbox inactivated is the temporary matching winner, then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted; it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the Rx FIFO is disabled, the `CANFD_CFG.IRMQEN` is enabled, there are two MBs with the same ID, and the CANFD starts receiving messages with that ID. If the two MBs are the second and the fifth in the array, then when the first message arrives, the matching algorithm finds the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive," so it keeps looking, finds MB number 5 and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive," so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full-featured FIFO) to allow more time for the processor to service the MBs. By programming more than one MB with the same ID, received messages are queued into the MBs. The CPU can examine the TIME STAMP field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID acceptance masks. The CANFD module supports individual masking per MB. During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a "don't care." The `CANFD_IMSK1` and `CANFD_IMSK2` registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the CANFD module is in freeze mode; otherwise, they are blocked by hardware.

The CANFD also supports an alternate masking scheme with only four mask registers (`CANFD_RX_FIFO_GMSK`, `CANFD_RX_MB_GMSK`, `CANFD_RX_14_MSK`, and `CANFD_RX_15_MSK`) for backward compatibility with legacy applications. This alternate masking scheme is enabled when `CANFD_CFG.IRMQEN` is enabled.

Receive Process Pretending Network Mode

Pretended networking mode adds specific wake up functionality in low power modes (doze mode). When pretended network (PN) mode is enabled by asserting the CANFD_CFG.PNETEN bit, the CANFD module continues processing Rx CAN messages under low power mode, able to detect specific wake up messages by filtering them against ID and payload target values using preselected matching criteria. Wake up functionality is not available for messages in CAN FD format. While in pretended networking mode, CAN FD format messages are ignored.

PN registers are in the 0x0B00–0x0B7C address range and can only be written only in freeze mode. These registers are used for writing PN configuration (both control and target values) prior entering into pretended networking mode, and for reading wake up flags and the received message ID and data when returning to normal mode after waking. The processor waits for the CANFD_CFG.LPMACK bit to be disabled before performing any access to CANFD PN registers.

PN control registers are CANFD_PN_CTL1 and CANFD_PN_CTL2. The control bit fields that configure the filtering criteria are:

- CANFD_PN_CTL1.PLFSEL: Payload filtering selection.
- CANFD_PN_CTL1.IDFSEL: ID filtering selection.
- CANFD_PN_CTL1.FCSEL: Filtering combination selection.

PN target values are:

- CANFD_FLTR_ID1.IDE: IDE target value used to filter the incoming message by its format (standard or extended).
- CANFD_FLTR_ID1.RTR: RTR target value used to filter the incoming message by its type (data or remote frame).
- CANFD_FLTR_DLC.HI and CANFD_FLTR_DLC.LO: Target DLC range used to filter the size of payload part of an incoming message.
- CANFD_FLTR_ID1.VALUE: ID target value used to filter the incoming message ID (equal to, smaller than or equal, greater than or equal, or the lower limit value in an ID range).
- CANFD_FLTR_ID2_IDMSK: ID target value used as the upper limit in an ID range.
- PL1: Payload target value used to filter the incoming message payload (equal to, smaller than or equal, greater than or equal, or the lower limit value in a payload range).
- PL2: Payload target value used as the upper limit in a payload range.

The IDE, RTR, ID, and payload filters have their respective masks. These masks determine which bits are considered in equality comparisons (1 in certain mask positions) and which ones are don't care (0 in other mask positions). ID and payload masks are used only for exact ID and/or exact payload comparisons.

The ID of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact ID value by detecting the equality between the ID field of the incoming message and the content of target CANFD_FLTR_ID1 register. The ID mask is used.
- A match with the maximum range of ID; in other words, any message with ID value smaller than or equal to the content of target CANFD_FLTR_ID11 register is accepted. The ID mask is not used.
- A match with the minimum range of ID; in other words, any message with ID value greater than or equal to the content of target CANFD_FLTR_ID1 register is accepted. The ID mask is not used.
- A match inside a range of IDs; in other words, any message with an ID value that is greater than or equal to the content of target CANFD_FLTR_ID1 register and smaller than or equal to the content of target CANFD_FLTR_ID2_IDMSK register is accepted. The ID mask is not used.

The above criteria for ID filtering must be coherent with CANFD_FLTR_ID1.IDE and CANFD_FLTR_ID1.RTR target values. Only Rx frames that match the respective IDE and RTR bits to the contents of FLT_IDE and FLT_RTR bit fields are compared. When range of IDs is selected (CANFD_PN_CTL1.IDFSEL = 11), both the CANFD_FLTR_ID1.VALUE field and the CANFD_FLTR_ID2_IDMSK register are referred to the same CANFD_FLTR_ID1.IDE and CANFD_FLTR_ID1.RTR bits.

The ID mask is applied only to the exact ID comparison filtering option (CANFD_PN_CTL1.IDFSEL = 00) to determine which bits are considered in the comparison. For the exact match option, the mask can select any bit within the ID field. For maximum range, minimum range and inside range comparisons, the ID mask is not considered.

The IDE and RTR masks are applied in both exact and range ID comparison filtering options to determine which bits are considered in comparison.

Similarly to the ID criteria, 64-bit data or payloads (PL) of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact payload value by detecting the equality between the payload field of the incoming message and the content of PL1 register. The payload mask is used.
- A match with the maximum range of payload–in other words, any message with payload value smaller than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match with the minimum range of payload–in other words, any message with payload value greater than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match inside a range of payloads–in other words, any message with a payload value that is greater than or equal to the content of PL1 register and smaller than or equal to the content of PL2 register is accepted. The payload mask is not used.

The above criteria for payload filtering must be coherent with CANFD_FLTR_DLC.HI and CANFD_FLTR_DLC.LO limit values. The payload of a Rx incoming message is filtered in accordance with the selected criteria only when the DLC value of the Rx incoming message is inside a DLC range:

- Greater than or equal to the CANFD_FLTR_DLC.LO (lower limit) and

- Lower than or equal to the CANFD_FLTR_DLC.HI (upper limit)

Conversely, a DLC value out of the specified range results in mismatch. By making CANFD_FLTR_DLC.LO = CANFD_FLTR_DLC.HI, only payloads of specified quantity of bytes will be filtered. DLC is not maskable.

When the inside range of payloads option is selected (CANFD_PN_CTL1.PLFSEL = 11), both PL1 and PL2 are considered with the 8-byte data length. All the data bytes excluded by the DLC of the received message are considered with value zero.

Payload mask is only used in the exact match option (CANFD_PN_CTL1.PLFSEL = 00) to select which bits or bytes in the 8-byte data field of both Rx incoming message and the contents of PL1 register are selected for matching. Mask length must be in accordance with the expected range of DLC values. For maximum range, minimum range, and inside range comparisons, the payload mask is not considered.

When a remote frame is received by the CANFD module and the CANFD_PN_CTL1.FCSEL bit is configured to select the payload comparison, the payload filtering is not considered and the comparison results in a mismatch.

Rx incoming messages can also be filtered based upon the quantity and rate of message reception, specifically:

- Several messages that match the filtering criteria for ID or payload a predefined quantity of times. This quantity can be configured in the 1 to 255 range. See the CANFD_CTL1 register.
- No message matching the filtering criteria for ID or payload up to a timeout trigger.

  That is, non-reception of a matching message for a defined quantity of time. See the CANFD_CTL2 register.

The CANFD module can generate a wakeup timeout event from an internal timer with associated comparator circuitry capable of generating a timeout flag when the counting reaches the predefined timeout value, as specified in the CANFD_PN_CTL2.MATCHTO bit field.

The above filtering criteria can be used together as follows:

- Message ID filtering only.
- Message ID filtering and payload filtering.
- Message ID filtering only occurring N times.
- Message ID filtering and payload filtering occurring N times.

The timeout counter runs concurrently with the reception filtering process. Both engines, timeout counter and message filtering, are independent. If an incoming message matches the selected filter criteria, the timeout counter keeps counting until the processor wakes up.

Conversely, if the timeout counter reaches the target value, then the message filtering process continue to filter incoming messages until the processor wakes up. The CANFD_WUM.MCNT field reports the number of matched messages that occurred under pretended networking mode up to the moment the processor wakes up.

Under pretended networking mode, a wakeup event that occurs sets the respective wake up flag (see the CANFD_WUM register.

- In case of a successful match with the selected filtering criteria, the CANFD_WUM.WUMF bit field.

- In case of a timeout trigger, the CANFD_WUM.WTOF bit field.

Either of these flags generates an interrupt to the processor if the respective mask bit is asserted (CANFD_PN_CTL1.WUMFMSK or CANFD_PN_CTL1.WTOFMSK).

There are four wake up message buffers (WMBs) for storing incoming messages in pretended networking mode. Up to four messages can be stored per the CANFD_WMB[n]_ID register. When the CANFD_PN_CTL1.MATCHCNT value is one, just one message is received if matching the filtering criteria, and this message is stored in the CANFD_WMB0 register. If the CANFD_PN_CTL1.MATCHCNT value is between two and four, CANFD_WMB1, CANFD_WMB2, and CANFD_WMB3 are used to store the second, third, and fourth matching messages, respectively. If CANFD_PN_CTL1.MATCHCNT is greater than four, the last four matching messages are stored in the WMBs; respecting the WMB index to indicate the arrival order, the latest is stored in CANFD_WMB3. Only the valid data bytes of the incoming match message is stored in data field of WMBs. The non-valid data bytes are read as zero. In case the of DLC = 0 and RTR = 1, the data field is filled with zero. In any of the above cases, the wake up interrupt is generated just when the filtering criteria is completed and the CANFD_PN_CTL1.WUMFMSK bit is set.

When a non-match wakeup event occurs (timeout or external) and the CANFD_WUM.MCNT value is equal or greater than four, the message stored in WMB0 does not have a valid content. The CANFD_WMB0 is used as the buffer for the current message on the CAN bus. Messages received during pretended networking mode do not have timestamps and respective field in the WMB structure must be ignored.

Under low-power mode (doze or stop), all processes are shut down except for the PN functionality in the PE submodule, which is kept clocked by the oscillator clock. The CANFD continues to receive Rx incoming messages and just compares them against the predefined target values and according to the selected filtering criteria. The matching, arbitration, move-in, and move-out processes, normally available in normal mode, are not performed under pretended networking mode.

Under pretended networking, the CANFD reacts to messages on the CAN bus in the same manner as in normal mode (generates acknowledge bits, detects and counts errors, etc.).

Move Process

There are two types of move process: move-in and move-out.

Move-In

The move-in process is the copy of a message received by an Rx SMB to an Rx mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the CANFD_RX_FIFO register. Each Rx SMB has its own move-in process, but only one is performed at a given time as described below. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (See Matching Process) and all of the following conditions are true:

- The CAN bus has reached or let past either of the following:
  - The second bit of the Intermission field next to the frame that carried the message that is in the Rx SMB
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB

- There is no ongoing matching process.
- The destination mailbox is not locked by the processor.
- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time, both are performed and the newest substitutes the oldest.

The term pending *move-in* is used throughout the documentation and stands for a *move-to- be* that still does not satisfy all the previously mentioned conditions.

The move-in is canceled and the Rx SMB can receive another message when any of the following conditions is satisfied:

- The destination mailbox is inactivated after the CAN bus has reached the first bit of intermission field next to the frame that carried the message and its matching process has finished.
- There is a previous pending move-in to the same destination mailbox.
- The Rx SMB is receiving a frame transmitted by the CANFD itself and the self-reception is disabled (`CANFD_CFG.SRXDIS` bit is enabled)
- Any CAN protocol error is detected.

Note that the pending move-in is not canceled if the CANFD module enters freeze or low-power mode. The pending move-in stays on hold waiting for exit from Freeze or low-power mode and to be unlocked. If an MB is unlocked during freeze mode, the move-in happens immediately.

The move-in process is the execution by the CANFD module of the following steps:

1. Push `CANFD_RX_FIFO.IDHIT` into the `CANFD_RX_FIFO` FIFO if the message is destined to the Rx FIFO.
2. Read all data words from the Rx SMB according to the selected payload size for the Rx storage element.
3. Write all data words to the Rx mailbox according to the selected payload size for the Rx storage element. If the data size of the storage element is smaller than the original payload size described in the message DLC field, the payload is truncated and the high order bytes that do not fit the destination size are lost.
4. Read the C/S and ID words from the Rx SMB.
5. Write C/S and ID words to the Rx mailbox and update the CODE field.

The move-in process is not atomic, in such a way that it is immediately canceled by the inactivation of the destination mailbox (See Mailbox Inactivation). In this case, the mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO message buffer, in which case the process cannot be canceled.

The BUSY bit (least significant bit of the CODE field) of the destination message buffer is asserted while the move-in is being performed to alert the processor that the message buffer content is temporarily incoherent.

Move-Out

The move-out process is the copy of the content from a Tx mailbox to the Tx SMB when a message for transmission is available (See Arbitration Process). The move-out occurs in the following conditions:

- The first bit of intermission field.
- During the bus off state when TX error counter is in the 124 to 128 range.
- During bus idle state.
- During the wait for bus idle state.

The move-out process is not atomic. Only the processor has priority to access the memory concurrently out of bus idle state. In bus idle, the move-out has the lowest priority to the concurrent memory accesses.

Data Coherence

In order to maintain data coherency and proper operation of the CANFD module, the processor must obey the rules described in the Transmit Process and Receive Process sections.

Transmission Abort Mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the processor if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- The `CANFD_CFG.ABORTEN` bit is enabled.
- The first processor action must be the writing of abort code (1001) into the CODE field of the C/S word.

Active MBs configured for transmission must be aborted first before they can be updated. If the abort code is written to a mailbox that is currently being transmitted or to a mailbox that was already loaded into the Tx SMB for transmission, the write operation is blocked, and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The CANFD module loses the bus arbitration.
- There is an error during the transmission.
- The CANFD module is put into freeze mode.
- The CANFD enters the bus off state.
- There is an overload frame.

When none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the respective `CANFD_IFLG1` or `CANFD_IFLG2` register, and an interrupt to the processor is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, when one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the

interrupt flag is set in the respective CANFD_IFLG1 or CANFD_IFLG2 register, and an interrupt is (optionally) generated.

When the processor writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated, and the interrupt flag is set. The processor just needs to read the abort code to make sure the active MB was safely inactivated. Although the CANFD_CFG.ABORTEN bit is asserted and the processor wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- The processor checks the corresponding IFLAG and clears it, if asserted.
- The processor writes 1001 into the CODE field of the C/S word.
- The processor waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- The processor reads the CODE field to check if the frame was either transmitted (CODE = 1000) or aborted (CODE = 1001).
- It is necessary to clear the corresponding IFLAG to allow the MB to be reconfigured.
- It is necessary to reconfigure the EDL, BRS, and ESI fields of the aborted MB before transmitting it again.

Mailbox Inactivation

Inactivation is a mechanism provided to protect the mailbox against updates by the CANFD internal processes, thus allowing the processor to rely on mailbox data coherence after having updated it, even in normal mode.

Inactivation of transmission mailboxes must be performed only when the CANFD_CFG.ABORTEN bit is deasserted.

If a mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated.

To inactivate a mailbox, the processor must update its CODE field to INACTIVE (either 0000 or 1000).

Because the processor is not able to synchronize the CODE field update with the CANFD internal processes, an inactivation has the following consequences:

- A frame on the bus that matches the filtering of the inactivated Rx mailbox may be lost without notice, even if there are other mailboxes with the same filter.
- A frame containing the message within the inactivated Tx mailbox may be transmitted without setting the respective IFLAG.

To perform a safe inactivation and avoid the above consequences for Tx mailboxes, the processor must use the transmission abort mechanism .

The inactivation automatically unlocks the mailbox (See Mailbox Lock Mechanism).

NOTE: Message buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region. The processor must maintain data coherency in the FIFO region when the CANFD_CFG.RFEN bit is enabled.

Mailbox Lock Mechanism

Other than mailbox inactivation, the CANFD module has another data coherence mechanism for the receive process. When the processor reads the C/S word of an Rx MB with codes FULL or OVERRUN, the CANFD assumes that the processor wants to read the whole MB in an atomic operation and therefore it sets an internal lock flag for that MB. The lock is released when the processor reads the CANFD_TIMING register (global unlock operation), or when it reads the C/S word of another MB regardless of its code. A processor write into the C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending move and potentially may lose a received message. The MB locking prevents a new frame from being written into the MB while the processor is reading it.

NOTE: The locking mechanism applies only to Rx MBs that are not part of the FIFO and have a code other than INACTIVE (0000) or EMPTY (0100). Tx MBs can not be locked. When the CANFD_CFG.IRMQEN bit is disabled, reading the C/S word locked the MB even if it was EMPTY.

When the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and the CANFD module has already received and stored messages into these two MBs. If the processor then attempts to read MB number five and at the same time another message with the same ID is arriving, then when the processor reads the C/S word of MB number five, this MB is locked. The new message arrives, and the matching algorithm finds out that there are no “free-to-receive” MBs, it decides to override MB number five. However, this MB is locked, so the new message cannot be written there. It remains in the Rx SMB waiting for the MB to be unlocked, and only then will be written to the MB.

When the MB is not unlocked in time and yet another new message with the same ID arrives, the new message overwrites the message in the Rx SMB and there is no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the Rx SMB to the MB, the BUSY bit on the CODE field is enabled. If the processor reads the C/S word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is cleared.

NOTE: If the BUSY bit is enabled or if the MB is empty, then reading the C/S word does not lock the MB.

Inactivation takes precedence over locking. If the processor inactivates a locked Rx MB, then the lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the Rx SMB is transferred to the MB. An MB is unlocked when the processor reads the CANFD_TMR register, or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both normal and freeze modes.

An unlock during normal or freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode, and it takes place only when the module resumes to normal or freeze mode.

Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the CANFD_CFG.RFEN bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the CANFD module that did not have the FIFO feature.

NOTE: The Rx FIFO must not be enabled when the CAN FD feature is enabled.

The Rx FIFO is 6 messages deep. The memory region occupied by the FIFO structure (both message buffers and FIFO engine) is described in the Rx FIFO Structure section. The processor can read the received messages sequentially, in the order they were received, by repeatedly reading a message buffer structure at the output of the FIFO.

The frames available bit (CANFD_IFLG1.MB05)is enabled when there is at least one frame available to be read from the Rx FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the processor reads the message (accessing the output of the FIFO as a message buffer) and the CANFD_RX_FIFO register, then clears the interrupt. If there are more messages in the Rx FIFO, the act of clearing the interrupt updates the output of the FIFO with the next message and updates the CANFD_RX_FIFO register with the attributes of that message, reissuing the interrupt to the processor. Otherwise, the flag remains negated. The output of the Rx FIFO is only valid while the CANFD_IFLG1.MB05 bit enabled.

The Rx FIFO warning bit (CANFD_IFLG1.MB06) is enabled when the number of unread messages in the Rx FIFO is increased to five from four due to the reception of a new message, meaning that the Rx FIFO is almost full. The flag remains enabled until the processor clears it.

The Rx FIFO overflow bit (CANFD_IFLG1.MB07) is enabled when an incoming message is lost because the Rx FIFO is full. The CANFD_IFLG1.MB07 bit is not set when the Rx FIFO is full, and the message is captured by a mailbox. The CANFD_IFLG1.MB07 remains set until the processor clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated when an IFLAG bit asserts and the corresponding mask bit is enabled.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing workload. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to the CANFD_CTL2.RFFNUM bit setting, which is configurable to one of the following formats:

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

NOTE: A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding identifier acceptance filter hit indicator (IDHIT) that can read in the IDHIT field from C/S word. Another way the processor can obtain this information is by accessing the CANFD_RX_FIFO register. The CANFD_RX_FIFO.IDHIT field refers to the message at the output of the FIFO

and is valid while the CANFD_IFLG1.MB05 flag is enabled. The CANFD_RX_FIFO register must be read only before clearing the flag, to guarantee that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the individual mask registers (CANFD_IMSK1 and CANFD_IMSK2), according to the setting of the CANFD_CTL2.RFFNUM bit, allowing very powerful filtering criteria to be defined. If the CANFD_CFG.IRMQEN bit is disabled, then the FIFO filter table is affected by the CANFD_RX_FIFO_GMSK register value.

Rx FIFO Under DMA Operation

The Rx FIFO can support DMA. Using the Rx FIFO with the DMA feature is enabled by enabling both CANFD_CFG.RFEN and CANFD_CFG.DMAEN bits. The DMA controller reads the received message by reading a message buffer structure at the FIFO output port at the 0x80–0x8C address range.

The reset value of the CANFD_CFG.DMAEN bit is zero. When the CANFD_CFG.DMAEN is enabled, the processor must not access the FIFO output port address range. Before enabling the CANFD_CFG.DMAEN bit, the processor must service the IFLAGs asserted in the Rx FIFO region. Otherwise, these IFLAGs may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before disabling the CANFD_CFG.DMAEN, the processor must perform a clear FIFO operation.

The Frames available in Rx FIFO bit (CANFD_IFLG1.MB05) is asserted when there is at least one frame available to be read from the FIFO; consequently, a DMA request is generated simultaneously. Upon receiving the request, the DMA controller reads the message (accessing the output of the FIFO as a message buffer). The DMA reading process must end by reading address 0x8C, which clears the CANFD_IFLG1.MB05 bit and updates both the FIFO output with the next message (if FIFO is not empty) and the CANFD_RX_FIFO register with the attributes of the new message. If there are more messages stored in the FIFO, the CANFD_IFLG1.MB05 bit is re-enabled, and another DMA request is issued.

NOTE: The CANFD_RX_FIFO register contents cannot be read after the DMA completes the FIFO read. The IDHIT information is also available in the C/S word at address 0x080.

The CANFD_IFLG1.MB06 and CANFD_IFLG1.MB07 flags are not used when the DMA feature is enabled.

When the CANFD module is working with DMA, the processor does not receive any Rx FIFO interrupts and must not clear the related IFLAGs. In addition, the related IMASKs are not used to mask the generation of DMA requests.

There is no dedicated DMA channel for reading the CANFD FIFO, however, the MDMA along with the TRU can read from the FIFO RAM. This works by configuring the CANFD module to generate a trigger whenever the CANFD_IFLG1.MB05 bit is set. The MDMA is configured to wait for the trigger from the CANFD module and on receiving the trigger, read from the CANFD FIFO start address at 0x80 offset. Once 16 bytes of FIFO data are read, the CANFD_IFLG1.MB05 bit is cleared.

Only MDMA2 and MDMA6 streams can access the CANFD FIFO.

Clear FIFO Operation

When the CANFD_CFG.RFEN is enabled, the clear FIFO operation empties the FIFO contents. With the CANFD_CFG.RFEN bit enabled, the clear FIFO occurs when the processor writes a one to the

CANFD_IFLG1.MB01 bit. This operation only can only occur in freeze mode and is blocked by hardware in other modes. This operation does not clear the FIFO IFLAGs; consequently, the processor must service all FIFO IFLAGs before executing the clear FIFO task.

When the Rx FIFO is working with DMA, the clear FIFO operation clears the CANFD_IFLG1.MB05 flag and the DMA request is canceled.

NOTE: The clear FIFO operation does not clear IFLAGs, except when the CANFD_CFG.DMAEN bit is enabled. When the CANFD_CFG.DMAEN bit is enabled, only the CANFD_IFLG1.MB05 flag is cleared.

CAN Protocol Features

This section describes the CAN protocol related features.

CAN FD Frames

The ISO 11898-1 standard specifies the classical Frame format compliant to ISO 11898-1 (2003) and introduces the CAN flexible data rate frame format. The classical frame format allows bit rates up to 1 Mbit/s and payloads up to 8 bytes per frame. The flexible data rate frame format allows bit rates higher than 1 Mbit/s and payloads longer than 8 bytes per frame. The CANFD module can receive and transmit CAN FD messages interleaved with classical CAN messages.

There are three additional control bits in the CAN FD frame:

- The extended data length (EDL) bit enables a longer data payload with different data length coding.
- The bit rate switch (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame.
- The error state indicator (ESI) flag is transmitted dominant by error active nodes, and recessive by error passive nodes.

There are no remote frames in the CAN FD format. A message configured to transmit a remote frame is always sent out in the classical CAN format. When a FD frame is received and matches a mailbox, the RTR bit in the receiving message buffer is cleared. The RTR bit is considered in only in classical CAN frames.

CAN FD messages can be formatted as long frames, in which the data field exceeds 8 bytes and may range from 12 up to 64 bytes. CAN FD messages can also be configured to support bit rate switching, where the control field, the data field, and the CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

Messages in the classical CAN format are limited to transport a maximum payload of 8 bytes at nominal rate. The *CAN Message Format* figure illustrates the message formats for classical and FD frames with either a standard or extended ID.

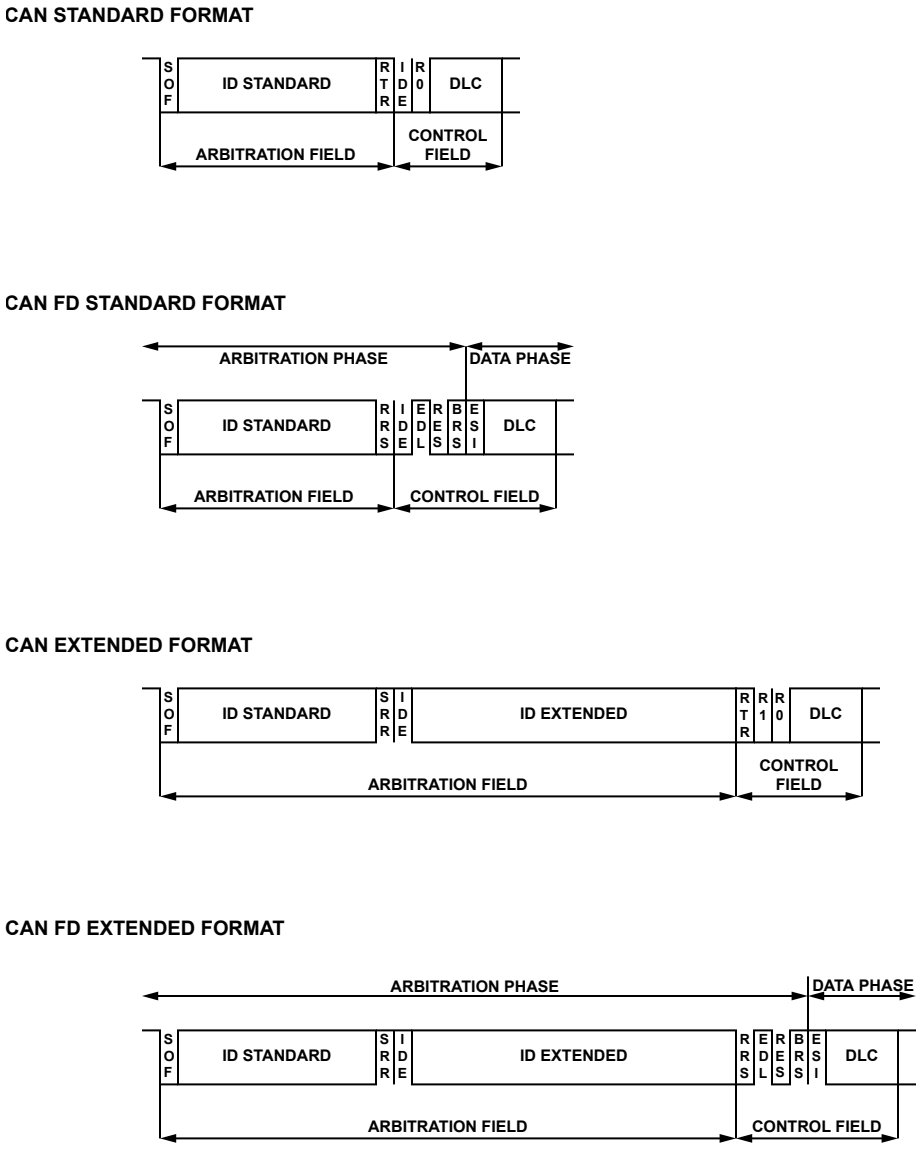


Figure 16-5: CAN Message Format

The ability to receive and transmit CAN FD messages is enabled by the `CANFD_CFG.FDEN` bit. Either a recessive R0 bit in CAN frames with 11-bit identifiers or a recessive R1 bit in CAN frames with 29-bit identifiers are decoded as an EDL bit (not a reserved one). A CAN FD frame is recognized by a recessive EDL bit, while a classical CAN frame is recognized by a dominant EDL bit. The BRS bit specifies whether this frame switches the bit rate in its data phase. A long frame is decoded according to the DLC field value.

CANFD messages can be transmitted with two different bit rates. The first part of a CAN FD frame, from the start of frame (SOF) bit until the bit rate switch (BRS) bit, also called the arbitration phase, is transmitted with the nominal bit rate based on a set of nominal CAN bit timing configuration values. The second part, from the BRS bit until the CRC delimiter bit, also named the data phase, is transmitted with the data bit rate defined by a second set of CAN data bit timing configuration values. Finally, from the CRC delimiter until the intermission bits, the transmission resumes to nominal bit rate. In CAN FD frames with bit rate switching, the bit timing is changed inside the frame at the sample point of the BRS bit if this bit is recessive. Before the BRS bit, in the CAN FD arbitration

phase, the nominal CAN bit timing defined by the CANFD_TIMING register and by the CANFD_CTL1 register for backward compatibility. Upon detecting a recessive BRS bit, the CAN data bit timing is used as defined by the CANFD_FD_TIMING register.

NOTE: If the length of the time quantum in the nominal bit timing and the length of the time quantum in the data bit timing are not identical, a quantization error of up to one time quantum of the arbitration phase may be present as a phase error. This situation can occur after the switch from arbitration to data phase and lasts until the next synchronization event. Thus, the length of the time quantum should be the same in nominal and data bit timing to minimize the chance of error frames on the CAN bus, and to optimize the clock tolerance in networks that use CAN FD frames.

The CANFD_FD_CTL.BRSEN bit enables the transmission of all frames with bit rate switching if the BRS bit in the selected Tx MB is set. When the bit is cleared, the transmission is performed at nominal rate regardless of the BRS bit value. The CANFD_FD_CTL.BRSEN bit can be written any time but takes effect only for the next message transmitted or received.

The nominal bit timing is resumed at either the sample point of the CRC delimiter bit or when an error is detected, whichever occurs first. The *CAN FD Message Bit Rate Switching* figure describes the mechanism for entering and leaving the data phase when the BRS bit is recessive.

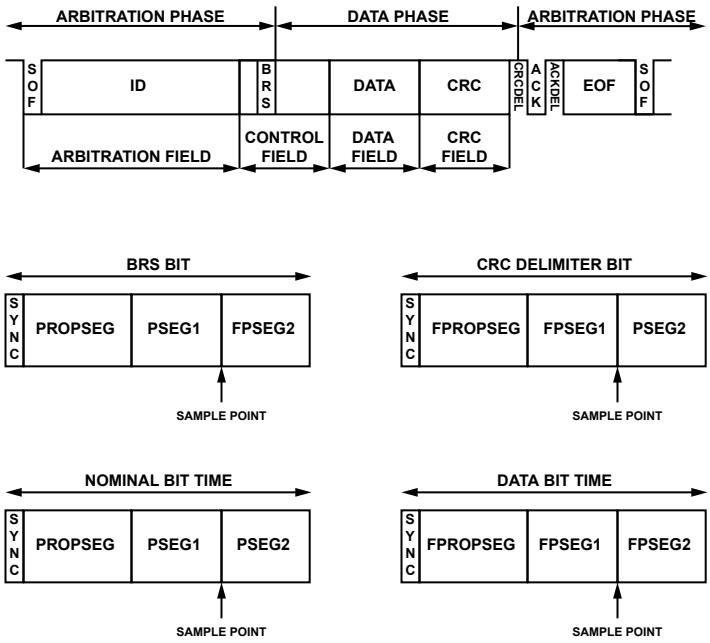


Figure 16-6: CAN FD Message Bit Rate Switching

NOTE: In classical CAN frames, the CRC delimiter is one single recessive bit. In CAN FD frames, the CRC delimiter may consist of one or two recessive bits. The CANFD module sends only one recessive bit as the CRC delimiter, but it accepts two recessive bits before the edge from recessive to dominant that starts the acknowledge slot. As a receiver, the CANFD module sends its acknowledge bit after the first CRC delimiter bit. In CAN FD frames, the CANFD module accepts a two-bit dominant ACK slot as a valid ACK to compensate for phase shifts between the receivers.

The value of the ESI bit is determined either by the error state of the transmitter at the start of the transmission, if the frame is originated in the CANFD node, or by the original transmitting node if the CANFD module is acting as a gateway for the message. If the transmitter is error passive, the ESI is transmitted recessive; otherwise, it is transmitted dominant.

There are different CRC polynomials for different CAN frame formats:

- The first polynomial, CRC_15, is for all frames in classical CAN format.
- The second, CRC_17, is for frames in CAN FD format with a data field up to 16 bytes long.
- The third, CRC_21, is for frames in CAN FD format with a data field longer than 16 bytes.

Each polynomial results in a hamming distance of 6. At the start of the frame, all three CRC polynomials are calculated concurrently. The CRC sequence to be transmitted is selected by the values of the EDL bit and the DLC bit field. When receiving a message, the CANFD module decodes the EDL and DLC bits to select the proper CRC polynomial to check for a CRC error.

In CAN FD format frames, stuff bits are included in the bit stream for CRC calculation. In classical CAN format frames, stuff bits are not included. After the transmission of the last bit relevant to the CRC calculation, the CANFD_FD_CRC register stores the calculated CRC for the transmitted message, with the adequate length for the type of message, for both CAN FD and non-FD messages. The CANFD_CRC register reports a valid CRC for classical CAN messages only.

In CAN FD format frames, the CAN bit stuffing method is changed for the CRC sequence so that the stuff bits are inserted at fixed positions. When the CANFD module is transmitting a CAN FD frame, a fixed stuff bit is inserted just before the first bit of the CRC sequence, even if the last bits of the preceding field do not fulfill the CAN stuff condition. Additional stuff bits are inserted after each fourth bit of the CRC sequence. The value of any fixed stuff bit is the inverse value of its preceding bit. When the CANFD module is receiving a CAN FD frame, it discards the fixed stuff bits from the bit stream for the CRC check. A stuff error is detected if the fixed stuff bit has the same value as its preceding bit.

The CANFD module detects errors in CAN FD frames the same way as in Classical CAN frames. The error counters CANFD_ECR.RXERRCNT and CANFD_ECR.TXERRCNT in the CANFD_ECR register accumulate the counts of Rx and Tx errors, respectively, for both FD and non-FD frames indistinctly. There are two extra error counters (CANFD_ECR.RXERRCNTF and CANFD_ECR.TXERRCNTF) that accumulate Rx and Tx errors occurring in the data phase of CAN FD frames with the BRS bit set. The rules for updating the error counters are the same for both CAN FD and non-FD frames.

The error flags CANFD_ESR1.B1ERR, CANFD_ESR1.B0ERR, CANFD_ESR1.ACKERR, CANFD_ESR1.CRCERR, CANFD_ESR1.FRMERR, and CANFD_ESR1.STFERR report errors in both CAN FD and non-FD frames. They also generate the ERRINT interrupt if the CANFD_CTL1.ERRMSK bit is enabled. The CANFD_ESR1 register has additional error flags (CANFD_ESR1.B1ERRF, CANFD_ESR1.B0ERRF, CANFD_ESR1.CRCERRF, CANFD_ESR1.FRMERRF, and CANFD_ESR1.STFERRF) to individually indicate the occurrence of errors in the data phase of CAN FD frames with the BRS bit set. There is no ACKERR detected in the data phase of a CAN FD frame. Fault confinement status reported with the CANFD_ESR1.FLTCONF bit is the same for both CAN FD and classical CAN frames, and is based on CANFD_ECR.RXERRCNT and

CANFD_ECR.TXERRCNT error counters only. Information contained in CANFD_ECR.RXERRCNTF and CANFD_ECR.TXERRCNTF counters may be considered as status to help detect the error nature related to the bit rate value. The *CANFD State Related Bit Encoding* table shows how the CANFD state is related to the CANFD_ESR1.SYNC, CANFD_ESR1.IDLE, CANFD_ESR1.TXINPROG, and CANFD_ESR1.RXINPROG bits.

Table 16-19: CANFD State Related Bit Encoding

SYNC	IDLE1	TXINPROG	RXINPROG	CANFD State
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

When the CANFD is in the data phase, either transmitting or receiving a CAN FD message, and detects an error, it immediately switches back to the arbitration phase and to the nominal rate to start an error flag

Resynchronization and hard synchronization occur in CAN FD frames in the same way as in classical CAN frames. A hard synchronization is also performed at the recessive-to-dominant edge from EDL to R0 in CAN FD format frames. The CANFD does not resynchronize while transmitting in the CAN FD data phase.

Transceiver Delay Compensation

The CAN FD protocol allows the transmission and reception of data at a higher bit rate than the nominal rate used in the arbitration phase when the BRS bit of the message is set. This feature enables the use of rates up to 8 Mbps.

During the data phase of a CAN FD frame, the transmitter detects a bit error if it cannot receive its own latest transmitted bit at the sample point of that bit. When bit rate switching is enabled, the length of the CAN bit time in the data phase can become shorter than the transceiver's loop delay, thus impeding the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

The CANFD module supports an optional transceiver delay compensation (TDC) mechanism that defines a secondary sample point where the transmitted bit is correctly compared with the received bit to check for bit errors.

The TDC mechanism is enabled by the CANFD_FD_CTL.TDCOMPEN bit and is effective only during the data phase of CAN FD frames having the BRS bit set. It has no effect either on non-FD frames, or on FD frames transmitted at normal bit rate. The TDC mode is active from the sample point of the BRS bit until the sample point of the CRC delimiter bit, provided the respective message under transmission has the BRS bit set. When it is active, a comparison is done between the real received bit and the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

NOTE: The actual value of the CRC delimiter bit is disregarded by transmitters using the TDC mechanism. A global error at the end of the CRC field will cause the receivers to send error frames that the transmitter will detect during an acknowledge or end of frame.

For every transmitted FD frame having the BRS bit set, the delay measurement is triggered by the transition from the recessive EDL bit to the dominant R0 bit, as shown in the *Transceiver Loop Delay Measurement* figure.

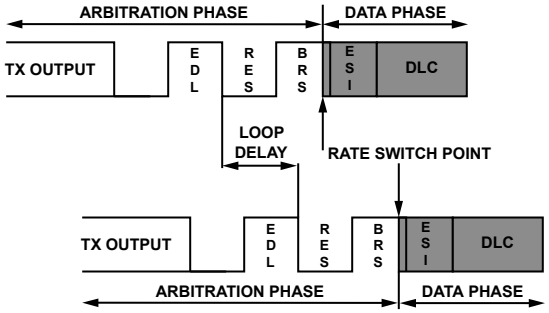


Figure 16-7: Transceiver Loop Delay Measurement

The loop delay is measured in PE clock periods (CANCLK) , from the transmitted EDL-R0 edge to the received EDL-R0 edge. The position of the secondary sample point is defined by the measured loop delay time added to an offset value specified with the `CANFD_FD_CTL.TDCOMPOFF` bit. The `CANFD_FD_CTL.TDCOMPVAL` bit field stores the result of this calculation. The `CANFD_FD_CTL.TDCOMPVAL` value saturates at the maximum value of 15 CANCLK when the delay measurement is too long.

The measured loop delay is not enough to be used to define the secondary sample point because it relates to the CAN bit edges. The transceiver delay compensation offset (`CANFD_FD_CTL.TDCOMPOFF`) shifts the secondary sample point from the edge to an intermediate point inside the bit time (for example, half of the bit time in the data phase), far away from its edges. Therefore, the `CANFD_FD_CTL.TDCOMPOFF` value cannot be larger than the CAN bit duration in the data phase.

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every Tx CAN bit, the transmitted Tx bit value is temporarily stored in a buffer and a time countdown based on the `CANFD_FD_CTL.TDCOMPVAL` value is started which ends with the comparison of the received Rx bit (delayed by the external loop delay plus the specified offset) with the stored Tx bit. If a bit error is detected at the secondary sample point, the CANFD module issues an error flag to the CAN bus at the next sample point.

During the arbitration phase, the delay compensation is always disabled. The maximum delay that can be compensated by the transceiver delay compensation during the data phase is 3 CAN bit times - 2 Tq. Beyond this limit, the `CANFD_FD_CTL.TDCOMPFAIL` flag is set to indicate when the TDC mechanism is out of range and unable to compensate the transceiver loop delay.

Remote Frames

A remote frame is a special kind of frame. The processor can program a mailbox to be a remote request frame by configuring the mailbox to transmit with the RTR bit set to one. After the remote request frame is transmitted successfully, the mailbox becomes a receive message buffer, with the same ID as before.

When a remote request frame is received by the CANFD module, it is treated in one of the following ways, depending on the `CANFD_CTL2.RRS` and `CANFD_CFG.RFEN` bits:

- When RRS is disabled, the frame ID is compared to the IDs of the transmit message buffers with the CODE field 1010. If there is a matching ID, then this mailbox frame is transmitted. Note that if the matching mailbox has the RTR bit set, then the CANFD module transmits a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame match. In the case that a remote request frame is received and matches a mailbox, this message buffer immediately enters the internal arbitration process, but is considered a normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- When RRS is enabled, the frame ID is compared to the IDs of the receive mailboxes with the CODE field 0100, 0010, or 0110. If there is a matching ID, then this mailbox stores the remote frame in the same fashion of a data frame. No automatic remote response frame is generated. The mask registers are used in the matching process.
- When RFEN is asserted, the CANFD does not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it is stored in the FIFO and presented to the processor. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote request frames are considered normal frames,and generate a FIFO overflow when a successful reception occurs, and the FIFO is already full.

NOTE: There is no remote frame in the CAN FD format. The RTR bit is replaced by a fixed dominant RRS bit. The CANFD receives and transmits remote frames in the classical CAN format.

Overload Frames

The CANFD module transmits overload frames due to the detection of the following CAN bus conditions:

- Detection of a dominant bit in the first/second bit of intermission.
- Detection of a dominant bit at the 7th bit (last) of end of frame field (Rx frames).
- Detection of a dominant bit at the 8th bit (last) of error frame delimiter or overload frame delimiter.

Time Stamp

The value of the CANFD_TMR register is sampled at the beginning of the identifier field on the CAN bus and stored at the end of the move-in process in the TIME STAMP field, providing network behavior with respect to time. The free running timer is clocked by the CANFD bit-clock, which defines the baud rate on the CAN bus. During a message transmission or reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate.

The CANFD_TMR register is not incremented during module disable, doze, stop, and freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization.

Protocol Timing

The *CAN Engine Clocking Scheme* figure shows the structure of the clock generation circuitry that prescales the CAND module clock to generate the Sclock.

Figure 16-8: CAN Engine Clocking Scheme

The CANFD module supports a variety of means to set up bit timing parameters that are required by the CAN protocol. The CANFD_CTL1 register has fields to control bit timing parameters: CANFD_CTL1.PRESDIV, CANFD_CTL1.PROPSEG, CANFD_CTL1.PSEG1, CANFD_CTL1.PSEG2, and CANFD_CTL1.RJW.

The CANFD_TIMING register extends the range of the CAN bit timing variables in the CANFD_CTL1 register. The CANFD_TIMING register provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the BRS bit set.

NOTE: When the CAN FD feature is enabled, always set the CANFD_TIMING.BTF bit and configure the CAN bit timing variables in the CANFD_TIMING register.

The CANFD_CTL1.PRESDIV bit field (as well as the extended range CANFD_TIMING.EPRESDIV and CANFD_FD_TIMING.FPRESDIV for the data phase bits of CAN FD messages) defines the prescaler value that generates Sclock. The Sclock period defines the time quantum used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time handled by the CAN engine.

$Tq = (PRESDIV + 1) / f_{CANCLK}$

The bit rate, which defines the rate at which the CAN message is either received or transmitted, is given by the formula:

CAN Bit Time = (Number of Time Quanta in 1 bit time) * Tq

Bit Rate = 1/CAN Bit Time

A bit time is subdivided into three segments:

- SYNC_SEG – Has a fixed length of one time quantum. Signal edges are expected to occur within this segment.
- Time segment 1 – Includes the propagation segment and the phase segment 1 of the CAN standard. Time segment 1 is programmed by setting the CANFD_CTL1.PROPSEG and the CANFD_CTL1.PSEG1 fields so that their sum (plus 2) is in the range of 2 to 16 time quanta. When the CANFD_TIMING.BTF bit is enabled, the CANFD module uses the CANFD_TIMING.EPROPSEG and CANFD_TIMING.EPSEG1 bit fields so that their sum (plus 2) is in the range of 2 to 96 time quanta. For messages in CAN FD format with the BRS bit set, the CANFD uses CANFD_FD_TIMING.FPROPSEG and CANFD_FD_TIMING.FPSEG1 bit fields, so that their sum (plus 1) is in the range of 2 to 39 time quanta.
- Time Segment 2 – Represents the phase segment 2 of the CAN standard. It can be programmed by setting the CANFD_CTL1.PSEG2 bit field (plus 1) to be 2 to 8 time quanta long. When the CANFD_TIMING.BTF

bit is enabled, the CANFD module configures the CANFD_TIMING.EPSEG2 bit field with a value (plus 1) is in the range of 2 to 32 time quanta. For messages in CAN FD format with the BRS bit set, the CANFD uses the CANFD_FD_TIMING.FPSEG2 bit field instead, so that the value (plus 1) is in the range of 2 to 8 time quanta. The time segment 2 cannot be smaller than the information processing time (IPT), which is 2 time quanta.

See the *Bit Time Segment Example One* figure for an example using CANFD_CTL1 register bit timing variables for classical can format. See the *Bit Time Segment Example Two* figure for an example using CANFD_TIMING and CANFD_FD_TIMING register bit timing variables for CAN FD format. See the *Time Segment Syntax* table for syntax descriptions.

NOTE: For further explanation of the underlying concepts, see the ISO 11898-1 standard and the CAN 2.0A/B protocol specification for bit timing.

The bit time defined by the above time segments must not be smaller than 5 time quanta. For bit time calculations, use an IPT of 2, which is the value implemented in the CANFD module.

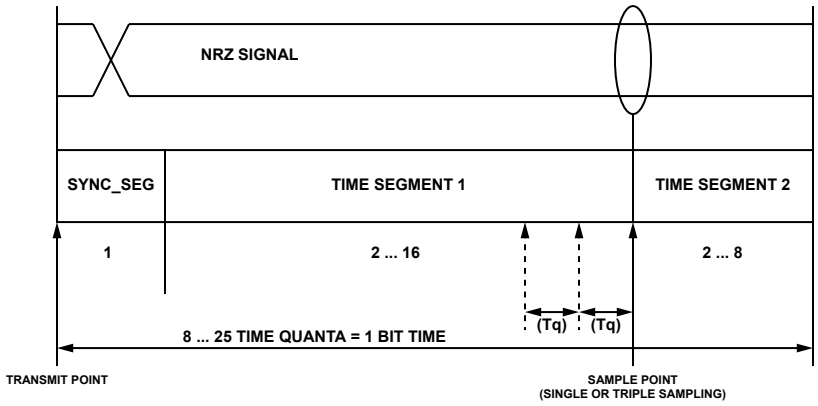


Figure 16-9: Bit Time Segment Example One

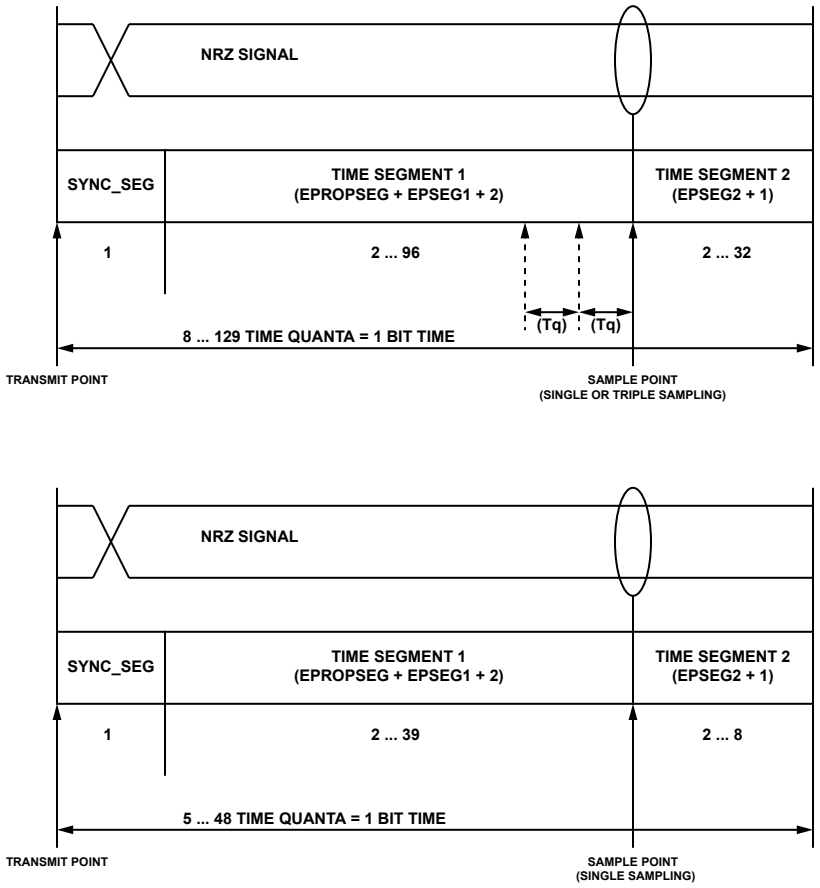


Figure 16-10: Bit Time Segment Example Two

Table 16-20: Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
TSEG1	Corresponds to the sum of PROPSEG and PSEG1.
TSEG2	Corresponds to the PSEG2 value.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The *BOSCH CAN 2.0B Standard Compliant Bit Time Segment Settings* table gives some examples of the CAN compliant segment settings for classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

Table 16-21: BOSCH CAN 2.0B Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Resynchronization Jump Width
	2	1 .. 2

Table 16-21: BOSCH CAN 2.0B Standard Compliant Bit Time Segment Settings (Continued)

Time Segment 1	Time Segment 2	Resynchronization Jump Width
5 .. 10		
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

NOTE: Ensure the bit time settings comply with the CAN Protocol standard (ISO 11898-1).

Whenever a CAN bit is used as a measure of time duration (for example, estimating the occurrence of a CAN bit event in a message), the number of peripheral clocks in one CAN bit (NumClkBit) is calculated as NumClkBit = (f_{SYS}/f_{CANCLK}) x (PRESDIV + 1) x (PROPSEG + PSEG1 + PSEG2 + 4), where:

- NumClkBit is the number of peripheral clocks in one CAN bit.
- f_{CANCLK} is the PE clock in Hz
- f_{SYS} is the frequency of operation of the system CHI clock in Hz.
- PSEG1 is the `CANFD_CTL1.PSEG1` value.
- PSEG2 is the `CANFD_CTL1.PSEG2` value.
- PROPSEG is the `CANFD_CTL1.PROPSEG` value.
- PRESDIV is the `CANFD_CTL1.PRESDIV` value.

The formula above is also applicable to the alternative CAN bit timing variables described in the `CANFD_TIMING` register and the `CANFD_FD_TIMING` register. For example, 180 CAN bits = (180 x NumClkBit) peripheral clock periods.

Arbitration and Matching Timing

During normal reception and transmission, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

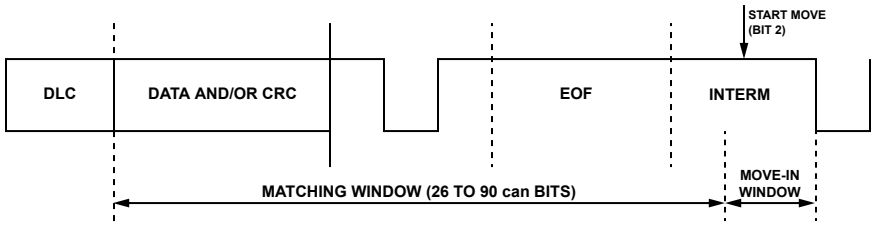


Figure 16-11: Matching and Move-In Time Windows

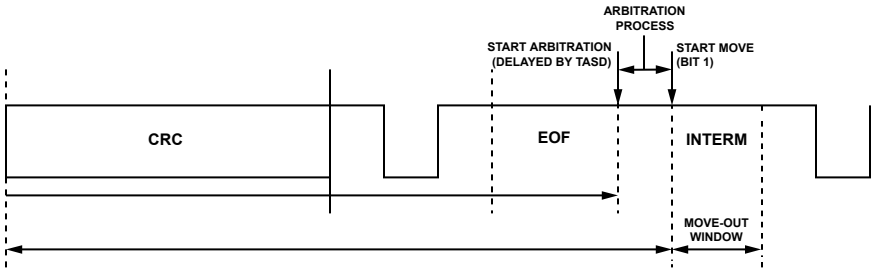


Figure 16-12: Arbitration And Move-out Time Windows

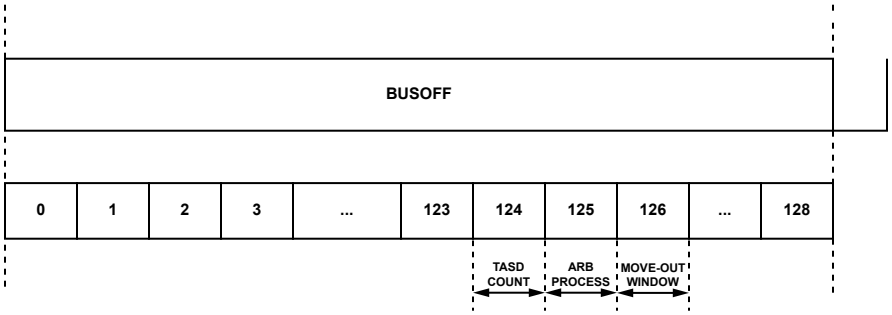


Figure 16-13: Arbitration At The End Of Bus Off And Move-out Time Windows

NOTE: In the previous figures, the matching and arbitration timing does not consider the delay caused by the concurrent memory access due to the processor or other internal CANFD submodules.

Tx Arbitration Start Delay

The Tx arbitration start delay (`CANFD_CTL2.TXASDLY` bit field) is a variable that indicates the number of CAN bits used by the CANFD to delay the Tx arbitration process start point from the first bit of the CRC field of the current frame.

The `CANFD_CTL2.TXASDLY` bit field can only be written only in freeze mode and is blocked by hardware in other modes.

The transmission performance is impacted by the ability of the processor to reconfigure MBs for transmission after the end of the internal arbitration process, in which the CANFD module finds the winner MB for transmission according to the standard arbitration process. If the arbitration ends before the first bit of the intermission field, then there is a chance the processor reconfigures some Tx MBs and the winner MB is no longer the best candidate to be transmitted.

The `CANFD_CTL2.TXASDLY` bit field is useful to optimize the transmission performance by defining the arbitration start point, as shown in the next figure, based on factors such as the CAN bit timing variables that determine the CAN bit rate and the number of MBs in use by the matching and arbitration processes.

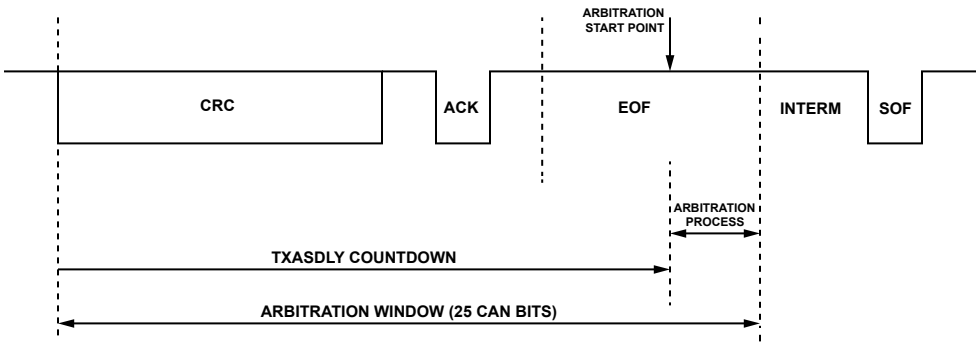


Figure 16-14: Optimal Tx Arbitration Start Point

The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and to the CAN bit rate, and inversely proportional to the peripheral clock frequency.

The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the intermission field of a CAN frame. For instance, if there are few MBs and the peripheral/oscillator clock ratio is high and the CAN baud rate is low, then the arbitration is placed closer to the frame's end, adding more delay to its start point, and vice-versa.

When the `CANFD_CTL2.TXASDLY` field is set to zero, the arbitration start is not delayed, and more time is reserved for arbitration. On the other hand, when the `CANFD_CTL2.TXASDLY` value is close to 24, then the processor can configure a Tx MB later and less time is reserved for arbitration. When too little time is reserved for arbitration, the CANFD module may not be able to find a winner MB in time to be transmitted with the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal `CANFD_CTL2.TXASDLY` value is calculated as follows:

For CAN FD frames and (MAXMB + 1) $\leq NMB_{END}$, TXASDLY = 31 – ((2*(MAXMB + 1) + 4)/$CPCB_N$)

For CAN FD frames and (MAXMB + 1) > NMB_{END}, TXASDLY = 22 – ((2*(MAXMB + 1) - NMB_{END})/ $CPCB_F$)

For non-FD frames, TXASDLY = 25 – (((2*(MAXMB + 1) + 4)) / CPCB) where:

NMB_{END} = ((9 * CPCBN) - 4) / 2

$BITRATE_N$= f_{CANCLK} / ([1 + (EPSEG1 + 1) + (EPSEG2 + 1) + (EPROPSEG + 1)] x (EPRESDIV + 1))

$BITRATE_F$= f_{CANCLK} / ([1 + (FPSEG1 + 1) + (FPSEG2 + 1) + FPROPSEG] x (FPRESDIV + 1))

$CPCB_N$ = f_{SYS} / $BITRATE_N$

$CPCB_F$ = f_{SYS} / $BITRATE_F$

CPCB = $CPCB_N$

- MAXMB is the `CANFD_CFG.MAXMB` value.

- NMBEND is the number of MBs that can be scanned by the arbitration process during the 9 last CAN bits at the end of a frame.
- BITRATEN is the CAN bit rate in bits per second calculated by the nominal CAN bit time variables.
- BITRATEF is the CAN bit rate in bits per second calculated by the data CAN bit time variables.
- CPCBN is the number of peripheral clocks per CAN bit in nominal bit rate for CAN FD frames.
- CPCBF is the number of peripheral clocks per CAN bit in data bit rate for CAN FD frames.
- CPCB is the number of peripheral clocks per CAN bit for non-FD frames.
- f_{CANCLK} is the oscillator clock, in Hz.
- f_{SYS} is the peripheral clock, in Hz.
- EPSEG1 is the `CANFD_TIMING.EPSEG1` or `CANFD_CTL1.PSEG1` value.
- EPSEG2 is the `CANFD_TIMING.EPSEG2` or `CANFD_CTL1.PSEG2` value.
- EPROPSEG is the `CANFD_TIMING.EPROPSEG` or `CANFD_CTL1.PROPSEG` value.
- EPRESDIV is the `CANFD_TIMING.EPRESDIV` or `CANFD_CTL1.PRESDIV` value.
- FPSEG1 is the `CANFD_FD_TIMING.FPSEG1` value.
- FPSEG2 is the `CANFD_FD_TIMING.FPSEG2` value.
- FPROPSEG is the `CANFD_FD_TIMING.FPROPSEG` value.
- FPRESDIV is the `CANFD_FD_TIMING.FPRESDIV` value.

NOTE: The f_{SYS} and f_{CANCLK} is same clocked from CDU CLKO4.

The following table give the TXASDLY value calculated for some configuration cases.

- f_{CANCLK} = 40 MHz
- Bit rate in arbitration phase = 1 Mbaud

Table 16-22: TXASDLY Values For Case Example

Number of MBs	TXASDLY Value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	23	6.67
54	22	5.0
64	21	3.33
96	20	1.6

CANFD Clock Domains and Restrictions

When doing matching and arbitration, the CANFD module needs to scan the entire MB memory during the time slot of one CAN frame, comprising a number of CAN bits. To have sufficient time to do that, there must be a minimum number of peripheral clocks per CAN bit, as specified in the *CAN Bit Minimum Peripheral Clock* table.

Table 16-23: CAN Bit Minimum Peripheral Clock

Number of Mailboxes	**CANFD_CFG.RFEN** Bit Value	Minimum Peripheral Clocks per CAN Bit
16	0	16
32	0	16
64	0	25
96	0	37
128	0	49
16	1	16
32	1	17
64	1	30

For the classical CAN frame format, the minimum number of peripheral clocks per CAN bit specified in the *TXASDLY Values For Case Example* table determines the minimum peripheral clock frequency for a given number of mailboxes and for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit, that can be defined by adjusting one or more of the bit timing values contained in either the CANFD_CTL1 register or the CANFD_TIMING register. The time quantum (Tq) is defined in the Protocol Timing section. The minimum number of time quanta per CAN bit must be 8, so the CAN clock frequency is at least 8 times the CAN bit rate.

For the CAN FD frame format, the number of peripheral clocks per CAN bit in nominal bit rate (NumClkNom-Bit) is calculated with the following equation:

NumClkNomBit =(fSYS/fCANCLK) X (PRESDIV + 1) X (PROPSEG + PSEG1 + PSEG2 + 4)

= fSYS/NomBitRate

where PRESDIV, PSEG1 and PSEG2 are CAN bit time values in the CANFD_CTL1 register. Alternatively, the EPRESDIV, EPSEG1 and EPSEG2 values in the CANFD_TIMING register can be used instead. NumClkNomBit is also calculated as a function of the expected nominal bit rate used in the arbitration phase (NomBitRate).

The number of CAN bits in the data phase of a FD frame with the BRS bit set depends on the number of data bytes in the payload. The number of fast CAN bits (NumOfFastBits) is determined as shown in the *CAN FD Frame Fast CAN Bits* table. The less the number of data bytes, the less the number of fast CAN bits, and less time is available for the CANFD module to scan the whole MB memory during the internal matching and arbitration processes.

Table 16-24: CAN FD Frame Fast CAN Bits

Minimum Data Bytes	DLC Field	Number of Fast Bits
0	0x0	21
1	0x1	29
2	0x2	37
3	0x3	45
4	0x4	53
5	0x5	61
6	0x6	69
7	0x7	77
8	0x8	85
12	0x9	117
16	0xA	149
20	0xB	186
24	0xC	218
32	0xD	282
48	0xE	410
64	0xF	538

The critical part of a CAN FD frame is during the data phase, where the CAN bit rate is faster than in the arbitration phase. The minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) is calculated so that enough time is available for the CANFD module to scan the MB memory during reception and transmission. The equation below calculates this constraint:

MinNumClkFastBit = ((8.5 x MaxNumOfMb) + 64 - (9 x NumClkNomBit)) / NumOfFastBits

where MaxNumOfMb is the maximum number of available mailboxes defined in the `CANFD_CFG.MAXMB` bit field.

The clock domain crossing circuit between the CHI and PE submodules also imposes a minimum number of peripheral clocks per fast CAN bit for the handshake mechanism to work properly without losing status information through the interface, as shown in the following equation:

MinNumClkFastBit = 3 x (1 + (f_{SYS}/f_{CANCLK}))

Therefore, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) is determined by the larger of the two values calculated above:

MinNumClkFastBit = Maximum (MinNumClkFastBit , MinNumClkFastBit)

Then, the maximum CAN bit rate in the data phase of CAN FD frames (DataBitRateMAX) is calculated as follows:

DataBitRate = f_{CANCLK} / (ROUNDUP ((MinNumClkFastBit x f_{CANCLK}) / f_{SYS}

The peripheral and oscillator clock frequencies, the maximum number of mailboxes, and the expected nominal bit rate affect the maximum data bit rate attainable by the CANFD module in CAN FD mode. In addition, the data bit rate depends on the minimum payload size of FD frames used in each application.

To illustrate how the CAN FD bit rate is affected by the configuration of CANFD variables, an consider an application example with the peripheral and oscillator clock frequencies set to 50 MHz.

1. Considering the nominal bit rate as 1 Mbps, the number of peripheral clocks per CAN bit in nominal bit rate is calculated as follows: NumClkNomBit = (50 x 10^6)/(1 x 10^6) = 50
2. The number of fast CAN bits (NumOfFastBits) is determined as shown in the *CAN FD Frames Maximum CAN Bit Rate In Data Phase* table. For example, if the minimum payload in FD frames is 8 bytes, then there are 85 CAN bits in the data phase.
3. Assuming the maximum number of mailboxes is 96, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated as follows:

   MinNumClkFastBit = ((8.5 X 96) + 64 - (9 X 50)) / 85 = 5.06

   MinNumClkFastBit = 3 x (1 + 50/50) = 6.0

   MinNumClkFastBit = Maximum (5.06 , 6.0) = 6.0

As demonstrated in this example, even though the oscillator clock frequency (50 MHz) is adequate to generate a data rate of 8 Mbps in CAN FD mode, the specific CANFD configuration limits this rate to 6.667 Mbps. This limitation is due to the low peripheral clock frequency that imposes the $MinNumClkFastBit_B$ limit.

The *CAN FD Frames Maximum CAN Bit Rate In Data Phase* table shows the maximum data rate for CAN FD according to the clock frequencies, payload size, and number of available mailboxes. As shown in the *CAN FD Frames Maximum CAN Bit Rate In Data Phase* table, for some cases, if the number of available mailboxes is reduced, the CANFD module achieves a data rate up to 8 Mbps.

Table 16-25: CAN FD Frames Maximum CAN Bit Rate In Data Phase

Peripheral Clock Frequency (MHz)	Payload Size	Number of Available Mailboxes	Maximum Data Rate (Mbps)
40	8	94	6.667
40	8	114	5.0
40	12	117	6.667
40	12	128	5.714
50	12 to 64	128	6.667
60	8	128	8.0
60	12	128	8.0

Table 16-25: CAN FD Frames Maximum CAN Bit Rate In Data Phase (Continued)

Peripheral Clock Frequency (MHz)	Payload Size	Number of Available Mailboxes	Maximum Data Rate (Mbps)
67	6	128	8.0
80	3	128	8.0
100	0	128	8.0

CANFD Operating Modes

The CANFD module has functional modes and low-power modes.

CAUTION: The CANFD does not support the permanent dominant failure on the CAN bus line. If a low-power request or freeze mode request is done during a permanent dominant, the corresponding acknowledge can never be asserted.

The CANDFD module has the following main functional modes:

- Normal Mode
- Freeze Mode
- Loop-Back Mode
- Listen-Only Mode
- CAN FD Active Mode

For low-power operation, the CANDFD module has the following modes:

- Module Disable Mode
- Doze Mode
- Stop Mode
- Pretended Network Mode

NOTE: The CANFD does not support the permanent dominant failure on the CAN bus line. If there is a low-power request or freeze mode request during a permanent dominant, the corresponding acknowledge will not be asserted.

Normal Mode

In normal mode, the CANFD module operates by receiving and transmitting message frames. In this mode, errors are handled normally, and all CAN protocol functions are enabled.

Freeze Mode

In freeze mode, there is no transmission or reception of frames and synchronicity to the CAN bus is lost. Freeze mode is enabled by setting the CANFD_CFG.FRZ bit.

Freeze mode is requested either by the processor by enabling the CANFD_CFG.HALT bit or when it is put into debug mode. In both cases, when the CANFD_CFG.FRZ bit is enabled, the CANFD module must not be in a low-power mode. Freeze mode is also requested through the automatic assertion of both the CANFD_CFG.HALT and CANFD_CFG.FRZACK bits when the CANFD_MEC.NCERRFRZEN bit is set and a non-correctable error is detected in a memory read access performed by CANFD internal processes. The acknowledgement is obtained through the assertion by the CANFD_CFG.FRZACK bit. The CANFD module is only in freeze mode when both the request and acknowledge conditions are satisfied.

In response to a freeze mode request, the CANFD module performs the following operations:

- Waits to be in either the intermission, passive error, bus off, or idle state.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in does not prevent going to freeze mode.
- Ignores the Rx input pin and drives the Tx pin as recessive.
- Stops the prescaler, thus halting all CAN protocol activities.
- Grants write access to the CANFD_ECR register, which is read-only in other modes.
- Sets the CANFD_CFG.NOTRDY and CANFD_CFG.FRZACK bits.

After requesting freeze mode, wait for the CANFD_CFG.FRZACK bit to be set before executing any other action, or the CANFD module will not operate properly.

To exit freeze mode, disable the CANFD_CFG.FRZ bit, remove the processor from debug mode, or disable the CANFD_CFG.HALT bit.

The CANFD_CFG.FRZACK bit is disabled after the PE recognizes the negation of the freeze request. Once out of freeze mode, the CANFD module tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

Loop-Back Mode

The CANFD module enters loop-back mode when the CANFD_CTL1.LBEN is set. In loop-back mode, the CANFD module performs an internal loop back for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). The CANFD module behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In loop-back mode, the CANFD ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

Listen-Only Mode

The CANFD module enters listen-only mode when the CANFD_CTL1.LOMEN bit is enabled. In listen-only mode, transmission is disabled, all error counters are frozen, and the CANFD module operates in a CAN error passive mode. The module only receives messages acknowledged by another CAN station. If the CANFD detects a message that has not been acknowledged, it flags a BIT0 error (without changing the receive error counter), as if it were trying to acknowledge the message.

CAN FD Active Mode

In CAN FD active mode, the CANFD module can transmit and receive all messages formatted according to the CAN FD protocol and CAN 2.0 (protocol 2.0) in an interleaved fashion. The CANFD module is put into CAN FD active mode by setting the CANFD_CFG.FDEN bit when in freeze mode.

Module Disable Mode

The module disable mode is a low-power mode normally used to temporarily disable a complete CANFD block, with no power consumption. To enter module disable mode, enable the CANFD_CFG.DIS bit. Once the processor is in module disable mode, the CANFD_CFG.LPMACK bit is set.

Do not use module disable mode under pretended networking mode. Disable the CANFD_CFG.DIS bit and wait for CANFD_CFG.LPMACK bit to clear before enabling the CANFD_CFG.PNETEN bit.

If the CNAFD module is disabled during transmission or reception, it will perform the following operations:

- Waits to be in either the idle or bus off state, or else waits for the third bit of intermission and then checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not considered.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Shuts down the clocks to the PE and CHI submodules.
- Sets the CANFD_CFG.NOTRDY and CANFD_CFG.LPMACK bits.

The bus interface unit continues to operate, enabling the processor to access memory mapped registers, except the CANFD_RX_FIFO_GMSK, CANFD_RX_14_MSK, CANFD_RX_15_MSK, and CANFD_RX_FIFO_GMSK registers. The CANFD_RX_FIFO, the message buffers, the Rx individual mask registers, and the reserved words within RAM are not accessible when the CANFD module is in the module disable mode.

To exit module disable mode, disable the CANFD_CFG.DIS bit. If the CANFD module is disabled during freeze mode, it sends a requests to disable the clocks to the PE and CHI submodules, sets the CANFD_CFG.LPMACK bit, and clears the CANFD_CFG.FRZACK bit.

Doze Mode

The doze mode is a system low-power mode in which the processor bus is kept alive, and a global doze mode request is sent to all peripherals asking them to enter low-power mode. Before sending a global doze mode request, set the CANFD_CFG.DOZEN bit. The global doze request is set with the MISCREG_CAN_SYSCTL.CAN1_IPG_DOZE or MISCREG_CAN_SYSCTL.CAN0_IPG_DOZE bit. Once the processor requests the doze mode, the CANFD_CFG.LPMACK bit is set.

To exit doze mode, disable the CANFD_CFG.DOZEN bit or clear the IPG_DOZE bits in the MISCREG_CAN_SYSCTL register. The CANFD module also exits doze mode when activity is detected on the CAN bus and the self-wake up mechanism is enabled using the CANFD_CFG.SLFWAKEN bit.

If the doze mode is triggered during freeze mode, the CANFD module requests to disable the clocks to the PE and CHI submodules, sets the CANFD_CFG.LPMACK bit, and clears the CANFD_CFG.FRZACK bit.

If doze mode is triggered during transmission or reception, the CANFD module performs the following operations:

- Waits to be in either idle or bus off state, or else waits for the third bit of intermission and checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not considered.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Shuts down the clocks to the PE and CHI submodules.
- Sets the CANFD_CFG.NOTRDY and CANFD_CFG.LPMACK bits.

The bus interface unit continues to operate, enabling the processor to access memory mapped registers, except the CANFD_RX_FIFO_GMSK, CANFD_RX_14_MSK, CANFD_RX_15_MSK, and CANFD_RX_FIFO_GMSK registers. The CANFD_RX_FIFO, the message buffers, the Rx individual mask registers, and the reserved words within RAM are not accessible when the CANFD module is in the doze mode.

Exiting Doze mode is done in one of the following ways:

- Disabling the doze mode request in the MISCREG_CAN_SYSCTL register.
- Disabling the CANFD_CFG.DOZEN bit.
- When the CANFD module detects activity on the CAN bus and the self-wake up mechanism is enabled using the CANFD_CFG.SLFWAKEN bit.

When the self-wake up mechanism is enabled at the time that the CANFD module entered doze mode, then upon detection of a recessive-to-dominant transition on the CAN bus, the CANFD module clears the CANFD_CFG.DOZEN bit, requests to resume clocks, and disables the CANFD_CFG.LPMACK bit after the CAN PE recognizes the doze mode is disabled.

The CANFD module also sets the CANFD_ESR1.WAKINT bit, so if the CANFD_CFG.WAKMSK bit is enabled, it generates a wake up interrupt to the processor. The CANFD module then waits for 11 consecutive recessive bits to

synchronize to the CAN bus. Consequently, the CANFD module does not receive the frame that woke it up. The *Wake Up From Doze Mode* table details the effect of the CANFD_CFG.SLFWAKEN and the CANFD_CFG.WAKMSK bits upon waking from the doze mode.

Table 16-26: Wake Up From Doze Mode

SLFWAKEN	WAKINT	WAKMSK	CANFD Clocks Enabled	Wakeup Interrupt Generated
0	N/A	N/A	No	No
0	N/A	N/A	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	Yes	No
1	1	1	Yes	Yes

Applying a low-pass filter function to the Rx CAN input line while in doze mode will modify sensitivity to CAN bus activity as described in the CANFD_CFG.WSFLTREN bit description. This feature also protects the CANFD module from waking up due to short glitches on the CAN bus lines that can result from electromagnetic interference in noisy environments.

Stop Mode

The stop mode is a system low-power mode in which all clocks are stopped for maximum power savings. A global stop request is set with the MISCREG_CAN_SYSCTL.CAN0_IPG_STOP or MISCREG_CAN_SYSCTL.CAN1_IPG_STOP bit. Once the processor is in the stop mode, the CANFD_CFG.LPMACK bit is set.

If the CANFD module receives a global stop mode request during freeze mode, it sets the CANFD_CFG.LPMACK bit, clears the CANFD_CFG.FRZACK bit, and then sends the stop acknowledge signal to the processor to shut down the clocks globally.

If stop mode is requested during transmission or reception, the CANFD module performs the following operations:

- Waits to be in either the idle or bus off state, or else waits for the third bit of intermission and checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not considered.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Sets the CANFD_CFG.NOTRDY and CANFD_CFG.LPMACK bits.
- Sends a stop acknowledge signal to the processor to shut down the clocks globally.

Stop mode is exited when the processor resumes the clocks and removes the stop mode request, or when activity is detected on the CAN bus and the self-wake up mechanism is enabled using the CANFD_CFG.SLFWAKEN bit.

When the self-wake up mechanism is enabled at the time that the CANFD module entered stop mode, then upon detection of a recessive-to-dominant transition on the CAN bus, the CANFD module sets the CANFD_ESR1.WAKINT bit, so if the CANFD_CFG.WAKMSK bit is enabled, it generates a wake up interrupt to the processor. Upon receiving the interrupt, the processor resumes the clocks and removes the stop mode request. The CANFD module then waits for 11 consecutive recessive bits to synchronize to the CAN bus. Consequently, the CANFD module does not receive the frame that woke it up. The *Wakeup From Stop Mode* table details the effect of the CANFD_CFG.SLFWAKEN and the CANFD_CFG.WAKMSK bits upon wakeup from the stop mode. Note that wakeup from the stop mode only works when both bits are asserted.

Table 16-27: Wakeup From Stop Mode

SLFWAKEN	WAKINT	WAKMSK	Chip Clocks Enabled	Wakeup Interrupt Generated
0	N/A	N/A	No	No
0	N/A	N/A	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	No	No
1	1	1	Yes	Yes

Applying a low-pass filter function to the Rx CAN input line while in stop mode will modify sensitivity to CAN bus activity as described in the CANFD_CFG.WSFLTREN bit description. This feature also protects the CANFD module from waking up due to short glitches on the CAN bus lines that can result from electromagnetic interference in noisy environments.

Pretended Network Mode

Pretended networking is a special low-power mode used to receive wake up messages with low power consumption. Pretended networking mode can be selected to operate together with stop or doze mode. Before entering low-power mode, the CANFD_CFG.PNETEN bit must be asserted. Once in low-power mode, the CHI submodule clocks are shut down and the PE submodule is kept clocked, so that the receive process is still active to filter incoming messages as defined by the configuration.

Upon detecting a wakeup event, a wake up interrupt is issued to the system. When the CANFD_CFG.PNETEN bit is enabled, the processor disables the self-wake up feature by disabling the CANFD_CFG.SLFWAKEN bit. Wake up from matched message or time-out in pretended networking modes happens only when pretended networking mode is used along with doze mode. Wakeup is not supported in stop mode.

To enter pretended networking mode, the CANFD must be in normal mode and not in freeze or module disable mode. Under pretended networking mode, the CANFD module stays synchronized with the CAN bus in doze or stop mode. When either doze or stop is requested, the CANFD module performs the following operations:

- Waits to be in the idle state, or else waits for the third bit of intermission, and then checks it to be recessive.
- Sets the CANFD_CFG.LPMACK bit.

- Requests the shutdown of the CHI submodule clock, while keeping the PE submodule clock active.

The CANFD module exits pretended networking mode in the following ways:

- The processor removes the doze mode request.
- The processor disables the CANFD_CFG.DOZEN bit.
- The CPU removes the stop mode request.

The CANFD will wait until bus idle or the third bit of the intermission state to disable the CANFD_CFG.LPMACK bit.

The above exit methods are triggered either by the CANFD module detecting a wake up event and issuing the respective interrupt, or by the processor itself upon being woken up. The CANFD will wait until the bus idle state or until the third bit of the intermission state to disable the CANFD_CFG.LPMACK bit and resume normal mode. This procedure ensures that the CANFD module is synchronized to the CAN bus after exiting pretended networking mode. The processor must wait for the CANFD_CFG.LPMACK bit to be disabled before performing any access to the CANFD.

CANFD Event Control

The following section describe how the CANFD module generates and controls events.

Interrupts

The CANFD module has many interrupt sources: interrupts due to MBs and interrupts due to the OR'd interrupts from MBs, bus off, bus off done, error, error fast (errors detected in the data phase of CAN FD format messages with the CANFD_FD_CTL.BRSEN bit set), wake up, wake up match, wake up timeout, Tx warning, and Rx warning states.

Each one of the MBs can be an interrupt source if the corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has an assigned flag bit in the CANFD_IFLG1 and CANFD_IFLG2 registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the processor writes it to 1 (unless another interrupt is generated at the same time).

NOTE: The processor must only clear the bit causing the current interrupt, so bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause the accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (CANFD_CFG.RFEN = 1) and DMA is disabled (CANFD_CFG.DMAEN = 0), the interrupts corresponding to MBs 0 to 7 have different meanings:

- Bit 7 of the CANFD_IFLG1 register becomes the "FIFO Overflow" flag.
- Bit 6 of the CANFD_IFLG1 register becomes the "FIFO Warning" flag.

- Bit 5 of the CANFD_IFLG1 register becomes the “Frames Available in FIFO” flag.
- Bit 4-0 of the CANFD_IFLG1 register are unused.

If both Rx FIFO and DMA are enabled (CANFD_CFG.RFEN =1 and CANFD_CFG.DMAEN = 1), the CANFD does not generate any FIFO interrupt. Bit 5 of the CANFD_IFLG1 register still indicates “Frames Available in FIFO” and generates a DMA request. Bits 7, 6, and 4–0 are unused.

NOTE: The Rx FIFO cannot be enabled when the CAN FD feature is enabled.

For a combined interrupt where multiple MB interrupt sources are OR’d together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the processor must read the CANFD_IFLG1 and CANFD_IFLG2 registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for bus off, bus off done, error, error fast, wake up, Tx warning ,and Rx warning generate interrupts like the MB interrupt sources, and are read from the CANFD_ESR1 register. The bus off, error, Tx warning, and Rx warning interrupt mask bits are located in the CANFD_CTL1 register; the wake up interrupt mask bit is located in CANFD_CFG register.

The interrupt sources for pretended networking (wake up by match flag and wake up by timeout flag) are read in the CANFD_WUM register and the respective interrupts masks bits are located in CANFD_PN_CTL1 register.

Bus Interface

The processor accesses to CANFD registers are subject to the following rules:

- Read and write accesses to reserved address space result in access errors.
- Write accesses to positions with bits that are all currently read-only results in access errors.

  If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to the register and bit descriptions for details.
- Read and write accesses to unimplemented address space results in access errors.
- Read and write accessed to RAM located positions during low-power modes results in access errors.
- If CANFD_CFG.MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Reserved words within RAM cannot be used.

  For example, suppose the CANFD RAM can support up to 16 MBs, the CANFD_CTL2.RFFNUM bit field is 0x0, and the CANFD_CFG.MAXMB bit field is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one individual mask and the available memory in the mask registers space is from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF, there are reserved words for internal use which cannot be used as general purpose RAM. Generally, free memory space for general-purpose usage depends only on the CANFD_CFG.MAXMB bit field.

Detection and Correction of Memory Errors

The CANFD module supports detection and correction of errors in memory read accesses. Each byte of CANFD memory is associated with five parity bits, which ensures a Hamming distance of four. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors), and errors in 2 bits can be detected but not corrected (non-correctable errors). Errors in more than 2 bits may not be detected. In case of non-correctable errors, the corrupted data is not changed by the error correction logic. When a read access is performed, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

The CANFD module detects a non-correctable error in the event either an all-zeros or an all-ones read occurs and updates the CANFD_ERR_RSYN register. The CANFD_ERR_RSYN register holds the syndrome detected in a memory read with an error. It reports the bytes that are read in the 32-bit read transaction.

The *Syndrome Definition* table shows the SYNDn field in the CANFD_ERR_RSYN register, indicates the type of error, and the bit in byte (n) affected by the error. Each BEn field in the CANFD_ERR_RSYN register indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, even for the non-read ones. Errors detected in non-read bytes are indicated and reported as described in the Error Indication and Error Reporting sections.

Table 16-28: Syndrome Definition

SYNDn	Type	Bit Affected
0x00	N/A	None (no error)
0x01	Code	0
0x02	Code	1
0x04	Code	2
0x07	Data	5
0x08	Code	3
0x0E	Data	7
0x10	Code	4
0x13	Data	2
0x15	Data	6
0x16	Data	1
0x19	Data	3
0x1A	Data	4
0x1C	Data	0
0x06	N/A	All-zeros non-correctable error
0x1F	N/A	All-ones non-correctable error
All others	N/A	Non-correctable error

Memory errors are indicated to the host through the CANFD_ERR_STAT register and bus transfer errors, and reported through the CANFD_ERR_RADDR,CANFD_ERR_RDAT, and CANFD_ERR_RSYN registers.

The error detection and correction mechanism is activated by the CANFD_MEC.ECCDIS bit. When error detection and correction is disabled, updates on indications and reporting registers are stopped, but the parity bits are still calculated and written along with data in memory write operations to ensure that memory has consistent parity bits associated to the data.

NOTE: All CANFD memory must be initialized before any operation for the parity bits in memory to properly update. The CANFD_CTL2.WRMFRZEN bit grants write access to all memory positions that require initialization, ranging from 0x080 to 0xADF (and from 0xF28 to 0xFFF when the CAN FD feature is enabled). The CANFD_RX_MB_GMSK, CANFD_RX_14_MSK, CANFD_RX_15_MSK, and CANFD_RX_FIFO_GMSK registers also need to be initialized. The CANFD_CFG.RFEN bit must not be set during memory initialization. The *Identifier Acceptance Filter Fields for Global Bits* table shows the configurations for the CANFD_RX_FIFO_GMSK register.

Table 16-29: Identifier Acceptance Filter Fields for Global Mask Bits

Rx FIFO ID Filter Table Element Format **CANFD_CFG.IDAM**	Identifier Acceptance Filter Fields **CANFD_RX_FIFO_GMSK.VALUE**					
RTR	IDE	RXIDA	RXIDB	RXIDC	Reserved	
A	VALUE[31]	VALUE[30]	VALUE[29:1]	N/A	N/A	VALUE[0]
B	VALUE[31], VALUE[15]	VALUE[30], VALUE[14]	N/A	VALUE[29:16], VALUE[13:0]	N/A	N/A
C	N/A	N/A	N/A	N/A	VALUE[31:24], VALUE[23:16], VALUE[15:8], VALUE[7:0]	N/A

To avoid accidentally changing the critical error correction configuration, use the following protocol to enable the CANFD_MEC register update:

1. By default, the CANFD_CTL2.ECRWREN bit is zero and the CANFD_MEC.ECRWRDIS bit is one.
2. Set the CANFD_CTL2.ECRWREN bit.
3. Clear the CANFD_MEC.ECRWRDIS bit.
4. All writes to the CANFD_MEC register must keep the CANFD_MEC.ECRWRDIS bit cleared.
5. After configuration is done, lock the CANFD_MEC register by either setting the CANFD_MEC.ECRWRDIS bit or clearing the CANFD_CTL2.ECRWREN bit.

Sources of Memory Access

The CANFD memory can be accessed by two major sources (or requesters):

- Host (processor): The largest word accessed is 32 bits wide.
- CANFD internals processes (Rx matching, Tx arbitration, move-in on reception, move-out on transmission): The largest word accessed is 64 bits wide.

The way that non-correctable errors are indicated and reported depends on the source of access.

Error Indication

Memory errors are indicated by the CANFD_ERR_STAT.HNCEI, CANFD_ERR_STAT.INCEI, and CANFD_ERR_STAT.CEI flag bits. Non-correctable errors detected in memory reads requested by the host are indicated separately from the errors detected in requests by CANFD internal processes. The CANFD makes no distinction of the source of the access when correctable errors are detected. There are three independent flags for these three cases, and each flag raises an interrupt unless it is masked in the CANFD_MEC register. If both non-correctable and correctable errors are found in different bytes of the same read operation, both flags are set.

A non-correctable error detected in host access is also indicated as a bus transfer error. A bus wait request asserts to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked. If the CANFD_ERR_STAT.HNCEI bit is not masked, the same non-correctable error will raise a bus transfer error and an interrupt request.

Each indication flag has one overrun flag in the CANFD_ERR_STAT register. The overrun flags do not request interrupts. Overrun flags for non-correctable errors indicate that other errors of the same nature were detected after current error being treated, while overrun flags for correctable errors indicate that other errors of the same nature were detected before the current error being treated. The recommended handling sequence for error indication is as follows:

1. Get the error report information from the report registers CANFD_ERR_RADDR, CANFD_ERR_RDAT, and CANFD_ERR_RSYN.
2. Use the error report information to take proper measures in the application.
3. Clear the CANFD_ERR_STAT.HNCEI, CANFD_ERR_STAT.INCEI, CANFD_ERR_STAT.CEI flag bits.
4. If the overrun flag is active:
    a. Alert that application that at least one error could not be handled.
    b. Clear the overrun flag bit.

The CANFD internal processes can access memory in transactions larger than 32 bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses. If errors are found in 2 or more 32-bit words, the interrupt and overrun flags are set simultaneously.

Error Reporting

The report registers CANFD_ERR_RADDR, CANFD_ERR_RDAT, and CANFD_ERR_RSYN provide detailed information about the address read, raw data, and syndrome read with error and indicated by the flags.

The address, data, and syndrome registers are updated simultaneously along with the error flags, according to these rules:

1. If any of the two non-correctable error flags is currently set, the report registers are not updated (the previous non-correctable error reporting is preserved).
2. Otherwise (either no error flag is currently set or only the correctable error flag is currently set), the report registers are updated according to the new error, or according to the most severe of new errors if non-correctable and correctable errors are simultaneously detected.

Reporting of errors detected in accesses larger than 32 bits follows the rules described in the CANFD_ERR_RADDR register.

The address reported in the CANFD_ERR_RADDR register and defined in the CANFD_ERR_IADDR register are not the same as those listed in the module memory map. The relation between the reported addresses and the respective ones in the module memory map is shown in the Error Injection section.

Addresses reported when reading memory portions organized as FIFOs, such as the CANFD_RX_FIFO register, refer to the address of the specific entry accessed in the FIFO, not to the FIFO base address.

To ensure coherence of the error report registers, turn off the report update by setting the CANFD_MEC.RERDIS bit before reading the report registers.

Response to Errors

Correctable errors have no effect on CANFD operation because affected data is corrected before use by the host or CANFD internal processes.

For host-initiated reads, a non-correctable error affects the host, but does not affect CANFD operation.

Non-correctable errors detected on memory reads requested by the CANFD internal processes result in incorrect operation depending on the state of the CANFD_MEC.NCERRFRZEN bit, as follows:

- During reception (either matching or move-in processes), when a non-correctable error occurs, an incorrect destination is selected to store the incoming frame, a corrupted frame is stored in the correct destination, or both. If the CANFD_MEC.NCERRFRZEN is set, the CANFD module stops operation automatically and enters freeze mode to prevent corrupted data from being treated as valid by CANFD internal processes. If the CANFD_MEC.NCERRFRZEN bit is cleared, the CANFD continues working and a corrupted frame is received.
- During the arbitration process, when a non-correctable error occurs, either a non-highest priority Tx message buffer is mistakenly selected for transmission or the data is corrupted. If the CANFD_MEC.NCERRFRZEN bit is set, the CANFD module stops operation automatically and enters freeze mode before starting the move-out. If the CANFD_MEC.NCERRFRZEN bit is cleared, the CANFD proceeds to move-out with a corrupted frame that will transmit on the CAN bus.

- During the move-out process, when a non-correctable error occurs, a corrupted frame is copied from the selected Tx MB that won the arbitration to the Tx SMB for transmission. If the CANFD_MEC.NCERRFRZEN bit is set, the CANFD module stops operation automatically and enters freeze mode before starting the transmission. If the CANFD_MEC.NCERRFRZEN bit is cleared, the corrupted frame is transferred from the Tx SMB to the PE submodule and transmits on the CAN bus.
- A non-correctable error can be detected beyond the move-out process, when Tx data is read from the Tx SMB (buffer located in RAM) to be transferred to the PE submodule for transmission. In this case, a frame with corrupted ID and/or data is transmitted on the CAN bus. To prevent the frame from being successfully received by the external nodes, the CANFD module inverts all bits in the CRC field (CRC sequence plus CRC Delimiter) and transmits an error flag just after CRC delimiter because of a self-detecting bit1 error and a form error due to the CRC field inversion. When the CANFD_MEC.NCERRFRZEN bit is set, the CANFD module stops operation automatically and enters freeze mode just after the error frame. If the CANFD_MEC.NCERRFRZEN bit is negated, the CANFD module attempts to re-transmit the same frame, when no other higher priority Tx MB is subsequently configured for transmission. In the event the non-correctable error persists, the CANFD module eventually reaches the bus off state because of consecutive error detections. The CANFD_ECR.TXERRCNT field is updated every time the CANFD module inverts the CRC field causing errors as described above.

When the CANFD_MEC.NCERRFRZEN bit is set and the CANFD module enters freeze mode, only the host processor can cause then CANFD to exit freeze mode and resume normal mode. Assertion of the CANFD_MEC.NCERRFRZEN bit is the only way to prevent corrupted frames from transmitting on the CAN bus up to the move-out internal process.

The error report registers provide information to the application for customized handling of these situations.

Error Injection

The error injection registers CANFD_ERR_IADDR, CANFD_ERR_IDP, and CANFD_ERR_IPP are used to inject errors in memory reads in order to force errors and consequently update the indication and reporting registers.

The *Error Injection Address Mapping* table shows the relationship between the error injection addresses and the ones in the module memory map. Use this table to convert from the memory map address to the correct location in the physical CANFD RAM.

NOTE: Where pairs of values are provided in the table, the first is the address when the CANFD_CFG.FDEN bit is disabled and the second is when the CANFD_CFG.FDEN bit is enabled.

Table 16-30: Error Injection Address Mapping

RAM Contents	Injection Address	Memory Map
CANFD registers	Not mapped	N/A
MBs	0x0000	0x0080
Reserved	N/A	0x0480
RXIMRs	0x0400	0x0880

Table 16-30: Error Injection Address Mapping (Continued)

RAM Contents	Injection Address	Memory Map
Reserved	N/A	0x0980
RXFIR_0	0x0500	0x0A80
RXFIR_1	0x0504	0x0A84
RXFIR_2	0x0508	0x0A88
RXFIR_3	0x050C	0x0A8C
RXFIR_4	0x0510	0x0A90
RXFIR_5	0x0514	0x0A94
Reserved	N/A	0x0A98
CANFD_RX_MB_GMSK	0x0520	0x0010
CANFD_RX_FIFO_GMSK	0x0524	0x0048
CANFD_RX_14_MSK	0x0528	0x0014
CANFD_RX_15_MSK	0x052C	0x0018
Tx_SMB	0x0530	0x0AB0 / 0x0F28
Rx_SMB0	0x0540 / 0x0578	0x0AC0 / 0x0F70
Rx_SMB1	0x0550 / 0x05C0	0x0AD0 / 0x0FB8
ECC registers	Not mapped	0x0AE0
Pretended networking registers	Not mapped	0x0B00
CAN FD registers	Not mapped	0x0C00
Reserved	N/A	0x0C0C

The injection is done by flipping the data and parity bits corresponding to the bits set to 1 in the CANFD_ERR_IDP and CANFD_ERR_IPP registers. Injection can be selected specifically for memory accesses requested by host or by CANFD internal processes.

In case of accesses larger than 32 bits, the CANFD_MEC.EXTERRIEN bit extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

NOTE: It is very unlikely, but error injection may correct a bit with an error. This will not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, the CANFD_MEC.HAERRIEN and CANFD_MEC.IAERRIEN bits must be cleared while configuring the memory injection.

CANFD Programming Model

Following sections provide some guidelines for CANFD programming.

Initialization

The CANFD module is reset in one of the following ways:

- A chip level hard reset that resets all memory mapped registers asynchronously.
- Setting the CANFD_CFG.SOFTRST bit , which resets some of the memory mapped registers synchronously. The *Register Reset States* table shows the registers affected by a soft reset.
- Chip level soft reset, which has the same effect as setting the CANFD_CFG.SOFTRST bit.

A soft reset is synchronous and must follow an internal request and acknowledge procedure across clock domains and takes time to fully propagate the effects. The CANFD_CFG.SOFTRST bit remains set while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset cannot be applied while clocks are shut down in a low-power mode. Exit the low-power mode and resume the clocks before applying a soft reset.

Table 16-31: Register Reset States

Register	Affected by Hard Reset	Affected by Soft Reset
CANFD_TIMING	Yes	No
CANFD_CRC	Yes	Yes
CANFD_CTL1	Yes	No
CANFD_PN_CTL1	Yes	Yes
CANFD_CTL2	Yes	No
CANFD_PN_CTL2	Yes	Yes
CANFD_ECR	Yes	Yes
CANFD_ERR_IADDR	Yes	Yes
CANFD_ERR_IDP	Yes	Yes
CANFD_ERR_IPP	Yes	Yes
CANFD_ERR_STAT	Yes	Yes
CANFD_ESR1	Yes	Yes
CANFD_ESR2	Yes	Yes
CANFD_FD_TIMING	Yes	No
CANFD_FD_CRC	Yes	Yes
CANFD_FD_CTL	Yes	No
CANFD_FLTR_DLC	Yes	Yes
CANFD_FLTR_ID1	Yes	Yes
CANFD_FLTR_ID2_IDMSK	Yes	Yes

Table 16-31: Register Reset States (Continued)

Register	Affected by Hard Reset	Affected by Soft Reset
CANFD_IFLG1	Yes	Yes
CANFD_IFLG2	Yes	Yes
CANFD_IMSK1	Yes	Yes
CANFD_IMSK2	Yes	Yes
CANFD_CFG	Yes	Yes
CANFD_MEC	Yes	Yes
CANFD_FLTR_DATA1_HI	Yes	Yes
CANFD_FLTR_DATA1_LO	Yes	Yes
CANFD_FLTR_DATA2_DMSK_HI	Yes	Yes
CANFD_FLTR_DATA2_DMSK_LO	Yes	Yes
CANFD_ERR_RADDR	Yes	Yes
CANFD_ERR_RDAT	Yes	Yes
CANFD_ERR_RSYN	Yes	Yes
CANFD_RX_14_MSK	No	No
CANFD_RX_15_MSK	No	No
CANFD_RX_FIFO_GMSK	No	No
CANFD_RX_FIFO	No	No
CANFD_RX_IMSK[n]	No	No
CANFD_RX_MB_GMSK	No	No
CANFD_TMR	Yes	Yes
CANFD_WMB[n]_DATA_HI	Yes	No
CANFD_WMB[n]_DATA_LO	Yes	No
CANFD_WMB[n]_ID	Yes	No
CANFD_WMB[n]_STAT	Yes	No
CANFD_WUM	Yes	Yes

Note that the MBs and the Rx individual mask registers are not affected by reset, so they are not automatically initialized.

CANFD Initialization Sequence

For any configuration change or initialization, the CANFD must be is put into freeze mode. The following steps are a generic initialization sequence applicable to the CANFD module:

1. Initialize the Module Configuration Register (CANFD_CFG):

- Enable the individual filtering per MB and reception queue features by setting the CANFD_CFG.IRMQEN bit.
- Enable the warning interrupts by setting the CANFD_CFG.WRNEN bit.
- Optionally disable frame self reception by setting the CANFD_CFG.SRXDIS bit.
- Enable the Rx FIFO by setting the CANFD_CFG.RFEN bit.
- If Rx FIFO is enabled and DMA is required, set the CANFD_CFG.DMAEN bit.
- If Pretended Networking mode is required, set the CANFD_CFG.PNETEN bit.
- Enable the abort mechanism by setting the CANFD_CFG.ABORTEN bit.
- Enable the local priority feature by setting the CANFD_CFG.LPRIOEN bit.

2. Initialize the Control 1 Register (CANFD_CTL1) and optionally the CAN Bit Timing Register (CANFD_TIMING). Also, initialize the CAN FD CAN Bit Timing Register (CANFD_FD_TIMING):
   - Determine the bit timing parameters: CANFD_CTL1.PROPSEG, CANFD_CTL1.PSEG1, CANFD_CTL1.PSEG2, CANFD_CTL1.RJW.
   - Optionally determine the extended bit timing parameters: CANFD_TIMING.EPROPSEG, CANFD_TIMING.EPSEG1, CANFD_TIMING.EPSEG2, CANFD_TIMING.ERJW.
   - Determine the CAN FD bit timing parameters: CANFD_FD_TIMING.FPROPSEG, CANFD_FD_TIMING.FPSEG1, CANFD_FD_TIMING.FPSEG2, CANFD_FD_TIMING.FRJW.
   - Determine the bit rate by programming the CANFD_CTL1.PRESDIV field and optionally the CANFD_TIMING.EPRESDIV field.
   - Determine the CAN FD bit rate by programming the CANFD_FD_TIMING.FPRESDIV field. Determine the internal arbitration mode (CANFD_CTL1.LBUF bit).
3. Initialize the Message Buffers:
   - Initialize the C/S word of all MBs.
   - If the Rx FIFO is enabled, initialize the ID filter table.
   - Initialize other entries in each MB.
4. Initialize the individual masks in the CANFD_IMSK1 and CANFD_IMSK2 registers.
5. Set required interrupt mask bits in the CANFD_IMSK1 and CANFD_IMSK2 registers (for all MB interrupts), in the CANFD_CFG register for the wake up interrupt, and in the CANFD_CTL1 and CANFD_CTL2 registers for the bus off and error interrupts.
6. If pretended networking mode is enabled, configure the necessary registers for selective wake up.
7. Disable the CANFD_CFG.HALT bit.

After the last step listed, the CANFD module attempts to synchronize to the CAN bus.

Pretended Network - Doze Mode

Complete the following sequence to enter the pretended networking mode in doze mode:

1. Set the CANFD_CFG.PNETEN bit to enable the controller to enter the pretended networking mode, when one of the low power modes is requested.
2. Clear the CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK bits.
3. Request the CANFD to enter Doze mode.
4. Wait for the CANFD to set the CANFD_CFG.LPMACK bit.
5. Clear the CANFD_WUM.WTOF and/or CANFD_WUM.WUMF bits (if either bit is set).
6. Set the CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK bits.

Pretended Network - Stop Mode

Complete the following sequence to enter the pretended networking mode in stop mode:

1. Set the CANFD_CFG.PNETEN bit to enable the controller to enter pretended networking mode, when one of the low power modes is requested.
2. Clear the CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK bits.
3. Request the CANFD to enter Stop mode. Keep CAN0_IPG_STOP_CLK_EXT/ CAN1_IPG_STOP_CLK_EXT bits set to keep the clock to register block running.
4. Wait for the CANFD to set the CANFD_CFG.LPMACK bit.
5. Clear the CANFD_WUM.WTOF and/or CANFD_WUM.WUMF bits (if either bit is set).
6. Set the CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK bits.
7. Disable CANFD register clocks by clearing bits CAN0_IPG_STOP_CLK_EXT/ CAN1_IPG_STOP_CLK_EXT for CANFD0/CANFD1 respectively.

ADSP-SC59x CANFD Register Descriptions

Controller Access Network with Flexible Data Rate (CANFD) (CANFD) contains the following registers.

Table 16-32: ADSP-SC59x CANFD Register List

Name	Description
CANFD_TIMING	Can Bit Timing Register
CANFD_CRC	CRC Register
CANFD_CTL1	Control 1 Register
CANFD_PN_CTL1	Pretended Networking Control1 Register

Table 16-32: ADSP-SC59x CANFD Register List (Continued)

Name	Description
CANFD_CTL2	Control 2 Register
CANFD_PN_CTL2	Pretended Networking Control2 Register
CANFD_ECR	Error Counter Register
CANFD_ERR_IADDR	Error Injection Address Register
CANFD_ERR_IDP	Error Injection Data Pattern Register
CANFD_ERR_IPP	Error Injection Parity Pattern Register
CANFD_ERR_STAT	Error Status Register
CANFD_ESR1	Error and Status 1 Register
CANFD_ESR2	Error and Status 2 Register
CANFD_FD_TIMING	CANFD Bit Timing Register
CANFD_FD_CRC	CANFD CRC Register
CANFD_FD_CTL	CANFD Control Register
CANFD_FLTR_DLC	Pretended Networking DLC Filter Register
CANFD_FLTR_ID1	Pretended Networking ID Filter1 Register
CANFD_FLTR_ID2_IDMSK	Pretended Networking ID Filter2 / IDMask Register
CANFD_IFLG1	Mailbox Interrupt Flag 1 Register
CANFD_IFLG2	Mailbox Interrupt Flag 2 Register
CANFD_IMSK1	Mailbox Interrupt Mask 1 Register
CANFD_IMSK2	Mailbox Interrupt Mask 2 Register
CANFD_CFG	Module Configuration Register
CANFD_MEC	Memory Error Control Register
CANFD_FLTR_DATA1_HI	Pretended Networking Payload Low Filter2 Register
CANFD_FLTR_DATA1_LO	Pretended Networking Payload Low Filter1 Register
CANFD_FLTR_DATA2_DMSK_HI	Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register
CANFD_FLTR_DATA2_DMSK_LO	Pretended Networking Payload Low Filter2 / Payload Low Mask Register
CANFD_ERR_RADDR	Error Report Address Register
CANFD_ERR_RDAT	Error Report Data Register
CANFD_ERR_RSYN	Error Report Syndrome Register
CANFD_RX_14_MSK	Receive Mailbox14 Mask Register
CANFD_RX_15_MSK	Receive Mailbox15 Mask Register
CANFD_RX_FIFO_GMSK	Receive FIFO Global Mask Register

Table 16-32: ADSP-SC59x CANFD Register List (Continued)

Name	Description
CANFD_RX_FIFO	Receive FIFO Information Register
CANFD_RX_IMSK[n]	Receive Individual Mask Register
CANFD_RX_MB_GMSK	Receive Mailbox Global Mask Register
CANFD_TMR	Free Running Timer Register
CANFD_WMB[n]_DATA_HI	Wakeup Message Buffer Data 4-7 Register
CANFD_WMB[n]_DATA_LO	Wakeup Message Buffer Data 0-3 Register
CANFD_WMB[n]_ID	Wakeup Message ID Buffer Register
CANFD_WMB[n]_STAT	Wakeup Message Buffer Control/Status Register
CANFD_WUM	Pretended Networking Wakeup Match Register

Can Bit Timing Register

The CANFD_TIMING register is an alternative way to store the CAN bit timing variables described in the CANFD_CTL1 register. The CANFD_TIMING.EPRESDIV, CANFD_TIMING.EPROPSEG, CANFD_TIMING.EPSEG1, CANFD_TIMING.EPSEG2, and CANFD_TIMING.ERJW bit fields are extended versions of the CANFD_CTL1 register CANFD_CTL1.PRESDIV, CANFD_CTL1.PROPSEG, CANFD_CTL1.PSEG1, CANFD_CTL1.PSEG2 and CANFD_CTL1.RJW bit fields respectively.

The CANFD_TIMING.BTF bit selects the use of the timing variables defined in the CANFD_TIMING register. The contents of the CANFD_TIMING register are not affected by a soft reset.

NOTE: The CAN bit variables in the CANFD_CTL1 and CANFD_TIMING registers are stored in the same register.

NOTE: When the CAN FD feature is enabled (CANFD_CFG.FDEN is set), always set CANFD_TIMING.BTF.

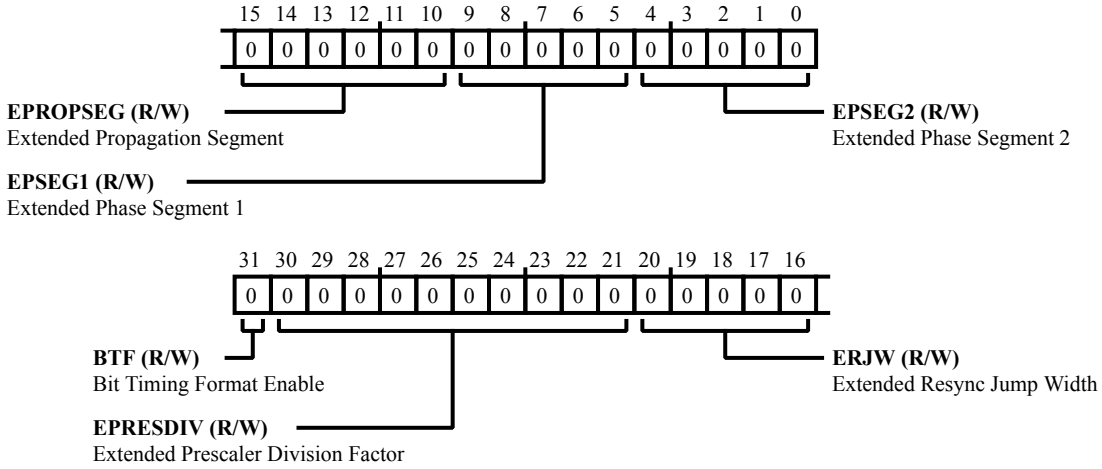


Figure 16-15: CANFD_TIMING Register Diagram

Table 16-33: CANFD_TIMING Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	BTF	Bit Timing Format Enable. The CANFD_TIMING.BTF bit enables the use of the extended CAN bit timing fields in the CANFD_TIMING register to replace the CAN bit time variables defined in the CANFD_CTL1 register. The CANFD_TIMING.BTF bit can only be written in freeze mode.	
		0	Disable
		1	Enable

Table 16-33: CANFD_TIMING Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30:21 (R/W)	EPRESDIV	Extended Prescaler Division Factor. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.EPRESDIV bit field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency. The CANFD_TIMING.EPRESDIV bit field extends the CANFD_CTL1.PRESDIV value range. Sclock frequency = PE clock frequency / (EPRESDIV + 1) The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.EPRESDIV bit field has no effect. The CANFD_TIMING.EPRESDIV bit field can only be written in freeze mode and is blocked by hardware in other modes.
20:16 (R/W)	ERJW	Extended Resync Jump Width. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.ERJW bit field defines the maximum number of time quanta that a bit time can be changed by one resynchronization. The CANFD_TIMING.ERJW bit field extends the CANFD_CTL1.RJW value range. Resync Jump Width = ERJW + 1 One time quantum is equal to the Sclock period. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.ERJW bit field has no effect. The CANFD_TIMING.ERJW bit field can only be written in freeze mode and is blocked by hardware in other modes.
15:10 (R/W)	EPROPSEG	Extended Propagation Segment. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.EPROPSEG bit field defines the length of the propagation segment in the bit time. The CANFD_TIMING.EPROPSEG bit field extends the CANFD_CTL1.PROPSEG value range. Propagation Segment Time = (EPROPSEG + 1) Time-Quanta One time quantum is equal to the Sclock period. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.EPROPSEG bit field has no effect. The CANFD_TIMING.EPROPSEG bit field can only be written in freeze mode and is blocked by hardware in other modes.

Table 16-33: CANFD_TIMING Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9:5 (R/W)	EPSEG1	Extended Phase Segment 1. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.EPSEG1 bit field defines the length of phase segment 1 in the bit time. The CANFD_TIMING.EPSEG1 bit field extends the CANFD_CTL1.PSEG1 value range. Phase Buffer Segment 1 = (EPSEG1 + 1) Time-Quanta One time quantum is equal to the Sclock period. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.EPSEG1 bit field has no effect. The CANFD_TIMING.EPSEG1 bit field can only be written in freeze mode and is blocked by hardware in other modes.
4:0 (R/W)	EPSEG2	Extended Phase Segment 2. When the CANFD_TIMING.BTF bit is enabled, the CANFD_TIMING.EPSEG2 bit field defines the length of phase segment 2 in the bit time. The CANFD_TIMING.EPSEG2 bit field extends the CANFD_CTL1.PSEG2 value range. Phase Buffer Segment 1 = (EPSEG1 + 1) Time-Quanta One time quantum is equal to the Sclock period. When the CANFD_TIMING.BTF bit is disabled, the CANFD_TIMING.EPSEG2 bit field has no effect. The CANFD_TIMING.EPSEG2 bit field can only be written in freeze mode and is blocked by hardware in other modes.

CRC Register

The CANFD_CRC register provides information about the CRC of transmitted messages for non-FD messages. For messages in CAN FD format that require either 17 or 21 bits, the CANFD_CRC register only reports the 15 low order bits of CRC calculations. For CAN FD format frames, the CANFD_FD_CRC register must be used. The CANFD_CRC register updates at the same time the Tx Interrupt Flag is set.

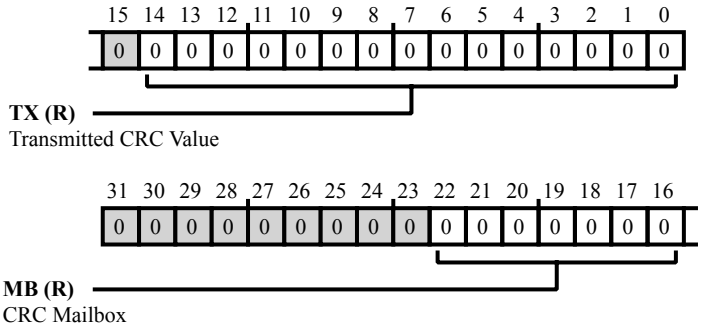


Figure 16-16: CANFD_CRC Register Diagram

Table 16-34: CANFD_CRC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
22:16 (R/NW)	MB	CRC Mailbox. The CANFD_CRC.MB bit field indicates the number of the mailbox corresponding to the value in the CANFD_CRC.TX field.
14:0 (R/NW)	TX	Transmitted CRC Value. The CANFD_CRC.TX bit field indicates the CRC value of the last transmitted message for non-FD frames. For CAN FD frames, the CRC value is reported in the CANFD_FD_CRC register.

Control 1 Register

The CANFD_CTL1 register is defined for specific CANFD control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, loop-back mode, listen-only mode, bus off recovery behavior, and interrupt enabling (bus off, error, warning). It also determines the division factor for the clock prescaler.

The CAN bit timing variables (PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW) are also configurable in the CANFD_TIMING register, which extends the range of all these variables. If the CANFD_TIMING.BTF bit is set, the PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW fields of the CANFD_CTL1 register become read-only.

NOTE: When the CAN FD feature is enabled, do not use the CANFD_CTL1.PRESDIV, CANFD_CTL1.RJW, CANFD_CTL1.PSEG1, CANFD_CTL1.PSEG2, and CANFD_CTL1.PROPSEG fields of the CANFD_CTL1 register for CAN bit timing. Instead, use the CANFD_TIMING.EPRESDIV, CANFD_TIMING.ERJW, CANFD_TIMING.EPSEG1, CANFD_TIMING.EPSEG2, and CANFD_TIMING.EPROPSEG fields of the CANFD_TIMING register.

The contents of the CANFD_CTL1 register are not affected by a soft reset.

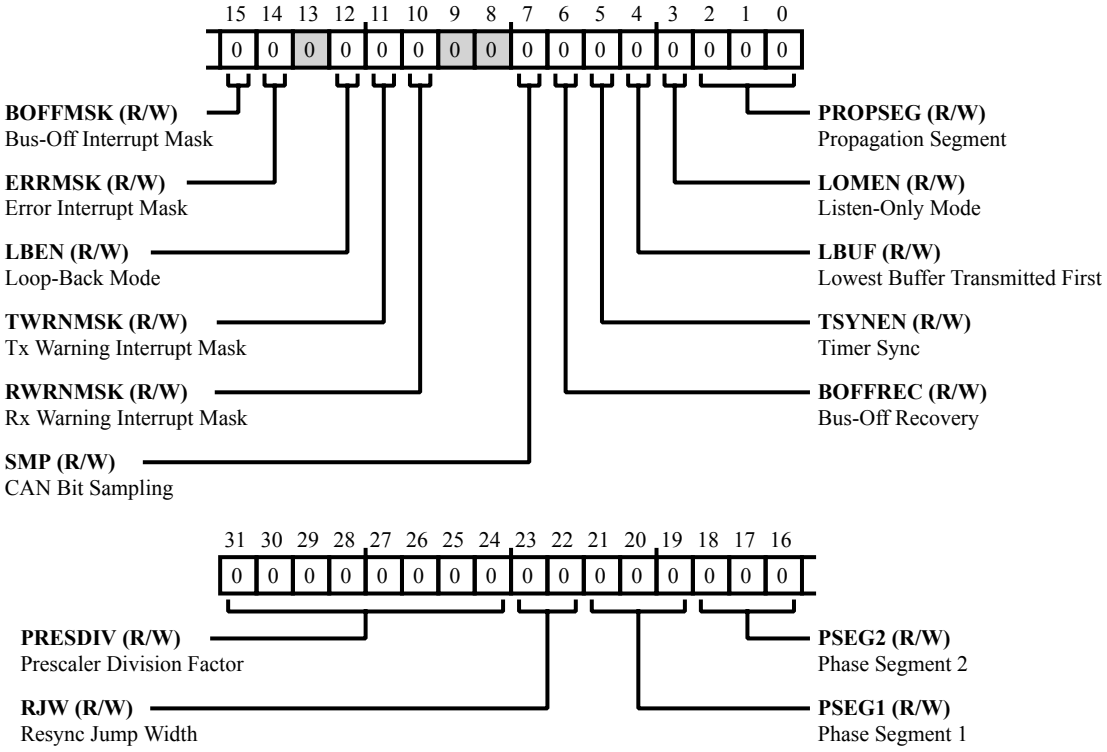


Figure 16-17: CANFD_CTL1 Register Diagram

Table 16-35: CANFD_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	PRESDIV	Prescaler Division Factor. The CANFD_CTL1.PRESDIV bit field defines the ratio between the PE clock frequency and the serial clock (SCLOCK) frequency. The SCLOCK period defines the time quantum of the CAN protocol. For the reset value, the SCLOCK frequency is equal to the PE clock frequency. The maximum value of PRESDIV is 0xFF, which gives a minimum SCLOCK frequency equal to the PE clock frequency divided by 256. The CANFD_CTL1.PRESDIV bit can only be written in freeze mode and is blocked by hardware in other modes. SCLOCK Frequency = PE Clock Frequency / (PRESDIV + 1)	
23:22 (R/W)	RJW	Resync Jump Width. The CANFD_CTL1.RJW bit field defines the maximum number of time quanta that a bit time can change with one re-synchronization. One time quantum is equal to the SCLOCK period. The valid programmable values for the CANFD_CTL1.RJW bits are 03. This bit field can only be written in freeze mode and is blocked by hardware in other modes. Resync Jump Width = RJW + 1	
21:19 (R/W)	PSEG1	Phase Segment 1. The CANFD_CTL1.PSEG1 bit field defines the length of the phase segment 1 of the bit time. The valid programmable values are 07. This bit field can only be written in freeze mode and is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) Time-Quanta	
18:16 (R/W)	PSEG2	Phase Segment 2. The CANFD_CTL1.PSEG2 bit field defines the length of phase segment 2 of the bit time. The valid programmable values are 17. This bit field can only be written in freeze mode and is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) Time-Quanta	
15 (R/W)	BOFFMSK	Bus-Off Interrupt Mask. The CANFD_CTL1.BOFFMSK bit provides a mask for the bus off interrupt (CANFD_ESR1.BOFFINT).	
		0	Disable
		1	Enable

Table 16-35: CANFD_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	ERRMSK	Error Interrupt Mask. The CANFD_CTL1.ERRMSK bit provides a mask for the error interrupt (CANFD_ESR1.ERRINT).	
		0	Disable
		1	Enable
12 (R/W)	LBEN	Loop-Back Mode. The CANFD_CTL1.LBEN bit configures the CANFD module to operate in loop-back mode. In loop-back mode, the CANFD modules performs an internal loop back to use for self-test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). The CANFD module behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In loop-back mode, the CANFD module ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. The CANFD_CTL1.LBEN bit can only be written in freeze mode and is blocked by hardware in other modes. Note that in loop-back mode, the CANFD_CFG.SRXDIS bit cannot be asserted, because it impedes the self reception of a transmitted message. Also, note that the CANFD_FD_CTL.TDCOMPEN bit must be 0 (transceiver delay compensation feature disabled) when the CANFD_CTL1.LBEN bit is asserted.	
11 (R/W)	TWRNMSK	Tx Warning Interrupt Mask. The CANFD_CTL1.TWRNMSK bit provides a mask for the Tx warning interrupt (CANFD_ESR1.TXWRNINT) The CANFD_CTL1.TWRNMSK bit is read as zero when CAN_MCR.WRNEN is negated. The CANFD_CTL1.TWRNMSK bit can only be written when the CANFD_CFG.WRNEN bit is enabled.	
		0	Tx Warning Interrupt Disabled
		1	Tx Warning Interrupt Enabled

Table 16-35: CANFD_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	RWRNMSK	Rx Warning Interrupt Mask. The CANFD_CTL1.RWRNMSK bit provides a mask for the Rx warning interrupt (CANFD_ESR1.RXWRNINT) The CANFD_CTL1.RWRNMSK bit is read as zero when the CANFD_CFG.WRNEN bit is negated. It can only be written only if the CANFD_CFG.WRNEN bit is enabled.	
		0	Rx Warning Interrupt Disabled
		1	Rx Warning Interrupt Enabled
7 (R/W)	SMP	CAN Bit Sampling. The CANFD_CTL1.SMP bit defines the sampling mode of CAN bits at the Rx input. When the CANFD_CTL1.SMP bit is disabled, one sample determines the bit value. When it is enabled three samples determine the value of the received bit: the regular sample (sample point) and the two preceding samples; a majority rule is used. The CANFD_CTL1.SMP bit can only be written in freeze mode and is blocked by hardware in other modes. Note that for proper operation, when enabling CAN bit sampling, guarantee a minimum value of 2 time quanta in the CANFD_CTL1.PSEG1 (or CANFD_TIMING.EPSEG1) bit field. SMP cannot be asserted when CAN FD is enabled (CANFD_CFG.FDEN = 1).	
		0	Disable
		1	Enable

Table 16-35: CANFD_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	BOFFREC	Bus-Off Recovery. The CANFD_CTL1.BOFFREC bit defines how the CANFD module recovers from bus off state. When the CANFD_CTL1.BOFFREC bit is enabled, automatic recovering from the bus off state occurs according to the CAN Specification 2.0B. When the CANFD_CTL1.BOFFREC bit is disabled, automatic recovering from the bus off state is disabled and the CANFD module remains in the bus off state until the CANFD_CTL1.BOFFREC bit is enabled by software. If this occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then the bus off recovery happens as if the CANFD_CTL1.BOFFREC bit was never disabled. If the bit is enabled after 128 sequences of 11 recessive bits occurred, then the CANFD module will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After being enabled, the CANFD_CTL1.BOFFREC bit can be disabled again during the bus off state, but it will be effective only the next time the CANFD module enters the bus off state. If CANFD_CTL1.BOFFREC bit was enabled when the CANFD module entered the bus off state, disabling the CANFD_CTL1.BOFFREC bit during the bus off state will not be effective for the current bus off recovery.	
		0	Enable
		1	Disable
5 (R/W)	TSYNEN	Timer Sync. The CANFD_CTL1.TSYNEN bit enables a mechanism that resets the free-running timer each time a message is received in message buffer 0. The timer sync feature provides a means to synchronize multiple CANFD stations with a special SYNC message (global network time). If the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the first available mailbox, according to the CANFD_CTL2.RFFNUM setting, is used for timer synchronization instead of MB0. The CANFD_CTL1.TSYNEN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable

Table 16-35: CANFD_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	LBUF	Lowest Buffer Transmitted First. The CANFD_CTL1.LBUF bit defines the ordering mechanism for message buffer transmission. When the CANFD_CTL1.LBUF bit is enabled, the lowest number buffer is transmitted first and when it is disabled the buffer with highest priority is transmitted first. When the CANFD_CTL1.LBUF bit is enabled, the CANFD_CFG.LPRIOEN bit does not affect the arbitration priority. The CANFD_CTL1.LBUF bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable
3 (R/W)	LOMEN	Listen-Only Mode. The CANFD_CTL1.LOMEN bit configures the CANFD module to operate in listen-only mode. When the CANFD_CTL1.LOMEN is enabled to be in listen-only mode, transmission is disabled, all error counters in the CANFD_ECR register are frozen, and the CANFD module operates in the CAN error passive mode. Only messages acknowledged by another CAN station will be received. If the CANFD module detects a message that has not been acknowledged, it flags a BIT0 error without changing the receive error counter (CANFD_ECR.RXERRCNT), as if it were trying to acknowledge the message. Listen-Only mode is acknowledged by the state of the CANFD_ESR1.FLTCONF bit field indicating a passive error. There can be some delay between the Listen-Only mode request and acknowledge. The CANFD_CTL1.LOMEN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable
2:0 (R/W)	PROPSEG	Propagation Segment. The CANFD_CTL1.PROPSEG bit field defines the length of the propagation segment in the bit time. The valid programmable values are 07. The CANFD_CTL1.PROPSEG bit can only be written in freeze mode and is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) Time-Quanta. Time-Quantum = One SCLOCK Period.	

Pretended Networking Control1 Register

The CANFD_PN_CTL1 register contains control bits for pretended networking mode filtering selection. Configure this register with the filter criteria to receive wakeup messages. Bits other than CANFD_PN_CTL1.WTOFMSK and CANFD_PN_CTL1.WUMFMSK can only be written in freeze mode.

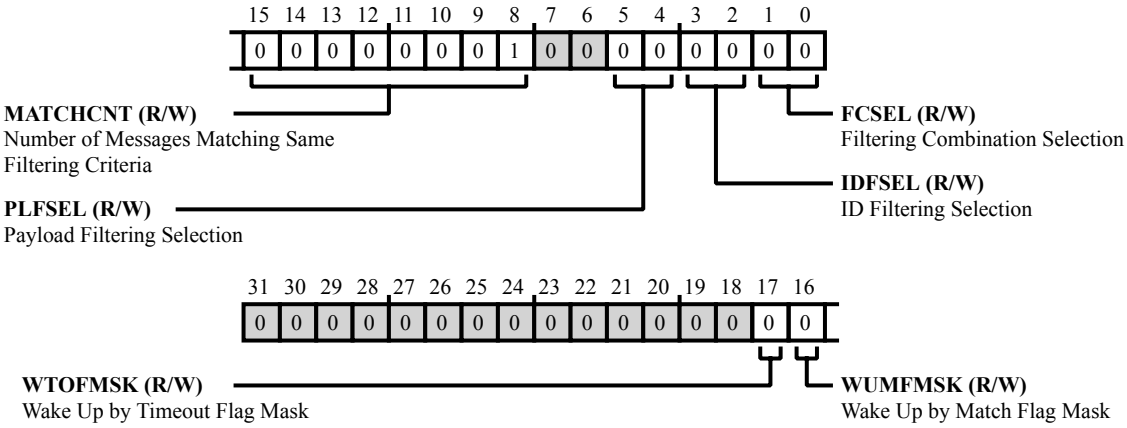


Figure 16-18: CANFD_PN_CTL1 Register Diagram

Table 16-36: CANFD_PN_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	WTOFMSK	Wake Up by Timeout Flag Mask. The CANFD_PN_CTL1.WTOFMSK bit masks the generation of a wakeup event originated by a timeout:	
		0	Timeout Event Disabled
		1	Timeout Event Enabled
16 (R/W)	WUMFMSK	Wake Up by Match Flag Mask. The CANFD_PN_CTL1.WUMFMSK bit masks the generation of a wakeup event originated by a successfully filtered Rx message.	
		0	Wakeup match disabled
		1	Wakeup match enabled

Table 16-36: CANFD_PN_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:8 (R/W)	MATCHCNT	Number of Messages Matching Same Filtering Criteria. The CANFD_PN_CTL1.MATCHCNT bit field defines the number of times a given message must match the predefined filtering criteria for ID and/or PL before generating a wakeup event. This quantity is configured in the 1 to 255 range. For example, if the received message must match the predefined filtering criteria for ID and/or PL once before generating a wakeup event, the CANFD_PN_CTL1.MATCHCNT bit field value is 0x01; if it must match twice before generating a wakeup event, the bit field value is 0x02; and if it must match 255 times generating a wakeup event, the CANFD_PN_CTL1.MATCHCNT bit field value is 0xFF.	
5:4 (R/W)	PLFSEL	Payload Filtering Selection. The CANFD_PN_CTL1.PLFSEL bit field selects the level of payload filtering to be applied when the CANFD module is under pretended networking mode. Filtering does not accept remote messages (RTR=1) when payload filtering is active.	
		0	Match the payload contents against an exact target value
		1	Match if a payload value is greater than or equal to a specific target value
		2	Match if a payload value is less than or equal to a specific target value
		3	Match if a payload value is inside a range, greater than or equal to a specified lower limit and smaller than or equal a specified upper limit
3:2 (R/W)	IDFSEL	ID Filtering Selection. The CANFD_PN_CTL1.IDFSEL bit field selects the level of ID filtering to be applied when the CANFD module is under pretended networking mode. In ID filtering, if the CANFD_FLTR_ID2_IDMSK.IDE and CANFD_FLTR_ID2_IDMSK.RTR bits are set, the IDE and RTR bits are also considered part of the reception filter.	
		0	Match the ID contents against an exact target value
		1	Match if the ID value is greater than or equal to a specific target value
		2	Match if the ID value is less than or equal to a specific target value
		3	Match if the ID value is inside a range, greater than or equal to a specified lower limit and smaller than or equal a specified upper limit

Table 16-36: CANFD_PN_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	FCSEL	Filtering Combination Selection. The `CANFD_PN_CTL1.FCSEL` bit field selects elects the filtering criteria to be applied when the CANFD module is in pretended networking mode.	
		0	Message ID filtering only
		1	Message ID filtering and payload filtering
		2	Message ID filtering occurring a specified number of times
		3	Message ID filtering and payload filtering a specified number of times

Control 2 Register

The CANFD_CTL2 register complements CANFD_CTL1, providing control bits for memory write accesses in freeze mode, for extending FIFO filter quantity, and for adjusting the operation of internal CANFD processes like matching and arbitration.

The contents of the CANFD_CTL2 register are not affected by soft reset.

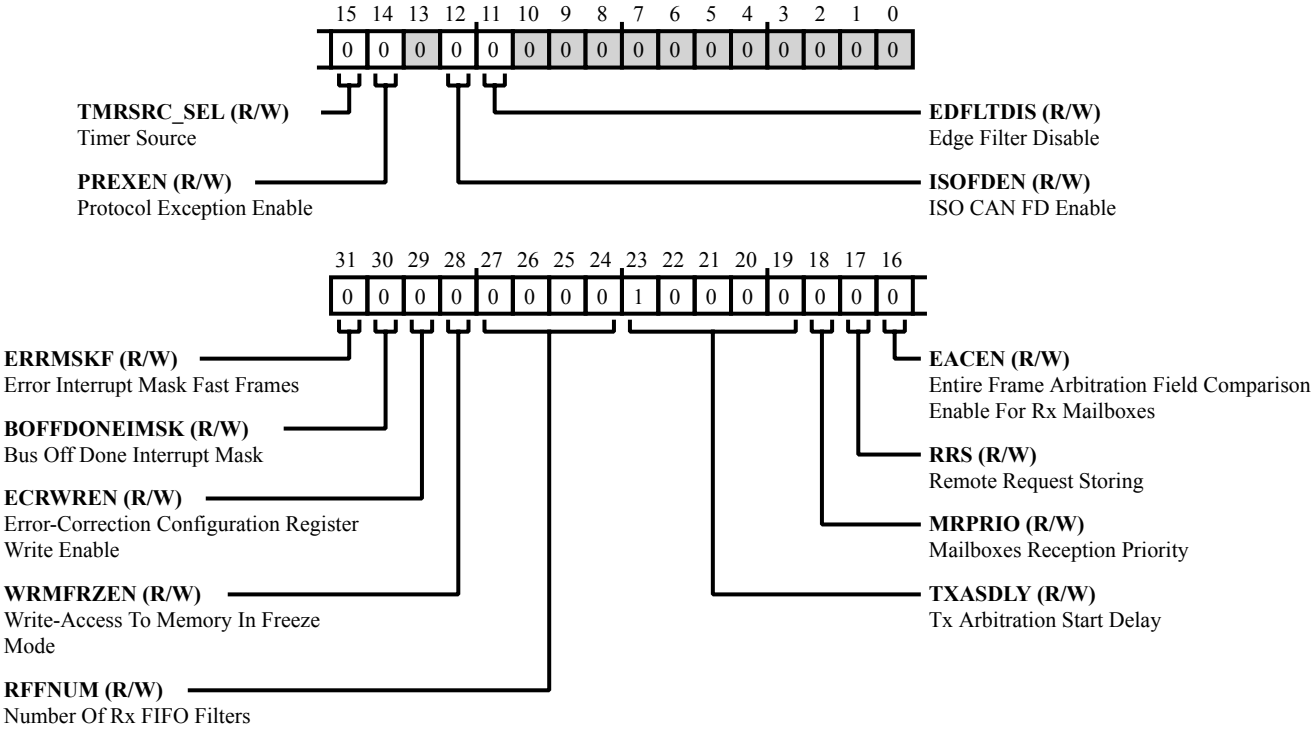


Figure 16-19: CANFD_CTL2 Register Diagram

Table 16-37: CANFD_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	ERRMSKF	Error Interrupt Mask Fast Frames. The CANFD_CTL2.ERRMSKF bit is the error interrupt mask for errors detected in the data phase of fast CAN FD frames. It provides a mask for the CANFD_ESR1.ERRINTF interrupt. When the CANFD_CTL2.ERRMSKF bit is set, the error interrupt is enabled. When the bit is cleared the error interrupt is disabled.

Table 16-37: CANFD_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W)	BOFFDONEIMSK	Bus Off Done Interrupt Mask. The CANFD_CTL2.BOFFDONEIMSK bit provides a mask for the bus off done interrupt in the CANFD_ESR1 register. When the CANFD_CTL2.BOFFDONEIMSK bit is set, the bus off done interrupt is enabled. When the bit is cleared the bus off done interrupt is disabled.
29 (R/W)	ECRWREN	Error-Correction Configuration Register Write Enable. The CANFD_CTL2.ECRWREN bit enables the MECR register to be updated. Setting the CANFD_CTL2.ECRWREN bit is set, enables updates. Clearing the bit disables updates. The CANFD_CTL2.ECRWREN bit is automatically set to zero if the protocol described in the Detection and Correction of Memory Errors section of the CANFD chapter is not followed.
28 (R/W)	WRMFRZEN	Write-Access To Memory In Freeze Mode. The CANFD_CTL2.WRMFRZEN bit enables unrestricted write access to the CANFD memory in freeze mode. Clear the CANFD_CTL2.WRMFRZEN bit to maintain the write access restrictions. Set the CANFD_CTL2.WRMFRZEN bit to enable unrestricted write access to the CANFD memory. The CANFD_CTL2.WRMFRZEN bit can only be written in freeze mode and has no effect out of freeze mode. The CANFD_CFG.RFEN bit must not be set during CANFD memory initialization.

Table 16-37: CANFD_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
27:24 (R/W)	RFFNUM	Number Of Rx FIFO Filters. The CANFD_CTL2.RFFNUM bit field defines the number of Rx FIFO filters. For more details, see the Number of Rx FIFO Filters table in the CANFD chapter. The maximum selectable number of filters is a function of configuration parameter NUMBER_OF_MB. Do not program the CANFD_CTL2.RFFNUM bit field with values that make the number of message buffers occupied by the Rx FIFO and ID Filter exceed the number of mailboxes present, defined by the CANFD_CFG.MAXMB bit field. Each group of eight filters occupies a memory space equivalent to two message buffers, which means that as more filters are implemented fewer mailboxes are available. Considering that the Rx FIFO occupies the memory space originally reserved for MB05, program the CANFD_CTL2.RFFNUM bit with a value corresponding to a number of filters not greater than the number of available memory words. Calculate this as follows: (SETUP_MB - 6) 4 where SETUP_MB is the lower value between parameter NUMBER_OF_MB and register field CANFD_CFG.MAXMB. The number of remaining mailboxes available is (SETUP_MB - 8) - (RFFN 2) If the number of Rx FIFO filters programmed through the CANFD_CTL2.RFFNUM bit field exceeds the SETUP_MB value (memory space available), the exceeding ones are not functional. Note that the number of the last remaining available mailbox is defined by the least value between (NUMBER_OF_MB - 1) and the CANFD_CFG.MAXMB field. If the individual Rx mask registers are not enabled, all Rx FIFO filters are affected by the Rx FIFO global mask. The CANFD_CTL2.RFFNUM bit field can only be written in freeze mode as it is blocked by hardware in other modes.
23:19 (R/W)	TXASDLY	Tx Arbitration Start Delay. The CANFD_CTL2.TXASDLY bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on the CAN bus. The CANFD_CTL2.TXASDLY bit field can only be written in freeze mode and it is blocked by hardware in other modes.
18 (R/W)	MRPRIO	Mailboxes Reception Priority. The CANFD_CTL2.MRPRIO bit defines the mailboxes reception priority. If the CANFD_CTL2.MRPRIO bit is set, the matching process starts from the mailboxes and if no match occurs the matching continues on the Rx FIFO. If the bit is cleared, matching starts from Rx FIFO and continues on mailboxes. The CANFD_CTL2.MRPRIO bit can only be written only in freeze mode and it is blocked by hardware in other modes.

Table 16-37: CANFD_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	RRS	Remote Request Storing. If the CANFD_CTL2.RRS bit is set, a remote request frame is submitted to a matching process and stored in the corresponding message buffer in the same fashion of a data frame. No automatic remote response frame is generated. If the CANFD_CTL2.RRS bit is cleared, the remote request frame is submitted to a matching process and an automatic remote response frame is generated if a message buffer with CODE=0b1010 is found with the same ID. The CANFD_CTL2.RRS bit can only be written in freeze mode and it is blocked by hardware in other modes.	
16 (R/W)	EACEN	Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes. The CANFD_CTL2.EACEN bit controls the comparison of IDE and RTR bits within Rx mailboxes filters with their corresponding bits in the incoming frame by the matching process. The CANFD_CTL2.EACEN bit does not affect matching for the Rx FIFO. When the CANFD_CTL2.EACEN bit is set it enables the comparison of both the IDE and RTR bits of an Rx mailbox filter with the corresponding bits of the incoming frame. Mask bits do apply. When the CANFD_CTL2.EACEN bit is cleared the IDE bit of the Rx mailbox filter is always compared and RTR is never compared, regardless of mask bits. EACEN can be only written only in freeze mode and it is blocked by hardware in other modes.	
15 (R/W)	TMRSRC_SEL	Timer Source. The CANFD_CTL2.TMRSRC_SEL bit selects the time tick source used for incrementing the free running timer counter. When the CANFD_CTL2.TMRSRC_SEL bit is set, the free running timer is clocked by an external time tick. The period can be either adjusted to be equal to the baud rate on the CAN bus or a different value as required. When the CANFD_CTL2.TMRSRC_SEL bit is cleared, the free running timer is clocked by the CAN bit clock, which defines the baud rate on the CAN bus. The CANFD_CTL2.TMRSRC_SEL bit can only be written in freeze mode.	
14 (R/W)	PREXEN	Protocol Exception Enable. The CANFD_CTL2.PREXEN bit enables the protocol exception feature. The CANFD_CTL2.PREXEN bit can only be written in freeze mode.	
		0	Disable
		1	Enable

Table 16-37: CANFD_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	ISOFDEN	ISO CAN FD Enable. The `CANFD_CTL2.ISOFDEN` bit controls ISO CAN FD compliant operation. Setting the bit enables ISO CAN FD compliant operation by enabling the following features, which are part of the ISO 11898 standard and not included in the original (Bosch) CAN FD protocol specification: The count of variable stuff bits inserted from the Start of Frame bit to the last bit of Data field. Also, the modulo 8 count of variable stuff bits plus the respective parity bit (even parity calculated over the 3-bit modulo 8 count) are combined as the 4-bit Stuff Count field and inserted before the CRC sequence field. CRC calculation extends beyond the end of data field and takes the stuff count field bits into account. Clearing the `CANFD_CTL2.ISOFDEN` bit disables ISO CAN FD specific features (non-ISO CAN FD operation). The `CANFD_CTL2.ISOFDEN` bit is only writable in freeze mode.	
11 (R/W)	EDFLTDIS	Edge Filter Disable. The `CANFD_CTL2.EDFLTDIS` bit controls the edge filter used during the bus integration state. When the edge filter is enabled, two consecutive nominal time quanta with dominant bus state are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of eleven consecutive recessive bits is restarted. The edge filter prevents the dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD frame) from being mistaken for an idle condition. The `CANFD_CTL2.EDFLTDIS` bit is only writable in freeze mode.	
		0	Enable
		1	Disable

Pretended Networking Control2 Register

The CANFD_PN_CTL2 register contains configuration bits for the timeout value under pretended networking mode. Only write to this register in freeze mode.

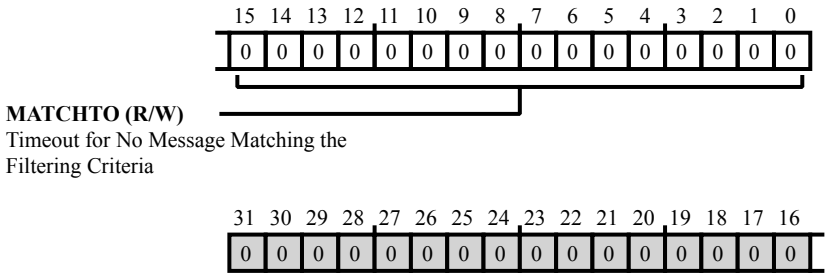


Figure 16-20: CANFD_PN_CTL2 Register Diagram

Table 16-38: CANFD_PN_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	MATCHTO	Timeout for No Message Matching the Filtering Criteria. The CANFD_PN_CTL2.MATCHTO bit field defines a timeout value that generates a wakeup event when the CANFD_CFG.PNETEN bit is enabled. If the timeout counter reaches the target value when the CANFD module is in pretended networking mode, a wakeup event is generated. The timeout limit value is configured from 1 to 65535 to control an internal 16-bit up-count timer to produce a trigger upon reaching this configured value. The internal timer is incremented based on periodic time ticks, with a period 64 times the CAN bit time unit. When the CANFD_PN_CTL2.MATCHTO bit field is 0x0000, the timeout is disabled.

Error Counter Register

The CANFD_ECR register has four 8-bit fields reflecting the value of the CANFD error counters:

• Transmit error counter (CANFD_ECR.TXERRCNT field)

• Receive error counter (CANFD_ECR.RXERRCNT field)

• Transmit error counter for errors detected in the data phase of CAN FD messages with the BRS bit set (CANFD_ECR.TXERRCNTF field)

• Receive error counter for errors detected in the data phase of CAN FD messages with the BRS bit set (CANFD_ECR.RXERRCNTF field)

The TXERRCNT and RXERRCNT counters take into account all errors in both CAN FD and non-FD message formats. The TXERRCNTF and RXERRCNT_FAST counters only count the errors that occur in the data phase of CAN FD frames with the BRS bit set.

The fault confinement state (CANFD_ESR1.FLTCONF) is only updated based on the TXERRCNT and RXERRCNT counters. The TXERRCNT and RXERRCNT counters can only be written in freeze mode. The TXERRCNTF and RXERRCNT_FAST counters are read-only except in freeze mode, where the processor can write a zero value. The rules for incrementing and decrementing these counters are described in the CAN protocol and are completely implemented in the CANFD module.

The basic rules for the CANFD bus state transitions are as follows:

• If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the CANFD_ESR1.FLTCONF field updates to reflect the error passive state.

• If the CANFD state is error passive, and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the CANFD_ESR1.FLTCONF field updates to reflect the error active state.

• If the value of TXERRCNT increases to be greater than 255, the CANFD_ESR1.FLTCONF field updates to reflect the bus off state and an interrupt may be issued. The value of TXERRCNT is then reset to zero.

• If the CANFD is in the bus off state, then TXERRCNT is cascaded together with another internal counter to count the 128 occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing TXERRCNT. When TXERRCNT reaches the value of 128, the CANFD_ESR1.FLTCONF field is updated to be error active and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value. The TXERRCNTF counter is frozen during bus off.

• If during system start-up, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the CANFD_ESR1.ACKERR bit). After the transition to the error passive state, TXERRCNT is no longer incremented by acknowledge errors. Therefore, the device never goes to the bus off state.

• If RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to the error active state.

• The TXERRCNTF and RXERRCNT_FAST error counters values increment and decrement based on errors detected only in the data phase of CAN FD frames with the BRS bit set, following the same increment and decrement rules as the TXERRCNT and RXERRCNT counters. These counters do not wrap around and get stuck at their maximum value (255). They stop counting and keep their values frozen while the CANFD is in the bus off state. They are reset when the CANFD leaves the bus off state and restart counting once the CANFD resumes to the error active state.

• When the CANFD module is in pretended networking mode, RXERRCNT and RXERRCNT_FAST keep counting errors and error flags are stored. TXERRCNT and TXERRCNTF preserve their values and do not change since no transmission occurs under pretended networking mode. Error counters and error flags that changed values while in pretended networking mode are updated in the CANFD_ECR and CANFD_ESR1 registers when the CANFD module resumes normal mode. The FAST error flags in the CANFD_ESR1 register are not set if the CANFD is in pretended networking mode.

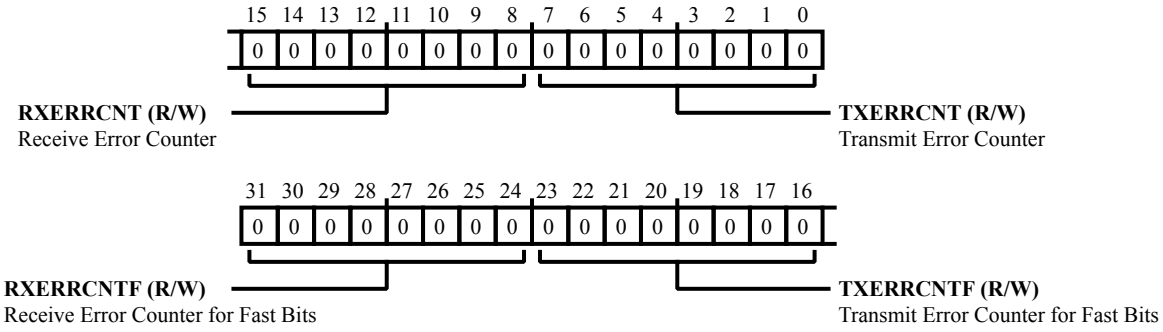


Figure 16-21: CANFD_ECR Register Diagram

Table 16-39: CANFD_ECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	RXERRCNTF	Receive Error Counter for Fast Bits. The CANFD_ECR.RXERRCNTF bit field is for errors detected in the data phase of received CAN FD messages with the BRS bit set. The CANFD_ECR.RXERRCNTF bit field is read-only except in freeze mode. In freeze mode the processor can only write a 8-bit zero value.
23:16 (R/W)	TXERRCNTF	Transmit Error Counter for Fast Bits. The CANFD_ECR.TXERRCNTF bit field is for errors detected in the data phase of received CAN FD messages with the BRS bit set. The CANFD_ECR.TXERRCNTF bit field is read-only except in freeze mode. In freeze mode the processor can only write a 8-bit zero value.

Table 16-39: CANFD_ECR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	RXERRCNT	Receive Error Counter. The `CANFD_ECR.RXERRCNT` bit field is for all errors detected in received messages. The `CANFD_ECR.RXERRCNT` bit field is read-only except in freeze mode, where the processor can write it.
7:0 (R/W)	TXERRCNT	Transmit Error Counter. The `CANFD_ECR.TXERRCNT` bit field is for all errors detected in transmitted messages. The `CANFD_ECR.TXERRCNT` bit field is read-only except in freeze mode, where the processor can write it.

Error Injection Address Register

The CANFD_ERR_IADDR register holds the address where a memory read error is injected.

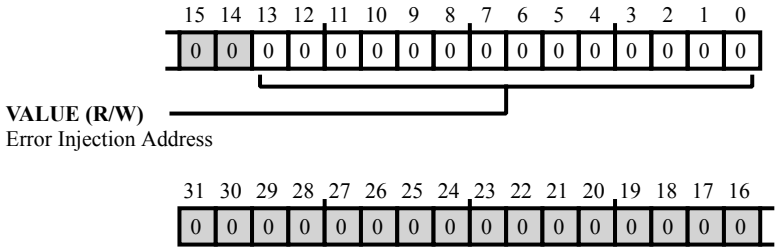


Figure 16-22: CANFD_ERR_IADDR Register Diagram

Table 16-40: CANFD_ERR_IADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:0 (R/W)	VALUE	Error Injection Address. The CANFD_ERR_IADDR.VALUE bit field defines the physical RAM address where a memory read error is to be injected. See the Error Injection Address Mapping table in the CANFD chapter for more details. Note that the two least significant bits are always read as zero.

Error Injection Data Pattern Register

The CANFD_ERR_IDP register holds the error pattern injected in the data word that is read from memory.

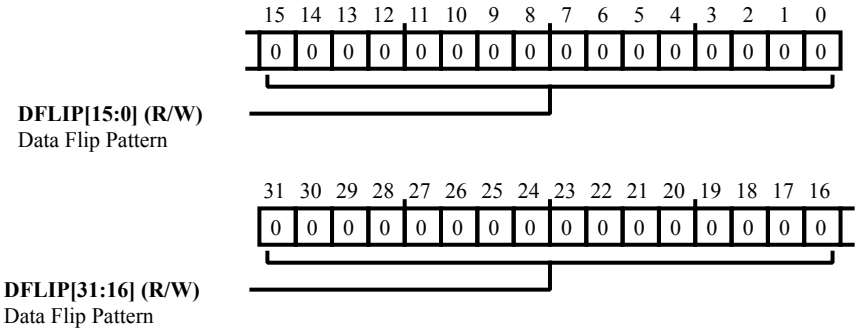


Figure 16-23: CANFD_ERR_IDP Register Diagram

Table 16-41: CANFD_ERR_IDP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DFLIP	Data Flip Pattern. The CANFD_ERR_IDP.DFLIP bit field determines the data flip pattern. Bits set to one in the flip pattern cause the corresponding data bit in the word read from memory to invert.

Error Injection Parity Pattern Register

The CANFD_ERR_IPP register holds the error pattern that is injected in the parity bits read from memory along with data word. Bits set to one in the flip pattern cause the corresponding parity bit, in the word read from memory, to invert.

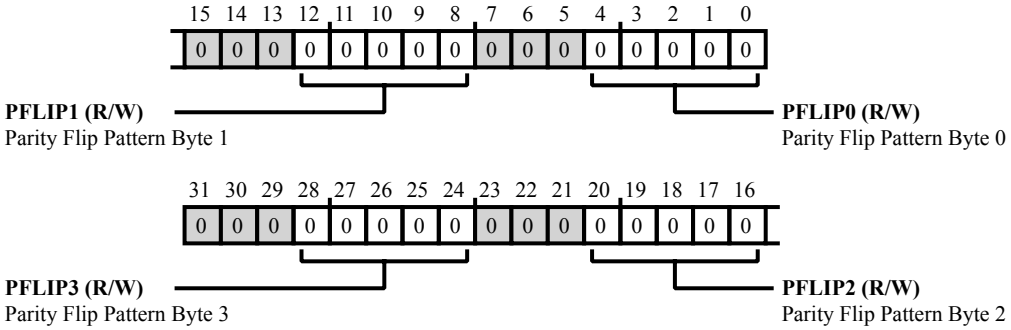


Figure 16-24: CANFD_ERR_IPP Register Diagram

Table 16-42: CANFD_ERR_IPP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:24 (R/W)	PFLIP3	Parity Flip Pattern Byte 3. The CANFD_ERR_IPP.PFLIP3 bit field is for the parity flip pattern for byte 3 (most significant).
20:16 (R/W)	PFLIP2	Parity Flip Pattern Byte 2. The CANFD_ERR_IPP.PFLIP2 bit field is for the parity flip pattern for byte 2.
12:8 (R/W)	PFLIP1	Parity Flip Pattern Byte 1. The CANFD_ERR_IPP.PFLIP1 bit field is for the parity flip pattern for byte 1.
4:0 (R/W)	PFLIP0	Parity Flip Pattern Byte 0. The CANFD_ERR_IPP.PFLIP0 bit field is for the parity flip pattern for byte 0 (least significant).

Error Status Register

The CANFD_ERR_STAT register holds the status bits of the error correction and detection operations. Clear these flags by writing them with a one. Writing a zero to the flags has no effect.

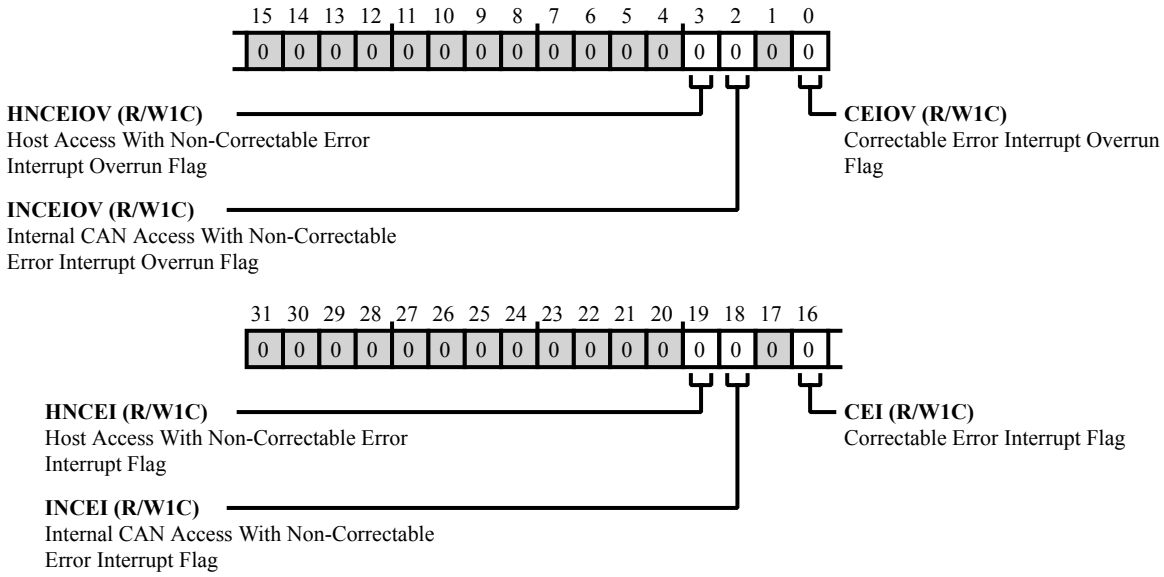


Figure 16-25: CANFD_ERR_STAT Register Diagram

Table 16-43: CANFD_ERR_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W1C)	HNCEI	Host Access With Non-Correctable Error Interrupt Flag. The CANFD_ERR_STAT.HNCEI bit indicates if a non-correctable error is detected in a host access. If the CANFD_ERR_STAT.HNCEI bit is set, a non-correctable error was detected in a memory read initiated by the host. A bus transfer error is asserted for that access. If the CANFD_MEC.HNCIMSK bit is set, the interrupt is asserted. If the CANFD_ERR_STAT.HNCEI bit is cleared, no non-correctable errors were detected in host accesses so far.	
		0	No Non-Correctable Error Detected
		1	Non-Correctable Error Detected

Table 16-43: CANFD_ERR_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W1C)	INCEI	Internal CAN Access With Non-Correctable Error Interrupt Flag. The CANFD_ERR_STAT.INCEI bit indicates if a non-correctable error is detected in a CAN access. If the CANFD_ERR_STAT.INCEI bit is set, a non-correctable error was detected in a memory read initiated by a CANFD internal process. A bus transfer error is asserted for that access. If the CANFD_MEC.INCEMSK bit is set, the interrupt is asserted. If the CANFD_ERR_STAT.INCEI bit is cleared, no non-correctable errors were detected in CAN accesses so far.	
		0	No Non-Correctable Error Detected
		1	Non-Correctable Error Detected
16 (R/W1C)	CEI	Correctable Error Interrupt Flag. The CANFD_ERR_STAT.CEI bit indicates if a correctable error is detected. If the CANFD_ERR_STAT.CEI bit is set, a correctable error was detected in a memory read. If the CANFD_MEC.CEIMSK bit is set, the interrupt is asserted. If the CANFD_ERR_STAT.CEI bit is cleared, no correctable errors were detected so far.	
		0	No Correctable Error Detected
		1	Correctable Error Detected
3 (R/W1C)	HNCEIOV	Host Access With Non-Correctable Error Interrupt Overrun Flag. The CANFD_ERR_STAT.HNCEIOV bit indicates if an overrun on non-correctable error is detected in a host access. If the CANFD_ERR_STAT.HNCEIOV bit is set, a non-correctable error was detected in a memory read initiated by the host when the CANFD_ERR_STAT.HNCEI bit is set. No interrupt is associated with this flag. If the CANFD_ERR_STAT.HNCEIOV bit is cleared, no overrun on non-correctable errors were detected in a host access.	
		0	No Overrun Detected
		1	Overrun Detected

Table 16-43: CANFD_ERR_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	INCEIOV	Internal CAN Access With Non-Correctable Error Interrupt Overrun Flag. The `CANFD_ERR_STAT.INCEIOV` bit indicates if an overrun on non-correctable error is detected in a CAN access. If the `CANFD_ERR_STAT.INCEIOV` bit is set, a non-correctable error was detected in a memory read initiated by a CANFD internal process when the `CANFD_ERR_STAT.INCEI` bit is set. No interrupt is associated with this flag. If the `CANFD_ERR_STAT.INCEIOV` bit is cleared, no overrun on non-correctable errors were detected in a CAN access.	
		0	No Overrun Detected
		1	Overrun Detected
0 (R/W1C)	CEIOV	Correctable Error Interrupt Overrun Flag. The `CANFD_ERR_STAT.CEIOV` bit indicates if an overrun on correctable errors is detected. If the `CANFD_ERR_STAT.CEIOV` bit is set, a overrun on a correctable error was detected in a memory read while the `CANFD_ERR_STAT.CEI` bit was set. No interrupt is associated with this flag. If the `CANFD_ERR_STAT.CEIOV` bit is cleared, no overrun on correctable errors were detected.	
		0	No Overrun Detected
		1	Overrun Detected

Error and Status 1 Register

The CANFD_ESR1 register reports various error conditions detected in the reception and transmission of a CAN frame, some general status of the device, and is the source of some interrupts to the processor.

The reported error conditions are:

• CANFD_ESR1.B1ERR, CANFD_ESR1.B0ERR, CANFD_ESR1.ACKERR, CANFD_ESR1.CRCERR, CANFD_ESR1.FRMERR, and CANFD_ESR1.STFERR, for errors detected in CAN frames of any format.

• CANFD_ESR1.B1ERRF, CANFD_ESR1.B0ERRF, CANFD_ESR1.CRCERRF, CANFD_ESR1.FRMERRF, and CANFD_ESR1.STFERRF for errors detected in the data phase of CAN FD frames with the BRS bit set.

An error detected in a single CAN frame is reported by one or more error flags. Error reporting is cumulative in case additional error events occur in subsequent frames before the processor reads the CANFD_ESR1 register.

CANFD_ESR1.TXWRN, CANFD_ESR1.RXWRN, CANFD_ESR1.IDLE, CANFD_ESR1.TXINPROG, CANFD_ESR1.FLTCONF, CANFD_ESR1.RXINPROG, and CANFD_ESR1.SYNC are status bits.

CANFD_ESR1.BOFFINT, CANFD_ESR1.BOFFDONEINT, CANFD_ESR1.ERRINT, CANFD_ESR1.ERRINTF, CANFD_ESR1.WAKINT,CANFD_ESR1.TXWRNINT, and CANFD_ESR1.RXWRNINT are interrupt bits. When servicing interrupt requests generated by these bits use the following procedure:

• Read the CANFD_ESR1 register to capture all error condition and status bits. This action clears the respective bits set since the last read access.

• Write 1 to clear the interrupt bit that triggered the interrupt request.

• Write 1 to clear the CANFD_ESR1.ERROVR bit if it is set.

Starting from all error flags cleared, a first error event sets either the CANFD_ESR1.ERRINT or the CANFD_ESR1.ERRINTF bits (provided the corresponding mask bit is asserted). If other error events in subsequent frames occur before the processor services the interrupt request, the CANFD_ESR1.ERROVR bit is set to indicate that errors from different frames have accumulated.

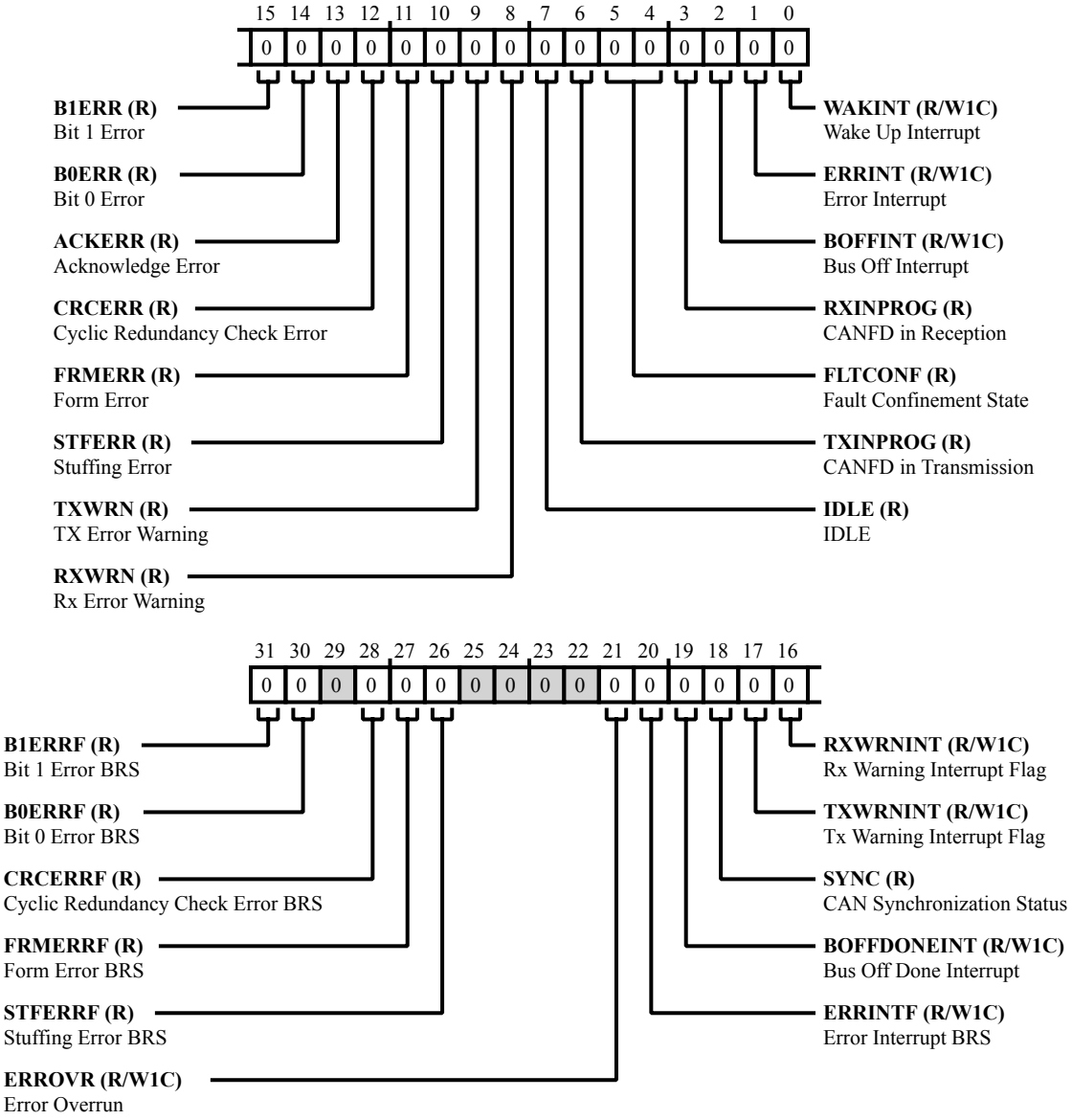


Figure 16-26: CANFD_ESR1 Register Diagram

Table 16-44: CANFD_ESR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	B1ERRF	Bit 1 Error BRS. The CANFD_ESR1.B1ERRF bit indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames with the BRS bit set. When the CANFD_ESR1.B1ERRF bit is enabled, at least one bit sent as recessive is received as dominant. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
30 (R/NW)	B0ERRF	Bit 0 Error BRS. The CANFD_ESR1.B0ERRF bit indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames with the BRS bit set. When the CANFD_ESR1.B0ERRF bit is enabled, at least one bit sent as recessive is received as dominant. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
28 (R/NW)	CRCERRF	Cyclic Redundancy Check Error BRS. The CANFD_ESR1.CRCERRF bit indicates if a CRC error is detected by the receiver node in the CRC field of CAN FD frames with the BRS bit set. The calculated CRC is different from the received CRC. When the CANFD_ESR1.CRCERRF bit is enabled, a CRC error occurred since the last read of the register. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
27 (R/NW)	FRMERRF	Form Error BRS. The CANFD_ESR1.FRMERRF bit indicates that a form error is detected by the receiver node in the data phase of CAN FD frames with the BRS bit set. A fixed-form bit field contains at least one illegal bit. When the CANFD_ESR1.FRMERRF bit is enabled, a form error occurred since the last read of the register. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/NW)	STFERRF	Stuffing Error BRS. The CANFD_ESR1.STFERRF bit indicates that a stuffing error is detected in the data phase of CAN FD frames with the BRS bit set. When the CANFD_ESR1.STFERRF bit is enabled, a stuffing error occurred since the last read of the register. When the bit is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
21 (R/W1C)	ERROVR	Error Overrun. The CANFD_ESR1.ERROVR bit indicates that an error condition occurred when any error flag is already set. When the CANFD_ESR1.ERROVR bit is enabled an overrun has occurred. When it is disabled, an overrun has not occurred. Clear the CANFD_ESR1.ERROVR bit by writing a one to it.	
		0	Disable
		1	Enable
20 (R/W1C)	ERRINTF	Error Interrupt BRS. The CANFD_ESR1.ERRINTF bit indicates an error interrupt is detected in the data phase of CAN FD frames with the BRS bit set. When the CANFD_ESR1.ERRINTF bit is enabled it indicates setting the of any error bit detected in the data phase of CAN FD frames with the BRS bit set. When it is disabled, there is no such occurrence. The CANFD_ESR1.ERRINTF bit indicates that at least one of the error bits detected in the data phase of CAN FD frames with the BRS bit set (CANFD_ESR1.B1ERRF, CANFD_ESR1.B0ERRF, CANFD_ESR1.CRCERRF, CANFD_ESR1.FRMERRF, or CANFD_ESR1.STFERRF) is set. If the CANFD_CTL2.ERRMSKF is set, an interrupt is generated to the processor. Clear the CANFD_ESR1.ERRINTF bit by writing a one to it. Writing a zero has no effect.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W1C)	BOFFDONEINT	Bus Off Done Interrupt. When the CANFD_ESR1.BOFFDONEINT bit is enabled it indicates the CANFD module has completed the bus off process. When it is disabled, there is no such occurrence. The CANFD_ESR1.BOFFDONEINT bit is set when the Tx Error Counter (CANFD_ECR.TXERRCNT) finishes counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave bus off mode. If the CANFD_CTL2.BOFFDONEIMSK is set, an interrupt is generated to the processor. Clear the CANFD_ESR1.BOFFDONEINT bit by writing a one to it. Writing a zero has no effect.	
		0	Disable
		1	Enable
18 (R/NW)	SYNC	CAN Synchronization Status. The CANFD_ESR1.SYNC bit is a read-only flag that indicates whether the CANFD module is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the CANFD module. When the CANFD_ESR1.SYNC bit is enabled, the CANFD module is synchronized to the CAN bus. When it is disabled, the CANFD module is not synchronized to the CAN bus.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	TXWRNINT	Tx Warning Interrupt Flag. When the CANFD_ESR1.TXWRNINT bit is enabled, the Tx error counter transitioned from less than 96 to greater than or equal to 96. When the bit is disabled, there is no such occurrence. If the CANFD_CFG.WRNEN bit is enabled, the CANFD_ESR1.TXWRNINT bit is set when the CANFD_ESR1.TXWRN bit transitions from 0 to 1, meaning that the Tx error counter reached 96. If the CANFD_CTL1.TWRNMSK bit is set, an interrupt is generated to the processor. The CANFD_ESR1.TXWRNINT bit is cleared by writing a one to it. Writing a zero has no effect. When the CANFD_CFG.WRNEN bit is disabled, the CANFD_ESR1.TXWRNINT flag is masked. The processor must clear the CANFD_ESR1.TXWRNINT flag before disabling the CANFD_CFG.WRNEN bit or the CANFD_ESR1.TXWRNINT bit will be set when the CANFD_CFG.WRNEN bit is enabled again. The CANFD_ESR1.TXWRNINT bit is not generated during the bus off state and is not updated during freeze mode. When the CANFD module returns to normal mode from pretended networking mode, the CANFD_ESR1.TXWRNINT bit does not update.	
		0	Disable
		1	Enable
16 (R/W1C)	RXWRNINT	Rx Warning Interrupt Flag. When the CANFD_ESR1.RXWRNINT bit is enabled, it indicates that the Rx error counter transitioned from less than 96 to greater than or equal to 96. When the bit is disabled, there is no such occurrence. If the CANFD_CFG.WRNEN bit is enabled, the CANFD_ESR1.RXWRNINT bit is set when the CANFD_ESR1.RXWRN bit transitions from 0 to 1, meaning that the Rx error counter reached 96. If the CANFD_CTL1.RWRNMSK bit is set, an interrupt is generated to the processor. The CANFD_ESR1.RXWRNINT bit is cleared by writing a one to it. Writing a zero has no effect. When the CANFD_CFG.WRNEN bit is disabled, the CANFD_ESR1.RXWRNINT flag is masked. The processor must clear the CANFD_ESR1.RXWRNINT flag before disabling the CANFD_CFG.WRNEN bit or the CANFD_ESR1.RXWRNINT bit will be set when the CANFD_CFG.WRNEN bit is enabled again. The CANFD_ESR1.RXWRNINT bit is not updated during freeze mode. When the CANFD module returns to normal mode from pretended networking mode, the CANFD_ESR1.RXWRNINT bit does not update to reflect the Rx error counter state.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	B1ERR	Bit 1 Error. The CANFD_ESR1.B1ERR bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. When the CANFD_ESR1.B1ERR bit is enabled, at least one bit sent as recessive is received as dominant. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.B1ERR bit updates when the CANFD module returns to normal mode from pretended networking mode. Note that the CANFD_ESR1.B1ERR bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits.	
		0	Disable
		1	Enable
14 (R/NW)	B0ERR	Bit 0 Error. The CANFD_ESR1.B0ERR bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. When the CANFD_ESR1.B0ERR bit is enabled, at least one bit sent as recessive is received as dominant. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.B0ERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable
13 (R/NW)	ACKERR	Acknowledge Error. The CANFD_ESR1.ACKERR bit indicates an acknowledge error is detected by the transmitter node (a dominant bit detected during the ACK SLOT) When the CANFD_ESR1.ACKERR bit is enabled an ACK error occurred since the last read of this register.When the bit is disabled, there is no such occurrence. The CANFD_ESR1.ACKERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/NW)	CRCERR	Cyclic Redundancy Check Error. The CANFD_ESR1.CRCERR bit indicates if a CRC error is detected by the receiver node in a non-FD message or in the arbitration or data phase of a frame in CAN FD format (the calculated CRC is different from the received CRC) When the CANFD_ESR1.CRCERR bit is enabled, a CRC error occurred since the last read of the register. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.CRCERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable
11 (R/NW)	FRMERR	Form Error. The CANFD_ESR1.FRMERR bit indicates that a form error is detected by the receiver node in a non-FD message or in a CAN FD message in the arbitration or data phase. A fixed-form bit field contains at least one illegal bit. When the CANFD_ESR1.FRMERR bit is enabled, a form error occurred since the last read of the register. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.FRMERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable
10 (R/NW)	STFERR	Stuffing Error. The CANFD_ESR1.STFERR bit indicates that a stuffing error is detected by the receiver node in a non-FD message or in a CAN FD message in the arbitration or data phase. When the CANFD_ESR1.STFERR bit is enabled, a stuffing error occurred since the last read of the register. When the bit is disabled, there is no such occurrence. The CANFD_ESR1.STFERR bit updates when the CANFD module returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/NW)	TXWRN	TX Error Warning. The CANFD_ESR1.TXWRN bit indicates when repetitive errors are occurring during message transmission. When the CANFD_ESR1.TXWRN bit is enabled, the CANFD_ECR.TXERRCNT value is greater than or equal to 96. When it is disabled, there is no such occurrence. The CANFD_ESR1.TXWRN bit is only affected by the CANFD_ECR.TXERRCNT value. It does not update during freeze mode.	
		0	Disable
		1	Enable
8 (R/NW)	RXWRN	Rx Error Warning. The CANFD_ESR1.RXWRN bit indicates when repetitive errors are occurring during message transmission. When the CANFD_ESR1.RXWRN bit is enabled, the CANFD_ECR.TXERRCNT value is greater than or equal to 96. When it is disabled, there is no such occurrence. The CANFD_ESR1.RXWRN bit is only affected by the CANFD_ECR.RXERRCNT value. It does not update during freeze mode. It updates when the CANFD returns to normal mode from pretended networking mode.	
		0	Disable
		1	Enable
7 (R/NW)	IDLE	IDLE. The CANFD_ESR1.IDLE bit indicates when the CAN bus is in the IDLE state. When the CANFD_ESR1.IDLE bit is enabled, the CAN bus is IDLE. When it is disabled, there is no such occurrence.	
		0	Disable
		1	Enable
6 (R/NW)	TXINPROG	CANFD in Transmission. The CANFD_ESR1.TXINPROG bit indicates if the CANFD module is transmitting a message. When the CANFD_ESR1.TXINPROG bit is enabled, the CANFD module is transmitting a message. When it is disabled, the CANFD module is not transmitting a message.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/NW)	FLTCONF	Fault Confinement State. The CANFD_ESR1.FLTCONF bit field indicates the fault confinement state of the CANFD module. If the CANFD_CTL1.LOMEN bit is enabled, after some delay that depends on the CAN bit timing, the CANFD_ESR1.FLTCONF bit field will indicate the error passive state. The same delay affects how the CANFD_ESR1.FLTCONF bit field reflects an update to the CANFD_ECR register by the processor. It may take up to one CAN bit time to get them coherent again. The CANFD_ESR1.FLTCONF bit field is affected by soft reset. However, if the CANFD_CTL1.LOMEN bit is enabled, the CANFD_ESR1.FLTCONF reset value lasts just one CAN bit. After this time, the CANFD_ESR1.FLTCONF bit field indicates the error passive state.	
		0	Error Active
		1	Error Passive
		2	Bus Off
		3	Bus Off
3 (R/NW)	RXINPROG	CANFD in Reception. The CANFD_ESR1.RXINPROG bit indicates if the CANFD module is receiving a message. When the CANFD_ESR1.RXINPROG bit is enabled the CANFD module is receiving a message and when it is disabled the CANFD module is not receiving a message.	
		0	Disable
		1	Enable
2 (R/W1C)	BOFFINT	Bus Off Interrupt. The CANFD_ESR1.BOFFINT bit is enabled when the CANFD module enters the bus off state. If the corresponding mask bit (CAN_CTRL1.BOFFMSK) is set, an interrupt is generated to the processor. When the CANFD_ESR1.BOFFINT bit is disabled, there is no such occurrence. The CANFD_ESR1.BOFFINT bit is cleared by writing one to it. Writing zero has no effect.	
		0	Disable
		1	Enable

Table 16-44: CANFD_ESR1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	ERRINT	Error Interrupt. The CANFD_ESR1.ERRINT bit indicates that at least one of the error bits (CANFD_ESR1.B1ERR, CANFD_ESR1.B0ERR, CANFD_ESR1.CRCERR, CANFD_ESR1.FRMERR, or CANFD_ESR1.STFERR) is set. If the corresponding mask bit (CANFD_CTL1.ERRMSK) is set, an interrupt is generated to the processor. The CANFD_ESR1.ERRINT bit is cleared by writing a one to it. Writing a zero has no effect.	
		0	Disable
		1	Enable
0 (R/W1C)	WAKINT	Wake Up Interrupt. The CANFD_ESR1.WAKINT bit applies when the CANFD module is in low-power mode under a self wake up mechanism (doze or stop mode). When the CANFD_ESR1.WAKINT bit is enabled it indicates a recessive-to-dominant transition was received on the CAN bus. When a recessive-to-dominant transition is detected on the CAN bus and the CANFD_CFG.WAKMSK bit is set, an interrupt is generated to the processor. This bit is cleared by writing it to one. When the bit is disabled, there is no such occurrence. When the CANFD_CFG.SLFWAKEN bit is disabled, the CANFD_ESR1.WAKINT flag is masked. The processor must clear the CANFD_ESR1.WAKINT flag before disabling the CANFD_CFG.SLFWAKEN bit. Otherwise, he CANFD_ESR1.WAKINT is set when the CANFD_CFG.SLFWAKEN bit is set again. Writing zero has no effect. The CANFD_ESR1.WAKINT bit is cleared by writing one to it. Writing zero has no effect.	
		0	Disable
		1	Enable

Error and Status 2 Register

The CANFD_ESR2 register reports general status information.

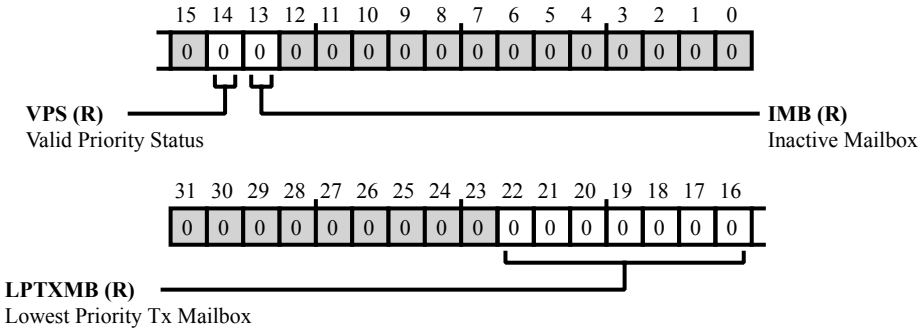


Figure 16-27: CANFD_ESR2 Register Diagram

Table 16-45: CANFD_ESR2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
22:16 (R/NW)	LPTXMB	Lowest Priority Tx Mailbox. If the CANFD_ESR2.VPS bit is set, the CANFD_ESR2.LPTXMB bit field indicates the lowest number inactive mailbox, per the CANFD_ESR2.IMB bit. If there is no inactive mailbox, then the mailbox indicated depends on the CANFD_CTL1.LBUF bit value. If the CANFD_CTL1.LBUF bit bit is cleared, the mailbox indicated is the one that has the greatest arbitration value. If the CANFD_CTL1.LBUF bit bit is set, the mailbox indicated is the highest number active Tx mailbox. If a Tx mailbox is being transmitted, it is not considered in the CANFD_ESR2.LPTXMB bit field calculation. If the CANFD_ESR2.IMB bit is not set and a frame is transmitted successfully, the CANFD_ESR2.LPTXMB bit field is updated with the mailbox number.

Table 16-45: CANFD_ESR2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/NW)	VPS	Valid Priority Status. The CANFD_ESR2.VPS bit indicates whether the CANFD_ESR2.IMB and CANFD_ESR2.LPTXMB contents are currently valid. If the CANFD_ESR2.VPS bit is set, the contents of the CANFD_ESR2.IMB and CANFD_ESR2.LPTXMB bits are valid. If the bit is cleared, the contents of the CANFD_ESR2.IMB and CANFD_ESR2.LPTXMB bits are invalid. The CANFD_ESR2.VPS bit is set upon every complete Tx arbitration process unless the processor writes to the C/S word of a mailbox that has already been scanned (is behind Tx Arbitration Pointer during the Tx arbitration process). If there is no inactive mailbox and only one Tx mailbox that is being transmitted, then the CANFD_ESR2.VPS bit is not set. The CANFD_ESR2.VPS bit is cleared upon the start of every Tx arbitration process or upon a write to the C/S word of any mailbox. Note that the CANFD_ESR2.VPS bit is not affected by any processor write into the C/S of a MB that is blocked by the abort mechanism. When the CANFD_CFG.ABORTEN bit is set, the abort code write into the C/S of an MB that is being transmitted (pending abort), or any write attempt into a Tx MB with the interrupt flag set is blocked.
13 (R/NW)	IMB	Inactive Mailbox. If the CANFD_ESR2.VPS bit is set, the CANFD_ESR2.IMB bit indicates whether there is any inactive mailbox (CODE field is either 0b1000 or 0b0000). If the CANFD_ESR2.IMB bit is set and the CANFD_ESR2.VPS bit is set and there is at least one inactive mailbox. The CANFD_ESR2.LPTXMB bit field content is the number of the first one. If the CANFD_ESR2.IMB bit is cleared and the CANFD_ESR2.VPS bit is set, the CANFD_ESR2.LPTXMB bit field is not an inactive mailbox. The CANFD_ESR2.IMB bit is set during arbitration, if a CANFD_ESR2.LPTXMB value is found and it is inactive. The CANFD_ESR2.IMB bit is also set if CAN_ESR2.MB is not set and a frame is transmitted successfully. The CANFD_ESR2.IMB bit is cleared at the start of arbitration. Note that if an MB is successfully transmitted and The CANFD_ESR2.IMB bit is zero (no inactive mailbox), then the CANFD_ESR2.VPS and CANFD_ESR2.IMB bits are set and the index related to the MB just transmitted is loaded into the CANFD_ESR2.LPTXMB bit field.

CANFD Bit Timing Register

The CANFD_FD_TIMING register stores the CAN bit timing variables for use in the data phase of CAN FD messages when the CANFD_FD_CTL.BRSEN bit is set, compatible with CAN FD specification. The CANFD_FD_TIMING.FPRESDIV, CANFD_FD_TIMING.FPROPSEG, CANFD_FD_TIMING.FPSEG1, CANFD_FD_TIMING.FPSEG2, and CANFD_FD_TIMING.FRJW bit fields define the time quantum duration, the number of time quanta per CAN bit, and the sample point position for the data bit rate portion of a CAN FD message with the BRS bit set.

The contents of the CANFD_FD_TIMING register are not affected by soft reset.

Note that the sum of values in the CANFD_FD_TIMING.FPROPSEG and CANFD_FD_TIMING.FPSEG1 bit fields must be at least two time quanta.

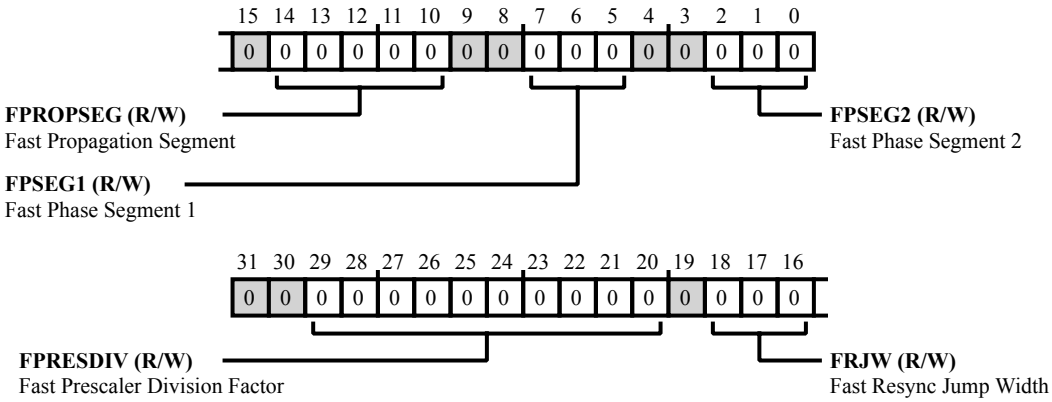


Figure 16-28: CANFD_FD_TIMING Register Diagram

Table 16-46: CANFD_FD_TIMING Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:20 (R/W)	FPRESDIV	Fast Prescaler Division Factor. The CANFD_FD_TIMING.FPRESDIV bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. This is in the data bit rate portion of a CAN FD message with the BRS bit set. The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. Sclock frequency = PE clock frequency / (FPRESDIV + 1). To minimize errors when processing FD frames, use the same value for the CANFD_FD_TIMING.FPRESDIV and CANFD_CTL1.PRESDIV bit fields The CANFD_FD_TIMING.FPRESDIV bit field can only be written in freeze mode and is blocked by hardware in other modes.

Table 16-46: CANFD_FD_TIMING Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
18:16 (R/W)	FRJW	Fast Resync Jump Width. The `CANFD_FD_TIMING.FRJW` bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization. This is in the data bit rate portion of a CAN FD message with the BRS bit set. Resync Jump Width = FSJW + 1. One time quantum is equal to the Sclock period. The `CANFD_FD_TIMING.FRJW` bit field can only be written in freeze mode and is blocked by hardware in other modes.
14:10 (R/W)	FPROPSEG	Fast Propagation Segment. The `CANFD_FD_TIMING.FPROPSEG` bit field defines the length of the propagation segment in the bit time. This is in the data bit rate portion of a CAN FD message with the BRS bit set. Propagation Segment Time = FPROPSEG Time-Quanta One time quantum is equal to the Sclock period. The `CANFD_FD_TIMING.FPROPSEG` bit field can only be written in freeze mode and is blocked by hardware in other modes.
7:5 (R/W)	FPSEG1	Fast Phase Segment 1. The `CANFD_FD_TIMING.FPSEG1` bit field defines the length of phase segment 1 in the bit time. This is in the data bit rate portion of a CAN FD message with the BRS bit set. Phase Segment 1 = (FPSEG1 + 1) Time-Quanta One time quantum is equal to the Sclock period. FPSEG1 can be written only in Freeze mode because it is blocked by hardware in other modes.
2:0 (R/W)	FPSEG2	Fast Phase Segment 2. The `CANFD_FD_TIMING.FPSEG2` bit field defines the length of phase segment 2. This is in the data bit rate portion of a CAN FD message with the BRS bit set. Phase Segment 2 = (FPSEG2 + 1) Time-Quanta One time quantum is equal to the Sclock period. FPSEG1 can be written only in Freeze mode because it is blocked by hardware in other modes.

CANFD CRC Register

The CANFD_FD_CRC register provides information about the CRC of transmitted messages. The CANFD module uses different CRC polynomials for different frame formats as follows:

- The CRC_15 polynomial is used for all frames in CAN format.
- The CRC_17 polynomial is used for frames in CAN FD format with a data field up to sixteen bytes.
- The CRC_21 polynomial is used for frames in CAN FD format with a data field longer than sixteen bytes.

Each polynomial shown below results in a Hamming distance of 6. The CANFD_FD_CRC register updates when the Tx interrupt flag is asserted.

CRC_15 = 0xC599:$(x^{15}+x^{14}+x^{10}+x^{8}+ x^{7}+x^{4}+x^{3}+1)$

CRC_17 = 0x3685B:$(x^{17}+x^{16}+x^{14}+x^{13}+x^{11}+x^{6}+x^{4}+x^{3}+x^{1}+1)$

CRC_21 = 0x302899:$(x^{21}+x^{20}+x^{13}+x^{11}+x^{7}+x^{4}+x^{3}+1)$

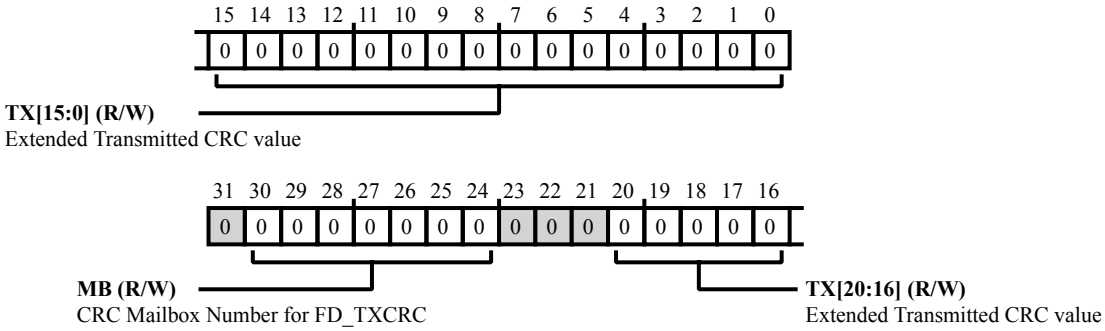


Figure 16-29: CANFD_FD_CRC Register Diagram

Table 16-47: CANFD_FD_CRC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30:24 (R/W)	MB	CRC Mailbox Number for FD_TXCRC. The CANFD_FD_CRC.MB bit field indicates the number of the mailbox corresponding to the value in the FD_TXCRC field, for both FD and non-FD frames.

Table 16-47: CANFD_FD_CRC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20:0 (R/W)	TX	Extended Transmitted CRC value. The `CANFD_FD_CRC.TX` bit field contains the CRC value calculated over the most recent transmitted message. There are different CRC polynomials for different frame formats. All frames in CAN format use a 15-bit polynomial, CRC_15. CRC_17 is for frames in CAN FD format with a data field up to sixteen bytes long. The 21-bit polynomial, CRC_21, is for frames in CAN FD format with a data field longer than sixteen bytes. For CRC_15 and CRC_17, the six most significant bits and the four most significant bits are reported as zeros, respectively. For CRC_15, this register has the same content as the `CANFD_CRC` register.

CANFD Control Register

The CANFD_FD_CTL register contains control bits for CAN FD operation. It also defines the data size of message buffers allocated in different partitions of RAM (memory blocks).

When 8 bytes of payload is selected, block R0 allocates MB0 to MB31 and block R1 allocates MB32 to MB63.

When more than 8 bytes of payload is selected, the maximum number of MBs in a block is limited as described as follows:

Payload Size => Maximum Number of Message Buffers per RAM Block

8 bytes => 32

16 bytes => 21

32 bytes => 12

64 bytes => 7

Note that one memory block fits exactly 32 MBs with 8 bytes payload. For the other options of payload sizes, empty memory may exist between last MB in a block and the beginning of the next block. This empty memory corresponds to less than one MB, and must not be used.

The contents of CANFD_FD_CTL register are not affected by soft reset.

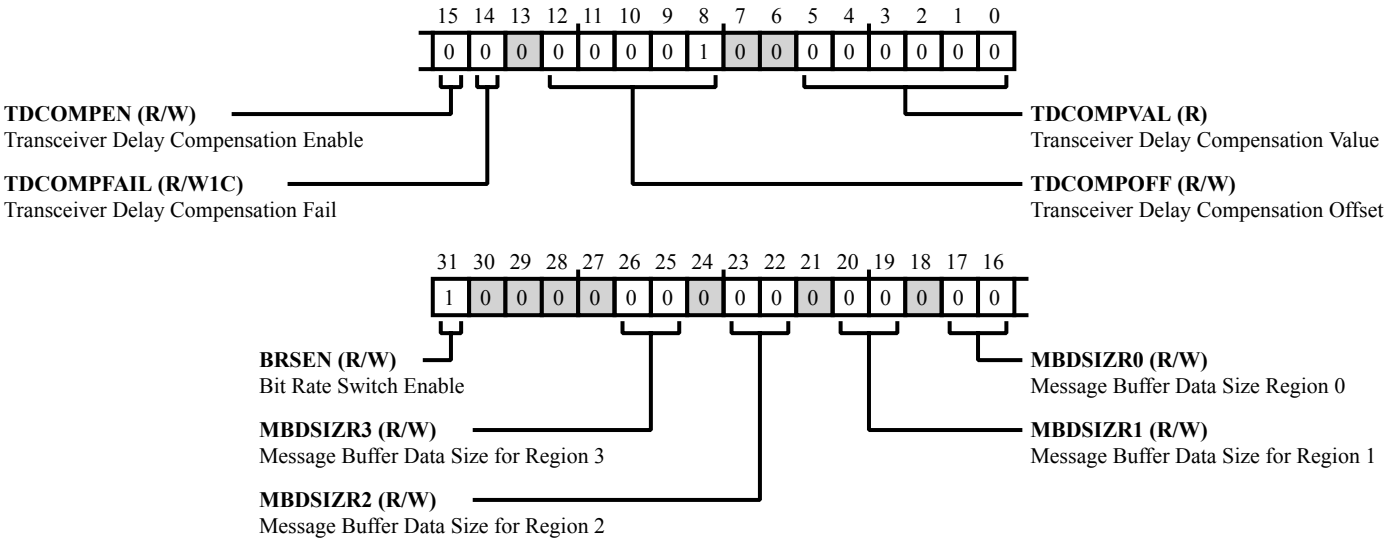


Figure 16-30: CANFD_FD_CTL Register Diagram

Table 16-48: CANFD_FD_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	BRSEN	Bit Rate Switch Enable. The CANFD_FD_CTL.BRSEN bit enables the effect of the bit rate switch (BRS) bit during the data phase of Tx messages. When the CANFD_FD_CTL.BRSEN bit is set, if the BRS bit in the Tx MB is recessive, the CANFD module transmits a frame with bit rate switching. When the CANFD_FD_CTL.BRSEN bit is clear, the BRS bit in the Tx MB has no effect and the CANFD module transmits a frame with the nominal rate. The processor can write the CANFD_FD_CTL.BRSEN bit any time, however, the effect only becomes active when the CAN bus is in the wait for bus idle, bus idle, bus off state, or when the current frame under reception or transmission reaches the inter-frame space. By negating the CANFD_FD_CTL.BRSEN bit, the processor can force all bits in CAN FD messages to be transmitted with a nominal bit rate, despite the value in the BRS bit of the Tx MBs.	
		0	Nominal Rate
		1	Bit Rate Switching
26:25 (R/W)	MBDSIZR3	Message Buffer Data Size for Region 3. The CANFD_FD_CTL.MBDSIZR3 field selects the data size for the region R3 of message buffers allocated in RAM. The CANFD_FD_CTL.MBDSIZR3 bit field can only be written in freeze mode.	
		0	8 bytes per message buffer
		1	16 bytes per message buffer
		2	32 bytes per message buffer
		3	64 bytes per message buffer
23:22 (R/W)	MBDSIZR2	Message Buffer Data Size for Region 2. The CANFD_FD_CTL.MBDSIZR2 field selects the data size for the region R2 of message buffers allocated in RAM. The CANFD_FD_CTL.MBDSIZR2 bit field can only be written in freeze mode.	
		0	8 bytes per message buffer
		1	16 bytes per message buffer
		2	32 bytes per message buffer
		3	64 bytes per message buffer

Table 16-48: CANFD_FD_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20:19 (R/W)	MBDSIZR1	Message Buffer Data Size for Region 1. The CANFD_FD_CTL.MBDSIZR1 bit field selects the data size for the region R1 of message buffers allocated in RAM. The CANFD_FD_CTL.MBDSIZR1 bit field can only be written in freeze mode.	
		0	8 bytes per message buffer
		1	16 bytes per message buffer
		2	32 bytes per message buffer
		3	64 bytes per message buffer
17:16 (R/W)	MBDSIZR0	Message Buffer Data Size Region 0. The CANFD_FD_CTL.MBDSIZR0 bit field selects the data size for the region R0 of message buffers allocated in RAM. The CANFD_FD_CTL.MBDSIZR0 bit field can only be written in freeze mode.	
		0	8 bytes per message buffer
		1	16 bytes per message buffer
		2	32 bytes per message buffer
		3	64 bytes per message buffer
15 (R/W)	TDCOMPEN	Transceiver Delay Compensation Enable. The CANFD_FD_CTL.TDCOMPEN bit enables or disables the TDC feature. The CANFD_FD_CTL.TDCOMPEN bit can only be written in freeze mode. Note that when loop-back mode is enabled (CANFD_CTL1.LBEN), the CANFD_FD_CTL.TDCOMPEN bit must be disabled.	
		0	Disable
		1	Enable
14 (R/W1C)	TDCOMPFAIL	Transceiver Delay Compensation Fail. The CANFD_FD_CTL.TDCOMPFAIL bit indicates when the transceiver delay compensation (TDC) mechanism is out of range. When the TDC is out of range it is unable to compensate for the loop delay of the transceiver and successfully compare the delayed received bits to the transmitted ones. The CANFD_FD_CTL.TDCOMPFAIL bit is set the first time the CANFD module detects the out of range condition. To clear the CANFD_FD_CTL.TDCOMPFAIL bit, write a one to it.	
		0	Measured loop delay is in range
		1	Measured loop delay is out of range

Table 16-48: CANFD_FD_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
12:8 (R/W)	TDCOMPOFF	Transceiver Delay Compensation Offset. The `CANFD_FD_CTL.TDCOMPOFF` bit field contains the offset value that must be added to the loop delay of the measured transceiver. This defines the position of the delayed comparison point when bit rate switching is active. The `CANFD_FD_CTL.TDCOMPOFF` bit field can only be written in freeze mode. The bit field value is defined in PE clock periods and must be smaller than the CAN bit duration in the data bit rate. Do not set the `CANFD_FD_CTL.TDCOMPOFF` bit field to zero.
5:0 (R/NW)	TDCOMPVAL	Transceiver Delay Compensation Value. The `CANFD_FD_CTL.TDCOMPVAL` bit field contains the value of the transceiver loop delay measured from the transmitted EDL to R0 transition edge to the respective received one added to the value in the `CANFD_FD_CTL.TDCOMPOFF` bit field. This value is an integer multiple of the PE clock period.

Pretended Networking DLC Filter Register

The CANFD_FLTR_DLC register contains the DLC inside range target values (CANFD_FLTR_DLC.HI and CANFD_FLTR_DLC.LO) for filtering incoming messages. The DLC range is only for payload filtering. The CANFD_FLTR_DLC register can only be written in freeze mode.

Note that when a fixed quantity of data bytes is required, both the CANFD_FLTR_DLC.HI and CANFD_FLTR_DLC.LO bit fields must have the same value; otherwise, a range of DLC is considered for filtering.

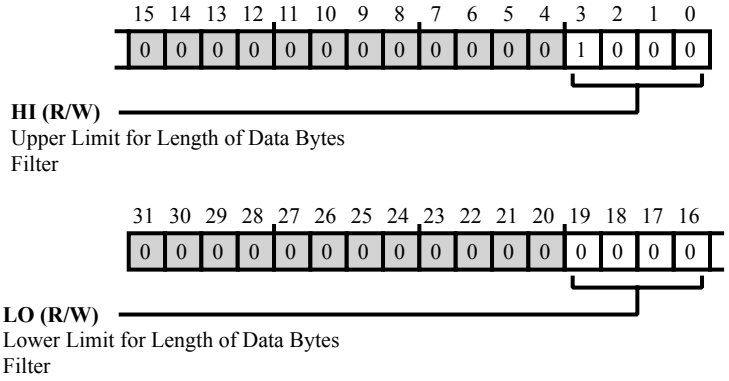


Figure 16-31: CANFD_FLTR_DLC Register Diagram

Table 16-49: CANFD_FLTR_DLC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:16 (R/W)	LO	Lower Limit for Length of Data Bytes Filter. The CANFD_FLTR_DLC.LO bit field specifies the lower limit on the number of data bytes considered valid for payload comparison. The CANFD_FLTR_DLC.LO bit field is part of payload reception filter.
3:0 (R/W)	HI	Upper Limit for Length of Data Bytes Filter. The CANFD_FLTR_DLC.HI bit field specifies the upper limit on the number of data bytes considered valid for payload comparison. The CANFD_FLTR_DLC.HI bit field is part of payload reception filter.

Pretended Networking ID Filter1 Register

The CANFD_FLTR_ID1 register contains the FLT_ID1 target value and the IDE and RTR target values for filtering the incoming message ID. This register is for equal to, smaller than, greater than comparisons, or as the lower limit value in an ID range detection. The CANFD_FLTR_ID1 register is only written in freeze mode.

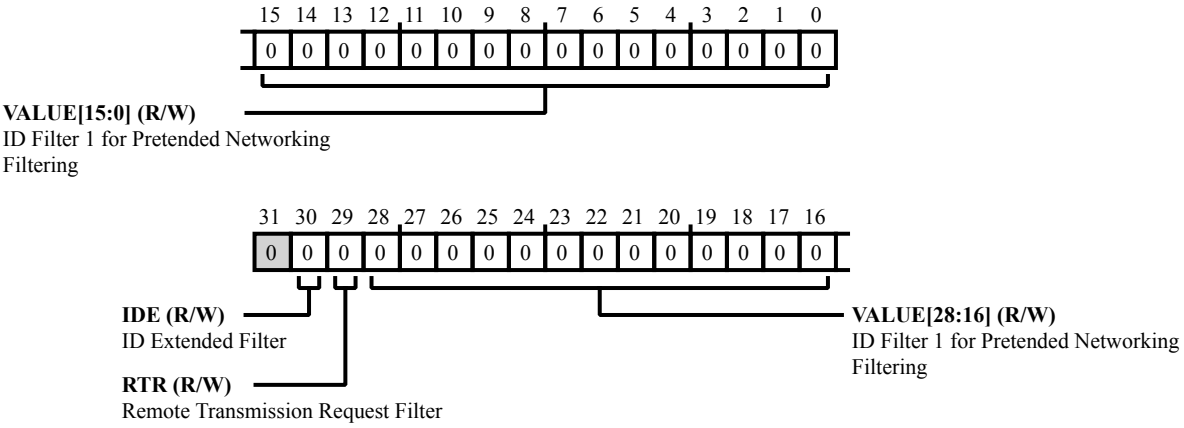


Figure 16-32: CANFD_FLTR_ID1 Register Diagram

Table 16-50: CANFD_FLTR_ID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	IDE	ID Extended Filter. The CANFD_FLTR_ID1.IDE bit identifies if the frame format is standard or extended. It is part of the ID reception filter.	
		0	Accept standard frame format
		1	Accept extended frame format
29 (R/W)	RTR	Remote Transmission Request Filter. The CANFD_FLTR_ID1.RTR bit identifies if the frame is remote. It is part of the ID reception filter.	
		0	Reject remote frame (accept data frame)
		1	Accept remote frame
28:0 (R/W)	VALUE	ID Filter 1 for Pretended Networking Filtering. The CANFD_FLTR_ID1.VALUE bit field defines either the 29 bits of a extended frame format, considering all bits, or the 11 bits of a standard frame format, considering just the 11 leftmost bits.	

Pretended Networking ID Filter2 / IDMask Register

The CANFD_FLTR_ID2_IDMSK register contains the FLT_ID2 target value for use as the upper limit value in ID range detection. When exact ID filtering criteria is selected, the CANFD_FLTR_ID2_IDMSK register is also for storing the ID mask. The CANFD_FLTR_ID2_IDMSK.IDE and CANFD_FLTR_ID2_IDMSK.RTR bits are for ID filtering (exact and range) as part of the ID reception filter.

The CANFD_FLTR_ID2_IDMSK register can only be written in freeze mode.

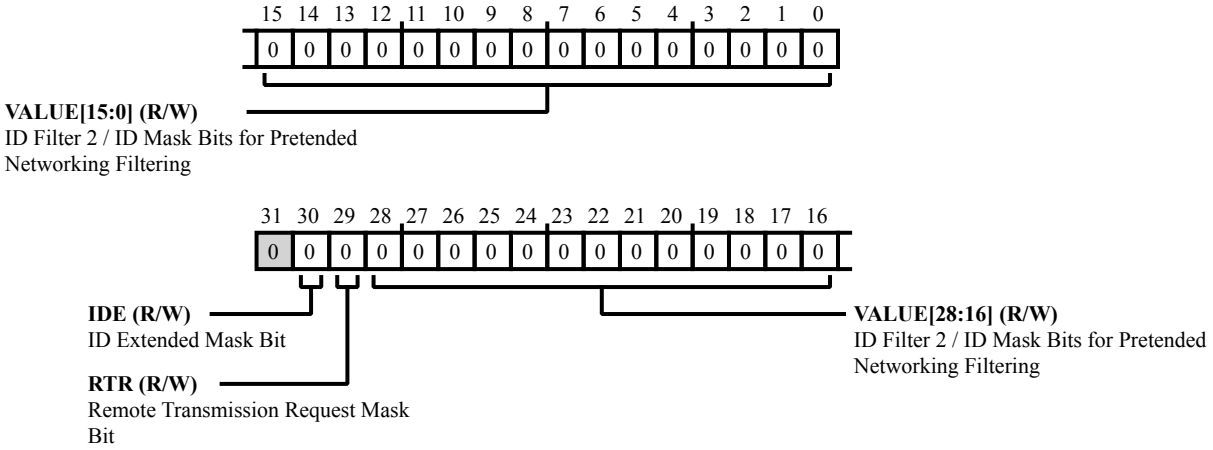


Figure 16-33: CANFD_FLTR_ID2_IDMSK Register Diagram

Table 16-51: CANFD_FLTR_ID2_IDMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	IDE	ID Extended Mask Bit. The CANFD_FLTR_ID2_IDMSK.IDE bit indicates whether the frame format (standard/extended) is used as part of the ID reception filter. If the CANFD_FLTR_ID2_IDMSK.IDE bit is set, the corresponding bit in the filter is checked. If it is cleared, the corresponding bit in the filter is a dont care.	
		0	Don't Care
		1	Checked
29 (R/W)	RTR	Remote Transmission Request Mask Bit. The CANFD_FLTR_ID2_IDMSK.RTR bit indicates if the frame type (data/remote) is part of the ID reception filter. If the CANFD_FLTR_ID2_IDMSK.RTR bit is set, the corresponding bit in the filter is checked. If it is cleared, the corresponding bit in the filter is a dont care.	
		0	Don't Care
		1	Checked

Table 16-51: CANFD_FLTR_ID2_IDMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	VALUE	ID Filter 2 / ID Mask Bits for Pretended Networking Filtering. The `CANFD_FLTR_ID2_IDMSK.VALUE` bit field defines the ID filter value (FLT_ID2). In extended frame format, use all 29 bits (FLT_ID2[28:0]) in the field. In standard frame format, use the upper 11 bits (FLT_ID2[28:18]) and the lower 18 bits (FLT_ID2[17:0]) have no meaning. The `CANFD_FLTR_ID2_IDMSK.VALUE` bit field can also be IDMASK in exact ID filtering to define the mask value. In extended frame format, use all 29 bits (IDMASK[28:0]) in the field. In standard frame format, use the upper 11 bits (IDMASK[28:18]) and the lower 18 bits (IDMASK[17:0]) have no meaning.

Mailbox Interrupt Flag 1 Register

The CANFD_IFLG1 register defines the flags for the 32 message buffer interrupts for MB31 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding bit in the CANFD_IFLG1 register. If the corresponding bit in the CANFD_IMSK1 register is set, an interrupt is generated. The interrupt flag must be cleared by writing a 1 to it. Writing a zero has no effect.

Before updating the CANFD_CFG.MAXMB field, the processor must service the CANFD_IFLG1 bits whose MB value is greater than the CANFD_CFG.MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

There is an exception is when DMA for Rx FIFO is enabled, as described below.

Rx FIFO:

The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the CANFD_CFG.RFEN bit is set and the CANFD_CFG.DMAEN bit is cleared, the function of the 8 least significant interrupt flags changes as follows:

- BUF7I, BUF6I, and BUF5I indicate operating conditions of the FIFO.
- BUF0I is used to empty the FIFO.
- BUF4I to BUF1I are reserved.

Before setting the CANFD_CFG.RFEN bit, the processor must service the IFLAG bits asserted in the Rx FIFO region or these IFLAG bits will mistakenly show the related MBs now belonging to the FIFO as having contents to be serviced. When the CANFD_CFG.RFEN bit is negated, the FIFO flags must be cleared. The same care must be taken when a CANFD_CTL2.RFFNUM value is selected extending Rx FIFO filters beyond MB7. For example, when the CANFD_CTL2.RFFNUM value is 0x8, the MB0–23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

The Rx FIFO must be disabled when the CANFD_CFG.FDEN bit is enabled.

Rx FIFO with DMA:

When both the CANFD_CFG.RFEN bit and the CANFD_CFG.DMAEN bits are set (DMA feature for Rx FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I–BUF0I) are changed to support the DMA operation:

- BUF7I, BUF6I, and BUF4I–BUF1I are not used.
- BUF5I indicates operating condition of FIFO.
- BUF5I does not generate a CPU interrupt, but generates a DMA request.
- BUF0I is used to empty FIFO.

The CANFD_IMSK1 register bits in the Rx FIFO region are not considered when the CANFD_CFG.DMAEN bit is asserted. In addition, the processor must not clear the flag BUF5I when DMA is enabled. Before setting the

CANFD_CFG.DMAEN bit, the processor must service the IFLAGs asserted in the Rx FIFO region. When the CANFD_CFG.DMAEN bit is cleared, the FIFO must be empty.

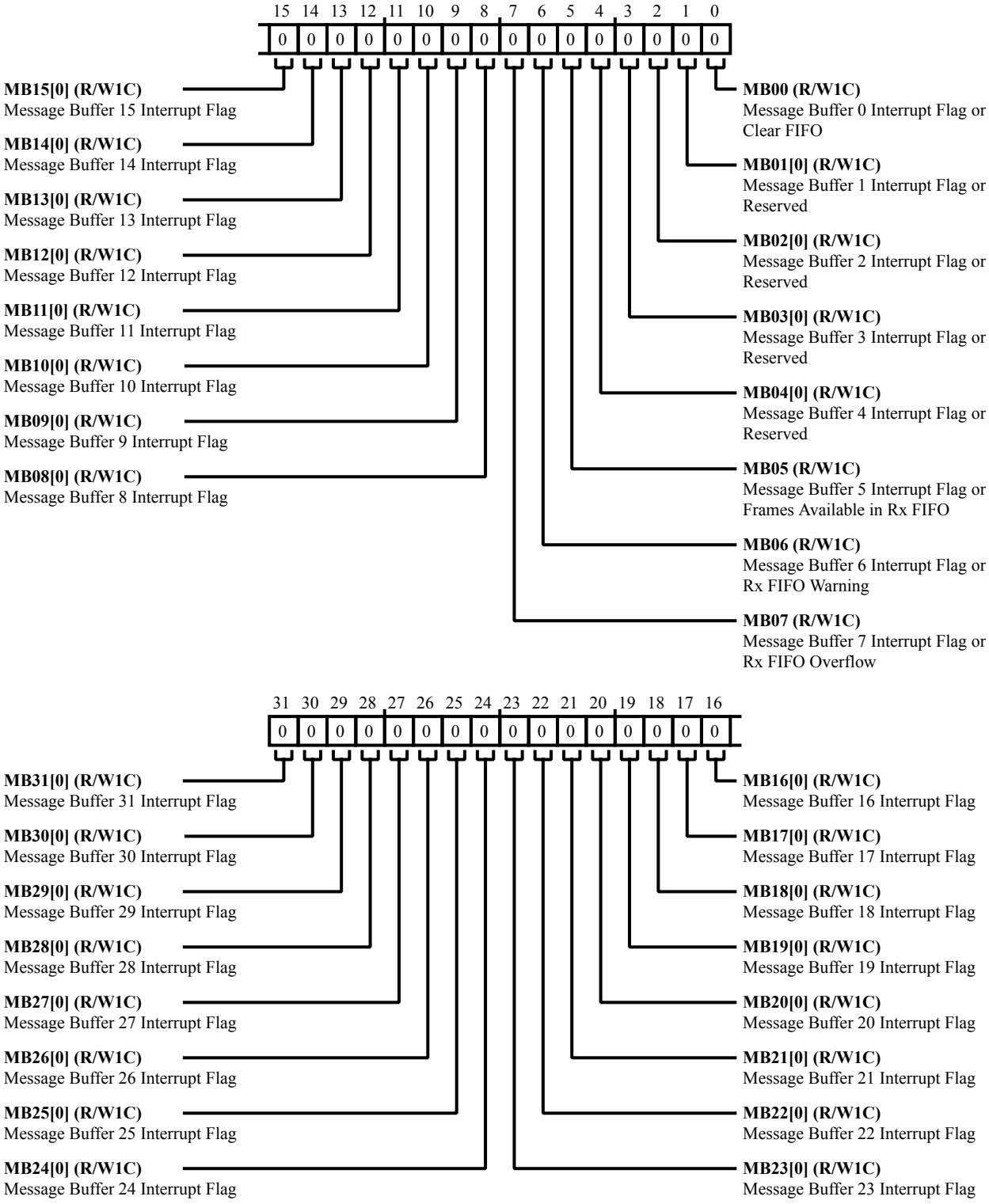


Figure 16-34: CANFD_IFLG1 Register Diagram

Table 16-52: CANFD_IFLG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	MB31	Message Buffer 31 Interrupt Flag. The CANFD_IFLG1.MB31 bit flags the CANFD module message buffer interrupt for MB31. When the CANFD_IFLG1.MB31 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
30 (R/W1C)	MB30	Message Buffer 30 Interrupt Flag. The CANFD_IFLG1.MB30 bit flags the CANFD module message buffer interrupt for MB30. When the CANFD_IFLG1.MB30 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
29 (R/W1C)	MB29	Message Buffer 29 Interrupt Flag. The CANFD_IFLG1.MB29 bit flags the CANFD module message buffer interrupt for MB29. When the CANFD_IFLG1.MB29 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
28 (R/W1C)	MB28	Message Buffer 28 Interrupt Flag. The CANFD_IFLG1.MB28 bit flags the CANFD module message buffer interrupt for MB28 When the CANFD_IFLG1.MB28 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
27 (R/W1C)	MB27	Message Buffer 27 Interrupt Flag. The CANFD_IFLG1.MB27 bit flags the CANFD module message buffer interrupt for MB27. When the CANFD_IFLG1.MB27 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
26 (R/W1C)	MB26	Message Buffer 26 Interrupt Flag. The CANFD_IFLG1.MB26 bit flags the CANFD module message buffer interrupt for MB26. When the CANFD_IFLG1.MB26 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
25 (R/W1C)	MB25	Message Buffer 25 Interrupt Flag. The CANFD_IFLG1.MB25 bit flags the CANFD module message buffer interrupt for MB25. When the CANFD_IFLG1.MB25 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
24 (R/W1C)	MB24	Message Buffer 24 Interrupt Flag. The CANFD_IFLG1.MB24 bit flags the CANFD module message buffer interrupt for MB24. When the CANFD_IFLG1.MB24 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
23 (R/W1C)	MB23	Message Buffer 23 Interrupt Flag. The CANFD_IFLG1.MB23 bit flags the CANFD module message buffer interrupt for MB23. When the CANFD_IFLG1.MB23 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
22 (R/W1C)	MB22	Message Buffer 22 Interrupt Flag. The CANFD_IFLG1.MB22 bit flags the CANFD module message buffer interrupt for MB22. When the CANFD_IFLG1.MB22 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
21 (R/W1C)	MB21	Message Buffer 21 Interrupt Flag. The CANFD_IFLG1.MB21 bit flags the CANFD module message buffer interrupt for MB21. When the CANFD_IFLG1.MB21 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
20 (R/W1C)	MB20	Message Buffer 20 Interrupt Flag. The CANFD_IFLG1.MB20 bit flags the CANFD module message buffer interrupt for MB20. When the CANFD_IFLG1.MB20 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
19 (R/W1C)	MB19	Message Buffer 19 Interrupt Flag. The CANFD_IFLG1.MB19 bit flags the CANFD module message buffer interrupt for MB19. When the CANFD_IFLG1.MB19 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
18 (R/W1C)	MB18	Message Buffer 18 Interrupt Flag. The CANFD_IFLG1.MB18 bit flags the CANFD module message buffer interrupt for MB18. When the CANFD_IFLG1.MB18 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
17 (R/W1C)	MB17	Message Buffer 17 Interrupt Flag. The CANFD_IFLG1.MB17 bit flags the CANFD module message buffer interrupt for MB17. When the CANFD_IFLG1.MB17 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W1C)	MB16	Message Buffer 16 Interrupt Flag. The CANFD_IFLG1.MB16 bit flags the CANFD module message buffer interrupt for MB16. When the CANFD_IFLG1.MB16 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
15 (R/W1C)	MB15	Message Buffer 15 Interrupt Flag. The CANFD_IFLG1.MB15 bit flags the CANFD module message buffer interrupt for MB15. When the CANFD_IFLG1.MB15 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
14 (R/W1C)	MB14	Message Buffer 14 Interrupt Flag. The CANFD_IFLG1.MB14 bit flags the CANFD module message buffer interrupt for MB14. When the CANFD_IFLG1.MB14 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
13 (R/W1C)	MB13	Message Buffer 13 Interrupt Flag. The CANFD_IFLG1.MB13 bit flags the CANFD module message buffer interrupt for MB13. When the CANFD_IFLG1.MB13 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
12 (R/W1C)	MB12	Message Buffer 12 Interrupt Flag. The CANFD_IFLG1.MB12 bit flags the CANFD module message buffer interrupt for MB12. When the CANFD_IFLG1.MB12 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/W1C)	MB11	Message Buffer 11 Interrupt Flag. The `CANFD_IFLG1.MB11` bit flags the CANFD module message buffer interrupt for MB11. When the `CANFD_IFLG1.MB11` bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
10 (R/W1C)	MB10	Message Buffer 10 Interrupt Flag. The `CANFD_IFLG1.MB10` bit flags the CANFD module message buffer interrupt for MB10. When the `CANFD_IFLG1.MB10` bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
9 (R/W1C)	MB09	Message Buffer 9 Interrupt Flag. The `CANFD_IFLG1.MB09` bit flags the CANFD module message buffer interrupt for MB09. When the `CANFD_IFLG1.MB09` bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.
8 (R/W1C)	MB08	Message Buffer 8 Interrupt Flag. The `CANFD_IFLG1.MB08` bit flags the CANFD module message buffer interrupt for MB08. When the `CANFD_IFLG1.MB08` bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16.

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W1C)	MB07	Message Buffer 7 Interrupt Flag or Rx FIFO Overflow. The CANFD_IFLG1.MB07 bit flags the CANFD module message buffer interrupt for MB7 or Rx FIFO overflow. When the CANFD_IFLG1.MB07 bit is set and CANFD_CFG.RFEN is disabled, MB7 completed transmission or reception. When the CANFD_IFLG1.MB07 bit is set and CANFD_CFG.RFEN is enabled, there is an Rx FIFO overflow. When the CANFD_IFLG1.MB07 bit is cleared and CANFD_CFG.RFEN is disabled, the corresponding buffer has no occurrence of successfully completed transmission or reception.When the CANFD_IFLG1.MB07 bit is cleared and CANFD_CFG.RFEN is enabled, there is no occurrence of Rx FIFO overflow. When the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled), the CANFD_IFLG1.MB07 bit flags the interrupt for MB7. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB07 bit represents the Rx FIFO overflow. In this case, the CANFD_IFLG1.MB07 bit indicates that a message was lost because the Rx FIFO is full. Note that the CANFD_IFLG1.MB07 bit is not asserted when the Rx FIFO is full and the message was captured by a mailbox. Note that the CANFD_IFLG1.MB07 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.
6 (R/W1C)	MB06	Message Buffer 6 Interrupt Flag or Rx FIFO Warning. The CANFD_IFLG1.MB06 bit flags the CANFD module message buffer interrupt for MB6 or Rx FIFO warning. When the CANFD_IFLG1.MB06 bit is set and CANFD_CFG.RFEN is disabled, MB6 completed transmission or reception. When the CANFD_IFLG1.MB06 bit is set and CANFD_CFG.RFEN is enabled, the Rx FIFO is almost full. When the CANFD_IFLG1.MB06 bit is cleared and CANFD_CFG.RFEN is disabled, the corresponding buffer has no occurrence of successfully completed transmission or reception.When the CANFD_IFLG1.MB06 bit is cleared and CANFD_CFG.RFEN is enabled, the Rx FIFO is not almost full. When the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled), the CANFD_IFLG1.MB06 bit flags the interrupt for MB6. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB06 bit represents the Rx FIFO warning. In this case, BUF6I indicates when the number of unread messages in the Rx FIFO increases from 4 to 5 due to the reception of a new message, meaning that the Rx FIFO is almost full. Note that if the CANFD_IFLG1.MB06 bit is cleared while the number of unread messages is greater than 4, the CANFD_IFLG1.MB06 bit is not asserted again until the number of unread messages in the Rx FIFO decreases to 4 or less. Note that the CANFD_IFLG1.MB06 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W1C)	MB05	Message Buffer 5 Interrupt Flag or Frames Available in Rx FIFO. The CANFD_IFLG1.MB05 bit flags the CANFD module message buffer interrupt for MB5 or Rx FIFO frames available. When the CANFD_IFLG1.MB05 bit is set and CANFD_CFG.RFEN is disabled, MB5 completed transmission or reception. When the CANFD_IFLG1.MB05 bit is set and CANFD_CFG.RFEN is enabled, the Rx FIFO has frames available. BUF5I generates a DMA request when both the CANFD_CFG.RFEN and CANFD_CFG.DMAEN bits are set. When the CANFD_IFLG1.MB05 bit is cleared and CANFD_CFG.RFEN is disabled, the corresponding buffer has no occurrence of successfully completed transmission or reception. When the CANFD_IFLG1.MB05 bit is cleared and CANFD_CFG.RFEN is enabled, no frames are available in the Rx FIFO. When the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled), CANFD_IFLG1.MB05 bit flags the interrupt for MB5. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB05 bit represents the frames available in the Rx FIFO and indicates that at least one frame is available to be read from the Rx FIFO. When the CANFD_CFG.DMAEN bit is set, the CANFD_IFLG1.MB05 bit generates a DMA request and the processor must not clear the CANFD_IFLG1.MB05 bit by writing one to it. Note that the CANFD_IFLG1.MB05 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.
4 (R/W1C)	MB04	Message Buffer 4 Interrupt Flag or Reserved. The CANFD_IFLG1.MB04 bit flags the interrupt for MB4, when the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled). When the CANFD_IFLG1.MB04 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB04 bit is reserved. Note that the CANFD_IFLG1.MB04 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W1C)	MB03	Message Buffer 3 Interrupt Flag or Reserved. The CANFD_IFLG1.MB03 bit flags the interrupt for MB3, when the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled). When the CANFD_IFLG1.MB03 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB03 bit is reserved. Note that the CANFD_IFLG1.MB03 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.
2 (R/W1C)	MB02	Message Buffer 2 Interrupt Flag or Reserved. The CANFD_IFLG1.MB02 bit flags the interrupt for MB2, when the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled). When the CANFD_IFLG1.MB02 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB02 bit is reserved. Note that the CANFD_IFLG1.MB02 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.
1 (R/W1C)	MB01	Message Buffer 1 Interrupt Flag or Reserved. The CANFD_IFLG1.MB01 bit flags the interrupt for MB1, when the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled). When the CANFD_IFLG1.MB01 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB01 bit is reserved. Note that the CANFD_IFLG1.MB01 bit is cleared by the CANFD module whenever the CANFD_CFG.RFEN bit is changed by a processor write.

Table 16-52: CANFD_IFLG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W1C)	MB00	Message Buffer 0 Interrupt Flag or Clear FIFO. The CANFD_IFLG1.MB00 bit flags the interrupt for MB0, when the CANFD_CFG.RFEN bit is cleared (Rx FIFO disabled). When the CANFD_IFLG1.MB00 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception. When the CANFD_CFG.RFEN bit is set (Rx FIFO enabled), the CANFD_IFLG1.MB00 bit triggers the clear FIFO operation. This operation empties FIFO contents. Before performing this operation, the processor must service all FIFO related IFLAGs. When the CANFD_CFG.DMAEN bit is set (DMA is enabled), this operation also clears the CANFD_IFLG1.MB05 flag and consequently aborts the DMA request. The clear FIFO operation occurs when the processor writes a one to the CANFD_IFLG1.MB00 bit, which is only allowed in freeze mode and is blocked by hardware in other conditions.

Mailbox Interrupt Flag 2 Register

The CANFD_IFLG2 register defines the flags for the 32 message buffer interrupts for MB63 to MB32. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective bit in the CANFD_IFLG2 bit. If the corresponding bit is set, an interrupt is generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating the CANFD_CFG.MAXMB field, the processor must service the CANFD_IFLG2 bits whose MB value is greater than the CANFD_CFG.MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

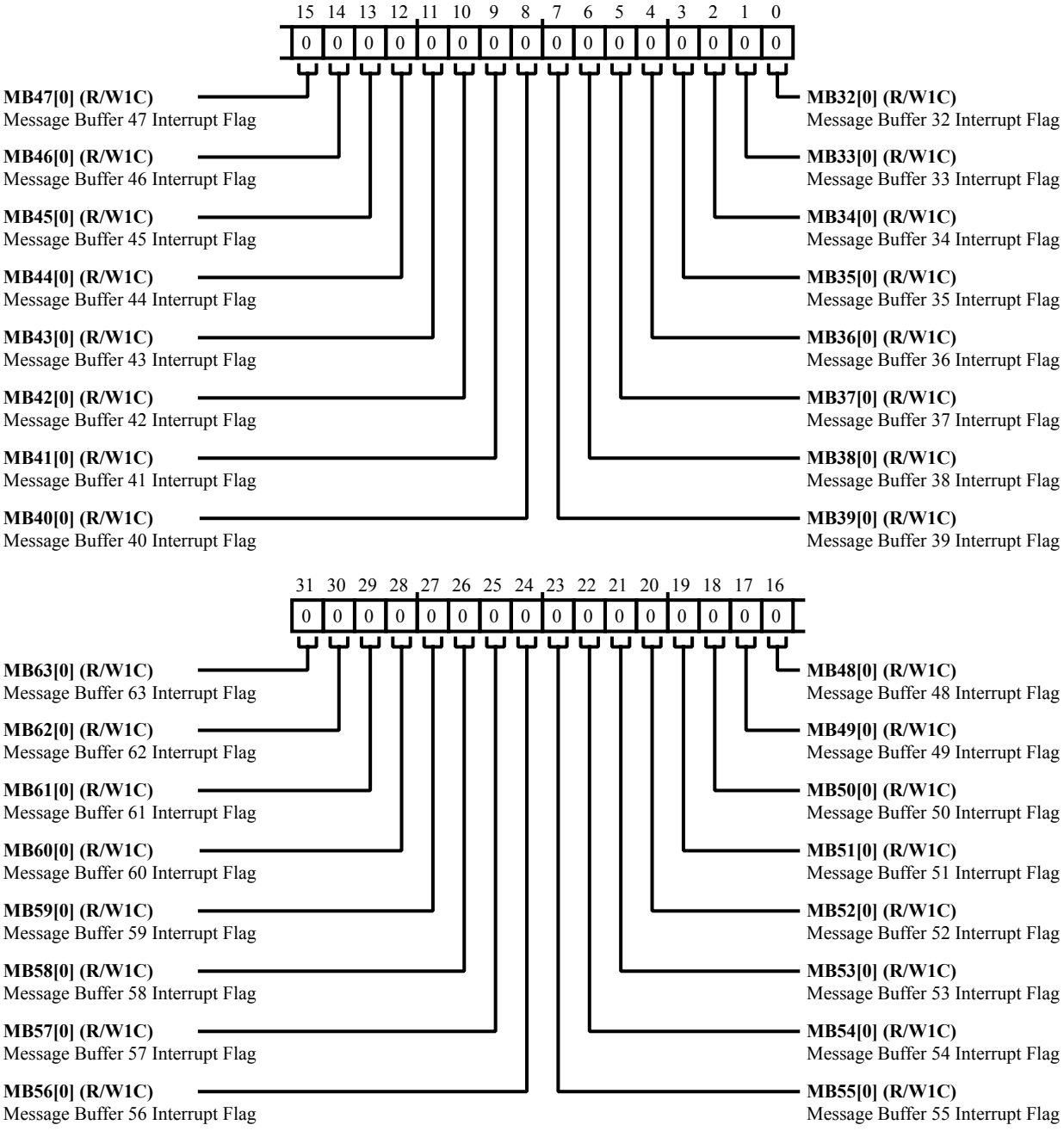


Figure 16-35: CANFD_IFLG2 Register Diagram

Table 16-53: CANFD_IFLG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	MB63	Message Buffer 63 Interrupt Flag. The CANFD_IFLG2.MB63 bit flags the CANFD module message buffer interrupt for MB63. When the CANFD_IFLG2.MB63 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
30 (R/W1C)	MB62	Message Buffer 62 Interrupt Flag. The CANFD_IFLG2.MB62 bit flags the CANFD module message buffer interrupt for MB62. When the CANFD_IFLG2.MB62 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
29 (R/W1C)	MB61	Message Buffer 61 Interrupt Flag. The CANFD_IFLG2.MB61 bit flags the CANFD module message buffer interrupt for MB61. When the CANFD_IFLG2.MB61 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
28 (R/W1C)	MB60	Message Buffer 60 Interrupt Flag. The CANFD_IFLG2.MB60 bit flags the CANFD module message buffer interrupt for MB60. When the CANFD_IFLG2.MB60 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
27 (R/W1C)	MB59	Message Buffer 59 Interrupt Flag. The CANFD_IFLG2.MB59 bit flags the CANFD module message buffer interrupt for MB59. When the CANFD_IFLG2.MB59 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
26 (R/W1C)	MB58	Message Buffer 58 Interrupt Flag. The CANFD_IFLG2.MB58 bit flags the CANFD module message buffer interrupt for MB58. When the CANFD_IFLG2.MB58 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.

Table 16-53: CANFD_IFLG2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
25 (R/W1C)	MB57	Message Buffer 57 Interrupt Flag. The CANFD_IFLG2.MB57 bit flags the CANFD module message buffer interrupt for MB57. When the CANFD_IFLG2.MB57 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
24 (R/W1C)	MB56	Message Buffer 56 Interrupt Flag. The CANFD_IFLG2.MB56 bit flags the CANFD module message buffer interrupt for MB56. When the CANFD_IFLG2.MB56 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
23 (R/W1C)	MB55	Message Buffer 55 Interrupt Flag. The CANFD_IFLG2.MB55 bit flags the CANFD module message buffer interrupt for MB55. When the CANFD_IFLG2.MB55 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
22 (R/W1C)	MB54	Message Buffer 54 Interrupt Flag. The CANFD_IFLG2.MB54 bit flags the CANFD module message buffer interrupt for MB54. When the CANFD_IFLG2.MB54 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
21 (R/W1C)	MB53	Message Buffer 53 Interrupt Flag. The CANFD_IFLG2.MB53 bit flags the CANFD module message buffer interrupt for MB53. When the CANFD_IFLG2.MB53 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
20 (R/W1C)	MB52	Message Buffer 52 Interrupt Flag. The CANFD_IFLG2.MB52 bit flags the CANFD module message buffer interrupt for MB52. When the CANFD_IFLG2.MB52 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.

Table 16-53: CANFD_IFLG2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W1C)	MB51	Message Buffer 51 Interrupt Flag. The CANFD_IFLG2.MB51 bit flags the CANFD module message buffer interrupt for MB51. When the CANFD_IFLG2.MB51 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
18 (R/W1C)	MB50	Message Buffer 50 Interrupt Flag. The CANFD_IFLG2.MB50 bit flags the CANFD module message buffer interrupt for MB50. When the CANFD_IFLG2.MB50 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
17 (R/W1C)	MB49	Message Buffer 49 Interrupt Flag. The CANFD_IFLG2.MB49 bit flags the CANFD module message buffer interrupt for MB49. When the CANFD_IFLG2.MB49 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
16 (R/W1C)	MB48	Message Buffer 48 Interrupt Flag. The CANFD_IFLG2.MB48 bit flags the CANFD module message buffer interrupt for MB48. When the CANFD_IFLG2.MB48 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
15 (R/W1C)	MB47	Message Buffer 47 Interrupt Flag. The CANFD_IFLG2.MB47 bit flags the CANFD module message buffer interrupt for MB47. When the CANFD_IFLG2.MB47 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
14 (R/W1C)	MB46	Message Buffer 46 Interrupt Flag. The CANFD_IFLG2.MB46 bit flags the CANFD module message buffer interrupt for MB46. When the CANFD_IFLG2.MB46 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.

Table 16-53: CANFD_IFLG2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W1C)	MB45	Message Buffer 45 Interrupt Flag. The CANFD_IFLG2.MB45 bit flags the CANFD module message buffer interrupt for MB45. When the CANFD_IFLG2.MB45 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
12 (R/W1C)	MB44	Message Buffer 44 Interrupt Flag. The CANFD_IFLG2.MB44 bit flags the CANFD module message buffer interrupt for MB44. When the CANFD_IFLG2.MB44 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
11 (R/W1C)	MB43	Message Buffer 43 Interrupt Flag. The CANFD_IFLG2.MB43 bit flags the CANFD module message buffer interrupt for MB43. When the CANFD_IFLG2.MB43 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
10 (R/W1C)	MB42	Message Buffer 42 Interrupt Flag. The CANFD_IFLG2.MB42 bit flags the CANFD module message buffer interrupt for MB42. When the CANFD_IFLG2.MB42 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
9 (R/W1C)	MB41	Message Buffer 41 Interrupt Flag. The CANFD_IFLG2.MB41 bit flags the CANFD module message buffer interrupt for MB41. When the CANFD_IFLG2.MB41 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
8 (R/W1C)	MB40	Message Buffer 40 Interrupt Flag. The CANFD_IFLG2.MB40 bit flags the CANFD module message buffer interrupt for MB40. When the CANFD_IFLG2.MB40 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.

Table 16-53: CANFD_IFLG2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W1C)	MB39	Message Buffer 39 Interrupt Flag. The CANFD_IFLG2.MB39 bit flags the CANFD module message buffer interrupt for MB39. When the CANFD_IFLG2.MB39 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
6 (R/W1C)	MB38	Message Buffer 38 Interrupt Flag. The CANFD_IFLG2.MB38 bit flags the CANFD module message buffer interrupt for MB38. When the CANFD_IFLG2.MB38 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
5 (R/W1C)	MB37	Message Buffer 37 Interrupt Flag. The CANFD_IFLG2.MB37 bit flags the CANFD module message buffer interrupt for MB37. When the CANFD_IFLG2.MB37 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
4 (R/W1C)	MB36	Message Buffer 36 Interrupt Flag. The CANFD_IFLG2.MB36 bit flags the CANFD module message buffer interrupt for MB36. When the CANFD_IFLG2.MB36 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
3 (R/W1C)	MB35	Message Buffer 35 Interrupt Flag. The CANFD_IFLG2.MB35 bit flags the CANFD module message buffer interrupt for MB35. When the CANFD_IFLG2.MB35 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
2 (R/W1C)	MB34	Message Buffer 34 Interrupt Flag. The CANFD_IFLG2.MB34 bit flags the CANFD module message buffer interrupt for MB34. When the CANFD_IFLG2.MB34 bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.

Table 16-53: CANFD_IFLG2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W1C)	MB33	Message Buffer 33 Interrupt Flag. The `CANFD_IFLG2.MB33` bit flags the CANFD module message buffer interrupt for MB33. When the `CANFD_IFLG2.MB33` bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.
0 (R/W1C)	MB32	Message Buffer 32 Interrupt Flag. The `CANFD_IFLG2.MB32` bit flags the CANFD module message buffer interrupt for MB32. When the `CANFD_IFLG2.MB32` bit is set, the corresponding buffer successfully completed transmission or reception. When the bit is cleared, the corresponding buffer has no occurrence of successfully completed transmission or reception.

Mailbox Interrupt Mask 1 Register

The CANFD_IMSK1 register allows any number of the 32 message buffer interrupts to be enabled or disabled for MB31 to MB. The CANFD_IMSK1 register contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (as indicated by the corresponding bit in the register).

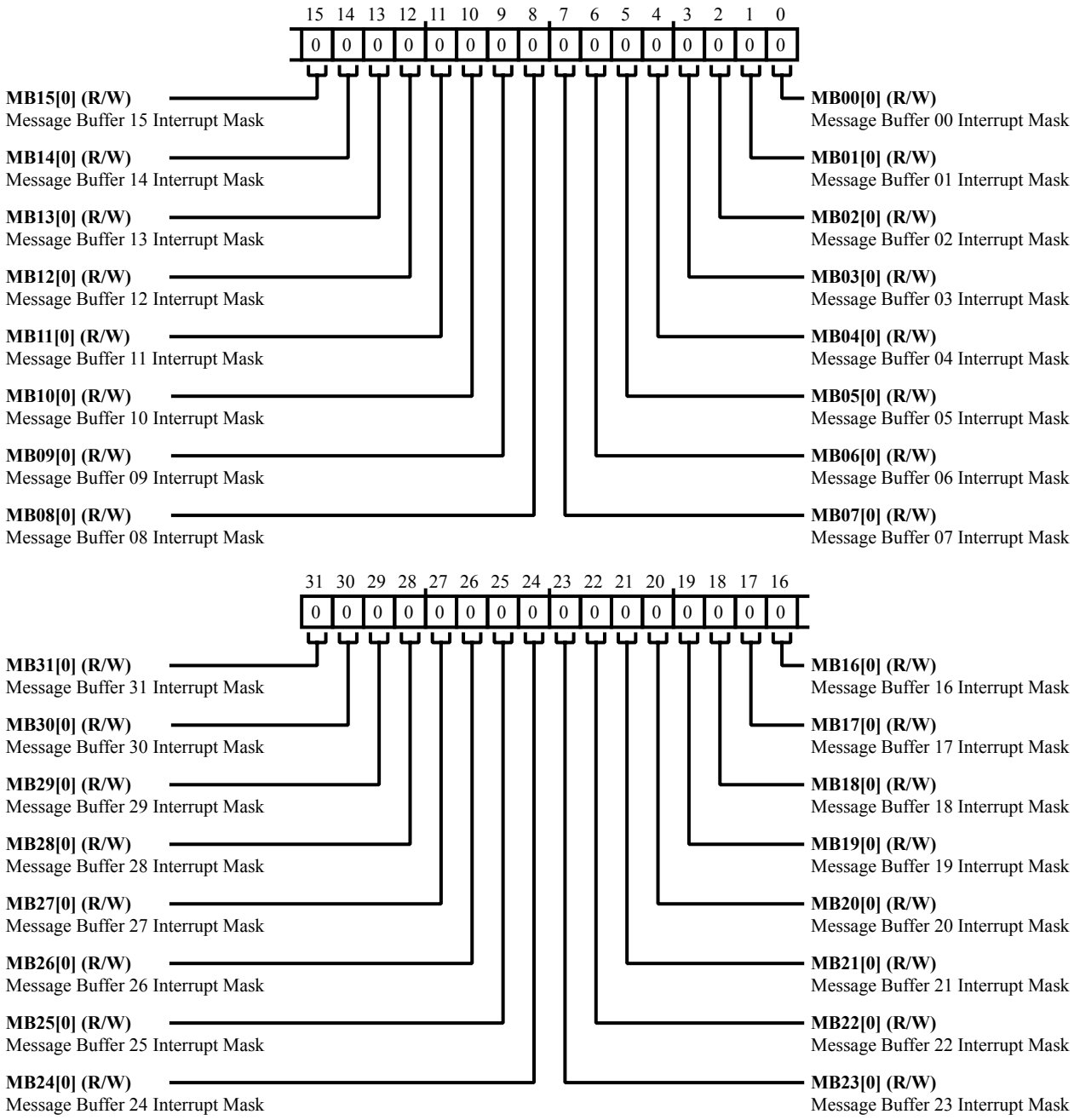


Figure 16-36: CANFD_IMSK1 Register Diagram

Table 16-54: CANFD_IMSK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	MB31	Message Buffer 31 Interrupt Mask. The CANFD_IMSK1.MB31 bit is the CANFD module message buffer interrupt for MB31. When the CANFD_IMSK1.MB31 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
30 (R/W)	MB30	Message Buffer 30 Interrupt Mask. The CANFD_IMSK1.MB30 bit is the CANFD module message buffer interrupt for MB30. When the CANFD_IMSK1.MB30 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
29 (R/W)	MB29	Message Buffer 29 Interrupt Mask. The CANFD_IMSK1.MB29 bit is the CANFD module message buffer interrupt for MB29. When the CANFD_IMSK1.MB29 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
28 (R/W)	MB28	Message Buffer 28 Interrupt Mask. The CANFD_IMSK1.MB28 bit is the CANFD module message buffer interrupt for MB28. When the CANFD_IMSK1.MB28 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
27 (R/W)	MB27	Message Buffer 27 Interrupt Mask. The CANFD_IMSK1.MB27 bit is the CANFD module message buffer interrupt for MB27. When the CANFD_IMSK1.MB27 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.

Table 16-54: CANFD_IMSK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
26 (R/W)	MB26	Message Buffer 26 Interrupt Mask. The CANFD_IMSK1.MB26 bit is the CANFD module message buffer interrupt for MB26. When the CANFD_IMSK1.MB26 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
25 (R/W)	MB25	Message Buffer 25 Interrupt Mask. The CANFD_IMSK1.MB25 bit is the CANFD module message buffer interrupt for MB25. When the CANFD_IMSK1.MB25 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
24 (R/W)	MB24	Message Buffer 24 Interrupt Mask. The CANFD_IMSK1.MB24 bit is the CANFD module message buffer interrupt for MB24. When the CANFD_IMSK1.MB24 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
23 (R/W)	MB23	Message Buffer 23 Interrupt Mask. The CANFD_IMSK1.MB23 bit is the CANFD module message buffer interrupt for MB23. When the CANFD_IMSK1.MB23 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
22 (R/W)	MB22	Message Buffer 22 Interrupt Mask. The CANFD_IMSK1.MB22 bit is the CANFD module message buffer interrupt for MB22. When the CANFD_IMSK1.MB22 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.

Table 16-54: CANFD_IMSK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
21 (R/W)	MB21	Message Buffer 21 Interrupt Mask. The CANFD_IMSK1.MB21 bit is the CANFD module message buffer interrupt for MB21. When the CANFD_IMSK1.MB21 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
20 (R/W)	MB20	Message Buffer 20 Interrupt Mask. The CANFD_IMSK1.MB20 bit is the CANFD module message buffer interrupt for MB20. When the CANFD_IMSK1.MB20 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
19 (R/W)	MB19	Message Buffer 19 Interrupt Mask. The CANFD_IMSK1.MB19 bit is the CANFD module message buffer interrupt for MB19. When the CANFD_IMSK1.MB19 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
18 (R/W)	MB18	Message Buffer 18 Interrupt Mask. The CANFD_IMSK1.MB18 bit is the CANFD module message buffer interrupt for MB18. When the CANFD_IMSK1.MB18 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
17 (R/W)	MB17	Message Buffer 17 Interrupt Mask. The CANFD_IMSK1.MB17 bit is the CANFD module message buffer interrupt for MB17. When the CANFD_IMSK1.MB17 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.

Table 16-54: CANFD_IMSK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	MB16	Message Buffer 16 Interrupt Mask. The CANFD_IMSK1.MB16 bit is the CANFD module message buffer interrupt for MB16. When the CANFD_IMSK1.MB16 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
15 (R/W)	MB15	Message Buffer 15 Interrupt Mask. The CANFD_IMSK1.MB15 bit is the CANFD module message buffer interrupt for MB15. When the CANFD_IMSK1.MB15 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
14 (R/W)	MB14	Message Buffer 14 Interrupt Mask. The CANFD_IMSK1.MB14 bit is the CANFD module message buffer interrupt for MB14. When the CANFD_IMSK1.MB14 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
13 (R/W)	MB13	Message Buffer 13 Interrupt Mask. The CANFD_IMSK1.MB13 bit is the CANFD module message buffer interrupt for MB13. When the CANFD_IMSK1.MB13 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
12 (R/W)	MB12	Message Buffer 12 Interrupt Mask. The CANFD_IMSK1.MB12 bit is the CANFD module message buffer interrupt for MB12. When the CANFD_IMSK1.MB12 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.

Table 16-54: CANFD_IMSK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/W)	MB11	Message Buffer 11 Interrupt Mask. The CANFD_IMSK1.MB11 bit is the CANFD module message buffer interrupt for MB11. When the CANFD_IMSK1.MB11 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
10 (R/W)	MB10	Message Buffer 10 Interrupt Mask. The CANFD_IMSK1.MB10 bit is the CANFD module message buffer interrupt for MB10. When the CANFD_IMSK1.MB10 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
9 (R/W)	MB09	Message Buffer 09 Interrupt Mask. The CANFD_IMSK1.MB09 bit is the CANFD module message buffer interrupt for MB09. When the CANFD_IMSK1.MB09 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
8 (R/W)	MB08	Message Buffer 08 Interrupt Mask. The CANFD_IMSK1.MB08 bit is the CANFD module message buffer interrupt for MB08. When the CANFD_IMSK1.MB08 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
7 (R/W)	MB07	Message Buffer 07 Interrupt Mask. The CANFD_IMSK1.MB07 bit is the CANFD module message buffer interrupt for MB07. When the CANFD_IMSK1.MB07 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.

Table 16-54: CANFD_IMSK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/W)	MB06	Message Buffer 06 Interrupt Mask. The CANFD_IMSK1.MB06 bit is the CANFD module message buffer interrupt for MB06. When the CANFD_IMSK1.MB06 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
5 (R/W)	MB05	Message Buffer 05 Interrupt Mask. The CANFD_IMSK1.MB05 bit is the CANFD module message buffer interrupt for MB05. When the CANFD_IMSK1.MB05 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
4 (R/W)	MB04	Message Buffer 04 Interrupt Mask. The CANFD_IMSK1.MB04 bit is the CANFD module message buffer interrupt for MB04. When the CANFD_IMSK1.MB04 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
3 (R/W)	MB03	Message Buffer 03 Interrupt Mask. The CANFD_IMSK1.MB03 bit is the CANFD module message buffer interrupt for MB03. When the CANFD_IMSK1.MB03 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.
2 (R/W)	MB02	Message Buffer 02 Interrupt Mask. The CANFD_IMSK1.MB02 bit is the CANFD module message buffer interrupt for MB02. When the CANFD_IMSK1.MB02 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK1 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG1 register is set.

Table 16-54: CANFD_IMSK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	MB01	Message Buffer 01 Interrupt Mask. The `CANFD_IMSK1.MB01` bit is the CANFD module message buffer interrupt for MB01. When the `CANFD_IMSK1.MB01` bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the `CANFD_IMSK1` register enables or disables an interrupt request if the corresponding bit in the `CANFD_IFLG1` register is set.
0 (R/W)	MB00	Message Buffer 00 Interrupt Mask. The `CANFD_IMSK1.MB00` bit is the CANFD module message buffer interrupt for MB00. When the `CANFD_IMSK1.MB00` bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the `CANFD_IMSK1` register enables or disables an interrupt request if the corresponding bit in the `CANFD_IFLG1` register is set.

Mailbox Interrupt Mask 2 Register

The CANFD_IMSK2 register allows any number of the 32 message buffer interrupts to be enabled or disabled for MB63 to MB32. The CANFD_IMSK2 register contains one interrupt mask bit per buffer, enabling the processor to determine which buffer generates an interrupt after a successful transmission or reception (as indicated by the corresponding bit in the register).

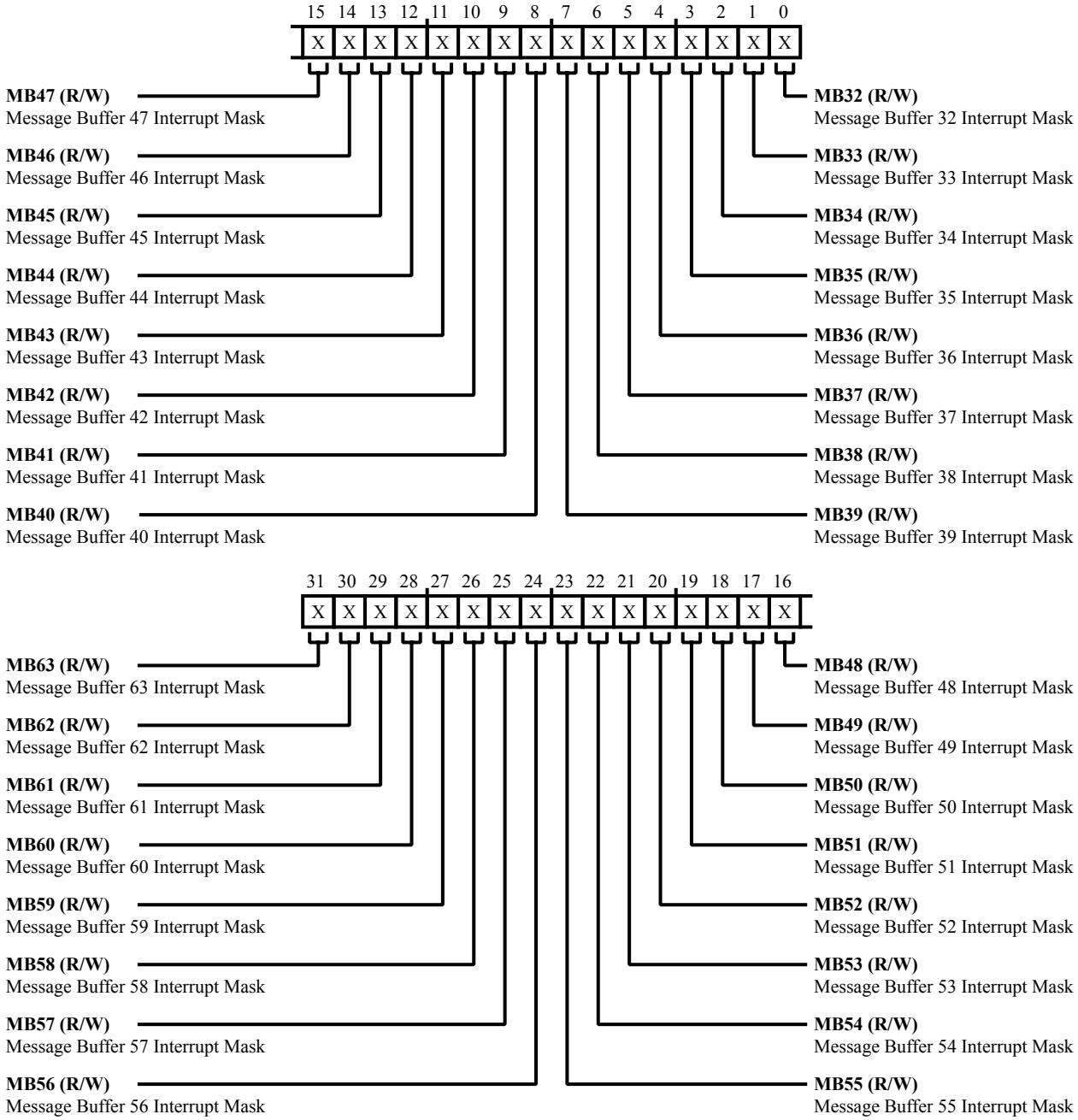


Figure 16-37: CANFD_IMSK2 Register Diagram

Table 16-55: CANFD_IMSK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	MB63	Message Buffer 63 Interrupt Mask. The CANFD_IMSK2.MB63 bit is the CANFD module message buffer interrupt for MB63. When the CANFD_IMSK2.MB63 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
30 (R/W)	MB62	Message Buffer 62 Interrupt Mask. The CANFD_IMSK2.MB62 bit is the CANFD module message buffer interrupt for MB62. When the CANFD_IMSK2.MB62 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
29 (R/W)	MB61	Message Buffer 61 Interrupt Mask. The CANFD_IMSK2.MB61 bit is the CANFD module message buffer interrupt for MB61. When the CANFD_IMSK2.MB61 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
28 (R/W)	MB60	Message Buffer 60 Interrupt Mask. The CANFD_IMSK2.MB60 bit is the CANFD module message buffer interrupt for MB60. When the CANFD_IMSK2.MB60 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
27 (R/W)	MB59	Message Buffer 59 Interrupt Mask. The CANFD_IMSK2.MB59 bit is the CANFD module message buffer interrupt for MB59. When the CANFD_IMSK2.MB59 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.

Table 16-55: CANFD_IMSK2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
26 (R/W)	MB58	Message Buffer 58 Interrupt Mask. The CANFD_IMSK2.MB58 bit is the CANFD module message buffer interrupt for MB58. When the CANFD_IMSK2.MB58 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
25 (R/W)	MB57	Message Buffer 57 Interrupt Mask. The CANFD_IMSK2.MB57 bit is the CANFD module message buffer interrupt for MB57. When the CANFD_IMSK2.MB57 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
24 (R/W)	MB56	Message Buffer 56 Interrupt Mask. The CANFD_IMSK2.MB56 bit is the CANFD module message buffer interrupt for MB56. When the CANFD_IMSK2.MB56 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
23 (R/W)	MB55	Message Buffer 55 Interrupt Mask. The CANFD_IMSK2.MB55 bit is the CANFD module message buffer interrupt for MB55. When the CANFD_IMSK2.MB55 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
22 (R/W)	MB54	Message Buffer 54 Interrupt Mask. The CANFD_IMSK2.MB54 bit is the CANFD module message buffer interrupt for MB54. When the CANFD_IMSK2.MB54 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.

Table 16-55: CANFD_IMSK2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
21 (R/W)	MB53	Message Buffer 53 Interrupt Mask. The CANFD_IMSK2.MB53 bit is the CANFD module message buffer interrupt for MB53. When the CANFD_IMSK2.MB53 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
20 (R/W)	MB52	Message Buffer 52 Interrupt Mask. The CANFD_IMSK2.MB52 bit is the CANFD module message buffer interrupt for MB52. When the CANFD_IMSK2.MB52 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
19 (R/W)	MB51	Message Buffer 51 Interrupt Mask. The CANFD_IMSK2.MB51 bit is the CANFD module message buffer interrupt for MB51. When the CANFD_IMSK2.MB51 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
18 (R/W)	MB50	Message Buffer 50 Interrupt Mask. The CANFD_IMSK2.MB50 bit is the CANFD module message buffer interrupt for MB50. When the CANFD_IMSK2.MB50 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
17 (R/W)	MB49	Message Buffer 49 Interrupt Mask. The CANFD_IMSK2.MB49 bit is the CANFD module message buffer interrupt for MB49. When the CANFD_IMSK2.MB49 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.

Table 16-55: CANFD_IMSK2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	MB48	Message Buffer 48 Interrupt Mask. The CANFD_IMSK2.MB48 bit is the CANFD module message buffer interrupt for MB48. When the CANFD_IMSK2.MB48 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
15 (R/W)	MB47	Message Buffer 47 Interrupt Mask. The CANFD_IMSK2.MB47 bit is the CANFD module message buffer interrupt for MB47. When the CANFD_IMSK2.MB47 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
14 (R/W)	MB46	Message Buffer 46 Interrupt Mask. The CANFD_IMSK2.MB46 bit is the CANFD module message buffer interrupt for MB46. When the CANFD_IMSK2.MB46 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
13 (R/W)	MB45	Message Buffer 45 Interrupt Mask. The CANFD_IMSK2.MB45 bit is the CANFD module message buffer interrupt for MB45. When the CANFD_IMSK2.MB45 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
12 (R/W)	MB44	Message Buffer 44 Interrupt Mask. The CANFD_IMSK2.MB44 bit is the CANFD module message buffer interrupt for MB44. When the CANFD_IMSK2.MB44 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.

Table 16-55: CANFD_IMSK2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/W)	MB43	Message Buffer 43 Interrupt Mask. The CANFD_IMSK2.MB43 bit is the CANFD module message buffer interrupt for MB43. When the CANFD_IMSK2.MB43 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
10 (R/W)	MB42	Message Buffer 42 Interrupt Mask. The CANFD_IMSK2.MB42 bit is the CANFD module message buffer interrupt for MB42. When the CANFD_IMSK2.MB42 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
9 (R/W)	MB41	Message Buffer 41 Interrupt Mask. The CANFD_IMSK2.MB41 bit is the CANFD module message buffer interrupt for MB41. When the CANFD_IMSK2.MB41 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
8 (R/W)	MB40	Message Buffer 40 Interrupt Mask. The CANFD_IMSK2.MB40 bit is the CANFD module message buffer interrupt for MB40. When the CANFD_IMSK2.MB40 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
7 (R/W)	MB39	Message Buffer 39 Interrupt Mask. The CANFD_IMSK2.MB39 bit is the CANFD module message buffer interrupt for MB39. When the CANFD_IMSK2.MB39 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.

Table 16-55: CANFD_IMSK2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/W)	MB38	Message Buffer 38 Interrupt Mask. The CANFD_IMSK2.MB38 bit is the CANFD module message buffer interrupt for MB39. When the CANFD_IMSK2.MB38 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
5 (R/W)	MB37	Message Buffer 37 Interrupt Mask. The CANFD_IMSK2.MB37 bit is the CANFD module message buffer interrupt for MB37. When the CANFD_IMSK2.MB37 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
4 (R/W)	MB36	Message Buffer 36 Interrupt Mask. The CANFD_IMSK2.MB36 bit is the CANFD module message buffer interrupt for MB36. When the CANFD_IMSK2.MB36 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
3 (R/W)	MB35	Message Buffer 35 Interrupt Mask. The CANFD_IMSK2.MB35 bit is the CANFD module message buffer interrupt for MB35. When the CANFD_IMSK2.MB35 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
2 (R/W)	MB34	Message Buffer 34 Interrupt Mask. The CANFD_IMSK2.MB34 bit is the CANFD module message buffer interrupt for MB34. When the CANFD_IMSK2.MB34 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.

Table 16-55: CANFD_IMSK2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	MB33	Message Buffer 33 Interrupt Mask. The CANFD_IMSK2.MB33 bit is the CANFD module message buffer interrupt for MB33. When the CANFD_IMSK2.MB33 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.
0 (R/W)	MB32	Message Buffer 32 Interrupt Mask. The CANFD_IMSK2.MB32 bit is the CANFD module message buffer interrupt for MB32. When the CANFD_IMSK2.MB32 bit is set, the corresponding buffer interrupt is enabled. When the bit is cleared, the corresponding buffer interrupt is disabled. Note that setting or clearing a bit in the CANFD_IMSK2 register enables or disables an interrupt request if the corresponding bit in the CANFD_IFLG2 register is set.

Module Configuration Register

The CANFD_CFG register configures the operation modes and settings for the CANFD module. This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

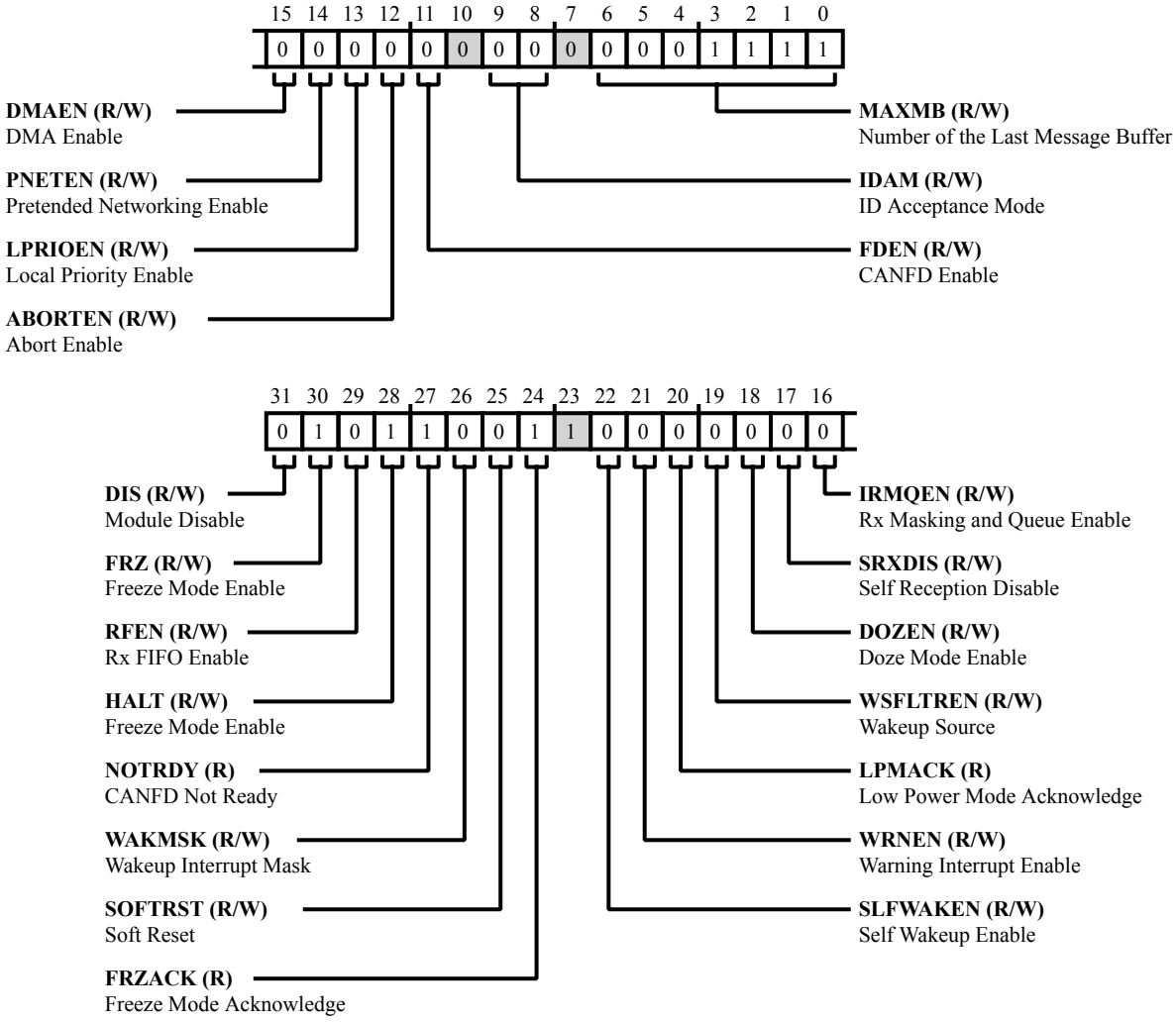


Figure 16-38: CANFD_CFG Register Diagram

Table 16-56: CANFD_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	DIS	Module Disable. The CANFD_CFG.DIS bit disables or enables the CANFD module. When the CANFD_CFG.DIS bit is set, the CANFD module disables the clocks to the PE and CHI submodules. This bit is not affected by a soft reset.	
		0	Enable
		1	Disable
30 (R/W)	FRZ	Freeze Mode Enable. The CANFD_CFG.FRZ bit specifies the CANFD module behavior when the CANFD_CFG.HALT bit is set or when there is a debug mode request at the chip level. When the CANFD_CFG.FRZ bit is set, the CANFD module enters freeze mode. Negation of the CANFD_CFG.FRZ bit causes the CANFD module to exit from freeze mode. The CANFD_CFG.FRZ bit is set by hardware when a non-correctable error is detected and the CANFD_MEC.NCERRFRZEN bit is set.	
		0	Disable
		1	Enable
29 (R/W)	RFEN	Rx FIFO Enable. The CANFD_CFG.RFEN bit enables the Rx FIFO. When the CANFD_CFG.RFEN bit is set, do not use MBs 05 for normal reception and transmission. The corresponding memory region (0x800xDC) is used by the FIFO engine and additional MBs (up to 32, depending on the CANFD_CTL2.RFFNUM bit setting) are used as Rx FIFO ID filter table elements. The CANFD_CFG.RFEN bit also impacts the minimum number of peripheral clocks per CAN bit. The CANFD_CFG.RFEN bit can only be written in freeze mode because it is blocked by hardware in other modes. Note that the CANFD_CFG.RFEN bit cannot be set when the CANFD_CFG.FDEN bit is set.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	HALT	Freeze Mode Enable. Setting the CANFD_CFG.HALT bit puts the CANFD module into freeze mode. The processor clears the CANFD_CFG.HALT bit after initializing the message buffers, the CANFD_CTL1 register, and the CANFD_CTL2 register. The CANFD module does not receive or transmit data before the CANFD_CFG.HALT bit is cleared. The CANFD module cannot enter freeze mode while it is in a low-power mode. The CANFD_CFG.HALT bit is set by hardware when a non-correctable error is detected and the CANFD_MEC.NCERRFRZEN bit is set.	
		0	Disable
		1	Enable
27 (R/NW)	NOTRDY	CANFD Not Ready. The CANFD_CFG.NOTRDY bit is a read-only bit that indicates the CANFD module is either in module disable, doze, stop, or freeze mode. The CANFD_CFG.NOTRDY bit is negated once the CANFD module exits these modes. Soft reset does not affect the CANFD_CFG.NOTRDY bit.	
		0	Normal, Listen-Only, or Loop-Back Mode
		1	Module Disable, Doze, Stop, or Freeze Mode
26 (R/W)	WAKMSK	Wakeup Interrupt Mask. The CANFD_CFG.WAKMSK bit enables the wakeup interrupt using the self wakeup mechanism.	
		0	Disable
		1	Enable
25 (R/W)	SOFTRST	Soft Reset. When the CANFD_CFG.SOFTRST bit is set, the internal state machines and some of the memory mapped registers of the CANFD module reset. For the processor to directly assert a soft reset, write to the CANFD_CFG.SOFTRST bit. A soft reset can also occur when a global soft reset is requested at the chip level. Soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, so it takes time to fully propagate the effect. The CANFD_CFG.SOFTRST bit remains set while reset is pending and is automatically negated when the reset completes. Poll the CANFD_CFG.SOFTRST bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in a low-power mode. Remove the CANFD module from low-power mode before applying soft reset. A soft reset does not affect the CANFD_CFG.SOFTRST bit.	

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/NW)	FRZACK	Freeze Mode Acknowledge. The CANFD_CFG.FRZACK bit is a read-only bit that indicates the CANFD module is in freeze mode and the prescaler is stopped. A freeze mode request is only granted after the current transmission or reception processes completes. Poll the CANFD_CFG.FRZACK bit in software to know when the CANFD module enters freeze mode. If a freeze mode request is negated, the CANFD_CFG.FRZACK bit is negated after the CANFD module prescaler is running again. If freeze mode is requested while the CANFD module is in a low-power mode, the CANFD_CFG.FRZACK bit sets only when the module exits the low-power mode. Soft reset does not affect the CANFD_CFG.FRZACK bit. Note that the CANFD_CFG.FRZACK bit is set within 178 CAN bits of the freeze mode request. It is negated within 2 CAN bits after a freeze mode request removal.	
		0	No Freeze Mode Prescaler is running.
		1	Freeze Mode The prescaler stopped.
22 (R/W)	SLFWAKEN	Self Wakeup Enable. The CANFD_CFG.SLFWAKEN bit enables the self wakeup feature when the CANFD module is in a low-power mode other than module disable. When the self wakeup feature is enabled, the CANFD module monitors the bus for a wakeup event (a recessive-to-dominant transition). If a wake up event is detected during doze mode, the CANFD module requests resumption of the clocks and when the CANFD_CFG.SLFWAKEN bit is enabled generates a wakeup interrupt to the processor. If a wakeup event is detected during stop mode, the CANFD module, when the CANFD_CFG.SLFWAKEN bit is enabled, generates a wake up interrupt to the processor so that it can exit stop mode globally and the CANFD module can request to resume the clocks. When the CANFD module is in a low-power mode other than module disable, the CANFD_CFG.SLFWAKEN bit cannot be written as it is blocked by hardware. Disable self wakeup when pretended networking mode is set.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	WRNEN	Warning Interrupt Enable. The CANFD_CFG.WRNEN bit enables the generation of the TWRNINT and RWRNINT flags in the CANFD_ESR1 register. If the CANFD_CFG.WRNEN bit is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters. No warning interrupt is generated. The CANFD_CFG.WRNEN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable
20 (R/NW)	LPMACK	Low Power Mode Acknowledge. The CANFD_CFG.LPMACK bit is a read-only bit that indicates that the CANFD module is in a low-power mode (module disable, doze, or stop). The module can only enter a low-power mode when all current transmission or reception processes have finished. Poll the CANFD_CFG.LPMACK bit to know when the CANFD module is in low-power mode. The CANFD_CFG.LPMACK bit is not affected by soft reset. Note that the CANFD_CFG.LPMACK bit is set within 180 CAN bits of the low-power mode request by the processor. It is negated within 2 CAN bits after removing the low-power mode request. When the CANFD module is in pretended networking mode, the CANFD_CFG.LPMACK bit is negated within 180 CAN bits from the removal of the low-power mode request.	
		0	Disable Not in low-power mode
		1	Enable In low-power mode
19 (R/W)	WSFLTREN	Wakeup Source. The CANFD_CFG.WSFLTREN bit controls whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wakeup. When enabled, the CANFD module uses the filtered Rx input to detect recessive-to-dominant edges on the CAN bus. When disabled the CANFD module uses the unfiltered Rx input to detect recessive-to-dominant edges on the CAN bus. The CANFD_CFG.WSFLTREN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	DOZEN	Doze Mode Enable. The CANFD_CFG.DOZEN bit controls whether the CANFD module can enter low-power mode when doze mode is requested at the chip level. When the CANFD_CFG.DOZEN bit is set the CANFD module can enter low-power mode and when it is cleared the module cannot. The CANFD_CFG.DOZEN bit is automatically reset when the CANFD module wakes up from doze mode upon detecting activity on the CAN bus (CANFD_CFG.SLFWAKEN enabled).	
		0	Disable
		1	Enable
17 (R/W)	SRXDIS	Self Reception Disable. The CANFD_CFG.SRXDIS bit controls whether the CANFD module can receive frames transmitted from itself. When the CANFD_CFG.SRXDIS bit is set, frames transmitted by the CANFD module are not stored in any MB, regardless of whether the MB is programmed with an ID that matches the transmitted frame. No interrupt flag or interrupt signal is generated due to the frame reception. The CANFD_CFG.SRXDIS bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Enable
		1	Disable
16 (R/W)	IRMQEN	Rx Masking and Queue Enable. The CANFD_CFG.IRMQEN bit controls whether the Rx matching process is based on individual masking and queue or on a masking scheme with the CANFD_RX_MB_GMSK, CANFD_RX_14_MSK, CANFD_RX_15_MSK, and CANFD_RX_FIFO_GMSK registers. The CANFD_CFG.IRMQEN bit can only be written in freeze mode and is blocked by hardware in other modes. For backward compatibility, when CANFD_CFG.IRMQEN is disabled, reading a C/S word locks the MB even when it is empty.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	DMAEN	DMA Enable. The CANFD_CFG.DMAEN bit enables or disables the DMA feature. The DMA feature is only for use with the Rx FIFO and the CANFD_CFG.RFEN bit must be set. When both the CANFD_CFG.DMAEN and CANFD_CFG.RFEN bits are set, CANFD_IFLG1.MB05 generates the DMA request and no Rx FIFO interrupt is generated. The CANFD_CFG.DMAEN bit can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable
14 (R/W)	PNETEN	Pretended Networking Enable. The CANFD_CFG.PNETEN bit enables the pretended networking mode. When pretended networking is enabled, the PE submodule is kept operational in doze mode and stop mode. It is able to process Rx message filtering as defined by the pretended networking configuration registers. The CANFD_CFG.PNETEN bit can be only be written in freeze mode.	
		0	Disable
		1	Enable
13 (R/W)	LPRIOEN	Local Priority Enable. The CANFD_CFG.LPRIOEN bit controls whether the local priority feature is enabled and is provided for backward compatibility with legacy applications. Local priority expands the arbitration ID. With this expanded ID concept, the arbitration process is based on the full 32-bit word, but the actual transmitted ID still has 11 bits for standard frames and 29 bits for extended frames. The CANFD_CFG.LPRIOEN bit can only be written only in freeze mode and is blocked by hardware in other modes.	
		0	Disable
		1	Enable

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	ABORTEN	Abort Enable. The CANFD_CFG.ABORTEN bit enables the Tx abort mechanism. The Tx abort mechanism ensures a safe procedure for aborting a pending transmission, so that no frame is sent on the CAN bus without notification. The CANFD_CFG.ABORTEN bit can only be written only in freeze mode and is blocked by hardware in other modes. Note that when the CANFD_CFG.ABORTEN bit is set, the abort mechanism must only be used for updating mailboxes configured for transmission. Writing the abort code into Rx mailboxes can cause unpredictable results when the CANFD_CFG.ABORTEN bit is enabled.	
		0	Disable
		1	Enable
11 (R/W)	FDEN	CANFD Enable. The CANFD_CFG.FDEN bit enables CAN with Flexible Data rate (CAN FD) operation. When the CANFD_CFG.FDEN bit is enabled, the CANFD module can receive and transmit messages in both CAN FD and CAN 2.0 formats. When it is disabled, the CANFD module can only receive and transmit messages in CAN 2.0 format. The CANFD_CFG.FDEN bit can only be written in freeze mode only. Note that the CANFD_CFG.RFEN bit cannot be set if the CANFD_CFG.FDEN bit is set.	
		0	Disable
		1	Enable
9:8 (R/W)	IDAM	ID Acceptance Mode. The CANFD_CFG.IDAM bit field identifies the format of the Rx FIFO ID Filter Table elements (ID Acceptance Mode). All elements of the table are configured at the same time by the CANFD_CFG.IDAM bit field (they are all the same format). The CANFD_CFG.IDAM bit field can only be written in freeze mode and is blocked by hardware in other modes.	
		0	Format A. There is one full ID (standard and extended) per ID filter table element.
		1	Format B. There are two full standard IDs or two partial 14-bit (standard and extended) IDs per ID filter table element.
		2	Format C. There are four partial 8-bit standard IDs per ID filter table element.
		3	Format D. All frames are rejected.

Table 16-56: CANFD_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	MAXMB	Number of the Last Message Buffer. The CANFD_CFG.MAXMB bit field defines the number of the last message buffer that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 message buffer configuration. The CANFD_CFG.MAXMB bit field can only be written in freeze mode and is blocked by hardware in other modes. Note that the CANFD_CFG.MAXMB bit field must be programmed with a value smaller than or equal to the number of available message buffers. Additionally, the definition of the CANFD_CFG.MAXMB bit value must take into account the region of MBs occupied by Rx FIFO and the ID filters table space defined by the CANFD_CTL2.RFFNUM bit. The CANFD_CFG.MAXMB bit field impacts the minimum number of peripheral clocks per CAN bit.

Memory Error Control Register

The CANFD_MEC register contains control bits for memory error detection and correction (ECC).

Note that when the CANFD_CTL2.ECRWREN bit is disabled, writes to the CANFD_MEC register are blocked, with the exception of the CANFD_MEC.ECRWRDIS bit.

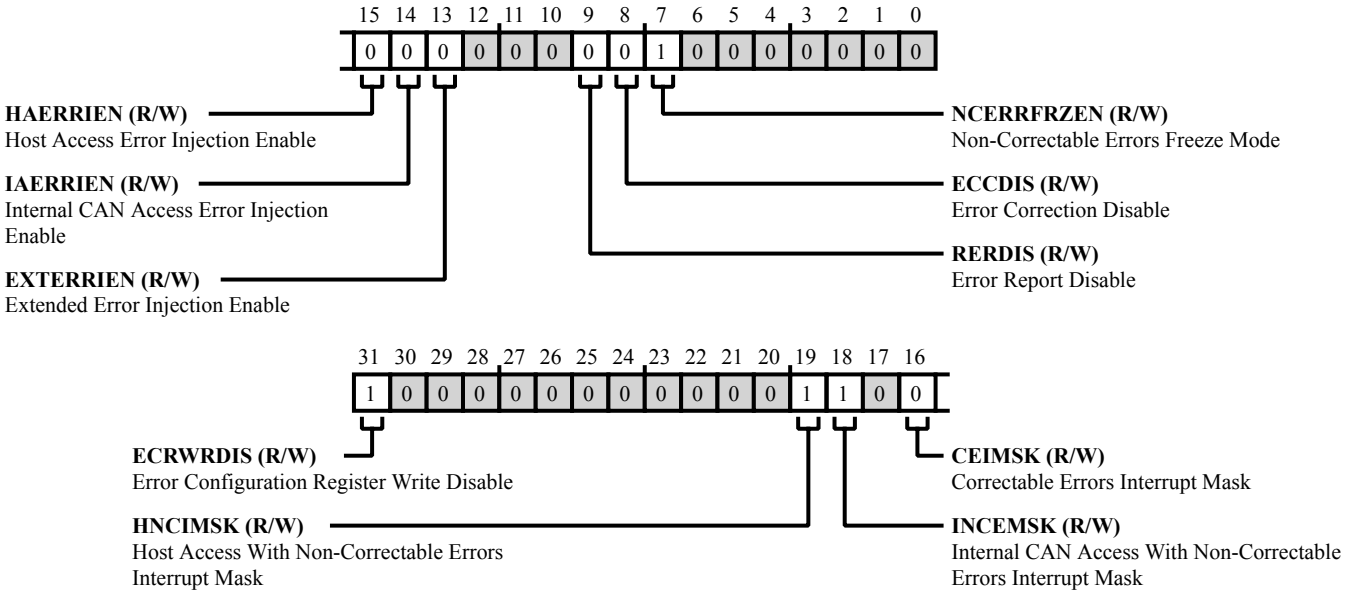


Figure 16-39: CANFD_MEC Register Diagram

Table 16-57: CANFD_MEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ECRWRDIS	Error Configuration Register Write Disable. The CANFD_MEC.ECRWRDIS bit disables writes to the CANFD_MEC register. The CANFD_MEC.ECRWRDIS bit is automatically set to 1 (disabled) when the CANFD_CTL2.ECRWREN bit is enabled. See the protocol described in the Detection and Correction of Memory Errors section of the CANFD chapter.	
		0	Enable
		1	Disable
19 (R/W)	HNCIMSK	Host Access With Non-Correctable Errors Interrupt Mask. The CANFD_MEC.HNCIMSK bit enables the interrupt in case of non-correctable errors detected in memory reads issued by the host.	
		0	Disable
		1	Enable

Table 16-57: CANFD_MEC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	INCEMSK	Internal CAN Access With Non-Correctable Errors Interrupt Mask. The `CANFD_MEC.INCEMSK` bit enables the interrupt in case of non-correctable errors detected in memory reads issued by the CAN internal processes.	
		0	Disable
		1	Enable
16 (R/W)	CEIMSK	Correctable Errors Interrupt Mask. The `CANFD_MEC.CEIMSK` bit enables the interrupt in case of correctable errors detected in memory reads issued by the host or CANFD module internal processes.	
		0	Disable
		1	Enable
15 (R/W)	HAERRIEN	Host Access Error Injection Enable. The `CANFD_MEC.HAERRIEN` bit enables the injection of errors for memory reads issued by the host.	
		0	Disable
		1	Enable
14 (R/W)	IAERRIEN	Internal CAN Access Error Injection Enable. The `CANFD_MEC.IAERRIEN` bit enables the injection of errors for memory reads issued by the CAN internal processes.	
		0	Disable
		1	Enable
13 (R/W)	EXTERRIEN	Extended Error Injection Enable. Memory accesses performed by internal CANFD processes are 64-bit accesses. The `CANFD_MEC.EXTERRIEN` bit extends the error injection on 32-bit memory accesses to the complementary 32-bit word using the same 32-bit error injection data and parity words. When the `CANFD_MEC.EXTERRIEN` bit is enabled, the error injection is applied to the 64-bit word. When the bit is disabled, error injection is only applied to the 32-bit word. See the descriptions of the `CANFD_ERR_IDP` and `CANFD_ERR_IPP` registers.	
		0	Disable
		1	Enable

Table 16-57: CANFD_MEC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	RERDIS	Error Report Disable. The `CANFD_MEC.RERDIS` bit disables the update of the error report registers. When the error report registers are disabled, the update of error-related flags and the generation of bus transfer errors are still active. Note that when reading the report registers, the `CANFD_MEC.RERDIS` bit must be set to ensure coherence on the consecutive register reads.	
		0	Enable
		1	Disable
8 (R/W)	ECCDIS	Error Correction Disable. The `CANFD_MEC.ECCDIS` bit disables the memory detection and correction mechanism. In addition to disabling the error report mechanism, when the `CANFD_MEC.ECCDIS` bit is set, the update of the error-related flags and generation of bus transfer errors also stops. The parity bits continue to be calculated and written into memory on write transactions.	
		0	Enable
		1	Disable
7 (R/W)	NCERRFRZEN	Non-Correctable Errors Freeze Mode. The `CANFD_MEC.NCERRFRZEN` bit determines the response when a non-correctable error is detected in a memory read performed by CANFD module internal processes. Enabling the `CANFD_MEC.NCERRFRZEN` bit puts the CANFD module in freeze mode, preventing corrupted data from being treated as valid by CANFD internal processes. Disabling the bit maintains normal operation.	
		0	Disable
		1	Enable

Pretended Networking Payload Low Filter2 Register

The CANFD_FLTR_DATA1_HI register contain Payload Filter 1 high order bits of the target value. This is for filtering the incoming message payload in pretended networking mode. It is for =, ≤, or ≥ to comparisons; or as the lower limit value in payload range detection.

The CANFD_FLTR_DATA1_HI register can only be written in freeze mode.

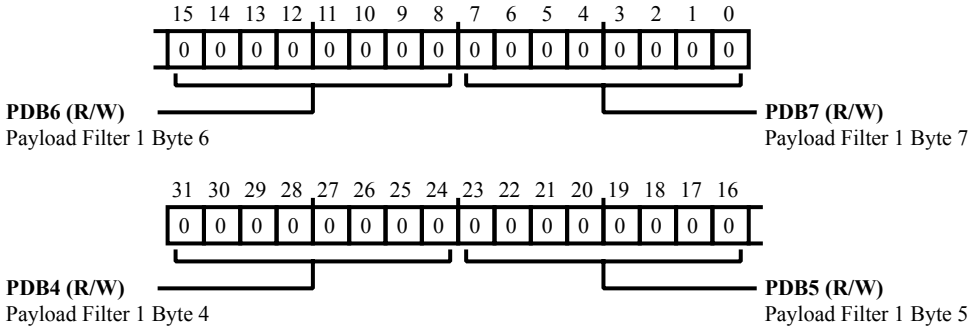


Figure 16-40: CANFD_FLTR_DATA1_HI Register Diagram

Table 16-58: CANFD_FLTR_DATA1_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PDB4	Payload Filter 1 Byte 4. The CANFD_FLTR_DATA1_HI.PDB4 bit field contains the payload filter 1 high order bits for pretended networking payload filtering corresponding to data byte 4.
23:16 (R/W)	PDB5	Payload Filter 1 Byte 5. The CANFD_FLTR_DATA1_HI.PDB5 bit field contains the payload filter 1 high order bits for pretended networking payload filtering corresponding to data byte 5.
15:8 (R/W)	PDB6	Payload Filter 1 Byte 6. The CANFD_FLTR_DATA1_HI.PDB6 bit field contains the payload filter 1 high order bits for pretended networking payload filtering corresponding to data byte 6.
7:0 (R/W)	PDB7	Payload Filter 1 Byte 7. The CANFD_FLTR_DATA1_HI.PDB7 bit field contains the payload filter 1 high order bits for pretended networking payload filtering corresponding to data byte 7.

Pretended Networking Payload Low Filter1 Register

The CANFD_FLTR_DATA1_LO register contain Payload Filter 1 low order bits of the target value. This is for filtering the incoming message payload in pretended networking mode. It is for =, ≤, or ≥ to comparisons; or as the lower limit value in payload range detection.

The CANFD_FLTR_DATA1_LO register can only be written in freeze mode.

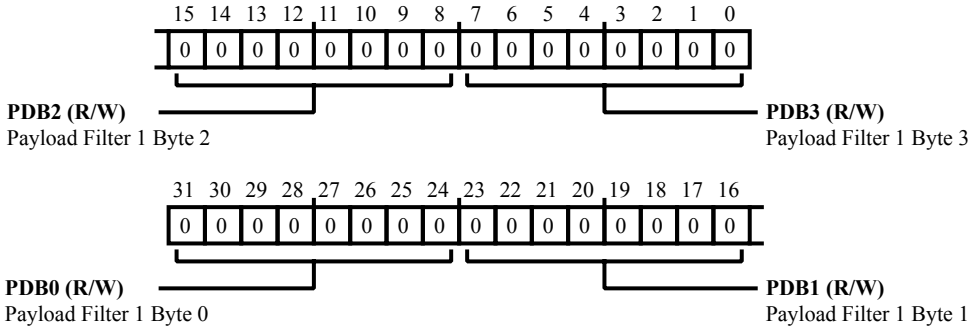


Figure 16-41: CANFD_FLTR_DATA1_LO Register Diagram

Table 16-59: CANFD_FLTR_DATA1_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PDB0	Payload Filter 1 Byte 0. The CANFD_FLTR_DATA1_LO.PDB0 bit field contains the payload filter 1 low order bits for pretended networking payload filtering corresponding to data byte 0.
23:16 (R/W)	PDB1	Payload Filter 1 Byte 1. The CANFD_FLTR_DATA1_LO.PDB1 bit field contains the payload filter 1 low order bits for pretended networking payload filtering corresponding to data byte 1.
15:8 (R/W)	PDB2	Payload Filter 1 Byte 2. The CANFD_FLTR_DATA1_LO.PDB2 bit field contains the payload filter 1 low order bits for pretended networking payload filtering corresponding to data byte 2.
7:0 (R/W)	PDB3	Payload Filter 1 Byte 3. The CANFD_FLTR_DATA1_LO.PDB3 bit field contains the payload filter 1 low order bits for pretended networking payload filtering corresponding to data byte 3.

Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register

The CANFD_FLTR_DATA2_DMSK_HI register has two functions:

It contains the high order bits for the Payload Filter 2 value for use as the upper limit in payload range detection.

When exact payload filtering criteria is selected, the CANFD_FLTR_DATA2_DMSK_HI register is the payload mask.

Otherwise, the CANFD_FLTR_DATA2_DMSK_HI register is not used. It can only be written in freeze mode.

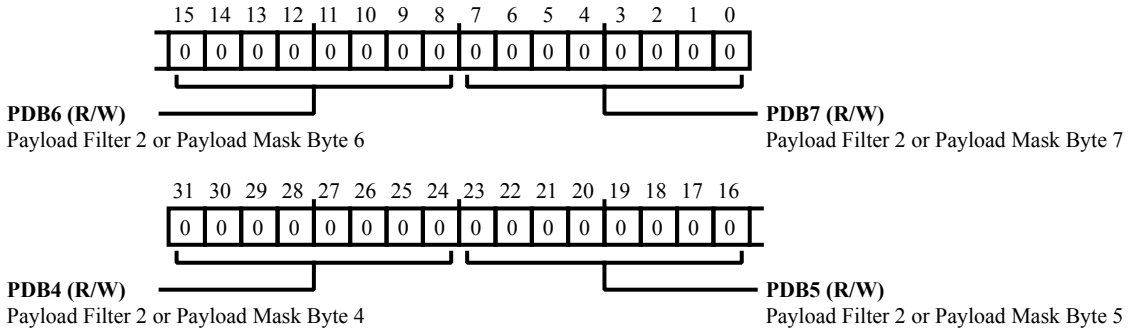


Figure 16-42: CANFD_FLTR_DATA2_DMSK_HI Register Diagram

Table 16-60: CANFD_FLTR_DATA2_DMSK_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PDB4	Payload Filter 2 or Payload Mask Byte 4. The CANFD_FLTR_DATA2_DMSK_HI.PDB4 bit field contains the payload filter 2 high order bits or payload mask high order bits for pretended networking payload filtering corresponding to data byte 4.
23:16 (R/W)	PDB5	Payload Filter 2 or Payload Mask Byte 5. The CANFD_FLTR_DATA2_DMSK_HI.PDB5 bit field contains the payload filter 2 high order bits or payload mask high order bits for pretended networking payload filtering corresponding to data byte 5.
15:8 (R/W)	PDB6	Payload Filter 2 or Payload Mask Byte 6. The CANFD_FLTR_DATA2_DMSK_HI.PDB6 bit field contains the payload filter 2 high order bits or payload mask high order bits for pretended networking payload filtering corresponding to data byte 6.
7:0 (R/W)	PDB7	Payload Filter 2 or Payload Mask Byte 7. The CANFD_FLTR_DATA2_DMSK_HI.PDB7 bit field contains the payload filter 2 high order bits or payload mask high order bits for pretended networking payload filtering corresponding to data byte 7.

Pretended Networking Payload Low Filter2 / Payload Low Mask Register

The CANFD_FLTR_DATA2_DMSK_LO register has two functions:

It contains the low order bits for the Payload Filter 2 value for use as the upper limit in payload range detection.

When exact payload filtering criteria is selected, the CANFD_FLTR_DATA2_DMSK_LO register is the payload mask.

Otherwise, the CANFD_FLTR_DATA2_DMSK_LO register is not used. It can only be written in freeze mode.

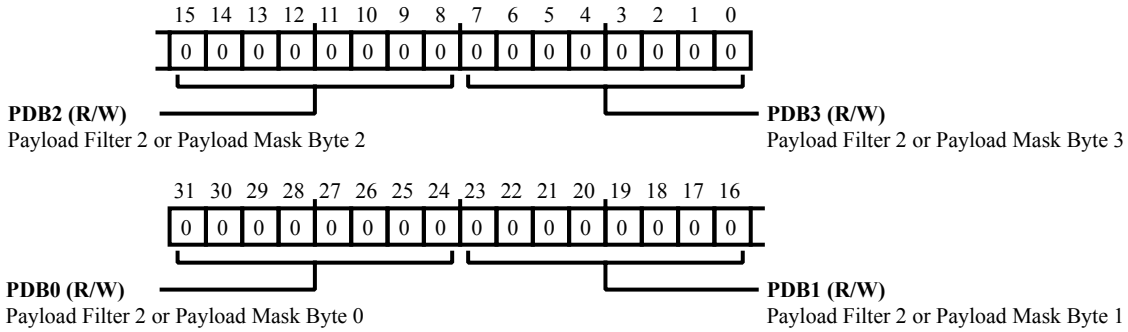


Figure 16-43: CANFD_FLTR_DATA2_DMSK_LO Register Diagram

Table 16-61: CANFD_FLTR_DATA2_DMSK_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PDB0	Payload Filter 2 or Payload Mask Byte 0. The CANFD_FLTR_DATA2_DMSK_LO.PDB0 bit field contains the payload filter 2 low order bits or payload mask low order bits for pretended networking payload filtering corresponding to data byte 0.
23:16 (R/W)	PDB1	Payload Filter 2 or Payload Mask Byte 1. The CANFD_FLTR_DATA2_DMSK_LO.PDB1 bit field contains the payload filter 2 low order bits or payload mask low order bits for pretended networking payload filtering corresponding to data byte 1.
15:8 (R/W)	PDB2	Payload Filter 2 or Payload Mask Byte 2. The CANFD_FLTR_DATA2_DMSK_LO.PDB2 bit field contains the payload filter 2 low order bits or payload mask low order bits for pretended networking payload filtering corresponding to data byte 2.
7:0 (R/W)	PDB3	Payload Filter 2 or Payload Mask Byte 3. The CANFD_FLTR_DATA2_DMSK_LO.PDB3 bit field contains the payload filter 2 low order bits or payload mask low order bits for pretended networking payload filtering corresponding to data byte 3.

Error Report Address Register

The CANFD_ERR_RADDR register reports the address of an access with an error (correctable or non-correctable) and reports the identification of the source of that access.

The address is 32-bit aligned. Non-aligned accesses, where the CANFD_ERRADDR[1:0] bits are non-zero, are reported with the address aligned and the data in the CANFD_ERR_RDAT register shifted accordingly. If there is an error in an access that is larger than 32 bits (as performed by CANFD internal processes), the address of the 32-bit word with the error is reported. In case of errors detected in more than one 32-bit word, only the least significant address is reported.

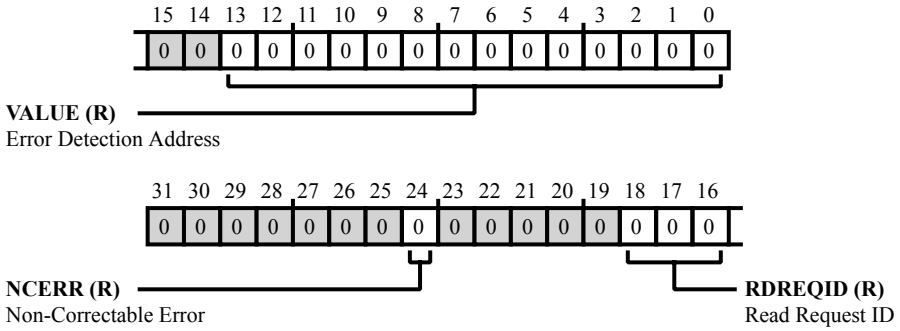


Figure 16-44: CANFD_ERR_RADDR Register Diagram

Table 16-62: CANFD_ERR_RADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/NW)	NCERR	Non-Correctable Error. The CANFD_ERR_RADDR.NCERR bit indicates that a report is due to an non-correctable error. When the CANFD_ERR_RADDR.NCERR bit is set, it is reporting a non-correctable error. When it is cleared, it is reporting a correctable-error.	
18:16 (R/NW)	RDREQID	Read Request ID. The CANFD_ERR_RADDR.RDREQID bit field identifies the details of a memory read request.	
		0	Move-Out CANFD Access
		1	Move-In CANFD Access
		2	TX Arbitration
		3	Rx Matching
		4	Move-Out Host Access
		5	Reserved
		6	Reserved
		7	Reserved

Table 16-62: CANFD_ERR_RADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
13:0 (R/NW)	VALUE	Error Detection Address. The `CANFD_ERR_RADDR.VALUE` bit field contains the address where the error was detected. See the description of the Error Injection Address Register (`CANFD_ERR_IADDR`).

Error Report Data Register

The CANFD_ERR_RDAT register reports the raw data (unmodified by the ECC logic correction) read from the memory with the error. The value reported does not represent the transient values of the BUSY bit when reading a message buffer.

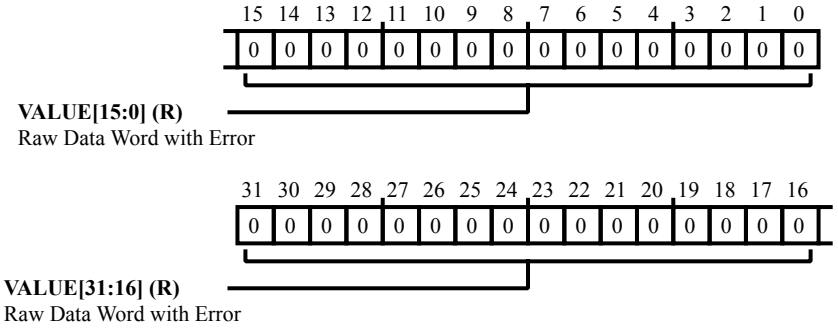


Figure 16-45: CANFD_ERR_RDAT Register Diagram

Table 16-63: CANFD_ERR_RDAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Raw Data Word with Error. The CANFD_ERR_RDAT.VALUE bit field contains the raw data word read from memory with error. This data word is unmodified by the correction performed by ECC logic and does not represent the transient values of the BUSY bit when reading the message buffer.

Error Report Syndrome Register

The CANFD_ERR_RSYN register holds the syndrome detected in a memory read with an error. It reports the bytes that are read in the 32-bit read transaction.

Each SYNDn field indicates the type of error and which bit in byte (n) is affected by the error. The CANFD_ERR_RSYN.SYND3 bit field corresponds to the most significant byte in the data word read from memory and the CANFD_ERR_RSYN.SYND0 bit field corresponds to the least significant.

Each BEn field indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, even for the non-read ones. Errors detected in non-read bytes are indicated and reported.

See the Detection and Correction of Memory Errors section of the CANFD chapter for more details.

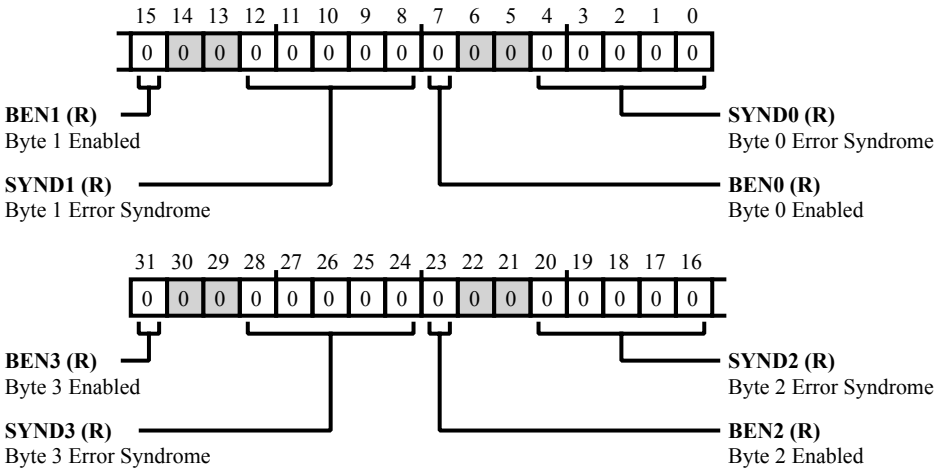


Figure 16-46: CANFD_ERR_RSYN Register Diagram

Table 16-64: CANFD_ERR_RSYN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	BEN3	Byte 3 Enabled. The CANFD_ERR_RSYN.BEN3 bit indicates if byte 3 (most significant) was read. If the CANFD_ERR_RSYN.BEN3 bit is enabled the byte was read, and if it is disabled then it was not read.	
		0	Disable
		1	Enable
28:24 (R/NW)	SYND3	Byte 3 Error Syndrome. The CANFD_ERR_RSYN.SYND3 bit field contains the error syndrome for byte 3. The Syndrome Definition table in the CANFD chapter defines the values for the CANFD_ERR_RSYN.SYND3 bit field.	

Table 16-64: CANFD_ERR_RSYN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/NW)	BEN2	Byte 2 Enabled. The CANFD_ERR_RSYN.BEN2 bit indicates if byte 2 was read. If the CANFD_ERR_RSYN.BEN2 bit is enabled the byte was read, and if it is disabled then it was not read.	
		0	Disable
		1	Enable
20:16 (R/NW)	SYND2	Byte 2 Error Syndrome. The CANFD_ERR_RSYN.SYND2 bit field contains the error syndrome for byte 2. The Syndrome Definition table in the CANFD chapter defines the values for the CANFD_ERR_RSYN.SYND2 bit field.	
15 (R/NW)	BEN1	Byte 1 Enabled. The CANFD_ERR_RSYN.BEN1 bit indicates if byte 1 was read. If the CANFD_ERR_RSYN.BEN1 bit is enabled the byte was read, and if it is disabled then it was not read.	
		0	Disable
		1	Enable
12:8 (R/NW)	SYND1	Byte 1 Error Syndrome. The CANFD_ERR_RSYN.SYND1 bit field contains the error syndrome for byte 1. The Syndrome Definition table in the CANFD chapter defines the values for the CANFD_ERR_RSYN.SYND1 bit field.	
7 (R/NW)	BEN0	Byte 0 Enabled. The CANFD_ERR_RSYN.BEN0 bit indicates if byte 2 (least significant) was read. If the CANFD_ERR_RSYN.BEN0 bit is enabled the byte was read, and if it is disabled then it was not read.	
		0	Disable
		1	Enable
4:0 (R/NW)	SYND0	Byte 0 Error Syndrome. The CANFD_ERR_RSYN.SYND0 bit field contains the error syndrome for byte 0 (least significant). The Syndrome Definition table in the CANFD chapter defines the values for the CANFD_ERR_RSYN.SYND0 bit field.	

Receive Mailbox14 Mask Register

The CANFD_RX_14_MSK register is located in RAM.

When the CANFD_CFG.IRMQEN bit is asserted, the CANFD_RX_14_MSK register has no effect.

The CANFD_RX_14_MSK register is used to mask the filter fields of message buffer 14.

This register can only be programmed while the CANFD module is in freeze mode as it is blocked by hardware in other modes.

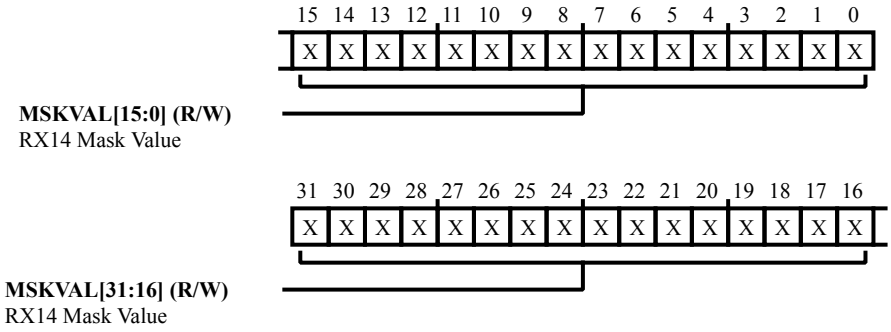


Figure 16-47: CANFD_RX_14_MSK Register Diagram

Table 16-65: CANFD_RX_14_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSKVAL	RX14 Mask Value. In the CANFD_RX_14_MSK.MSKVAL bit field, if a bit is a zero the corresponding bit in the filter is a don't care. If it is a one, the corresponding bit in the filter is checked. Each mask bit masks the corresponding mailbox 14 filter field in the same way that the CANFD_RX_MB_GMSK register masks other mailbox filters.

Receive Mailbox15 Mask Register

The CANFD_RX_15_MSK register is located in RAM.

When the CANFD_CFG.IRMQEN bit is asserted, the CANFD_RX_15_MSK register has no effect.

The CANFD_RX_15_MSK register is used to mask the filter fields of message buffer 15.

This register can only be programmed while the CANFD module is in freeze mode as it is blocked by hardware in other modes.

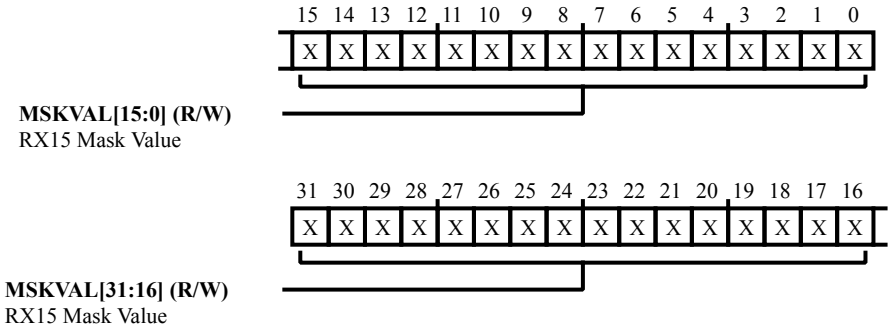


Figure 16-48: CANFD_RX_15_MSK Register Diagram

Table 16-66: CANFD_RX_15_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSKVAL	RX15 Mask Value. In the CANFD_RX_15_MSK.MSKVAL bit field, if a bit is a zero the corresponding bit in the filter is a don't care. If it is a one, the corresponding bit in the filter is checked. Each mask bit masks the corresponding mailbox 15 filter field in the same way that the CANFD_RX_MB_GMSK register masks other mailbox filters.

Receive FIFO Global Mask Register

The CANFD_RX_FIFO_GMSK register is located in RAM.

If the Rx FIFO is enabled, the CANFD_RX_FIFO_GMSK register is used to mask the Rx FIFO ID filter table elements that do not have a corresponding CANFD_RX_IMSK according to the CANFD_CTL2.RFFNUM bit field setting.

The CANFD_RX_FIFO_GMSK register can only be written in freeze mode as it is blocked by hardware in other modes.

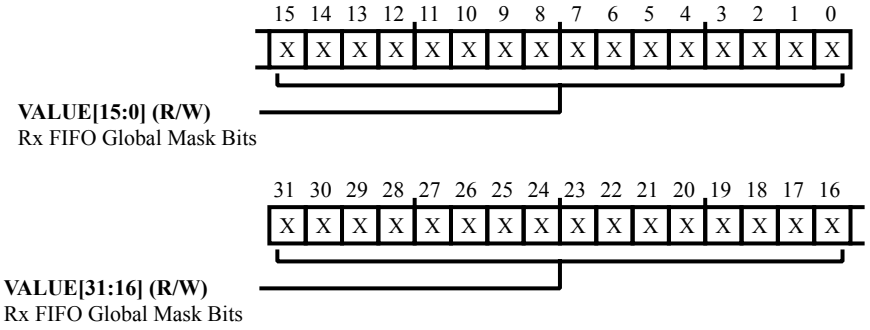


Figure 16-49: CANFD_RX_FIFO_GMSK Register Diagram

Table 16-67: CANFD_RX_FIFO_GMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Rx FIFO Global Mask Bits. The CANFD_RX_FIFO_GMSK.VALUE bits mask the ID filter table elements bits in a perfect alignment. If the CANFD_RX_FIFO_GMSK.VALUE bit is set, the corresponding bit in the filter is checked. If it is clear, the corresponding bit in the filter is a dont care. See the Identifier Acceptance Filter Fields for Global Mask Bits table in the CANFD chapter.

Receive FIFO Information Register

The CANFD_RX_FIFO register provides information on the Rx FIFO.

The CANFD_RX_FIFO register is the port through which the processor accesses the output of the RXFIR FIFO located in RAM. The CANFD_RX_FIFO register is written by the CANFD module whenever a new message is moved into the Rx FIFO and the register output is updated whenever the output of the Rx FIFO is updated with the next message.

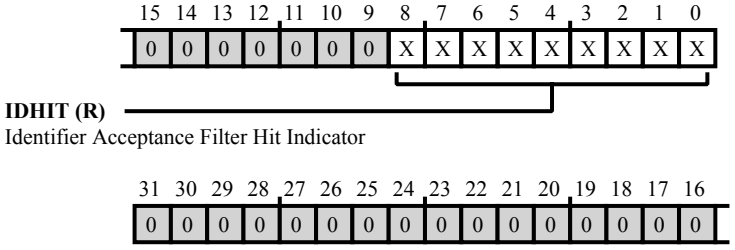


Figure 16-50: CANFD_RX_FIFO Register Diagram

Table 16-68: CANFD_RX_FIFO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/NW)	IDHIT	Identifier Acceptance Filter Hit Indicator. The CANFD_RX_FIFO.IDHIT bit field indicates which identifier acceptance filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID, the first matching IDAF found (lowest number) by the matching process is indicated. The CANFD_RX_FIFO.IDHIT bit field is valid only while the CANFD_IFLG1.MB05 bit is set.

Receive Individual Mask Register

The CANFD_RX_IMSK[n] registers are used to store the acceptance masks for ID filtering in Rx MBs and the Rx FIFO.

When the Rx FIFO is disabled (CANFD_CFG.RFEN is disabled), an individual mask is provided for each available Rx mailbox on a one-to-one correspondence. When the Rx FIFO is enabled (CANFD_CFG.RFEN is enabled), an individual mask is provided for each Rx FIFO ID Filter table element on a one-to-one correspondence depending on the setting of the CANFD_CTL2.RFFNUM bit.

The CANFD_RX_IMSK0 register stores the individual mask associated with either MB0 or ID Filter Table Element 0, CANFD_RX_IMSK1 stores the individual mask associated with either MB1 or ID Filter Table Element 1, and so on.

The CANFD_RX_IMSK[n] registers are only be accessed by the processor while the CANFD is in freeze mode and are otherwise blocked by hardware.

The CANFD_RX_IMSK[n] registers are not affected by reset. They are located in RAM and must be explicitly initialized prior to any reception.

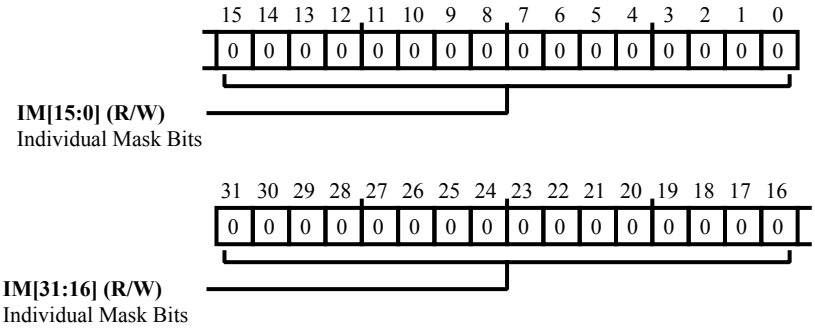


Figure 16-51: CANFD_RX_IMSK[n] Register Diagram

Table 16-69: CANFD_RX_IMSK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	IM	Individual Mask Bits. The CANFD_RX_IMSK[n].IM bits masks the corresponding bit in both the mailbox filter and Rx FIFO ID filter table element in distinct ways. For mailbox filters, see the CANFD_RX_MB_GMSK register description. For Rx FIFO ID filter table elements, see the CANFD_RX_FIFO_GMSK register description. If a bit in the CANFD_RX_IMSK[n].IM field is set, the corresponding bit in the filter is checked. If it is cleared, the corresponding bit in the filter is a don't care.

Receive Mailbox Global Mask Register

The CANFD_RX_MB_GMSK register is located in RAM.

When the CANFD_CFG.IRMQEN bit is disabled, the CANFD_RX_MB_GMSK register is always in effect (the bits in the MG field will mask the mailbox filter bits).

When the CANFD_CFG.IRMQEN bit is enabled, the CANFD_RX_MB_GMSK register has no effect (the bits in the MG field will not mask the mailbox filter bits).

The CANFD_RX_MB_GMSK register is used to mask the filter fields of all Rx MBs, excluding MBs 14–15, which have individual mask registers.

The CANFD_RX_MB_GMSK register can only be written in freeze mode and is blocked by hardware in other modes.

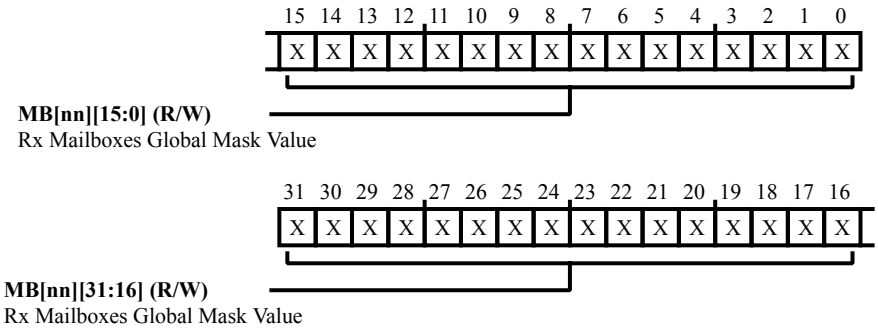


Figure 16-52: CANFD_RX_MB_GMSK Register Diagram

Table 16-70: CANFD_RX_MB_GMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	MB[nn]	Rx Mailboxes Global Mask Value. For all CANFD_RX_MB_GMSK.MB[nn] bits, write =0 for stop, and write =1 for start. Read =1 when timer is running. The bits in the CANFD_RX_MB_GMSK.MB[nn] bit field mask the mailbox filter bits. If a bit in the field is a zero, the corresponding bit in the filter is a don't care. If the bit is a one, the corresponding bit in the filter is checked. Note that the alignment with the ID word of the mailbox is not perfect as the two most significant bits affect the fields RTR and IDE, which are located in the control and status (C/S) word of the mailbox. For more details on this bit field, see the Mailbox Global Filter Field Mask table in the CANFD chapter.	
		0	Don't Care
		1	Checked

Free Running Timer Register

The CANFD_TMR register is a 16-bit free running counter. The timer starts from 0x0 after reset, counts linearly to 0xFFFF, and wraps around.

The timer is incremented by the CAN bit clock, which defines the baud rate on the CAN bus. During a message transmission/reception, the timer increments by one for each bit that is received or transmitted. When there is no message on the bus, the timer counts using the previously programmed baud rate. The timer is not incremented during module disable, doze, stop, pretended networking, and freeze modes.

The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the time stamp entry in a message buffer after the successful reception or transmission of a message.

If the CANFD_CTL1.TSYNEN bit is enabled, the timer is reset whenever a message is received in the first available mailbox, according to the CANFD_CTL2.RFFNUM setting.

The processor can write to this register anytime. However, if the write occurs at the same time that the timer is being reset by a reception in the first mailbox, then the write value is discarded.

Reading this register affects the mailbox unlocking procedure.

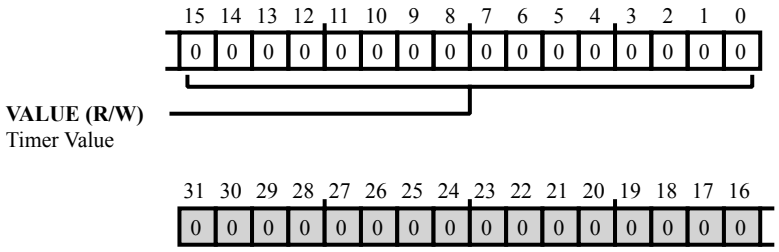


Figure 16-53: CANFD_TMR Register Diagram

Table 16-71: CANFD_TMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Timer Value. The CANFD_TMR.VALUE bit field contains the free-running counter value.

Wakeup Message Buffer Data 4-7 Register

The CANFD_WMB[n]_DATA_HI registers are the wakeup message buffer data 4-7 registers. Each of the four wake up message buffers contains a register to store the data bytes 4 to 7 of the payload information of an incoming Rx message. This register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 4-7 registers are located at 0xB4C for WMB0, 0xB5C for WMB1, 0xB6C for WMB2, and 0xB7C for WMB3.

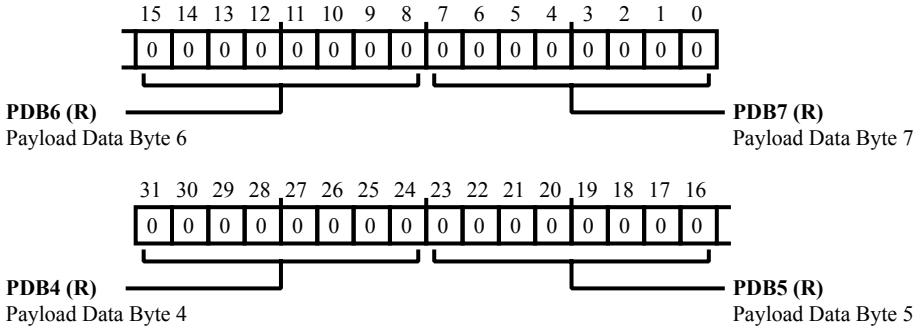


Figure 16-54: CANFD_WMB[n]_DATA_HI Register Diagram

Table 16-72: CANFD_WMB[n]_DATA_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB4	Payload Data Byte 4. The CANFD_WMB[n]_DATA_HI.PDB4 bit field contains the received payload corresponding to data byte 4 in pretended networking mode.
23:16 (R/NW)	PDB5	Payload Data Byte 5. The CANFD_WMB[n]_DATA_HI.PDB5 bit field contains the received payload corresponding to data byte 5 in pretended networking mode.
15:8 (R/NW)	PDB6	Payload Data Byte 6. The CANFD_WMB[n]_DATA_HI.PDB6 bit field contains the received payload corresponding to data byte 6 in pretended networking mode.
7:0 (R/NW)	PDB7	Payload Data Byte 7. The CANFD_WMB[n]_DATA_HI.PDB7 bit field contains the received payload corresponding to data byte 7 in pretended networking mode.

Wakeup Message Buffer Data 0-3 Register

The CANFD_WMB[n]_DATA_LO registers are the wakeup message buffer data 0-3 registers. Each of the four wake up message buffers contains a register to store the data bytes 0 to 3 of the payload information of an incoming Rx message. This register content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

Note that the data 0-3 registers are located at 0xB48 for WMB0, 0xB58 for WMB1, 0xB68 for WMB2, and 0xB78 for WMB3.

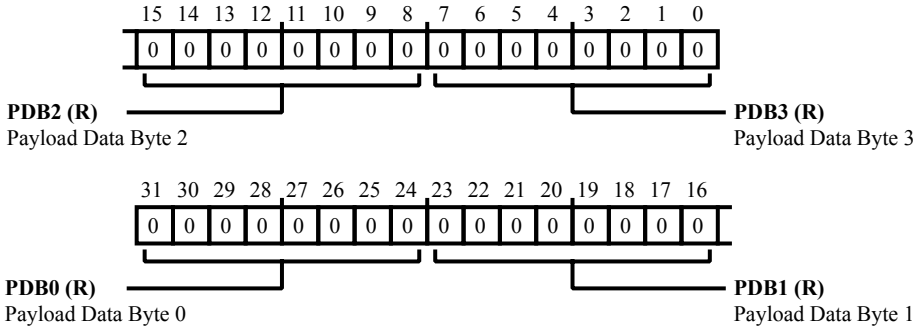


Figure 16-55: CANFD_WMB[n]_DATA_LO Register Diagram

Table 16-73: CANFD_WMB[n]_DATA_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PDB0	Payload Data Byte 0. The CANFD_WMB[n]_DATA_LO.PDB0 bit field contains the received payload corresponding to data byte 0 in pretended networking mode.
23:16 (R/NW)	PDB1	Payload Data Byte 1. The CANFD_WMB[n]_DATA_LO.PDB1 bit field contains the received payload corresponding to data byte 1 in pretended networking mode.
15:8 (R/NW)	PDB2	Payload Data Byte 2. The CANFD_WMB[n]_DATA_LO.PDB2 bit field contains the received payload corresponding to data byte 2 in pretended networking mode.
7:0 (R/NW)	PDB3	Payload Data Byte 3. The CANFD_WMB[n]_DATA_LO.PDB3 bit field contains the received payload corresponding to data byte 3 in pretended networking mode.

Wakeup Message ID Buffer Register

The CANFD_WMB[n]_ID registers are the wakeup message ID buffers. Each of the four wake up message buffers contains a register to store the ID information of an incoming Rx message.

Note that the ID registers are located at 0xB44 for WMB0, 0xB54 for WMB1, 0xB64 for WMB2, and 0xB74 for WMB3.

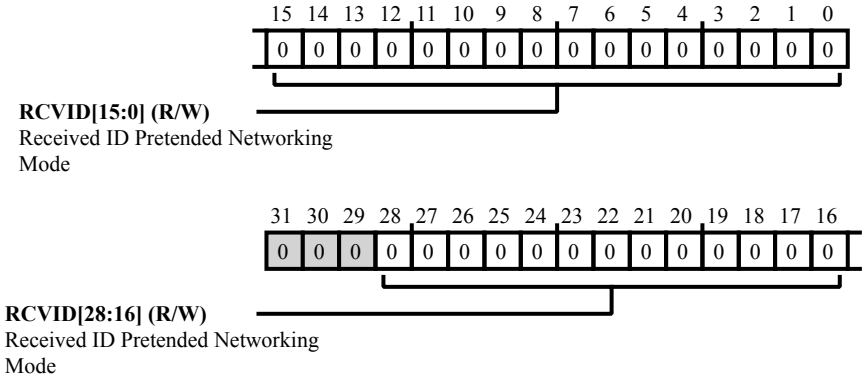


Figure 16-56: CANFD_WMB[n]_ID Register Diagram

Table 16-74: CANFD_WMB[n]_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	RCVID	Received ID Pretended Networking Mode. The CANFD_WMB[n]_ID.RCVID bit field stores the received ID as either, the 29 bits of the extended frame format (considering the ID[28:0] field) or the 11 bits of the standard frame format (considering the ID[28:18] field only. In the standard frame format, the remaining bits in the ID[17:0] range have no meaning.

Wakeup Message Buffer Control/Status Register

The CANFD_WMB[n]_STAT registers are the for wakeup message buffer control and status information. Each of the four wake up message buffers contains a register to store the control status information (IDE, RTR and DLC fields) of an incoming Rx message.

Note that the control status registers are located at 0xB40 for WMB0, 0xB50 for WMB1, 0xB60 for WMB2, and 0xB70 for WMB3.

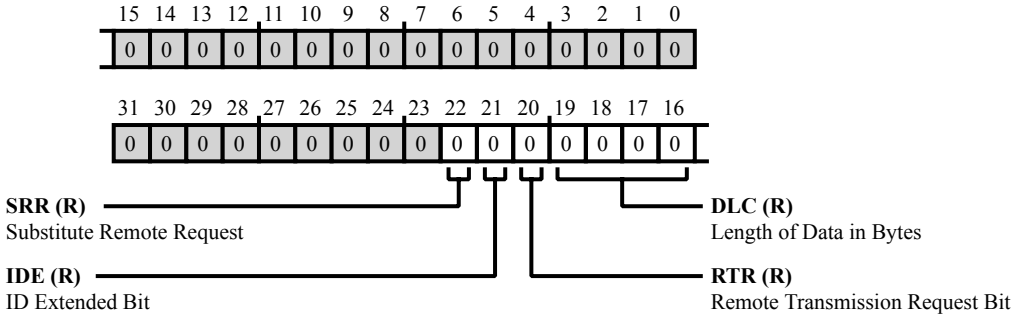


Figure 16-57: CANFD_WMB[n]_STAT Register Diagram

Table 16-75: CANFD_WMB[n]_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/NW)	SRR	Substitute Remote Request. The CANFD_WMB[n]_STAT.SRR bit is received as either recessive or dominant.	
21 (R/NW)	IDE	ID Extended Bit. The CANFD_WMB[n]_STAT.IDE bit identifies is the frame format is standard or extended.	
		0	Standard Frame Format
		1	Extended Frame Format
20 (R/NW)	RTR	Remote Transmission Request Bit. The CANFD_WMB[n]_STAT.RTR bit identifies if the frame is remote.	
		0	Data Frame (Not Remote)
		1	Remote Frame
19:16 (R/NW)	DLC	Length of Data in Bytes. The CANFD_WMB[n]_STAT.DLC bit field represents the length (in bytes) of the Rx data received when the CANFD module is in pretended networking mode. The CANFD_WMB[n]_STAT.DLC bit field is written by the CANFD module, copied from the data length code (DLC) field of the received frame. The DLC field indicates which data bytes are valid.	

Pretended Networking Wakeup Match Register

The CANFD_WUM register contains wake up information related to the matching processes performed while the CANFD module receives frames under pretended networking mode.

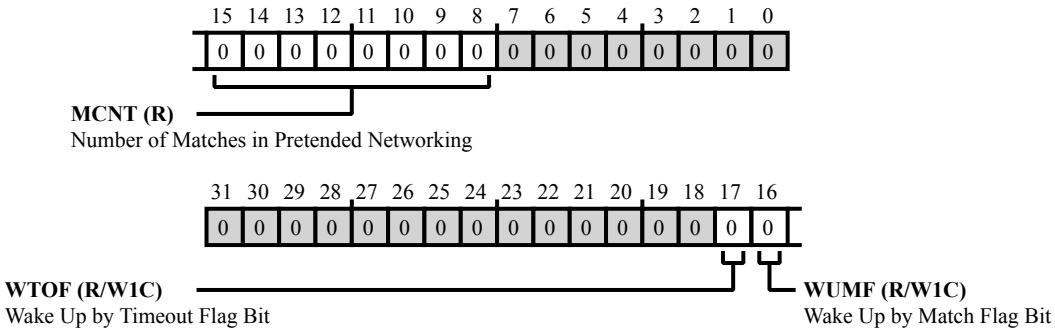


Figure 16-58: CANFD_WUM Register Diagram

Table 16-76: CANFD_WUM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W1C)	WTOF	Wake Up by Timeout Flag Bit. The CANFD_WUM.WTOF bit indicates if the CANFD module detects a timeout event during a time interval defined by the CANFD_PN_CTL2.MATCHTO bit field. WTOF generates a wakeup event if the CANFD_PN_CTL1.WTOFMSK enabled.	
		0	No event detected
		1	Wakeup event detected
16 (R/W1C)	WUMF	Wake Up by Match Flag Bit. The CANFD_WUM.WUMF bit indicates whether the CANFD module detects a matching Rx incoming message passing the filtering criteria specified in the CANFD_PN_CTL1 register. WUMF generates a wakeup event if the CANFD_PN_CTL1.WUMFMSK bit is enabled.	
		0	No event detected
		1	Wakeup event detected
15:8 (R/NW)	MCNT	Number of Matches in Pretended Networking. The CANFD_WUM.MCNT bit field reports the number of times a given message matches the predefined filtering criteria for ID and/or PL before a wake up event. The CANFD_WUM.MCNT bit field is reset by the CANFD module when it enters pretended networking mode. This bit field and is not affected by a soft reset.	

ADSP-SC59x MISCREG Register Descriptions

Misc registers for module are for integration purpose (MISCREG) contains the following registers.

Table 16-77: ADSP-SC59x MISCREG Register List

Name	Description
MISCREG_CAN_SYSCTL	CANFD Low Power Selection Mode
MISCREG_ECO_REG9	Error Register
MISCREG_PLL2_CONTROL	PLL2 Control Register
MISCREG_SH0_PFB_RANGE_SELECT	SH0 Prefetch Range Selection Register
MISCREG_SH1_PFB_RANGE_SELECT	SH1 Prefetch Range Selection Register
MISCREG_SHARC_BRIDGE_REMAP	SPORT Direct Interrupt Enable for SH0 and SH1

CANFD Low Power Selection Mode

The MISCREG_CAN_SYSCTL register contains the control bits for the different low power modes supported in the CANFD module.

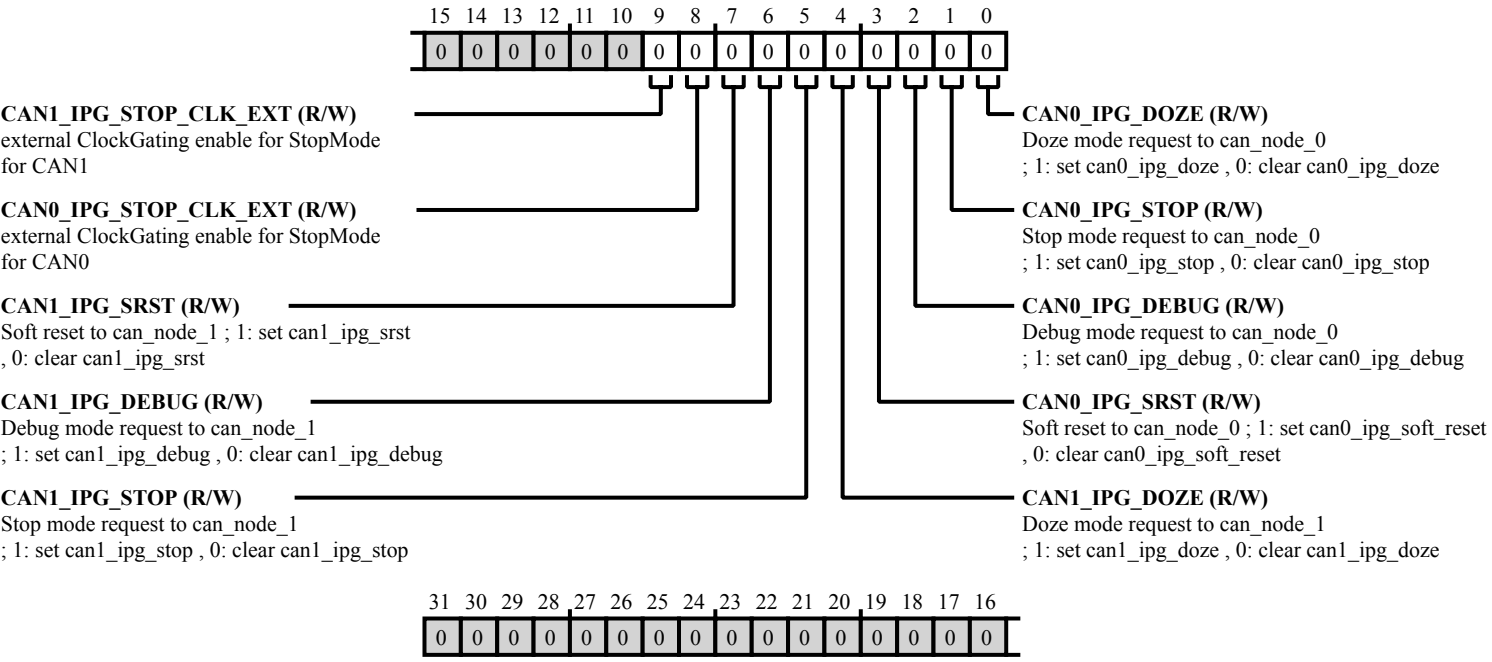


Figure 16-59: MISCREG_CAN_SYSCTL Register Diagram

Table 16-78: MISCREG_CAN_SYSCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	CAN1_IPG_STOP_CLK_EXT	external ClockGating enable for StopMode for CAN1.
8 (R/W)	CAN0_IPG_STOP_CLK_EXT	external ClockGating enable for StopMode for CAN0.
7 (R/W)	CAN1_IPG_SRST	Soft reset to can_node_1 ; 1: set can1_ipg_srst , 0: clear can1_ipg_srst.
6 (R/W)	CAN1_IPG_DEBUG	Debug mode request to can_node_1 ; 1: set can1_ipg_debug , 0: clear can1_ipg_debug.
5 (R/W)	CAN1_IPG_STOP	Stop mode request to can_node_1 ; 1: set can1_ipg_stop , 0: clear can1_ipg_stop.
4 (R/W)	CAN1_IPG_DOZE	Doze mode request to can_node_1 ; 1: set can1_ipg_doze , 0: clear can1_ipg_doze.

Table 16-78: MISCREG_CAN_SYSCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W)	CAN0_IPG_SRST	Soft reset to can_node_0 ; 1: set can0_ipg_soft_reset , 0: clear can0_ipg_soft_reset.
2 (R/W)	CAN0_IPG_DEBUG	Debug mode request to can_node_0 ; 1: set can0_ipg_debug , 0: clear can0_ipg_debug.
1 (R/W)	CAN0_IPG_STOP	Stop mode request to can_node_0 ; 1: set can0_ipg_stop , 0: clear can0_ipg_stop.
0 (R/W)	CAN0_IPG_DOZE	Doze mode request to can_node_0 ; 1: set can0_ipg_doze , 0: clear can0_ipg_doze.

Error Register

The MISCREG_ECO_REG9 register contains eight Tx channels, eight Rx channels, and DMA errors for both EMAC0 and EMAC1.

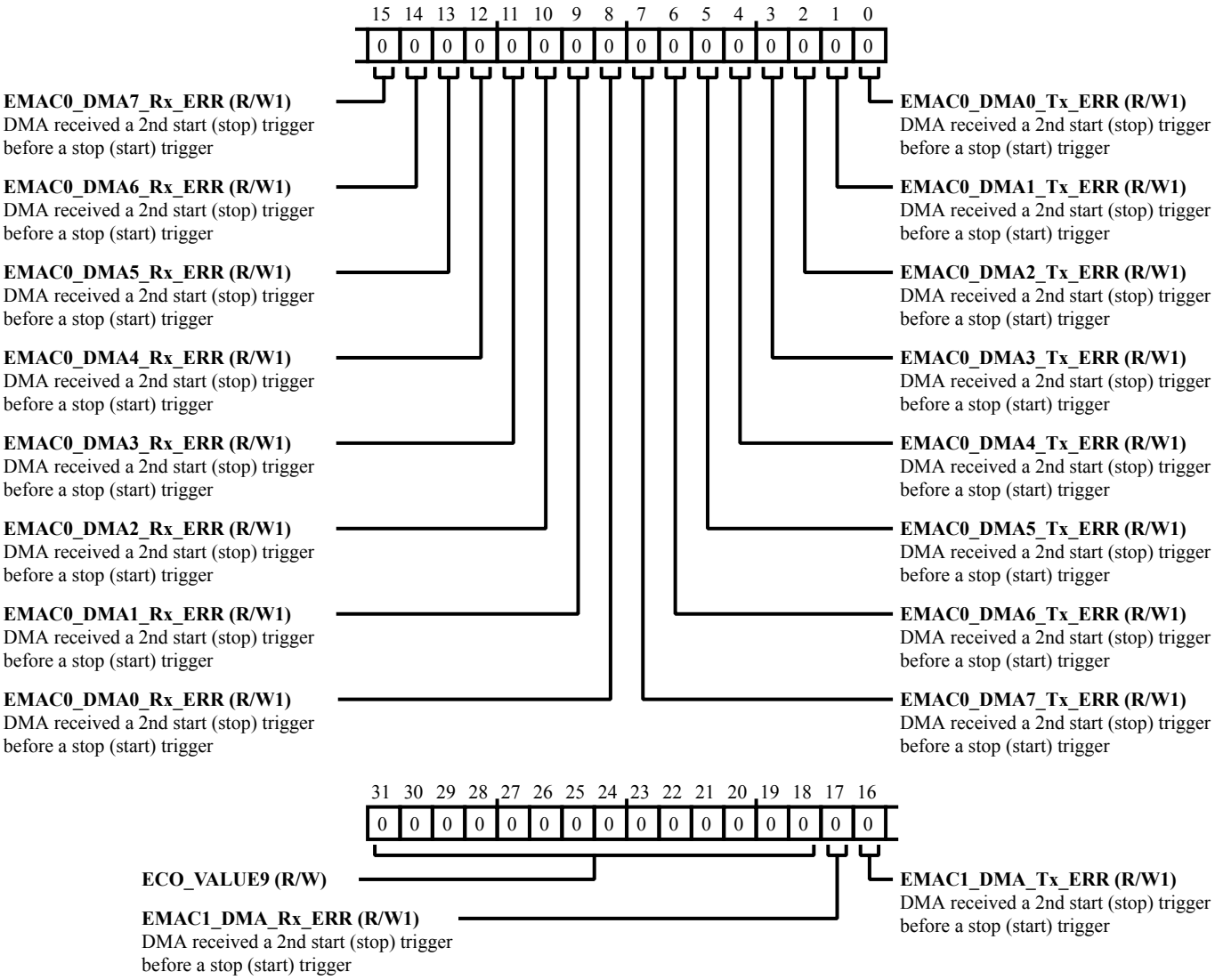


Figure 16-60: MISCREG_ECO_REG9 Register Diagram

Table 16-79: MISCREG_ECO_REG9 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:18 (R/W)	ECO_VALUE9	.

Table 16-79: MISCREG_ECO_REG9 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W1)	EMAC1_DMA_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
16 (R/W1)	EMAC1_DMA_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
15 (R/W1)	EMAC0_DMA7_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
14 (R/W1)	EMAC0_DMA6_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
13 (R/W1)	EMAC0_DMA5_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
12 (R/W1)	EMAC0_DMA4_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
11 (R/W1)	EMAC0_DMA3_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
10 (R/W1)	EMAC0_DMA2_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
9 (R/W1)	EMAC0_DMA1_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
8 (R/W1)	EMAC0_DMA0_RX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
7 (R/W1)	EMAC0_DMA7_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
6 (R/W1)	EMAC0_DMA6_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
5 (R/W1)	EMAC0_DMA5_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
4 (R/W1)	EMAC0_DMA4_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
3 (R/W1)	EMAC0_DMA3_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
2 (R/W1)	EMAC0_DMA2_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.

Table 16-79: MISCREG_ECO_REG9 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W1)	EMAC0_DMA1_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.
0 (R/W1)	EMAC0_DMA0_TX_ERR	DMA received a 2nd start (stop) trigger before a stop (start) trigger.

PLL2 Control Register

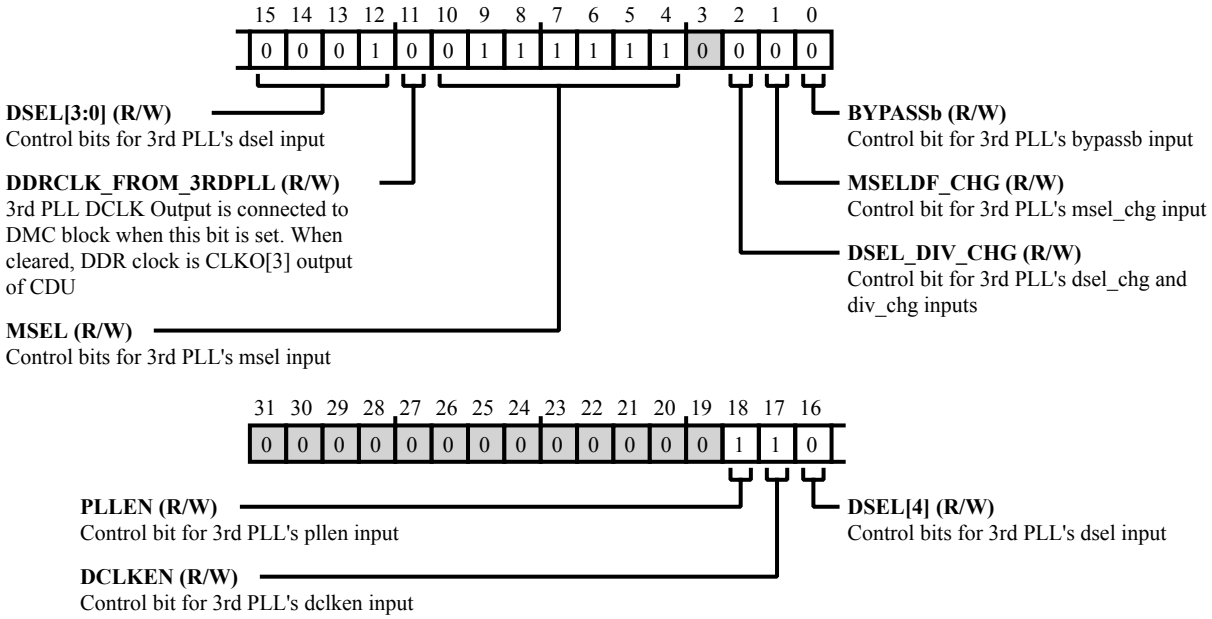


Figure 16-61: MISCREG_PLL2_CONTROL Register Diagram

Table 16-80: MISCREG_PLL2_CONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/W)	PLLEN	Control bit for 3rd PLL's pllen input. Internal signal from Dynamic Power Management Unit (DPM) which enables/disables the PLL system (e.g. in Active mode, during BYPASS.)
17 (R/W)	DCLKEN	Control bit for 3rd PLL's dclken input. Enables DCLK clock buffer
16:12 (R/W)	DSEL	Control bits for 3rd PLL's dsel input. Selects DCLK divide ratio
11 (R/W)	DDRCLK_FROM_3RDPLL	3rd PLL DCLK Output is connected to DMC block when this bit is set. When cleared, DDR clock is CLKO[3] output of CDU. 3rd PLL DCLK Output is connected to DMC block when this bit is set. When cleared, DDR clock is CLKO[3] output of CDU
10:4 (R/W)	MSEL	Control bits for 3rd PLL's msel input. The `MISCREG_PLL2_CONTROL.MSEL` selects the multiplier in the PLLCLK equation: PLLCLK frequency = (SYS_CLKIN frequency / (DF+1)) * MSEL * 2 Where the value of MSEL is between 1 and 127.

Table 16-80: MISCREG_PLL2_CONTROL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/W)	DSEL_DIV_CHG	Control bit for 3rd PLL's dsel_chg and div_chg inputs. Signal from PCU indicating a DDR clock frequency change
1 (R/W)	MSELDF_CHG	Control bit for 3rd PLL's msel_chg input. Signal from PLL Control Unit indicating a frequency change resulting from a change in either the DF or MSEL bits.
0 (R/W)	BYPASSB	Control bit for 3rd PLL's bypassb input. Active-low control bit to bypass the PLL which forces CLKIN on all output clocks

SH0 Prefetch Range Selection Register

In the MISCREG_SH0_PFB_RANGE_SELECT register, each bit field controls if the corresponding address range will be prefetched. When an address range is set it is prefetched and when it is zero there is no prefetch.

The MISCREG_SH0_PFB_RANGE_SELECT[31:16] bits control the range selection for the instruction cache, and MISCREG_SH0_PFB_RANGE_SELECT[15:0] bits control the range selection for the data cache.

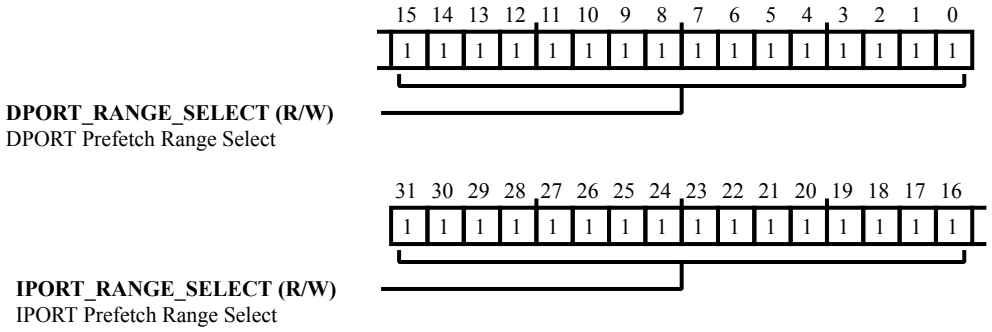


Figure 16-62: MISCREG_SH0_PFB_RANGE_SELECT Register Diagram

Table 16-81: MISCREG_SH0_PFB_RANGE_SELECT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	IPORT_RANGE_SELECT	IPORT Prefetch Range Select. Prefetch Range Selection for IPORT
15:0 (R/W)	DPORT_RANGE_SELECT	DPORT Prefetch Range Select. Prefetch Range Selection for DPORT

SH1 Prefetch Range Selection Register

Each bit field controls if the corresponding address range will be prefetched. When an address range is set it is prefetched an when it is zero there is no prefetch.

The MISCREG_SH1_PFB_RANGE_SELECT[31:16] bits control the range selection for the instruction cache, and MISCREG_SH1_PFB_RANGE_SELECT[15:0] bits control the range selection for the data cache.

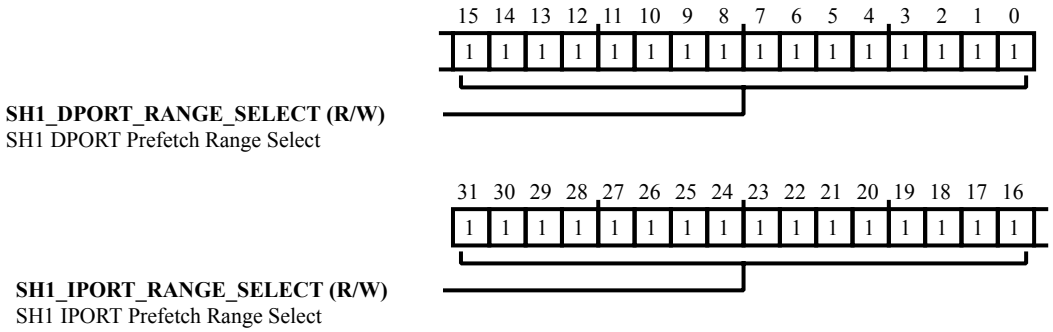


Figure 16-63: MISCREG_SH1_PFB_RANGE_SELECT Register Diagram

Table 16-82: MISCREG_SH1_PFB_RANGE_SELECT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	SH1_IPORT_RANGE_SELECT	SH1 IPORT Prefetch Range Select. Prefetch Range Selection for IPORT
15:0 (R/W)	SH1_DPORT_RANGE_SELECT	SH1 DPORT Prefetch Range Select. Prefetch Range Selection for DPORT

SPORT Direct Interrupt Enable for SH0 and SH1

The bits in the MISCREG_SHARC_BRIDGE_REMAP register control the direct connection of SPORT interrupts to the SHARC core.

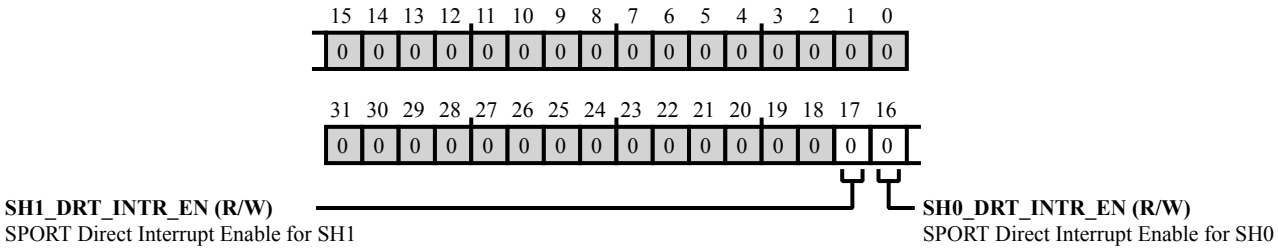


Figure 16-64: MISCREG_SHARC_BRIDGE_REMAP Register Diagram

Table 16-83: MISCREG_SHARC_BRIDGE_REMAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	SH1_DRT_INTR_EN	SPORT Direct Interrupt Enable for SH1. If the MISCREG_SHARC_BRIDGE_REMAP.SH1_DRT_INTR_EN bit is set a direct connection is enabled for SH1. Do not set the MISCREG_SHARC_BRIDGE_REMAP.SH1_DRT_INTR_EN bit when IIR1 of the SH1 is in use. If this bit is zero the direct connection is disabled.
16 (R/W)	SH0_DRT_INTR_EN	SPORT Direct Interrupt Enable for SH0. If the MISCREG_SHARC_BRIDGE_REMAP.SH0_DRT_INTR_EN bit is set a direct connection is enabled for SH0. Do not set the MISCREG_SHARC_BRIDGE_REMAP.SH0_DRT_INTR_EN bit when IIR1 of the SH0 is in use. If this bit is zero the direct connection is disabled.

17 Watchdog Timer (WDOG)

The processor has a 32-bit watchdog timer that can be used to verify system reliability by generating an event to the processor core when the watchdog expires before software updates it.

WDOG Features

The watchdog timer has the following features:

- Programmable 32-bit watchdog count value
- 8-bit disable bit pattern
- General-purpose core event generation

The watchdog timer can supervise system software stability by periodically reloading it to prevent expiration of the downward-counting timer (such that the count never becomes 0). When used in this fashion, an expiring timer can indicate the system software is not running normally.

Expiration of the WDOG counter generates a general-purpose interrupt, which can be used in a variety of ways:

- as an interrupt vector sent through the System Event Controller (SEC) to the core to be serviced by a handler function, providing full software control of device resources (for example, GPIO management and reset control).
- as a fault condition through the SEC to provide hardware-automated:
    - signaling of the fault condition on external pins to the system,
    - system reset requests to the Reset Control Unit (RCU), and/or
    - trigger outputs (SEC0_FAULT trigger generator) through the Trigger Routing Unit (TRU) to initiate activities in a variety of potential trigger receiver (for example, GPIO control).

To help debugging, the watchdog timer does not decrement (even when enabled) if the processor is in emulation mode.

WDOG Functional Description

When enabled, the 32-bit watchdog timer counts downward every SCLK0_0 cycle. If it reaches zero, the watchdog expiration event is generated, which can be used in many ways.

To start the watchdog timer:

1. Program the watchdog timeout period (in SCLK0_0 cycles) in the WDOG_CNT register. With the watchdog disabled, this write also preloads the WDOG_STAT register.
2. Enable the watchdog timer by writing any value other than 0xAD to the WDOG_CTL.WDEN field.

Once the watchdog is enabled, writes to the WDOG_CNT register are ignored. The counter begins decrementing, and the current counter value can be read from the 32-bit WDOG_STAT register at any time.

To prevent the counter from expiring, software must "kick" the watchdog by writing any value to the WDOG_STAT register while the current count is non-zero. While the value written is irrelevant and ignored, this action resets the current counter in the WDOG_STAT register to the programmed WDOG_CNT value, and decrementing continues. The internal counter continues decrementing until it reaches zero, at which point the expiration event is generated, and the WDOG_CTL.WDRO rollover bit is set.

Watchdog operation continues in this manner unless disabled by explicitly writing 0xAD to the WDOG_CTL.WDEN field.

The watchdog expiration event can be used in a variety of ways, as listed below.

- The watchdog expiration event itself is one of numerous interrupt sources that is managed by the System Event Controller. Like other peripheral sources, this event can be used to cause a vector to a handler function that executes based on interrupt priority.
- The watchdog expiration event can be used to initiate automated hardware response through the SEC Fault Interface (SFI).

For this, the WDOG expiry must be configured as the fault source in the SEC. Then the response to the WDOG expiry can be set to any of the following actions:

- Signaling through the external fault pin.
- System reset.
- Trigger outputs to numerous potential trigger completers.

For further details on how watchdog expiration event can be used with the SFI, see Configuring the WDOG Expiry Event to Issue a System Reset.

ADSP-SC59x WDOG Register List

The Watchdog Timer unit (WDOG) provides a software-based watchdog timer that can improve system reliability by generating an event to the processor core if the watchdog expires before being updated by software. A set of registers governs WDOG operations. For more information on WDOG functionality, see the WDOG register descriptions.

Table 17-1: ADSP-SC59x WDOG Register List

Name	Description
WDOG_CNT	Count Register
WDOG_CTL	Control Register
WDOG_STAT	Watchdog Timer Status Register
WDOG_WIN	Watchdog Timer Window Register

ADSP-SC59x WDOG Interrupt List

Table 17-2: ADSP-SC59x WDOG Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
3	WDOG0_EXP	WDOG0 Expiration	Level	
4	WDOG1_EXP	WDOG1 Expiration	Level	
5	WDOG2_EXP	WDOG2 Expiration	Level	

ADSP-SC59x WDOG Trigger List

Table 17-3: ADSP-SC59x WDOG Trigger List Generators

Trigger ID	Name	Description	Sensitivity
167	WDOG0_EXP	WDOG0 Expiration	Level
168	WDOG1_EXP	WDOG1 Expiration	Level
169	WDOG2_EXP	WDOG2 Expiration	Level

Table 17-4: ADSP-SC59x WDOG Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

WDOG Block Diagram

The ***Watchdog Timer Block Diagram*** figure shows the detailed block diagram for the watchdog timer.

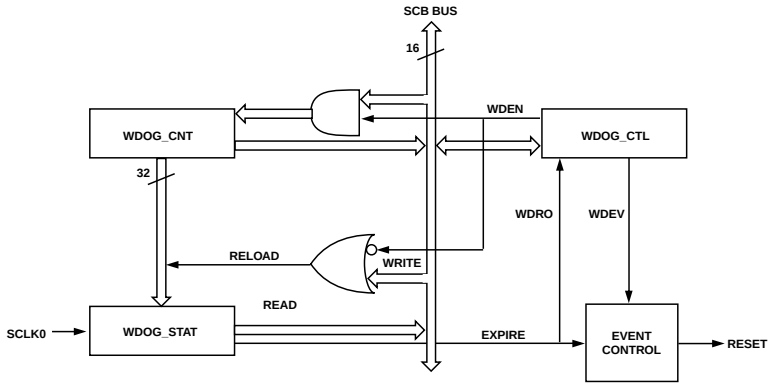


Figure 17-1: Watchdog Timer Block Diagram

Internal Interface

The system clock (SCLK0) clocks the watchdog timer. The registers are accessed through the peripheral MMR access bus. 32-bit read/write operations always access the 32-bit `WDOG_CNT` and `WDOG_STAT` registers. Hardware ensures that those accesses are atomic. When the counter expires, the WDOG expiration event is generated.

External Interface

The watchdog timer does not directly interact with any external pins.

ADSP-SC59x WDOG Register Descriptions

Watchdog Timer Unit (WDOG) contains the following registers.

Table 17-5: ADSP-SC59x WDOG Register List

Name	Description
`WDOG_CNT`	Count Register
`WDOG_CTL`	Control Register
`WDOG_STAT`	Watchdog Timer Status Register
`WDOG_WIN`	Watchdog Timer Window Register

Count Register

The WDOG_CNT register holds the programmable, unsigned count value. A valid write to this register also pre-loads the WDOG counter. For added safety, the WDOG_CNT register can be updated only when the WDOG timer is disabled. A write to the WDOG_CNT register while the timer is enabled does not modify the contents of this register. This register must be accessed with 32-bit read/writes only.

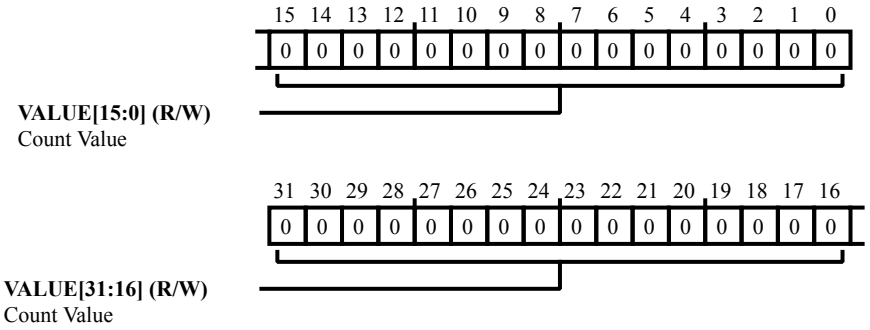


Figure 17-2: WDOG_CNT Register Diagram

Table 17-6: WDOG_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Count Value. The WDOG_CNT.VALUE bit field holds the programmable, unsigned count value.

Control Register

The WDOG_CTL register controls the watchdog timer. This register supports enabling/disabling the watchdog timer and supports checking the timer rollover status. Note that when the processor is in emulation mode, the watchdog timer counter will not decrement even if it is enabled.

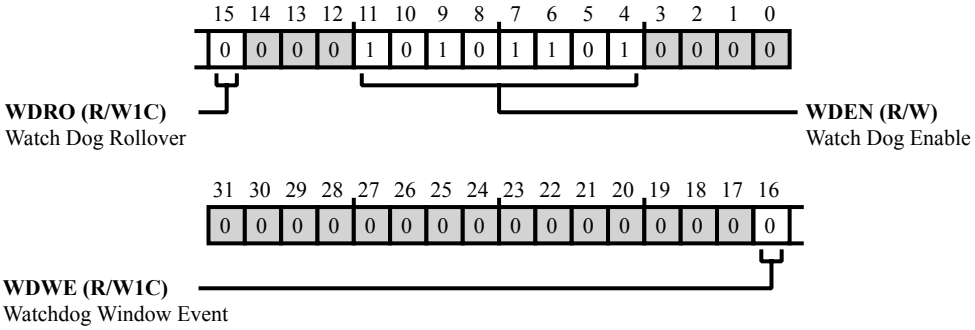


Figure 17-3: WDOG_CTL Register Diagram

Table 17-7: WDOG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W1C)	WDWE	Watchdog Window Event. Software can determine whether the timer has been serviced unexpectedly early by interrogating the WDOG_CTL.WDWE status bit. This is a sticky bit that is set whenever the watchdog is serviced and the WDOG_STAT value is greater than the WDOG_WIN value. It is cleared only by disabling the watchdog timer and by writing a 1 to the bit.	
		0	Window Event has not occurred
		1	Window Event has occurred
15 (R/W1C)	WDRO	Watch Dog Rollover. Software can determine whether the timer has rolled over by interrogating the WDOG_CTL.WDRO status bit. This is a sticky bit that is set whenever the watch dog timer count reaches 0 and cleared only by disabling the watch dog timer and then writing a 1 to the bit.	
		0	WDT Has Not Expired
		1	WDT Has Expired
11:4 (R/W)	WDEN	Watch Dog Enable. The WDOG_CTL.WDEN field is used to enable and disable the watchdog timer. Writing any value other than the disable value into this field enables the watchdog timer. This multi-bit disable key minimizes the chance of inadvertently disabling the watchdog timer.	
		173	Counter Disabled. All other values mean that the counter is enabled.

Watchdog Timer Status Register

The WDOG_STAT register contains the current count value of the watchdog timer. Reads of this register return the current count value. When the watchdog timer is enabled, the WDOG_STAT register is decremented by 1 on each SCLK0 cycle. When the count value reaches 0, the watchdog timer stops counting, and the expiry event is generated. The WDOG_STAT register is a 32-bit unsigned system MMR that must be accessed with 32-bit reads and writes.

Values cannot be stored directly in this register but are instead copied from the WDOG_CNT register. This copy process can happen in two ways:

- While the watchdog timer is disabled, writing the WDOG_CNT register pre-loads the WDOG_STAT register.
- While the watchdog timer is enabled, writing the WDOG_STAT register loads it with the value in the WDOG_CNT register.
- While the watchdog timer is disabled, writing to the WDOG_STAT register also reloads it with the value in the WDOG_CNT register.

When the processor executes a write (of an arbitrary value) to the WDOG_STAT register, the value in the WDOG_CNT register is copied into the WDOG_STAT register. Typically, software sets the value of WDOG_CNT at initialization, then periodically writes to the WDOG_STAT register before the watchdog timer expires. This reloads the watchdog timer with the value from WDOG_CNT and prevents generation of the expiry event.

If the program does not reload the counter before SCLK0 x count register cycles, an expiry event is generated, and the WDOG_CTL.WDRO bit is set. When this happens, the counter stops decrementing and remains at zero. If the counter is enabled with a zero loaded to the counter, the WDOG_CTL.WDRO bit is set immediately and the counter remains at zero and does not decrement.

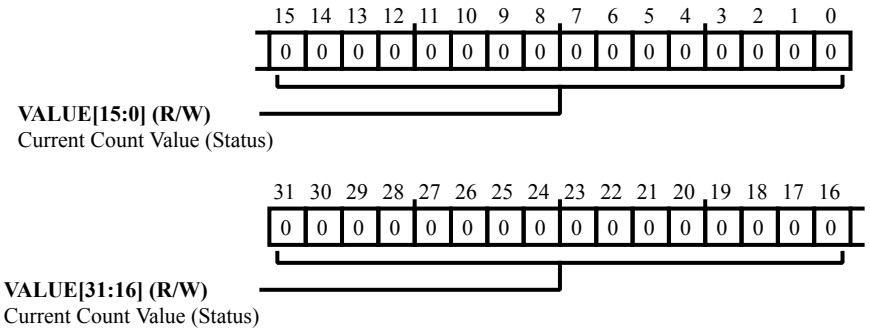


Figure 17-4: WDOG_STAT Register Diagram

Table 17-8: WDOG_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Current Count Value (Status). The `WDOG_STAT.VALUE` bit field contains the current count value of the watchdog timer.

Watchdog Timer Window Register

Watchdog window register holds the unsigned window value programmed. The window register can be programmed while the WDOG is disabled; any write to the register when WDOG is enabled doesn't alter the contents of the register. When the WDOG is enabled and the core services the WDOG by doing a write to `WDOG_STAT` register while the `WDOG_STAT` value is greater than `WDOG_WIN`, the WDWE event is generated and also the counter stops decrementing.

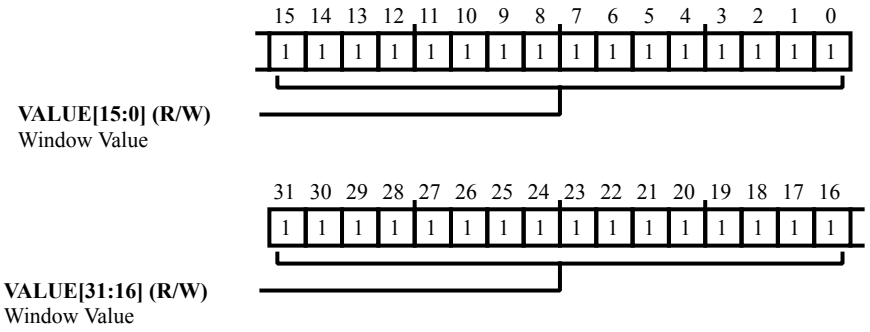


Figure 17-5: WDOG_WIN Register Diagram

Table 17-9: WDOG_WIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Window Value. The `WDOG_WIN.VALUE` bit field contains the unsigned window value.

18 Enhanced Mobile Storage Interface (eMSI)

The eMSI is a high performance, highly configurable and programmable, eMMC (embedded Multi-Media Card) and SD (secure digital) host controller. The eMSI is required to communicate with SD cards and eMMC devices targeted for the mobile/portable markets. The device memory to which the eMSI sends or receives data is typically a device for NAND flash storage. The eMSI is optimized for portable electronic devices and high-speed storage applications (for example embedded applications).

eMSI Features

The eMSI includes the following features.

General Features

- Supports eMMC protocols including JEDEC eMMC 5.1 specification
- Supported eMMC modes
  - Legacy (0-26 MHz SDR – 1-bit, 4-bit, 8-bit support)
  - HS SDR (0-50 MHz SDR –1-bit, 4-bit, 8-bit support)
  - HS DDR (0-50 MHz DDR – 4-bit, 8-bit support)
  - Supports boot operation and alternative boot operation
- Supports SD memory (SDSC, SDHC, SDXC) and wake up interrupt for SD cards
- Supported SD modes
  - Default Speed/DS (0-25 MHz SDR – 1-bit, 4-bit support)
  - High Speed/HS (0-44.4 MHz SDR – 1-bit, 4-bit support)
- Supports an internal DMA with the following data transfer types for eMMC modes:
  - SDMA (Single operation DMA)
  - ADMA2 (Advance DMA 2)

  - ADMA3 (Advance DMA 3)
- Maximum block size of 1024 bytes supported
- FIFO size is 2048 bytes (parity enabled)
- Supports Command Queuing Engine (CQE) and compliant with eMMC CQ HCI
- Programmable scheduler algorithm selection of task execution. Supports data prefetch for back-to-back write operations
- 10 MHz CQE timer clock

Supported Standards Compliance

- JEDEC eMMC 5.1 Specification - JESD84-B51, February 2015
- SD Specifications Part A2 SD Host Controller Standard Specification Version 4.20, August 2015
- SD Specifications Part A2 Host Controller Version 4.20 Supplementary Notes Version 1.00 Draft 0.60 December 2015
- SD Specifications Part 1 Physical layer Specification Version 6.00 February 2017

eMSI Functional Description

The following sections describe the function of the eMSI controller.

ADSP-SC59x EMSI Register List

The enhanced mobile storage interface (eMSI) is a high speed JEDEC eMMC 5.1 complaint controller. The eMSI controller can connect to single SD card or eMMC device at a time. A set of registers governs eMSI controller operations. For more information on eMSI functionality, see the eMSI controller Register Descriptions section.

Table 18-1: ADSP-SC59x EMSI Register List

Name	Description
EMSI_ADMA_ADDR_LO	ADMA System Address Register - Low
EMSI_ADMA_DESADDR_LO	ADMA3 Integrated Descriptor Address Register - Low
EMSI_ADMA_ERR_STAT	ADMA Error Status Register
EMSI_ARG	Argument Register
EMSI_AUTOCMD_STAT	Auto CMD Status Register
EMSI_BASEADDR0	Base Address 0 Register
EMSI_BASEADDR1	Base Address 1 Register
EMSI_BLKCNT	16-bit Block Count Register
EMSI_BLKSZ	Block Size Register

Table 18-1: ADSP-SC59x EMSI Register List (Continued)

Name	Description
EMSI_BOOT_CTL	eMMC Boot Control Register
EMSI_CAP1	Capabilities 1 Register - 0 to 31
EMSI_CAP2	Capabilities Register - 32 to 63
EMSI_CLK_CTL	Clock Control Register
EMSI_CMD	Command Register
EMSI_CQVER	Command Queuing Version Register
EMSI_CQ_CAP	Command Queuing Capabilities Register
EMSI_CQ_CFG	Command Queuing Configuration Register
EMSI_CQ_CRARG	CQ Command Response Argument Register
EMSI_CQ_CRDCT	Command Response for Direct Command Register
EMSI_CQ_CRI	CQ Command Response Index
EMSI_CQ_CTL	Command Queuing Control Register
EMSI_CQ_DPT	Device Pending Tasks Register
EMSI_CQ_DQSTAT	Device Queue Status Register
EMSI_CQ_IC	Command Queuing Interrupt Coalescing Register
EMSI_CQ_ISTAT	Command Queuing Interrupt Status Register
EMSI_CQ_ISTAT_EN	Command Queuing Interrupt Status Enable Register
EMSI_CQ_ISTAT_INTEN	Command Queuing Interrupt Signal Enable Register
EMSI_CQ_RMEM	Command Response Mode Error Mask Register
EMSI_CQ_SSCFG1	CQ Send Status Configuration 1 Register
EMSI_CQ_SSCFG2	CQ Send Status Configuration 2 Register
EMSI_CQ_TCLR	Command Queuing Task Clear Register
EMSI_CQ_TCN	Command Queuing Task Completion Notification Register
EMSI_CQ_TDBR	Command Queuing DoorBell Register
EMSI_CQ_TDL_BADDR	Command Queuing Task Descriptor List Base Address Register
EMSI_CQ_TERRINFO	CQ Task Error Information Register
EMSI_CTL1	EMSI Control 1 Register
EMSI_CTL2	EMSI Control 2 Register
EMSI_HOST_CNTRL_VERS	SD Host Controller Specification Version
EMSI_EMMC_CTL	eMMC Control Register
EMSI_ERR_STAT	Error Interrupt Status Register

Table 18-1: ADSP-SC59x EMSI Register List (Continued)

Name	Description
EMSI_ERR_STAT_EN	Error Interrupt Status Enable Register
EMSI_ERR_STAT_INTEN	Error Interrupt Signal Enable Register
EMSI_FRC_AUTOCMDSTAT	Force Event Register for Auto CMD Error Status Register
EMSI_FRC_ERRSTAT	Force Event Register for Error Interrupt Status
EMSI_ISTAT	Interrupt Status Register
EMSI_ISTAT_EN	Interrupt Status Enable Register
EMSI_ISTAT_INTEN	Interrupt Signal Enable Register
EMSI_PRESET_DS	Preset Value for Default Speed
EMSI_PRESET_HSDDR	Preset Value for HSDDR
EMSI_PRESET_HSSDR	Preset Value for HSSDR
EMSI_PRESET_INIT	Preset Value for Initialization
EMSI_PRESET_LEGACY	Preset Value for Legacy Mode
EMSI_PSTATE	Present State Register
EMSI_RESP0	Response Register 0
EMSI_RESP1	Response Register 1
EMSI_RESP2	Response Register 2
EMSI_RESP3	Response Register 3
EMSI_SDMA_ADDR	SDMA System Address Register
EMSI_SWRST	Software Reset Register
EMSI_TO_CTL	Timeout Control Register
EMSI_TRNSFRMODE	Transfer Mode Register
EMSI_VER_ID	EMSI Version
EMSI_VER_TYPE	EMSI Version Type
EMSI_WU_CTL	Wakeup Control Register

ADSP-SC59x EMSI Interrupt List

Table 18-2: ADSP-SC59x EMSI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
237	EMSI0_STAT	EMSI0 Status Interrupt	Level	
239	EMSI0_Wakeup	EMSI0 Wakeup Interrupt	Level	

ADSP-SC59x EMSI Trigger List

Table 18-3: ADSP-SC59x EMSI Trigger List Generators

Trigger ID	Name	Description	Sensitivity
None			

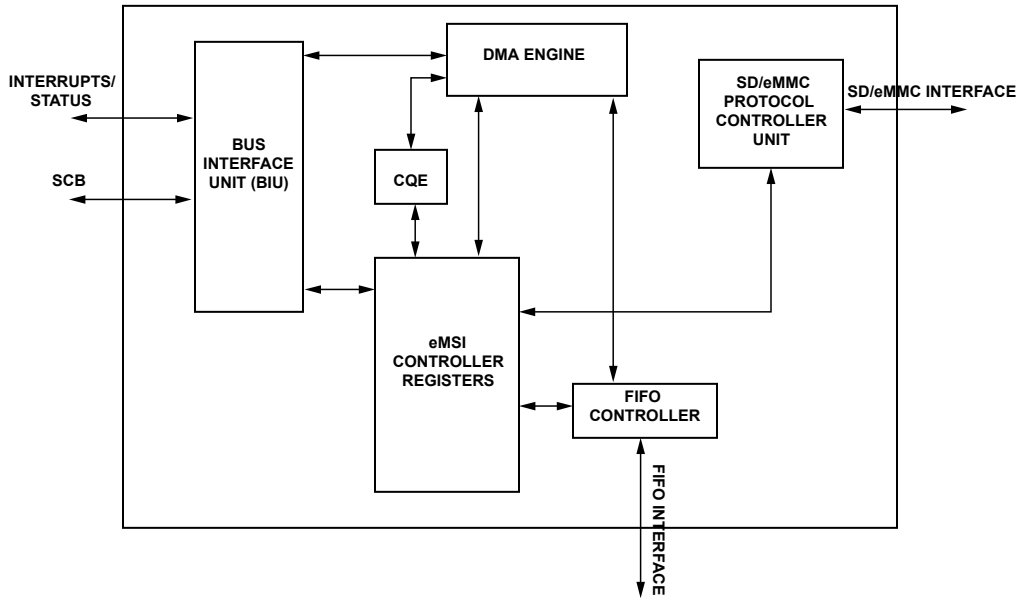
Table 18-4: ADSP-SC59x EMSI Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

eMSI Block Diagram

The eMSI Block Diagram shows the main functional blocks within the eMSI. These blocks are described in more detail in eMSI Architectural Concepts.

Figure 18-1: eMSI Block Diagram



NOTE: The card-detect and write-protect signals are from the SD card socket and not from the SD card.

eMSI Architectural Concepts

The following sections describe the functions and features of the eMSI controller as well as the eMMC and SD protocols.

The eMSI controller communicates with the device using a message-based bus protocol in which the eMSI controller sends commands serially using the EMSI_CMD signal. Some commands require response from the eMMC device. This response is also sent serially on the EMSI_CMD signal.

Data transfers, both to and from the eMMC device, occur using the data signals. The number of data lines used for the data transfer can be configured to 1 (EMSI_D0) bit, 4 (EMSI_D3–EMSI_D0) bit, or 8 (EMSI_D7–EMSI_D0) bit. All EMSI_CMD and EMSI_D[n] transfers are synchronous with EMSI_CLK. Cyclic redundancy codes (CRC) are used to protect commands, responses, and data transfers from transmission errors. A CRC7 code is generated for every command sent by the eMSI and every response returned by the eMMC device on the EMSI_CMD signal.

Bus Interface Unit (BIU)

The Bus Interface Unit (BIU) implements the logic to transfer data to SCB (system crossbars). The SCB transfers data to and from the system memory through the SCB, respectively. This unit has logic to primarily access the eMSI controller registers by using an SCB and MMRG fabric. This module supports only the little-endian scheme for register accesses.

DMA Engine

The DMA engine unit handles data transfer between controller and system memory. The key features of this unit are as follows.

- Supports SDMA/ADMA2/ADMA3 modes based on the configuration.
- Fetching the descriptor and data.
- Write back the received data packets to the system memory.

eMSI Controller Registers

The eMSI controller register unit is comprised of the standard SD host controller registers as specified in the SD Specifications Part A2 SD Host Controller Standard Specification Version 4.20. It also includes command queuing registers that are compliant to the JEDEC eMMC 5.1 specification.

The host ensures the clock supply to the controller. The clock supply is required for accessing the registers in eMSI controller.

FIFO Controller

The FIFO controller interfaces the packet buffer (FIFO) of the eMSI controller and the SD/eMMC protocol controller unit (SD/eMMC unit). The buffer (FIFO) depth is 2048 bytes. The FIFO stores data along with parity bits generated by the MEPU. The parity error is handled by the MEC.

SD/eMMC Protocol Controller Unit

The SD/eMMC protocol controller unit is responsible for handling SD and eMMC interface protocols. The key features of this unit are:

- Generate SD/eMMC command and data packets
- Generate CRC and check for command and data packets
- Handle packet timeouts
- Support 1-bit, 4-bit, and 8-bit bus width for eMMC data transfer
- Support 1-bit and 4-bit bus width for SD card data transfer

Command Queuing Engine (CQE)

The Command Queueing Engine (CQE) denotes the hardware unit executing the Command Queueing (CQ) activities. The CQE manages the interface between the host software and the eMMC device as well as the data transfers.

Generating SD/eMMC Commands

The eMSI controller generates SD/eMMC commands and sends a command to the eMSI bus (this includes the internally generated CRC7). When initiating a transaction using SDMA (Single Operation DMA) generated transfers, ADMA (Advanced DMA) generated transfers, and non-data transfers, the host driver programs registers sequentially from `EMSI_SDMA_ADDR` to `EMSI_CMD`. The beginning register may be chosen based on the type of transaction. Table 18-5 Register Programming Sequence Before Issuing Commands shows the register programming sequence needed before issuing the commands. The last written register is always `EMSI_CMD` because writing to the upper byte of this register triggers the issuance of an SD/eMMC command.

The host driver should not read the SDMA system address, block size and block count registers during a data transaction unless the transfer is stopped because the value is changing and not stable.

To prevent an overwrite of registers while issuing data transfer based commands, the 32-bit block count (`EMSI_SDMA_ADDR`), block size (`EMSI_BLKSZ`), 16-bit block count (`EMSI_BLKCNT`) and transfer mode (`EMSI_TRNSFRMODE`) registers is write protected by the eMSI controller, while command inhibit (`EMSI_PSTATE.CMD_INHIBIT_DAT`) is set to 1. (When host version 4 enable = 0, the SDMA system address is not protected by this signal.) The host driver must not write the argument and command registers while `EMSI_PSTATE.CMD_INHIBIT_DAT` is set to 1.

Table 18-5: Register Programming Sequence Before Issuing Commands

Command Type	Register to program before issuing the commands in sequence
Non Data transfer based commands	`EMSI_ARG`, `EMSI_CMD`
Data transfer based commands	`EMSI_SDMA_ADDR`, `EMSI_BLKSZ`, `EMSI_BLKCNT`, `EMSI_ARG`, `EMSI_TRNSFRMODE`, `EMSI_CMD`

Block Count

The block count parameter used by the eMSI controller to determine the total data length of the data transfer. It does this by multiplying the block length (block size). The data transfer length set to the SD/eMMC data transfer must be equivalent to data transfer length sets to ADMA2 and ADMA3 descriptors (in predefined data transfers).

Selection of 16-Bit or 32-bit Block Count

Due to the increased capacity of eMMC devices, the data that is available to transfer to and from these devices has also increased. To allow for this increase the eMSI controller extends block counts from 16-bit to 32-bit for all operations in SD/eMMC mode, SDMA and ADMA.

Because SDMA may use the ADMA system address register (EMSI_ADMA_ADDR_LO) to support 32-bit addressing, the SDMA system address register (EMSI_SDMA_ADDR) is re-assigned to the 32-bit block count register. Selection of the block count registers is either 16-bit (EMSI_BLKCNT) or 32-bit block count (EMSI_SDMA_ADDR) and is defined as follows, which allows mixed use of 16-bit or 32-bit block count.

- If the Host Version 4 Enable bit (EMSI_CTL2.HOST_VER4_EN) is cleared to 0 or the 16-bit EMSI_BLKCNT register is configured to non-zero, the 16-bit EMSI_BLKCNT register is selected.
- If Host Version 4 Enable is set to 1 and 16-bit EMSI_BLKCNT register is configured to 0000h, the 32-bit block count register is selected.

Use of block count is enabled by setting the EMSI_TRNSFRMODE.BLOCK_COUNT_EN bit during data transfers.

Preset Value Registers

As the operating eMSI bus clock frequency depends on the system implementation, it is difficult to determine the frequency parameters in using the standard host driver. When the preset value enable is set using the EMSI_CTL2.PRESET_VAL_EN bit, automatic eMSI bus clock frequency generation is performed without considering frequency dividers set by the host driver (EMSI_CLK_CTL.FREQ_SEL/ EMSI_CLK_CTL.UPPER_FREQ_SEL). The eMSI controller automatically selects the eMSI bus clock frequency as specified in the preset value (as per frequency dividers values set in preset value registers) depending on the selected bus speed mode (EMSI_CTL2.EMMC_MODE_SEL). For example, if the program selects the eMMC legacy speed mode then the eMSI controller chooses dividers from the EMSI_PRESET_LEGACY.FREQ_SEL bit field and the same dividers are set in the EMSI_CLK_CTL.FREQ_SEL bit field. If the bit is cleared to 0, the eMSI bus clock frequency is selected by the host driver frequency dividers that are set by the host driver using the EMSI_CLK_CTL.FREQ_SEL/EMSI_CLK_CTL.UPPER_FREQ_SEL bit field.

A preset value for initialization (EMSI_PRESET_INIT) is not selected by the bus speed mode. Before starting the initialization sequence, the host driver needs to set a clock preset value to EMSI_CLK_CTL.FREQ_SEL/ EMSI_CLK_CTL.UPPER_FREQ_SEL by reading the contents of the EMSI_PRESET_INIT register. Preset value enable (EMSI_CTL2.PRESET_VAL_EN) can be set after initialization is completed.

Automatic Command Generation (Auto CMD)

The eMSI controller internally generates Auto CMD12 (STOP_TRANSMISSION) or Auto CMD23 (SET_BLOCK_COUNT) commands as per configuration in the EMSI_TRNSFRMODE.AUTO_CMD_EN bit field. Automatic commands generation improve the performance as it eliminates the process of manually writing registers.

Auto CMD23

Auto CMD23 feature is useful in predefined multiple block operations. A block count is configured in the argument of CMD23 to specify a transfer length of following CMD18 (READ_MULTIPLE_BLOCK) or CMD25 (WRITE_MULTIPLE_BLOCK) commands. Auto CMD23 is a feature that automatically issues a CMD23 before a CMD18 or CMD25 is sent to the SD cards/eMMC device (when `EMSI_TRNSFRMODE.AUTO_CMD_EN` = 0x2). This avoids performance deterioration during memory accesses by removing the interrupt service of CMD23. The `EMSI_SDMA_ADDR` register is assigned to the 32-bit block count register for CMD23.

NOTE: CMD23 is supported in version 3.00 and above SD cards.

Auto CMD12

Open ended multiple block transfers for SD cards/eMMC require CMD12 (STOP_TRANSMISSION) command to stop the data transactions. The eMSI controller automatically issues CMD12 when the last block transfer is completed. The host driver can enable this command using the `EMSI_TRNSFRMODE.AUTO_CMD_EN` bit = 0x1 when issuing multiple block transfer (CMD18 and CMD25) commands. Auto CMD12 timing synchronization with the last data block is accomplished by hardware in the eMSI controller. Auto CMD12 is useful when block count is finite(>0, block count is enabled) and CMD23 is not used (for example in open ended transfers).

Data Transfer Modes

The eMSI controller supports data transfer using Direct Memory Access (DMA). There are three types pf DMA operations available: SDMA, ADMA2 and ADMA3.

Direct Memory Access (DMA)

The Direct Memory Access (DMA) transfers data from the host memory to the device port and conversely. The controller uses a descriptor (except in SDMA transfers) to move data efficiently from source to destination with minimal core intervention. Programs can configure the controller to interrupt the core in situations such as data transmit and receive transfer completion from the eMMC device, as well as other normal or error conditions. The DMA transfers the data received from the eMMC device to the data buffer in the host memory, and it transfers transmit data from the data buffer in the host memory to the eMSI FIFO. Descriptors that reside in the memory act as pointers to these buffers. DMA supports both single block and multi-block transfers.

Non Descriptor (Single Operation) DMA

SDMA (Single Operation DMA) performs a single read/write SD/eMMC command operation at a time. SDMA is suitable for short data transfers because SDMA requires an address update at the page boundary of system memory. A DMA interrupt generated at every page boundary uses core resources to reprogram the new system address. For short data transfers, writing a register is preferred to creating descriptors.

SDMA supports various size page boundaries (from 4 kB to 512 kB, configured using the `EMSI_BLKSZ.SDMA_BUF_BDARY` bit field). A page boundary is the size of a contiguous buffer in system memory. The SDMA transfer waits at every boundary specified by the `EMSI_BLKSZ.SDMA_BUF_BDARY` bit field

and the eMSI controller generates the DMA interrupt to request the host driver to update the SDMA system address register.

Descriptor (Advanced) DMA

A long data transfer should use ADMA (Advanced DMA) to avoid performance bottlenecks caused when an interrupt occurs at every page boundary. ADMA2 and ADMA3 adopt the scatter gather DMA algorithm so that higher data transfer speeds are available. The host driver can program a list of data transfers between system memory and an SD card eMMC using the descriptor table. ADMA2 performs one read/write command operation at a time. ADMA3 can program multiple read/write operations in a descriptor table. ADMA3 is used to perform large data transfers.

Advanced DMA 2 (ADMA2)

Figure 18-2 ADAM2 Block Diagram, shows the ADMA2 data transfer between the eMSI controller and the system memory as well as the descriptor table that is created by the host driver (for more information, see ADMA2 Descriptor Format). A 32-bit address descriptor table is used for a system with 32-bit addressing. In the descriptor table, each line contains information about the address, its length (descriptor data transfer size) and an attributes that specifies the operation of the descriptor line. ADMA2 uses the 32-bit ADMA address register (`EMSI_ADMA_ADDR_LO`) for the descriptor pointer. An ADMA2 transfer is triggered by writing to the command register and ADMA2 processes each line in the descriptor table. The descriptor table is created in the system memory by the host driver. ADMA2 fetches one descriptor line and executes it until the end of descriptor denoted by (End = 1 in attribute) is found.

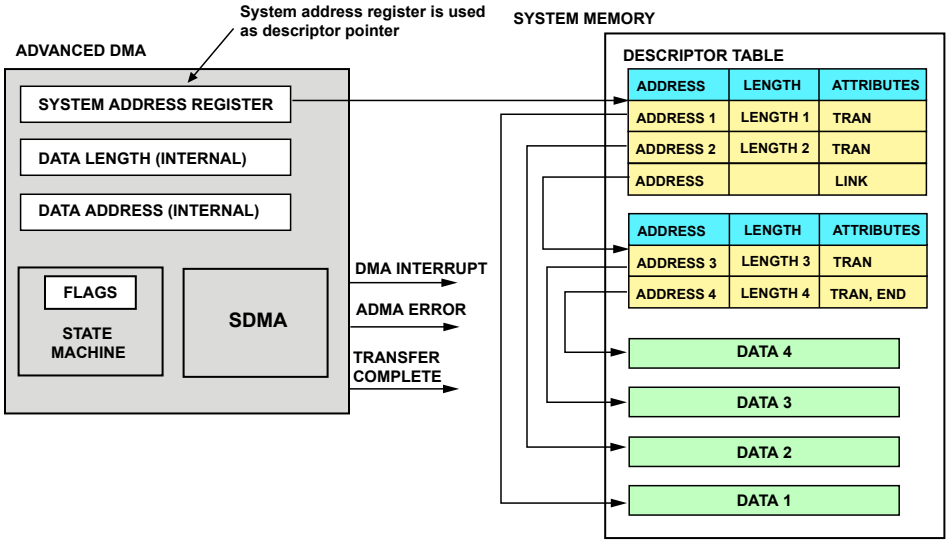


Figure 18-2: ADAM2 Block Diagram

Figure 18-3 ADMA2 Data Transfer Example shows a typical ADMA2 descriptor program. The host driver describes the descriptor table with each slice is placed somewhere in contiguous system memory. The host driver describes the descriptor table that contain the address, length, and attributes. Each sliced data is transferred in turn as programmed in the descriptor.

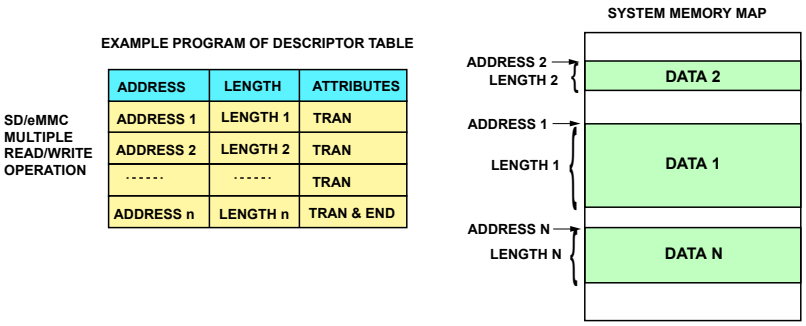


Figure 18-3: ADMA2 Data Transfer Example

Data Address and Data Length Requirements

There are three requirements to program the descriptor:

1. The minimum unit of address is 4 bytes (32-bits).
2. The maximum data length of each descriptor line is less than 64 KB (if data length set to 10-bit) or 64 MB (if data length set to 26-bit).
3. Total Length = Length 1 + Length 2 + Length 3 + ... + Length n = multiple of block size

Using 4 byte units of address simplifies byte enable control on the 32-bit data bus. If the total length of a descriptor were not a multiple of block size, ADMA2 transfers might not be terminated. In this case, a data timeout occurs and the transfer is stopped by an abort command. Therefore the total length should be a multiple of block size.

It is recommended that the total of the data lengths of all descriptors be equal to the total data transfer size where: total data transfer size = block count x block size.

General Descriptor Table Format

Figure 18-4 General Descriptor Table Format shows the general format of a descriptor table. One descriptor line consumes 64 bits (8 bytes) using 32-bit addressing mode. The attribute is used to control the descriptor. Act0 = 0 is assigned to ADMA2 descriptor. Act0 = 1 is assigned to the ADMA3 descriptor and is used to extend descriptors for ADMA3. Refer to Advanced DMA 3 (ADMA3) for more information.

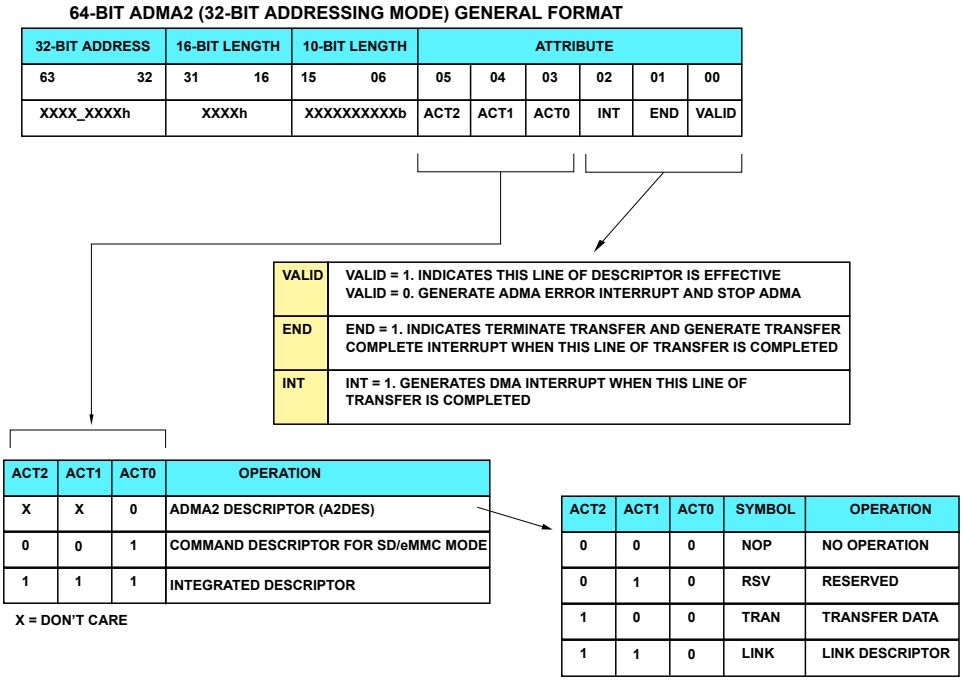


Figure 18-4: General Descriptor Table Format

ADMA2 Descriptor Format

Figure 18-5 ADMA2 Descriptor Table Format shows the ADMA2 descriptor table. Act0 = 0 is assigned to the ADMA2 descriptor. Three action symbols are specified by the combination of Act2 and Act1. A no operation (NOP) skips the current descriptor line and fetches the next one. A Tran operation transfers data designated by address and length field. A Link operation is used to connect separated two descriptors. The address field of link points to next descriptor table. The combination of Act2 = 0 and Act1 = 1 is reserved and defined in the same operation as a NOP. All other combinations are the same as No Operation (NOP).

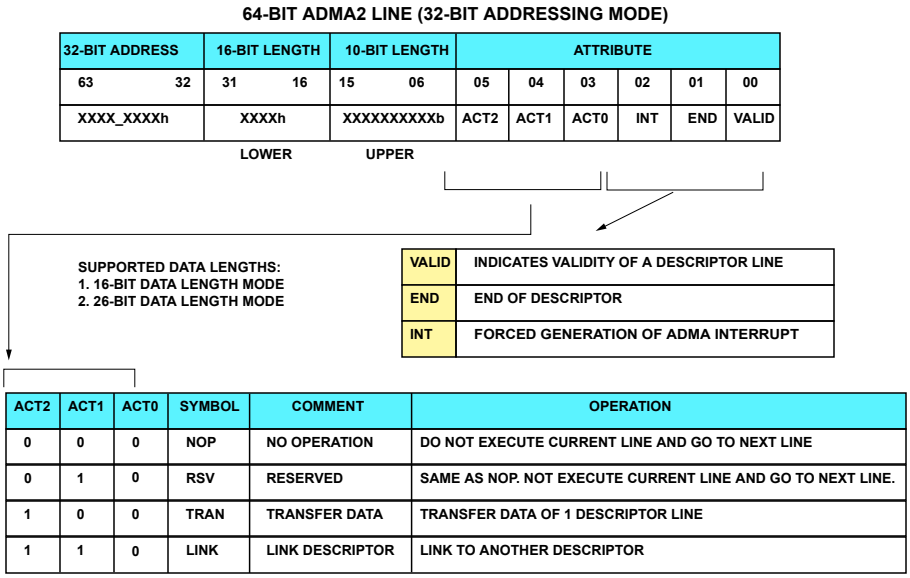


Figure 18-5: ADMA2 Descriptor Table Format

The Int attribute may be set only in an ADMA2 descriptor. The 26-bit data length mode is added to reduce the number of descriptor lines for large continuous data. The following tables shows the definition of 16-bit data length and 26-bit data length. 16-bit data length is selected when the `EMSI_CTL2.ADMA2_LEN_MODE` bit is cleared to 0. 26-bit data length is selected when the `EMSI_CTL2.ADMA2_LEN_MODE` bit is set to 1.

Table 18-6: ADMA2 16-bit Length Mode

16-bit Length (D31-D16)	Value of Length
000h	65536 bytes
0001h	1 byte
0002h	2 bytes
...	...
FFFFh	65535 bytes

Table 18-7: ADMA2 26-bit Length Mode

26-bit Length (D15-D06, D31-D16)	Value of Length
000_0000h	64M bytes
000_0001h	1 byte
000_0002h	2 bytes
000_0003h	3 bytes
...	...
3FF_FFFFh	64M-1 bytes

ADMA2 Error Status Register

An error that occurs during an ADMA2 transfer may stop ADMA2 operation and generate an ADMA error interrupt. The EMSI_ADMA_ERR_STAT.ERR_STATES bit field holds state of ADMA2 stopped.

Advanced DMA 3 (ADMA3)

As described above for ADMA2, SD/eMMC commands are issued when the host driver writes to the eMSI controller registers. For ADMA3, a host program performs multiple ADMA2 operations using command descriptors to issue SD/eMMC commands. A multi-block data transfer between system memory and an SD card/eMMC is programmed by using a pair of command descriptors and an ADMA2 descriptor. ADMA3 performs multiple multi-block data transfers by using an integrated descriptor. Figure 18-6 Example of ADMA3 Operation shows an example ADMA3 operation where three data blocks (data A, data B and data C) are written to different areas of the SD card/eMMC.

An integrated descriptor consists of pointers to command descriptors. Each command descriptor is followed by an ADMA2 descriptor. The first descriptor pair is programed to transfer data A, the second pair data B and the third pair data C. The location of the integrated descriptor is configured using the ADMA3 integrated descriptor address register (EMSI_ADMA_DESADDR_LO). ADMA3 fetches pointers one by one in the integrated descriptor and executes descriptors designated by the pointer. ADMA3 sets the contents of a command descriptor in the eMSI controller registers (EMSI_SDMA_ADDR to EMSI_CMD sequentially) to issue an SD/eMMC command and then executes the ADMA2 descriptor. The first operation transfers data A from system memory to the SD card/eMMC. The second operation transfers data B and the third operation transfers data C. When execution of all descriptors pointed by the integrated descriptor is completed, ADMA3 generates a transfer complete interrupt to inform the host driver that the operation is complete.

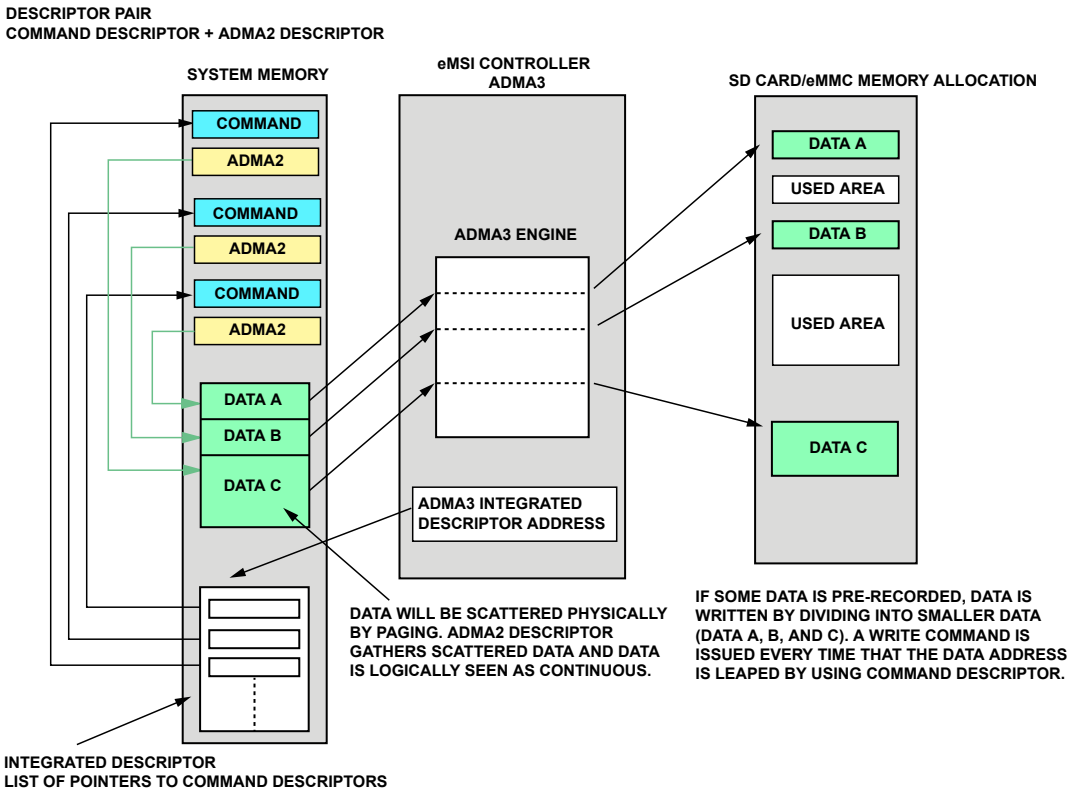


Figure 18-6: Example of ADMA3 Operation

Command Descriptor Format

As shown in Figure 18-7 Command Descriptor Format, 32-bit register data is configured in bits 63-32 of each descriptor line. Command descriptor types are distinguished by Attribute. If Attribute indicates the command descriptor for SD/eMMC mode (Act2-0 = 001b), the 32-bit register fields are written to the eMSI controller registers sequentially from the `EMSI_SDMA_ADDR` to `EMSI_CMD` registers. When the `EMSI_CMD` register is written, an SD/eMMC command is issued. The eMSI controller has a pointer to a descriptor line for the command descriptor and the ADMA2 descriptor. The pointer is incremented after reading every descriptor line. When the last line of the command descriptor is read, the pointer is assumed to point to the top of the ADMA2 descriptor, which is placed just after the command descriptor. The eMSI controller ignores the INT of Attribute in this descriptor.

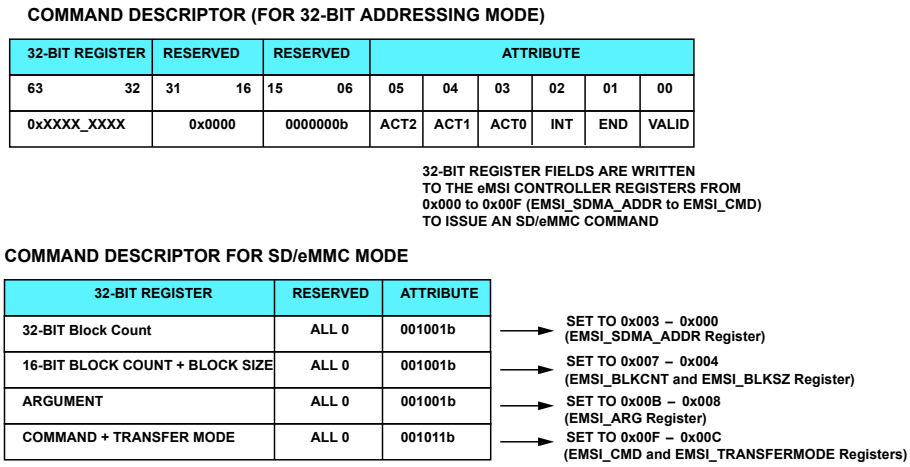


Figure 18-7: Command Descriptor Format

The programing requirements of command descriptors for SD/eMMC mode are:

- Setting infinite data transfer (`EMSI_TRNSFRMODE.BLOCK_COUNT_EN` = 0) is not allowed, except for single block transfers (`EMSI_TRNSFRMODE.MULTI_BLK_SEL` = 0).
- As ADMA3 only supports 32-bit block count mode, the `EMSI_TRNSFRMODE.BLOCK_COUNT_EN` bit must be set to 1 and the 16-bit block count must be set to 0x0000 (`EMSI_BLKCNT`). Clearing stop count (32-bit block count = 0) `EMSI_SDMA_ADDR` is not allowed, except for single block transfers (`EMSI_TRNSFRMODE.MULTI_BLK_SEL` = 0).
- Configuring no data transfer (block size = 0) is not allowed, except when using the no data transfer command in a command descriptor (where CDEnd set to 1).
- When using the memory data transfer command, The Auto CMD Auto Select of Auto CMD Enable (`EMSI_TRNSFRMODE.AUTO_CMD_EN`) must be used so that Auto CMD 23 is sent to the eMMC device if `EMSI_CTL2.CMD23_EN` = 1. For Auto CMD 12, Auto Command Disabled (`EMSI_TRNSFRMODE.AUTO_CMD_EN` = 0) is used.

Integrated Descriptor Format

Figure 18-8 Integrated Descriptor Format shows how multiple of pointers to command descriptors are configured in the integrated descriptor. The pointer of a 32-bit address is configured to bit 63-32 in the 64-bit integrated descriptor. The eMSI controller ignores the INT of Attribute in this descriptor.

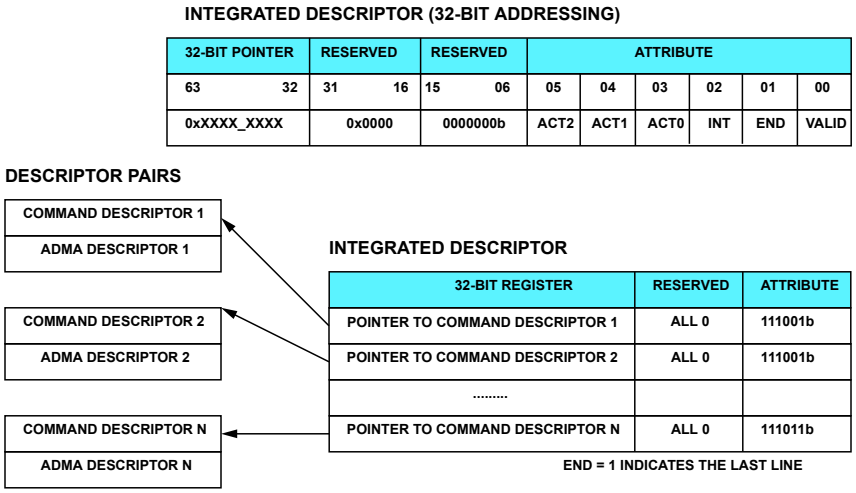


Figure 18-8: Integrated Descriptor Format

The ADMA3 integrated descriptor address register (EMSI_ADMA_DESADDR_LO) is used to designate the location of the integrated descriptor. The start of an ADMA3 is triggered by writing to offset EMSI_ADMA_DESADDR_LO in 32-bit addressing mode.

Response Error Check During ADMA3 Operation

The eMSI controller should perform response checks to prevent performance loss due to the host driver doing this task. When creating the command descriptor, configure the following 3 bits in the transfer mode register to: EMSI_TRNSFRMODE.RESP_INT_DIS = 1, EMSI_TRNSFRMODE.RESP_ERR_CHK_EN = 1. Clear EMSI_TRNSFRMODE.RESP_TYPE = 0 if the command is for an SD/eMMC enabled response check by the eMSI controller.

In SD/eMMC mode, if an error is detected in R1, a response error interrupt is generated in the EMSI_ERR_STAT register. ADMA3 is stopped and the host driver can read the error response from the EMSI_RESP0 register.

Command Queueing Engine

The Command Queueing Engine (CQE) denotes the hardware unit executing the Command Queueing (CQ) activities. The CQE manages the interface between the host software and the eMMC device, and the data transfers. The CQE receives tasks from the software though a Task Descriptor List (TDL) in the host memory and the doorbell register. The CQE issues CQ commands, CMD44 (QUEUED_TASK_PARAMS) and CMD45 (QUEUED_TASK_ADDRESS), to the eMMC device and stores the task information. The CQE also reads the queue status register of the device, decides the task to execute and issues the EXECUTE commands, CMD46 (EXECUTE_READ_TASK) and CMD47 (EXECUTE_WRITE_TASK).

The Command Queuing Engine has the following features:

- Non-blocking issuance of data transfer tasks using a task list in host memory
- Optional notification upon completion via interrupt, with interrupt coalescing

- Direct-Command (DCMD) tasks through which software can send any eMMC command, by its index and argument, to the device, using CQE
- Queue-Barrier (QBR) where software can control the execution order of tasks, and task management and error recovery executed by software, by halting the CQE at known locations

To implement queuing of tasks in the eMMC device, the host asynchronously issues tasks using the TDL, which is in a memory location known to the CQE and contains up to 32 fixed-size slots. Each slot contains one task descriptor and one transfer descriptor. To issue a task, the software selects an available TDL slot and constructs the following two descriptors in it.

- Task Descriptor - encodes all information that defines the task. For instance, the start address, block count, and priority. This information is later passed to the device. The task descriptor can alternatively encode any arbitrary command that is sent directly to the device.
- Transfer Descriptor - points either to one continuous data buffer to/from which data is transferred (TRAN Descriptor) or to a scatter/gather list of any length (LINK descriptor). Figure 18-9 Command Queuing Host Controller Interface (HCI) Structure shows the structure of the TDL in the host memory and the slot numbers in the TDL. The slot numbers in the figure store the following information:
  - 0 – Stores a data transfer task with a TRAN descriptor
  - 1 – Stores a data transfer task with a LINK descriptor, pointing to a scatter/gather list
  - 31 – Stores a DCMD descriptor

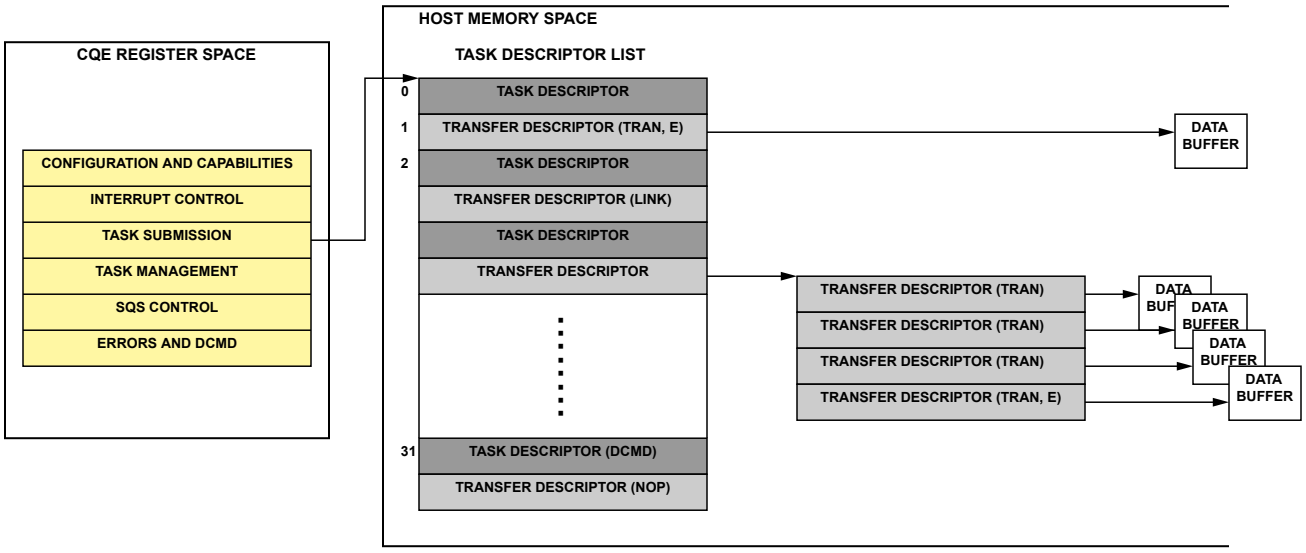


Figure 18-9: Command Queuing Host Controller Interface (HCI) Structure

Task Processing

Once the two descriptors are written in slot i, software issues the task by writing 1 to bit i in the `EMSI_CQ_TDBR` register. This action signals to the CQE to process the task. Software must not write the `EMSI_CQ_TDBR` register

to issue the task before valid descriptors are ready in the appropriate slot. The CQE then reads the descriptors. Based on the information encoded in the task descriptor, the CQE generates QUEUED_TASK_PARAMS (CMD44) and QUEUED_TASK_ADDRESS (CMD45), and sends them to the device.

Task Selection and Execution

The CQE is also in charge of reading the device's queue status register (QSR) to determine which tasks are ready for execution, selecting tasks for execution, and ordering their execution. These actions are described in this subsection.

Reading the QSR: When task(s) are queued in the device, the CQE reads the QSR to determine which tasks are ready for execution. The QSR is read using SEND_ QUEUE_STATUS (CMD13) commands. If a data transfer is ongoing, the CQE sends the SQS towards the end of the data transfer, as configured in the `EMSI_CQ_SSCFG1.SQSCMD_BLK_CNT` register field. If the bus is idle, the CQE periodically sends SQS commands, as configured in the `EMSI_CQ_SSCFG1.SQSCMD_IDLE_TMR` register field.

Selecting a Task: When one or more data transfer tasks are ready for execution and the bus is idle (or a previous data transfer is about to end), CQE is expected to select a task for execution. The selected task must be marked as ready for execution by the device.

Ordering Task Execution: When the bus is idle, the CQE sends EXECUTE_READ_TASK (CMD46) or EXECUTE_WRITE_TASK (CMD47), ordering the device to execute a task, identified by its task ID.

The CQE also feeds the task's transfer descriptor(s) to a DMA engine as a pointer to the data buffer in the host memory. A task's execution is considered completed once the data transfer phase is completed.

Task Completion: Interrupts and Interrupt Coalescing

Upon the completion of a data transfer task, after the data has been transferred, an interrupt may be generated, if requested. If the INT bit in the task descriptor is set, a task complete (TCC) interrupt is generated. If the INT bit is not set, the transfer is counted towards coalescing. DCMD tasks are not counted towards coalescing. The interrupt coalescing mechanism allows the host to moderate its interrupt load. The mechanism is also timer-protected to avoid high latency in low-load cases.

Direct Command (DCMD) Submission

When CQ is enabled, software still retains the ability to efficiently issue eMMC commands by their index and argument. To use the DCMD feature, it must be enabled where `EMSI_CQ_CFG.DCMD_EN` = 1. To issue a direct command, the software writes a DCMD task descriptor to slot #31 of the TDL and rings the doorbell register (`EMSI_CQ_TDBR`). The command's index and argument are encoded directly in the DCMD task descriptor (in the fields otherwise used for address and block count).

The DCMD feature is targeted for the transmission of non-data commands which are allowed when CQ mode is enabled, such as CMD0, CMD12, or CMD13.

When processing the task descriptor, the CQE constructs the command by its index and argument, sends it to the device, and stores the response in a dedicated register.

The DCMD task descriptor is identified by the CQE by its placement in slot #31 of the TDL. Therefore, it may only be issued using slot #31, and may only be sent one at a time. The transfer descriptor following the DCMD descriptor is ignored by CQE.

Queue-Barrier (QBR) Tasks

To enable a host's control on the ordering between tasks, a task can be marked as a Queue-Barrier (QBR) task by setting the QBR bit in the task descriptor. The following are guaranteed by the eMSI controller:

- A QBR task is only sent to the device after all the tasks issued before it have been executed (when the task queue in the device is empty).
- New tasks (tasks issued after the QBR task) are sent to the device after the QBR task execution is completed.

Specifically, the following rules apply:

- When the task marked with QBR is a data transfer task, the CQE may send the related QUEUED_ TASK_PARAMS (CMD44) and QUEUED_ TASK_ADDRESS (CMD45) commands to the device only after receiving the response for the EXECUTE_READ_TASK(CMD46)/ EXECUTE_WRITE_TASK(CMD47) for the last data transfer task in the device's queue (data transfer may still be on-going).
- When the task marked with QBR is a DCMD task, the CQE can send the related command only after all previously issued tasks have completed, including the associated data transfers.
- Commands related to tasks which are queued after the QBR task, may only be sent after the QBR task execution is completed.

After the CQE has halted, the device waits for the next command, so it is effectively halted as well. The CQE notifies software that it is halted by setting the `EMSI_CQ_CTL.HALT` bit and, optionally by an interrupt (`EMSI_CQ_ISTAT.HAC`). It is guaranteed that neither the device nor the host hardware (the CQE) will initiate any operation while the CQE is in the halt state. When software wants the CQE to resume operation, it clears the `EMSI_CQ_CTL.HALT` bit = 0. When 0 is written, the CQE continues its normal operation, based on the current state. In some error conditions, the halt bit may never actually set, because a response has not been received from the device. In these cases, software may assume the CQE is halted after waiting for a long enough time.

eMMC Command Queuing Data Structures

This section describes the descriptor structures used in eMMC Command Queuing. As mentioned in the previous section, the host writes a pair of descriptors—task descriptor and transfer descriptor, which are organized in the host memory in the task descriptor list. This section discusses the following structures:

- Task Descriptor Structure
- Transfer Descriptor Structure
- Task Descriptor for Direct-Command (DCMD) Tasks

Task Descriptor Structure

Table 18-8 Task Descriptor Structure; Lower 64 bits (Data Transfer Tasks) and Table 18-9 Task Descriptor Structure Upper 64 bits summarize the task descriptor structure for lower and upper 64 bits, and Table 18-10 Task Descriptor Fields describes task descriptor fields.

Table 18-8: Task Descriptor Structure; Lower 64 bits (Data Transfer Tasks)

Block Address	Block Count	Task Parameters Field							Attribute			
63:32	31:16	15	14	13	12	11	10:7	6	5:3	2	1	0
xxxx_xxxxh	xxxxh	Reliable Write	QBR	Priority	Data Direction	Tag request	Context	Forced programming	Act=101	Int	End= 1	Valid =1

Table 18-9: Task Descriptor Structure Upper 64 bits

Reserved	Task Parameters Field				Reserved
127:112	111	110:106	105:104	103:96	96:64
0x0000	0b	00000b	Transaction Type	0x00	X

Table 18-10: Task Descriptor Fields

Field Name	Bit Location	Description	Encoded in CMD
Valid	0	1 – The descriptor is effective and must be processed by hardware. 0 – The descriptor line must not be used.	
End	1	Must always be set to 1. Every Task Descriptor is standalone.	
Int	2	Indicates the interrupt generation policy required for this task. 1 – Hardware generates an interrupt upon the task completion. 0 – Hardware counts task completion for interrupt coalescing.	
Act	5:3	Must be configured to b101 to indicate that this is a Task Descriptor.	
Forced Programming (FP)	6	When 1, FP is enabled. Data is forcefully programmed to a non-volatile storage instead of volatile cache while cache is turned ON.	CMD44
Context ID	10:7	A context is an active session, configured for a specific read/write pattern. A device may support one or more concurrent contexts, defined by a Context ID. Each context ID (besides #0) has a configuration field in EXT_CSD to control its behavior.	CMD44
Tag Request	11	Indicates request to receive information (about specific data types) from the host.	CMD44
Data Direction	12	Indicates the direction of data transfer. 1 – Device to Host (Read) 0 – Host to Device(Write)	CMD44

Table 18-10: Task Descriptor Fields (Continued)

Field Name	Bit Location	Description	Encoded in CMD
Priority	13	1 – high 0 – simple	CMD44
Queue Barrier (QBR)	14	Indicates the control of the host on the ordering between tasks.	
Reliable Write	15	Indicates multiple block write with pre-defined block count and Reliable Write parameters.	CMD44
Block Count	31:16	Number of blocks to be read/written.	CMD44
Block Address	63:32	Data block address	CMD45
Reserved	95:64		N/A
Reserved	103:96		N/A
Transaction Type	105:104	Transaction Type (TT) Field value description: 00 - Simple Data Task 01 - Data Task w/ Immediate Partition Access 10 - Device Management Sequence (DMS) 11 - reserved **NOTE:** When 64 b descriptors are used, the implied value of this field is 0b	CMD44
Reserved	127:106		N/A

Transfer Descriptor Structure

Table 18-11 Transfer Descriptor Structure (32-Bit Addressing) summarizes the transfer descriptor structure for 32-bit addressing. Table 18-12 Transfer Descriptor Fields describes transfer descriptor fields.

Table 18-11: Transfer Descriptor Structure (32-Bit Addressing)

Address	Length	Reserved	Attribute			
63:32	31:16	15:6	5:3	2	1	0
0xXXXX_XXXX	xxxxh	0000000000	Act	Int	End	Valid

Table 18-12: Transfer Descriptor Fields

Field Name	Bit Location	Description
Valid	10	1 – The descriptor is effective and must be processed by hardware. 0 – The descriptor line must not be used.
End	1	1 – The descriptor is the last descriptor in a descriptor list. 0 – Additional descriptors follow this descriptor. In the case of a TRAN descriptor, the value of this bit is 1.

Table 18-12: Transfer Descriptor Fields (Continued)

Field Name	Bit Location	Description
Int	2	The value of this bit is cleared to 0 in command queuing.
Act	5:3	100 – TRAN – Address field of descriptor points to a data buffer. 110 – LINK – Address field of descriptor points to another descriptor. 000 – NOP no operation. Others: Reserved
Reserved	15:6	Reserved
Length	31:16	Length of data buffer in bytes. A value of 0000 means 64 kB.
Address (32- bit)	63:32	Data buffer address in host memory, in 32-bit addressing mode Address must be configured for 32-bit boundary (lower 2 bits cleared to 0)

Task Descriptor for Direct-Command (DCMD) Tasks

Table 18-13 Task Descriptor Structure summarizes the transfer descriptor structure for the DCMD tasks and Table 18-14 Transfer Descriptor Fields describes transfer descriptor fields.

Table 18-13: Task Descriptor Structure

Command Argument	Command Parameters				Task Parameters Field			Attribute			
63:32	31:25	24:23	22	21:16	15	14	13:6	5:3	2	1	0
xxxx_xxxxh	Reserved (0000000)	Response Type	CMD timing	CMD Index	Rsvd (0)	QBR	Reserved (0000000)	Act=101	Int	End=1	Valid =1

Table 18-14: Transfer Descriptor Fields

Field Name	Bit Location	Description
Valid	0	1 – The descriptor is effective and must be processed by hardware. 0 – The descriptor line must not be used.
End	1	Must always be set to 1. Every Task Descriptor is standalone.
Int	2	Indicates the interrupt generation policy required for this task. 1 – Hardware generates an interrupt upon the task completion. 0 – Hardware does not generate an interrupt upon the task completion. Interrupt coalescing is not used with DCMD.
Act	5:3	Is set to b101 to indicate that this is a Task Descriptor.
Reserved	13:6	
Queue Barrier (QBR)	14	Indicates the control of the host on the ordering between tasks.
Reserved	15	

Table 18-14: Transfer Descriptor Fields (Continued)

Field Name	Bit Location	Description
CMD Index	21:16	The index of the command to be sent to the device.
CMD Timing	22	1 – Command may be sent to device during data activity or busy time. 0 – Command may not be sent to device during data activity or busy time. **NOTE:** Software is 0 if response type is b11 (R1b).
Response Type	24:23	This field indicates to the eMSI controller the response expected to be received from the device. 00 – No Response Expected 01 – Reserved 10 – R1, R4 11 – R1b **NOTE:** R2 and R3 are not supported in DCMD.
Reserved	31:25	
CMD Argument	63:32	The argument of the command to be sent to the device.

NOTE: For more information on CQE please refer to *Annex B (Normative) Host Controller Interface for Command Queuing* from the JEDEC eMMC 5.1 specification.

eMMC Boot Operation

The eMSI controller supports both mandatory and alternate boot operations as per the JEDEC eMMC 5.1 specifications in 1-bit, 4-bit, and 8-bit SDR or legacy mode up to 50 MHz. In boot operation mode, the eMSI controller can read boot data from the eMMC by keeping the CMD line low (for mandatory boot) or sending CMD0 with the argument (0xFFFFFFFA) (alternate boot), before issuing CMD1. The data can be read from either the boot area or the user area depending on the extended CSD register configurations.

Mandatory Boot Operation

If the CMD line is held low for 74 clock cycles and more after power-up or reset operation (either through CMD0 with the argument of 0xF0F0F0F0 or assertion of hardware reset for eMMC, if it is enabled in extended CSD register byte [162], bits [1:0]) before the first command is issued, the eMMC recognizes that boot mode is being initiated and starts preparing boot data internally.

The partition that from the eMSI controller reads the boot data can be selected in advance using EXT_CSD byte [179], bits [5:3]. The eMSI controller can choose to use single data rate mode with backward-compatible interface timing or single data rate with high-speed interface timing by setting a proper value in EXT_CSD register byte [177] bits [4:3]. EXT_CSD register byte [228], bit 2 tells the eMSI controller if the high-speed timing during boot is supported by the device.

The configuration in eMMC device should be done before configuring eMSI controller for booting operation.

See eMMC Device Boot Programming Sequences for more information.

Refer to the JEDEC eMMC 5.1 specifications for more information. The mandatory boot timing is shown in the figure below.

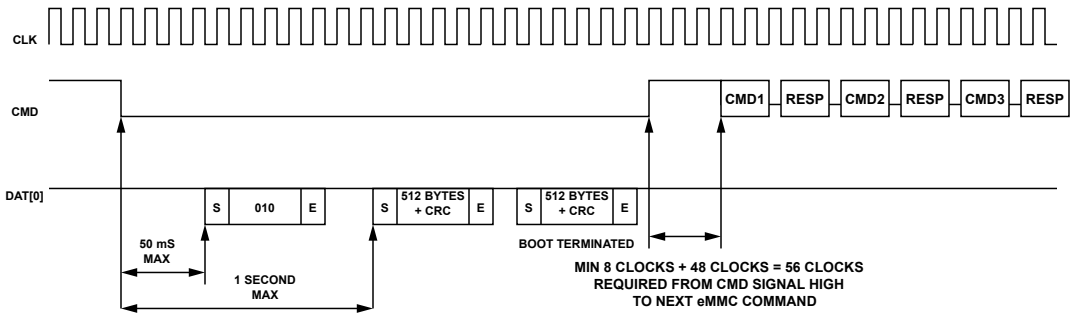


Figure 18-10: Mandatory Boot Timing

Alternate Boot Operation

Alternate boot is a command-based boot operation. After a power-up or reset operation (either assertion of CMD0 with the argument of 0xF0F0F0F0 or a hardware reset if it is enabled), the host issues CMD0 with the argument of 0xFFFFFFFA. After 74 clock cycles, before CMD1 is issued or the CMD line goes low, the eMMC recognizes that the boot mode is being initiated and starts preparing boot data internally.

All the mandatory boot operation EXT_CSD register configurations are applicable for alternate boot operations.

The configuration in eMMC device should be done before configuring eMSI controller for booting operation.

See eMMC Device Boot Programming Sequences for more information. The alternate boot timing is shown in the figure below.

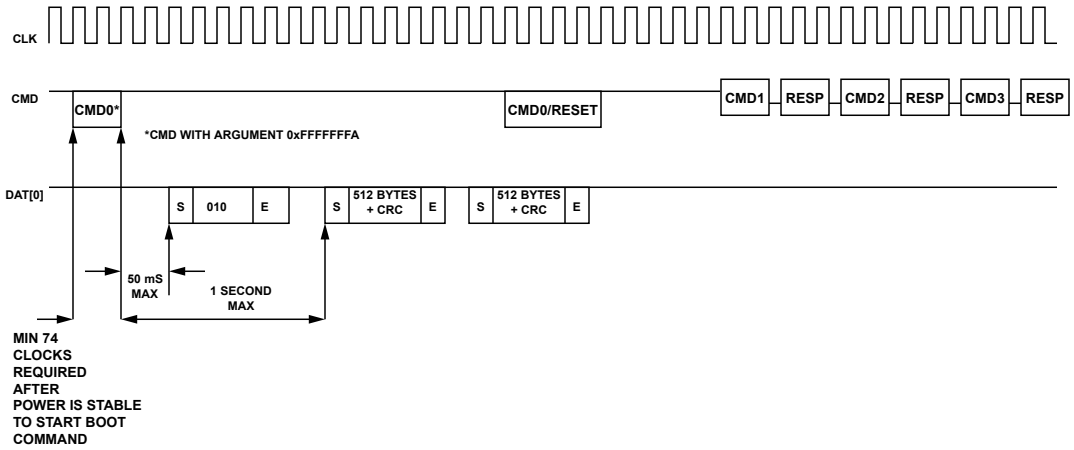


Figure 18-11: Alternate Boot Timing

eMSI Event Control and Interrupts

Interrupts

The eMSI controller generates interrupts based on various events. There are two interrupt outputs provided by the eMSI controller: EMSI_STAT and EMSI_WAKEUP. The interrupt signal must be used as the interrupt for different events. The interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the host or the software.

The interrupt status registers (EMSI_ERR_STAT, EMSI_ISTAT) indicate the events that caused the interrupt. Each event can be prevented from asserting the interrupt on the interrupt status by setting the corresponding mask bits in the EMSI_ERR_STAT_EN and EMSI_ISTAT_EN registers. A bit in interrupt status register is set only if the corresponding interrupt status enable is set and interrupt event is observed. The bit set in interrupt status register asserts an interrupt to the core only if the corresponding bit in the interrupt signal enable in the EMSI_ERR_STAT_INTEN and EMSI_ISTAT_INTEN registers is set.

During standby mode (when the SD card is not being accessed) the EMSI_WAKEUP signal can be used to identify any wakeup event, such as card removal or insertion.

NOTE: The host driver is responsible for enabling wakeup signals and disabling interrupt signals when the host system enters standby mode and for disabling wakeup signals and enabling interrupt signals when the host system goes into active mode, when the SD card is in use. The host driver must not enable both at same time. Interrupt signals are enabled using the interrupt signal enable and wakeup signals are enabled using the wakeup event enable.

For information about status interrupts, refer to the EMSI_ERR_STAT, EMSI_ISTAT, and EMSI_WU_CTL registers.

Error Detection

The eMSI controller can detect different types of errors in SD and eMMC transactions. Error are detected in either the command or data portion of the transaction. When an error is detected, the error interrupt in the interrupt status register (EMSI_ISTAT.ERR_INTERRUPT) is set. The controller uses the error interrupt status register (EMSI_ERR_STAT) to report errors in SD and eMMC mode. The abort command is used to recover from errors that are detected during data transfers. Additionally, the auto command status register (EMSI_AUTOCMD_STAT) is used to report errors during Auto CMD operation and the ADMA error status register (EMSI_ADMA_ERR_STAT) used to report errors in all ADMA based data transfer modes.

Response Error Check

The eMSI controller supports a R1 response error check function to avoid the overhead of response error checking by the host driver. If an error is detected, the response error interrupt is generated in the error interrupt status register (EMSI_ERR_STAT.RESP_ERR).

Table 18-15: Response Error Check

Bit	Error
Error status checked in R1	
31	OUT_OF_RANGE
30	ADDRESS_ERROR
29	BLOCK_LEN_ERROR
26	WP_VIOLATION
25	CARD_IS_LOCKED
23	COM_CRC_ERROR
21	CARD_ECC_FAILED
20	CC_ERROR
19	ERROR

NOTE: Please refer to Device Status section from JEDEC eMMC 5.1 specifications for more information.

eMSI Programming Model

This section provides detailed procedures for programming the eMSI controller.

Initial Programming Sequence

This section explains the programming steps required for the eMSI controller and the eMMC device before starting a data transfer with a connected eMMC device. Both the eMSI controller and the eMMC device need to be initialized.

1. Ensure clock supply to the eMSI controller. This clock is required for accessing most of the registers in eMSI controller.
2. Check if the card is already inserted by reading the present state register (`EMSI_PSTATE`). This step is required for a removable card. If the card is already inserted, then go to Step 4.
3. Wait for the card insertion interrupt. This step is required for a removable card. The card detection sequence is described in Card Detection.
4. Configure the basic settings for the eMSI controller as described in eMSI Controller Setup Sequence. This includes the timeout counter value and clock generation parameters. The setup sequence for an SD/eMMC device is provided.
5. Enable all clocks signals as discussed in the programming sequence in eMSI Controller Clock Setup Sequence. At this stage, the eMSI controller is ready to communicate with the eMMC device.

6. If the eMSI controller is connected to an SD card, then the programming sequence described in SD Card Interface Detection must be executed and if the eMSI controller is connected to an eMMC device, then the programming sequence described in eMMC Card Interface Setup must be executed. This is a simplified sequence especially for an eMMC card as the card type is always known.
7. The sequence for Initializing and Identifying an eMMC Device must be executed for initializing and identifying an eMMC device. For SD cards, the SD Card Initialization and Identification sequence must be executed.
8. When the eMSI controller and eMMC device are initialized, the controller can start sending commands to the SD/eMMC device to perform the data transfer. Control and data commands are two types of commands that can be issued to the SD card/eMMC device. The control command is used to read or write any register in the SD card/eMMC device. The data command is used for writing or reading data to or from the device. Below are the control and data command sequences for SD card/eMMC devices.
   - Control CMD – Issuing CMD Without Data Transfer
   - Data CMD – Issuing CMD With Data Transfer

   ADDITIONAL INFORMATION: Data transfer can be aborted using Abort Command Sequence.

Card Detection

The following figure and procedure shows the sequence for detecting an SD card. The procedure applies to removable cards (SD and cards that use external sockets). This programming sequence is not required for an eMMC device as it is non- removable.

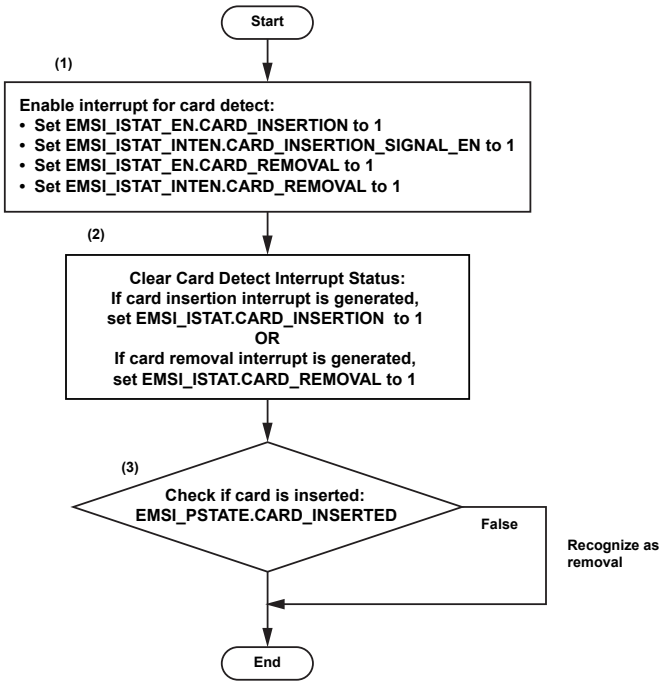


Figure 18-12: Card Detection

1. Enable interrupt for card detect:
    a. Set `EMSI_ISTAT_EN.CARD_INSERTION` to 1.
    b. Set `EMSI_ISTAT_INTEN.CARD_INSERTION` to 1.
    c. Set `EMSI_ISTAT_EN.CARD_REMOVAL` to 1.
    d. Set `EMSI_ISTAT_INTEN.CARD_REMOVAL` to 1.
2. Clear card detect interrupt status:
    - If card insertion interrupt is generated, set `EMSI_ISTAT.CARD_INSERTION` to 1.
    - If card removal interrupt is generated, set `EMSI_ISTAT.CARD_REMOVAL` to 1.
3. Check if card is inserted: `EMSI_PSTATE.CARD_INSERTED` (0 = reset, debouncing, or no card).

eMSI Controller Setup Sequence

This section discusses the eMSI controller setup sequence for an SD/eMMC device.

eMMC Device Setup

The *eMSI Controller Setup Sequence for eMMC Interface* figure and procedure shows the eMSI controller setup sequence for an eMMC device. For more information on bit fields set in this sequence please refer to register descriptions.

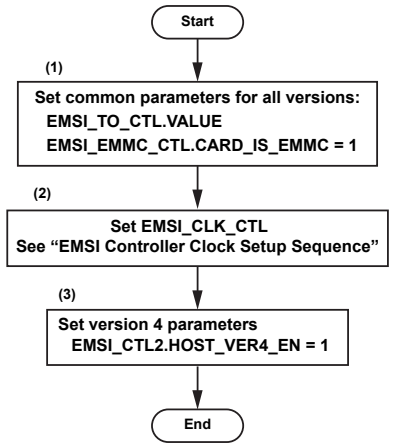


Figure 18-13: eMSI Controller Setup Sequence for eMMC Interface

1. Configure common parameters for all versions
    a. `EMSI_TO_CTL.VALUE`.
    b. `EMSI_EMMC_CTL.CARD_IS_EMMC` = 1.
2. Configure `EMSI_CLK_CTL` see eMSI Controller Clock Setup Sequence.

3. Configure version 4 parameters.
    a. `EMSI_CTL2.HOST_VER4_EN` = 1.

SD Interface Setup

The *eMSI Controller Setup Sequence for SD Interface* figure and procedure shows the eMSI controller setup sequence for an SD interface.

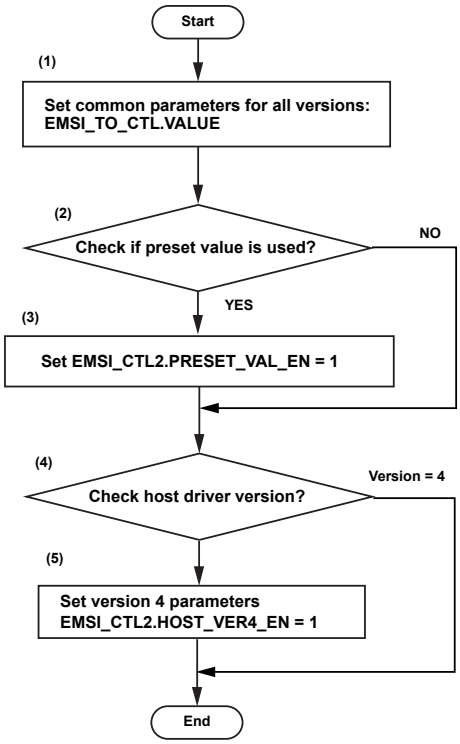


Figure 18-14: eMSI Controller Setup Sequence for SD Interface

NOTE: The eMSI controller supports version 4 mode.

1. Configure common parameters for all versions using the `EMSI_TO_CTL.VALUE` bit field.
2. Check if preset value is used?
    - If no go to step 4.
    - If yes go to step 3.
3. Set `EMSI_CTL2.PRESET_VAL_EN` = 1.
4. Check host driver version?
    - If version other than 4 go to end.
    - If version equal to 4 go to step 5.

5. Configure version 4 parameters, `EMSI_CTL2.HOST_VER4_EN` = 1.

Clock Control

This section discusses the programming sequence for setting up controller internal clocks and the eMSI bus clock which is provided to the eMMC device. This section also describes programming sequence for changing eMSI bus clock frequency after initial clock setup is completed.

This section includes the following sequences:

- eMSI Controller Clock Setup Sequence
- eMSI Bus Clock Supply and Stop Sequence
- eMSI Bus Clock Frequency Change Sequence

eMSI Controller Clock Setup Sequence

The *eMSI Controller Clock Setup Sequence* figure shows the sequence for setting up internal clocks for eMSI controller. The eMSI controller requires an eMSI input clock and eMSI TIMER CMQ clock from the system and an eMSI bus clock (clock supplied to eMMC device). The internal clocks should be initialized before accessing any eMSI controller registers.

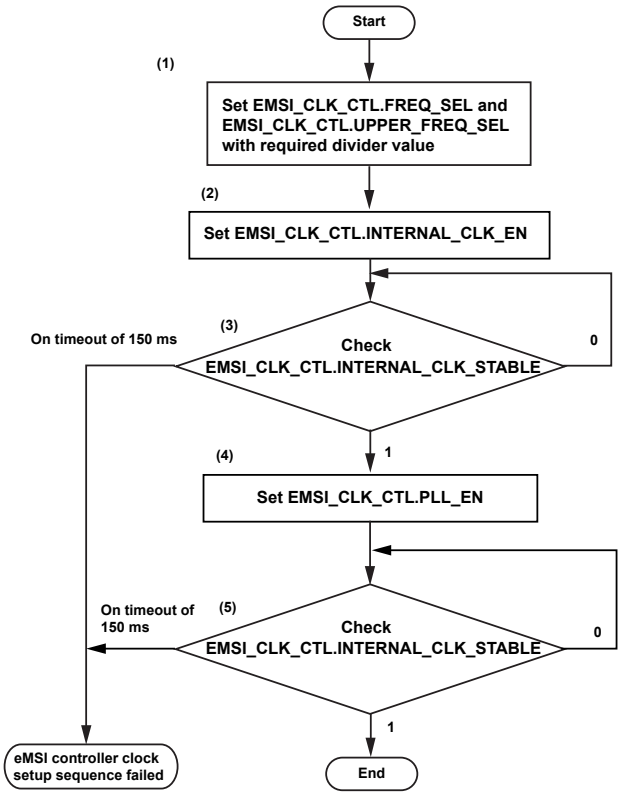


Figure 18-15: eMSI Controller Clock Setup Sequence

1. Configure the EMSI_CLK_CTL.UPPER_FREQ_SEL and EMSI_CLK_CTL.FREQ_SEL bits with the required divider value.
2. Configure the EMSI_CLK_CTL.INTERNAL_CLK_EN register.
3. Check the EMSI_CLK_CTL.INTERNAL_CLK_EN bit.
   - If 0 recheck the EMSI_CLK_CTL.INTERNAL_CLK_STABLE bit.
   - If 1 go to step 4.

   ADDITIONAL INFORMATION: On timeout of 150 ms eMSI controller clock setup sequence failed.
4. Configure the EMSI_CLK_CTL.PLL_EN bit.
5. Check the EMSI_CLK_CTL.INTERNAL_CLK_STABLE bit.
   - If 0 recheck the EMSI_CLK_CTL.INTERNAL_CLK_STABLE bit.
   - If 1 go to End

   ADDITIONAL INFORMATION: On timeout of 150 ms eMSI controller clock setup sequence failed.

eMSI Bus Clock Supply and Stop Sequence

The *eMSI Bus Clock Supply and Stop Sequence* figure shows the flow chart for stopping and supplying the eMSI bus clock. The procedure outlined in the figure applies to SD card/eMMC device.

NOTE: The clock supply and stop sequence is initiated, if required, only when there is no active data transfer between the eMSI controller and the eMMC device.

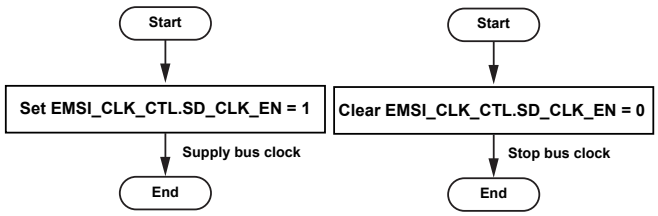


Figure 18-16: eMSI Bus Clock Supply and Stop Sequence

eMSI Bus Clock Frequency Change Sequence

The *eMSI Bus Clock Frequency Change* figure shows the sequence for changing eMSI bus clock frequency.

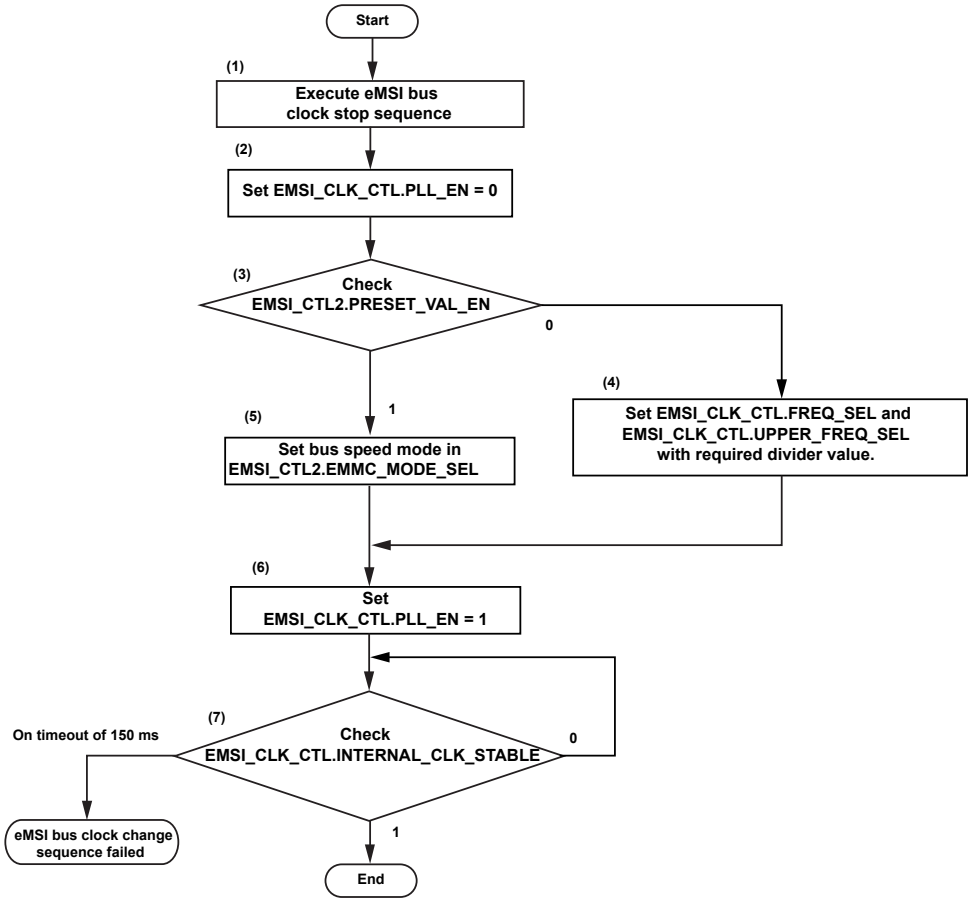


Figure 18-17: eMSI Bus Clock Frequency Change

1. Execute the eMSI bus clock stop sequence.
2. Clear the `EMSI_CLK_CTL.PLL_EN` bit = 0.
3. Check the `EMSI_CTL2.PRESET_VAL_EN` bit.
   - If 0 go to step 4.
   - If 1 go to step 5.
4. Configure bus speed mode in the `EMSI_CTL2.EMMC_MODE_SEL` bit field.
5. Configure `EMSI_CLK_CTL.FREQ_SEL` and `EMSI_CLK_CTL.UPPER_FREQ_SEL` with required divider value.
6. Set `EMSI_CLK_CTL.PLL_EN` = 1.
7. Check the `EMSI_CLK_CTL.INTERNAL_CLK_STABLE` bit.
   - If 0 recheck the `EMSI_CLK_CTL.INTERNAL_CLK_STABLE` bit.
   - If 1 go to End.

ADDITIONAL INFORMATION: On timeout of 150 ms eMSI bus clock frequency change sequence failed.

NOTE: Software must issue `EMSI_SWRST.DAT` and `EMSI_SWRST.CMD` after an eMSI bus clock frequency change sequence to avoid the effect of any glitch on the sampling clock. The program must also disable the CDU clock output for the eMSI, before executing an eMSI bus clock frequency change sequence. After sequence execution, the program should reenable the CDU clock. Additionally, enable the eMSI bus clock using the eMSI bus clock supply sequence (to ensure a clock is supplied to the eMMC device for further operations).

Card Interface Setup Sequence

This section discusses the programming sequence for setting up the card interface for an SD or eMMC card and contains the following topics:

- See SD Card Setup Sequence
- See eMMC Card Interface Setup

SD Card Setup Sequence

The *SD Card Setup Sequence* figure shows the programming sequence to set up the SD card. This sequence configures the eMSI controller for detection of an SD card.

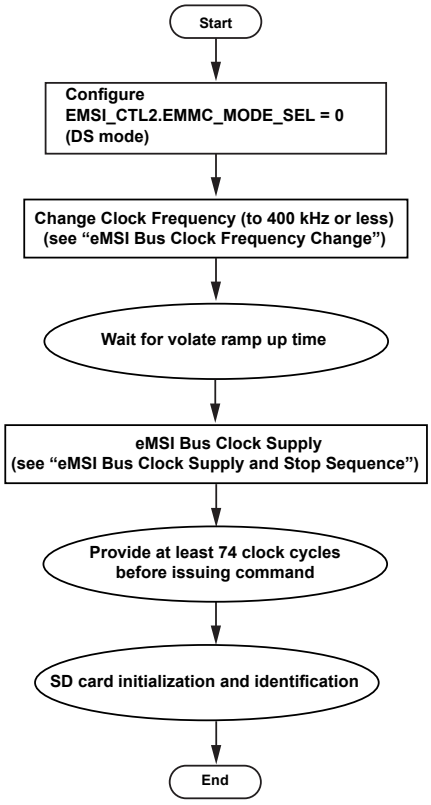


Figure 18-18: SD Card Setup Sequence

eMMC Card Interface Setup

The *eMMC Card Setup Sequence* figure and procedure shows the programming sequence to set up an eMMC device. This sequence configures the eMSI controller for detection of an eMMC device.

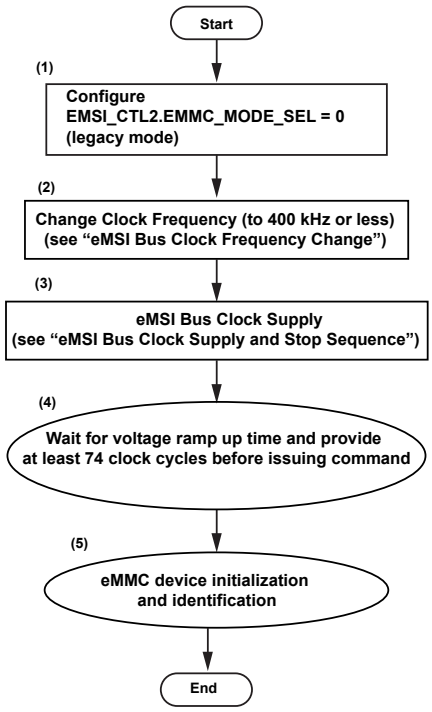


Figure 18-19: eMMC Card Setup Sequence

1. Configure `EMSI_CTL2.EMMC_MODE_SEL` = 0 bit for legacy mode.
2. Change the clock frequency (to 400 kHz or less). See eMSI Bus Clock Frequency Change Sequence.
3. Perform eMMC clock supply. See eMSI Bus Clock Supply and Stop Sequence.
4. Wait for voltage ramp up time and provide at least 74 clock cycles before issuing command.
5. eMMC device initialization and identification.

Setting Timeout on the eMSI Bus

The *Setting eMSI Bus Timeout* figure and procedure shows the sequence for setting timeout on the eMSI bus. This value determines the interval by which DAT line timeouts are detected. For more information please refer to the corresponding register descriptions.

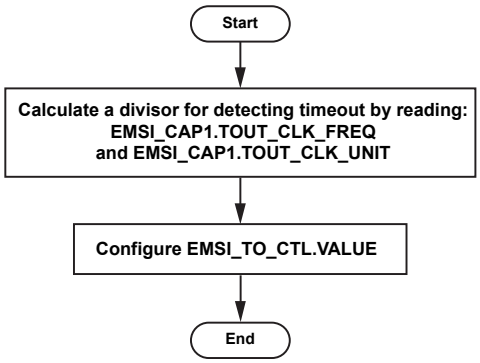


Figure 18-20: Setting eMSI Bus Timeout

1. Calculate a divisor for detecting timeout by reading the `EMSI_CAP1.TOUT_CLK_FREQ` and `EMSI_CAP1.TOUT_CLK_UNIT` registers.
2. Configure the `EMSI_TO_CTL.VALUE` register.

Abort Transaction

An abort command to the SD/eMMC device is issued to abort an ongoing data transaction. To abort a transaction on an SD/eMMC device, CMD12 is issued.

Following are the instances when an abort command must be issued:

- To stop an infinite block transfer
- To stop a transfer due to a request from an application
- To stop a transfer due to an error detection

The sequence to issue an abort command is shown in Abort Command Sequence. An abort command can be issued asynchronously during data transfer. In an asynchronous abort the abort command is issued at any time irrespective of the state of data transfer as a subcommand. The method is explained in Asynchronous Abort.

Abort Command Sequence

The *Abort Command Sequence* figure and procedure describe the abort command sequence.

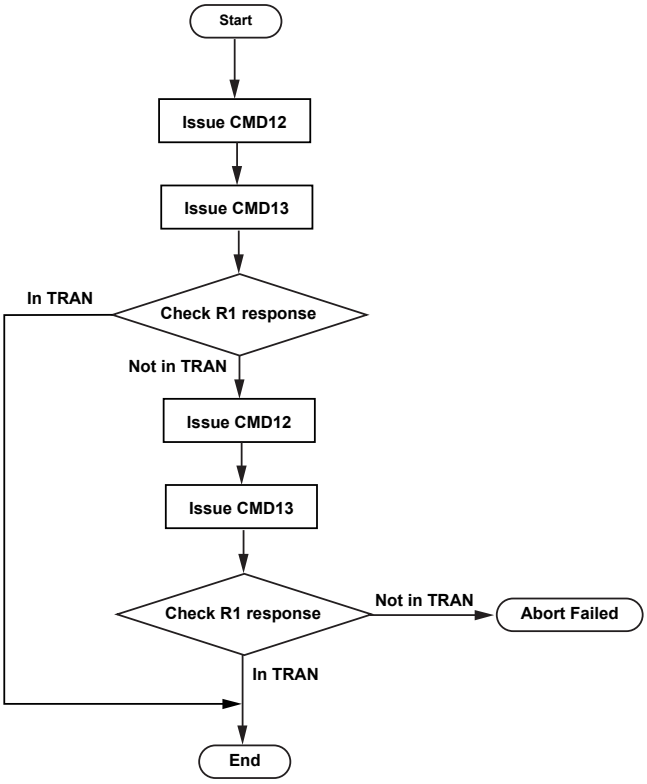


Figure 18-21: Abort Command Sequence

1. Issue CMD12.
2. Issue CMD13.
3. Check R1.
   - If in TRAN go to End.
   - If not in TRAN go to step 5.
4. Issue CMD12.
5. Issue CMD13.
6. Check R1.
   - If in TRAN go to End.
   - If not in TRAN abort failed.

   ADDITIONAL INFORMATION: Keep sending CMD13 until the card returns to the transfer state. In some cases it takes more than twice to send CMD13 for correctly detecting the transfer state. Applicable when CMD 12 is successfully sent.

Asynchronous Abort

The *Asynchronous Abort Sequence* figure and procedure describe the asynchronous abort command sequence.

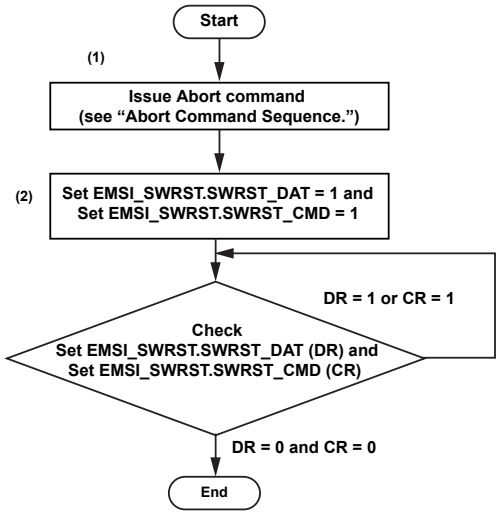


Figure 18-22: Asynchronous Abort Sequence

1. Issue abort command (see Abort Command Sequence).
2. Set the `EMSI_SWRST.DAT` and `EMSI_SWRST.CMD` bits = 1.
3. Check the `EMSI_SWRST.DAT` (DR) and `EMSI_SWRST.CMD` (CR) bits.
   - DR = 1 or CR = 1 recheck the `EMSI_SWRST.DAT` and `EMSI_SWRST.CMD` bits.
   - DR = 0 and CR = 0 go to End.

eMMC Transaction Mode

This section discusses the programming sequences for setting the eMSI controller for an eMMC interface. This section also discusses how to program and initiate different boot speed modes.

This section discusses the following programming sequences:

- Initializing and Identifying an eMMC Device
- Issuing CMD With or Without Data Transfers for an eMMC Device
- Switching to Various Speed Modes in an eMMC Device
- Changing the Data Bus Width for an eMMC Device
- Command Queueing
- eMMC Device Boot Programming Sequences

Initializing and Identifying an eMMC Device

The *Card Initialization and Identification Programming Sequence* figure shows the programming sequence to initialize and identify an eMMC device.

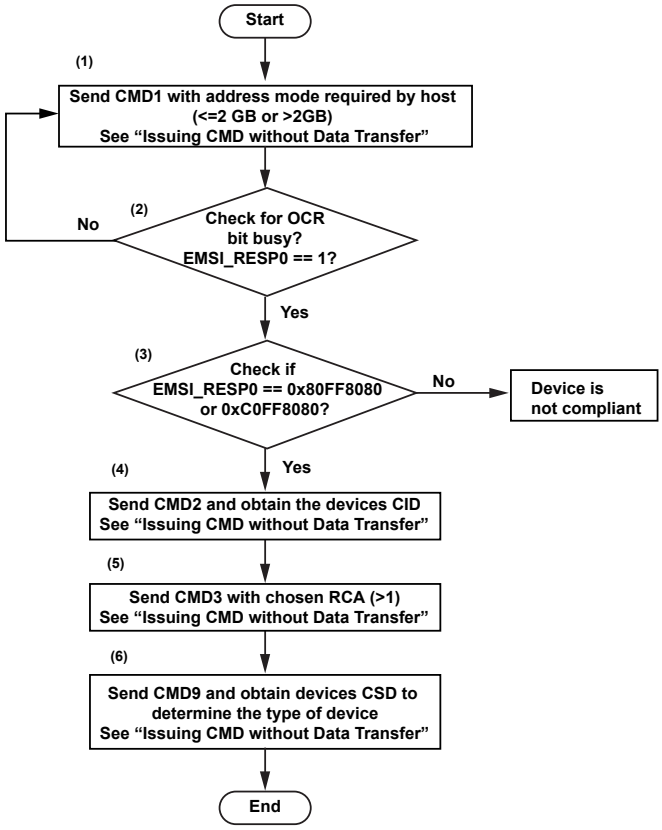


Figure 18-23: Card Initialization and Identification Programming Sequence

1. Send CMD1 with address mode required by host (≤2 GB or >2GB). See Issuing CMD Without Data Transfer.
2. Check for OCR bit busy `EMSI_RESP0` == 1?
   a. No - go back to step 1.
   b. Yes - Go to step 3.
3. Check if `EMSI_RESP0` == 0x80FF8080 or 0xC0FF8080? If not, the device is not compliant.
4. Send CMD2 and obtain the devices CID See Issuing CMD Without Data Transfer.
5. Send CMD3 with chosen RCA (>1) See Issuing CMD Without Data Transfer.
6. Send CMD9 and obtain devices CSD to determine the type of device See Issuing CMD Without Data Transfer.

Switching to Various Speed Modes in an eMMC Device

The *Programming Sequence to Switch to Various Speed Modes in an eMMC Device* figure and procedure shows the programming sequence to switch to various speed modes in an eMMC device. In the figure, note that CMD6 is effective only during the transfer state.

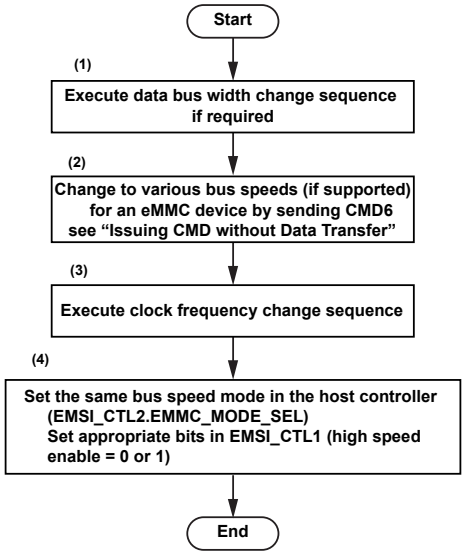


Figure 18-24: Programming Sequence to Switch to Various Speed Modes in an eMMC Device

1. Execute data bus width change sequence if required.
2. Change to various bus speeds (if supported) for an eMMC card by sending CMD6 see Issuing CMD Without Data Transfer.
3. Execute clock frequency change sequence
4. Configure the same bus speed mode in the host controller using the `EMSI_CTL2.EMMC_MODE_SEL` bit field. Configure the `EMSI_CTL1.HIGH_SPEED_EN` bit field (high speed enable = 0 or 1).

Changing the Data Bus Width for an eMMC Device

The *Programming Sequence to Change Data Bus Width for an eMMC Device* figure shows the programming sequence to change the data bus width for an eMMC device. In the figure, note that CMD6 is effective only during transfer state.

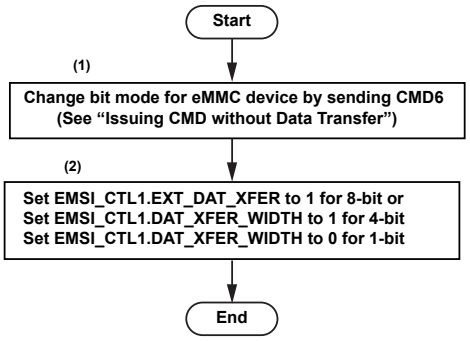


Figure 18-25: Programming Sequence to Change Data Bus Width for an eMMC Device

1. Change bit mode for eMMC card by sending CMD6 (See Issuing CMD Without Data Transfer).
2. Set the `EMSI_CTL1.EXT_DAT_XFER` bit to 1 for 8-bit or set the `EMSI_CTL1.DAT_XFER_WIDTH` bit to 1 for 4-bit or clear to 0 for 1-bit.

Error Recovery in eMMC Mode

The *Error Recovery Sequence in eMMC Mode* figure and procedure shows the error recovery sequence for eMMC mode.

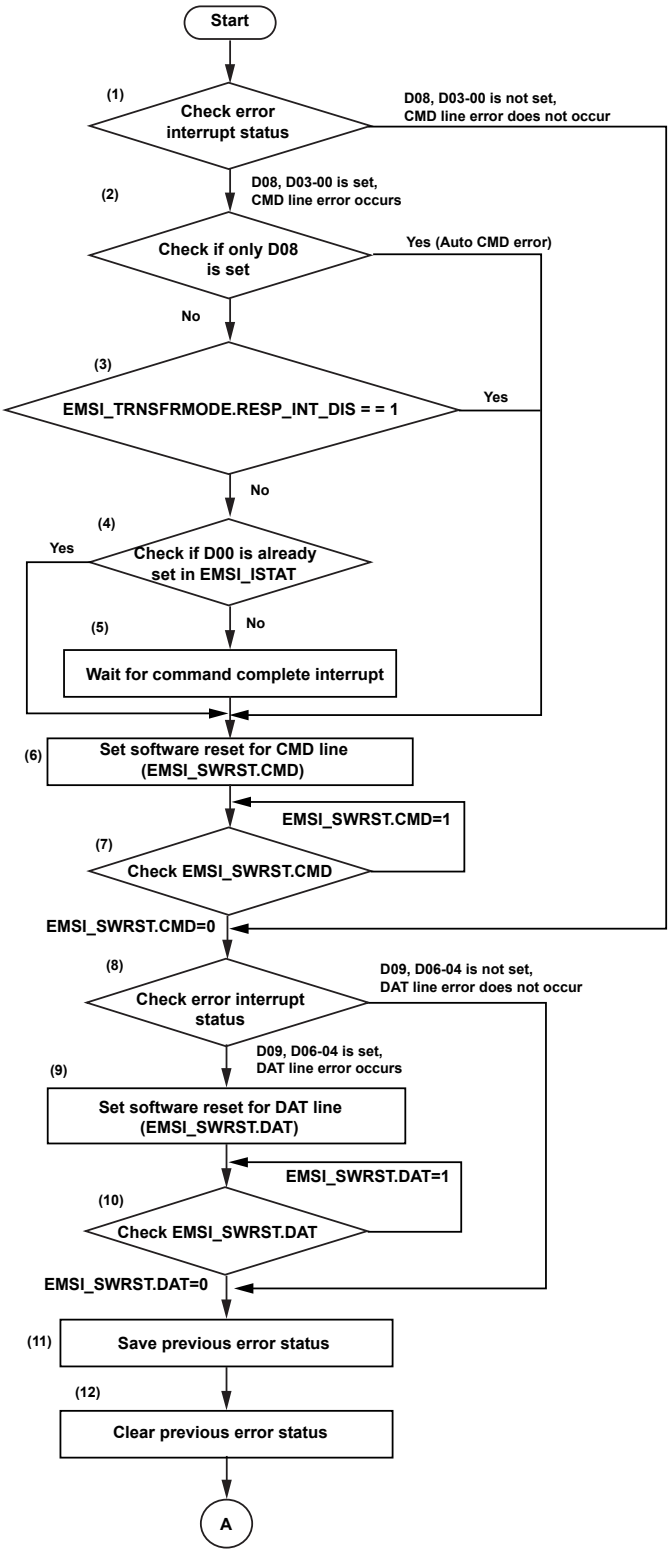


Figure 18-26: Error Recovery Sequence in eMMC Mode

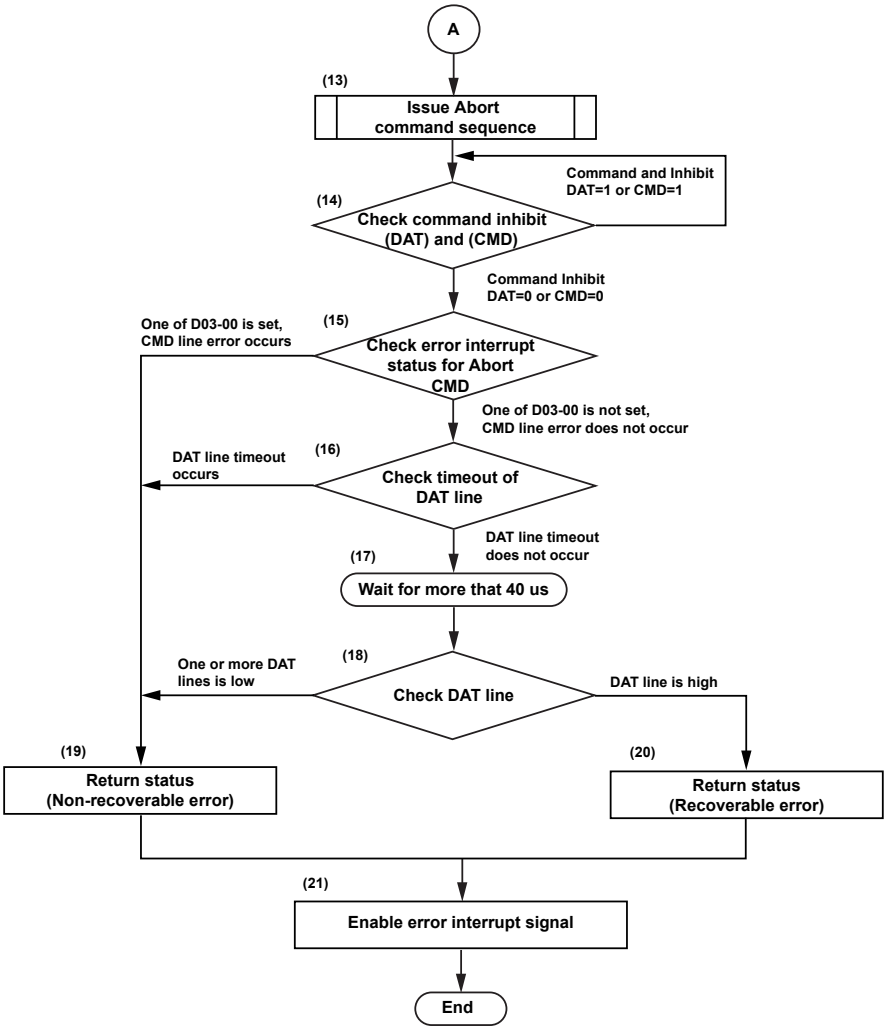


Figure 18-27: Error Recovery Sequence in eMMC Mode (continued)

1. Check error interrupt status.
   - D08, D03-00 is not set, CMD line error does not occur - go to step 8.
   - D08, D03-00 is set, CMD line error occurs - go to step 3.
2. Check if only D8 is set.
   - Yes (Auto CMD error) - go to step 6.
   - No - go to step 3.
3. `EMSI_TRNSFRMODE.RESP_INT_DIS` = = 1
   - Yes - go to step 6.
   - No - go to step 4.

4. Check if D00 is already set in the EMSI_ISTAT register.
    - Yes - go to step 6.
    - No - go to step 5.
5. Wait for command complete interrupt.
6. Set software reset for CMD line using the EMSI_SWRST.CMD bit.
7. Check the EMSI_SWRST.CMD bit.
    - If = 1 recheck.
    - If = 0 - go to step 8.
8. Check error interrupt status.
    - D09, D06-04 is not set, DAT line error does not occur - go to step 11.
    - D09, D06-04 is set, DAT line error occurs - go to step 9.
9. Set software reset for DAT line using the EMSI_SWRST.DAT bit.
10. Check the EMSI_SWRST.DAT bit.
    - If = 1 recheck.
    - If = 0 - go to step 11.
11. Save previous error status.
12. Clear previous error status.
13. Issue Abort command sequence.
14. Check command inhibit (DAT) and (CMD)
    - Command and Inhibit DAT = 1 or CMD = 1 - recheck.
    - Command Inhibit DAT = 0 or CMD = 0 - go to step 15.
15. Check error interrupt status for Abort CMD
    - One of D03-00 is set, CMD line error occurs - go to step 19.
    - One of D03-00 is not set, CMD line error does not occur - go to step 16.
16. Check timeout of DAT line.
    - DAT line timeout occurs - go to step 19.
    - DAT line timeout does not occur - go to step 17.
17. Wait for more that 40 us.
18. Check DAT line

- One or more DAT lines is low - go to step 19.
- DAT line is high - go to step 20.

19. Return status (Non-recoverable error).
20. Return status (Recoverable error).
21. Enable error interrupt signal.

NOTE: The error recovery sequence returns either a *Recoverable Error* or a *Non-Recoverable Error* status. When a recoverable error is returned, the error is recovered fully. When the non-recoverable error status returns, both the controller and the eMMC device must be re-initialized after performing a power cycle for the eMMC device.

Set the `EMSI_CMD.TYPE` bit field to 0x3 while issuing the abort CMD using CMD12. The software driver must issue a software reset for the DAT line while issuing a software reset for the CMD line. D00, D01, D02, D03, etc. corresponds to bit field 0 , 1, 2, 3 respectively from the `EMSI_ERR_STAT` register.

Issuing CMD With or Without Data Transfers for an eMMC Device

Programs can issue commands with or without data transfers.

- The programming sequence for issuing a command without a data transfer for an eMMC device is like the sequence described in Issuing CMD Without Data Transfer.
- The programming sequence for issuing a command with a data transfer for an eMMC device is like the programming sequence described in the sections following Issuing CMD With Data Transfer. Data transfers can be performed using DMA. The supported DMA modes are SDMA, ADMA2, and ADMA3.

NOTE: The application must ensure that the `EMSI_BOOT_CTL.BOOT_ACK_ENABLE` bit is cleared (when set due to a previous boot CMD) before initiating any data transfer CMD.

Issuing CMD Without Data Transfer

The *eMMC Command Issue and Complete* figure and procedure show the steps to issue a CMD without a data transfer.

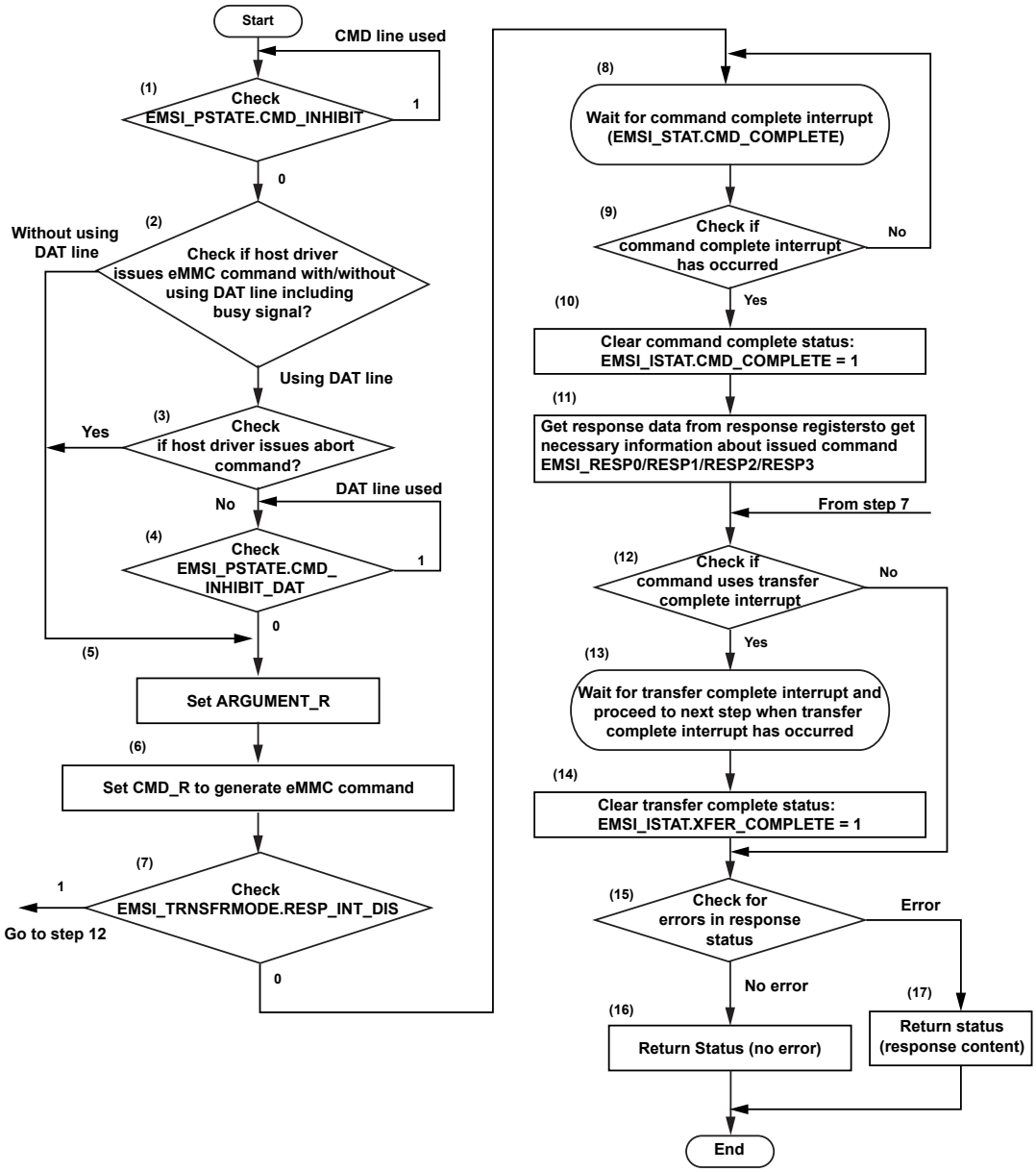


Figure 18-28: eMMC Command Issue and Complete

1. Check the `EMSI_PSTATE.CMD_INHIBIT` bit.
    a. If 1 - CMD line used. Recheck until the `EMSI_PSTATE.CMD_INHIBIT` bit = 0.
    b. If 0 - go to step 2.
2. Check if host driver issues an eMMC command with/without using DAT line including busy signal.
    a. Without using DAT line - go to step 5.
    b. Using DAT line - go to step 3.

3. Check if host driver issues abort command.
    a. Yes - go to step 5.
    b. No - go to step 4.
4. Check `EMSI_PSTATE.CMD_INHIBIT_DAT`.
    a. If 1 - DAT line used. Repeat until 0.
    b. If 0 - go to step 5.
5. Configure ARGUMENT_R to generate an eMMC command.
6. Configure CMD_R.
7. Check `EMSI_TRNSFRMODE.RESP_INT_DIS`
    a. If 1 - go to step 12.
    b. If 0 - go to step 8.
8. Wait for command complete interrupt (`EMSI_ISTAT.CMD_COMPLETE`).
9. Check if command complete interrupt has occurred.
    a. No - go back to step 8.
    b. Yes- go to step 10.
10. Set `EMSI_ISTAT.CMD_COMPLETE` = 1.
11. Get response data from response registers to get necessary information about issued command from the `EMSI_RESP0`/`EMSI_RESP1`/`EMSI_RESP2`/`EMSI_RESP3` registers.
12. Check if command uses transfer complete interrupt.
    a. No - go to step 15.
    b. Yes - go to step 13.
13. Wait for transfer complete interrupt and proceed to next step when transfer complete interrupt has occurred.
14. Clear transfer complete interrupt (`EMSI_ISTAT.XFER_COMPLETE` = 1).
15. Check for errors in response status.
    a. Error - go to step 17.
    b. No error - go to step 16.
16. Return Status (no error).
17. Return status (response contents).

NOTE: Before issuing next command for an extra check, the program can poll on the `EMSI_PSTATE.DAT_LINE_ACTIVE` bit until it is cleared for R1b response command types.

Issuing CMD With Data Transfer

This section includes the sequence to issues a eMMC command with data transfer using SDMA, ADMA, and ADMA3 modes.

NOTE: While using ADMA2 or ADMA3, the host memory data buffer size and start address must not exceed 128 MB. When greater than 128 MB, the data buffer must be split using two descriptors such that a transfer attempting to cross the limit (of 128 MB) is not generated.

This section discusses the following programming sequences:

- Issuing CMD with Data Transfer (Using SDMA)
- Issuing CMD with Data Transfer (Using ADMA2)
- Issuing CMD with Data Transfer (Using ADMA3)

The ADMA3 ID descriptor, command descriptor, and ADMA2 descriptor are organized in a location in the host memory. Please refer to Advanced DMA 2 (ADMA2) and Advanced DMA 3 (ADMA3) for more information on ADMA2 and ADMA3 descriptors.

NOTE: The ADMA3 ID address, command descriptor address, and ADMA2 descriptor address should be 32-bit aligned.

Issuing CMD with Data Transfer (Using SDMA)

The *Transaction Control with Data Transfer Using SDMA* figure and procedure shows the transaction that uses SDMA for data transfer using the DAT line. The `EMSI_ERR_STAT.ADMA_ERR` bit is set (=1) for all DMA (SDMA, ADMA2 and ADMA3) transfers for error response.

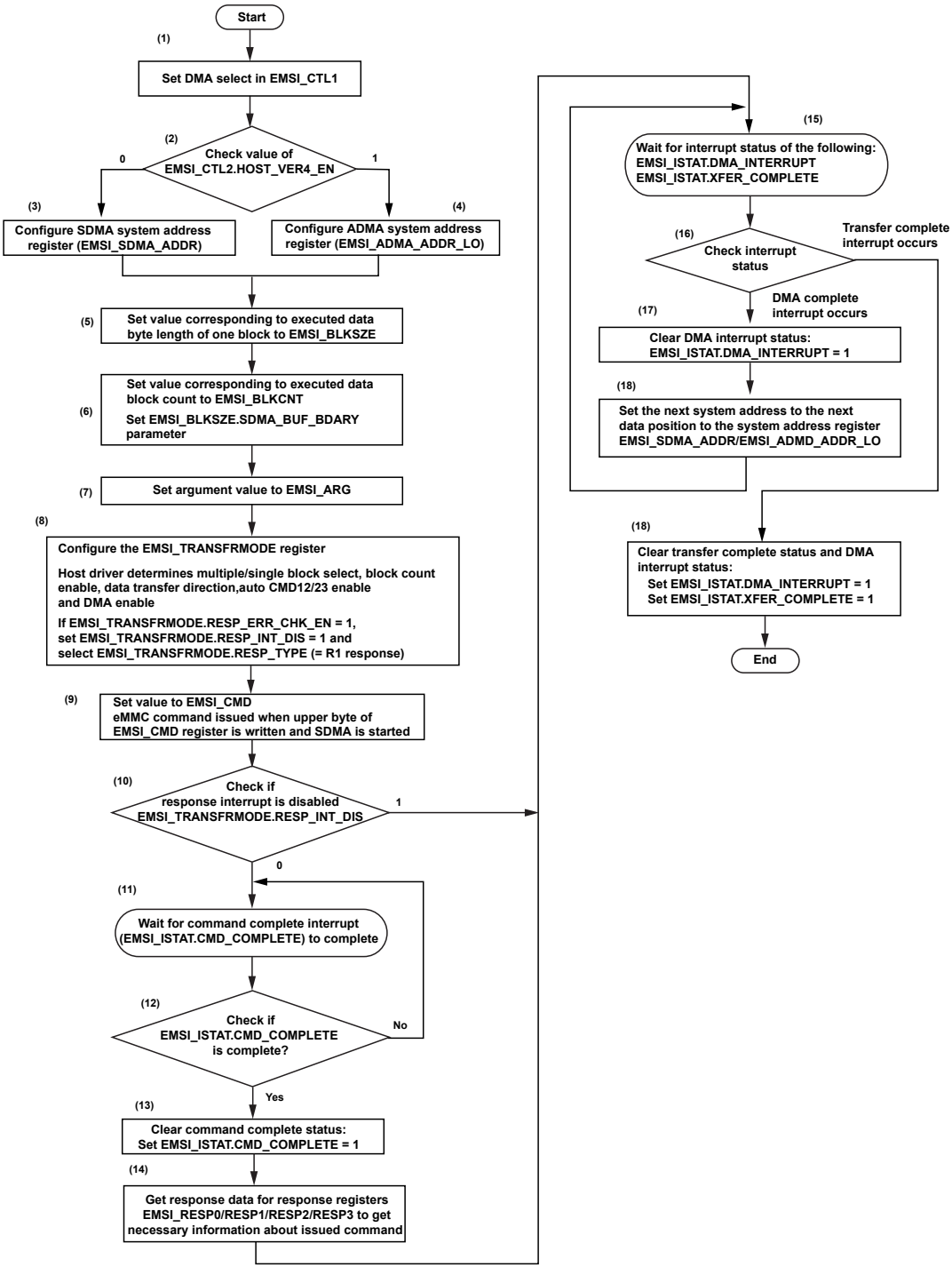


Figure 18-29: Transaction Control with Data Transfer Using SDMA

1. Configure the `EMSI_CTL1.DMA_SEL` bit field for SDMA.
2. Check the value of the `EMSI_CTL2.HOST_VER4_EN` bit.

a. If 0 - go to step 3.

b. If 1 - go to step 4.

3. Configure the SDMA system address register (EMSI_SDMA_ADDR).
4. Configure the ADMA system address register (EMSI_ADMA_ADDR_LO).
5. Configure the value corresponding to executed data byte length of one block to the EMSI_BLKSZ register.
6. Configure value corresponding to executed data block count to the EMSI_BLKCNT register. Configure the size of contiguous buffer in system memory using the EMSI_BLKSZ.SDMA_BUF_BDARY bit field.
7. Configure the argument value in the EMSI_ARG register.
8. Configure the EMSI_TRNSFRMODE register.

   The host driver determines multiple/single block select, block count enable, data transfer direction, auto CMD12/23 enable and DMA enable. If EMSI_TRNSFRMODE.RESP_ERR_CHK_EN = 1, set EMSI_TRNSFRMODE.RESP_INT_DIS = 1 and select the transfer mode type in the EMSI_TRNSFRMODE.RESP_TYPE register (= R1 response).
9. The eMMC command issued when the upper byte of the EMSI_CMD register is written and SDMA is started.
10. Check if the response interrupt is disabled in the EMSI_TRNSFRMODE.RESP_INT_DIS bit.

    a. If 1 - go to step 15.

    b. If 0 - go to step 11.
11. Wait for the command complete interrupt (EMSI_ISTAT.CMD_COMPLETE) to complete.
12. Check if the EMSI_ISTAT.CMD_COMPLETE interrupt is complete.

    a. If no - go back to step 11.

    b. If yes - go to step 13.
13. Clear the command complete status by setting EMSI_ISTAT.CMD_COMPLETE = 1.
14. Get response data for response registers EMSI_RESP0/EMSI_RESP1/EMSI_RESP2/EMSI_RESP3 to get the necessary information about the issued command.
15. Wait for the interrupt status of the EMSI_ISTAT.DMA_INTERRUPT and EMSI_ISTAT.XFER_COMPLETE bits.
16. Check the interrupt status.

    a. Transfer complete interrupt occurs - go to step 19.

    b. DMA interrupt occurs - go to step 17
17. Clear the DMA interrupt status by setting EMSI_ISTAT.DMA_INTERRUPT = 1.

18. Configure the next system address to the next data position to the system address register EMSI_SDMA_ADDR/EMSI_ADMA_ADDR_LO.
19. Clear the transfer complete status and DMA interrupt status by setting the EMSI_ISTAT.DMA_INTERRUPT and EMSI_ISTAT.XFER_COMPLETE bits = 1.

NOTE: As per the SDMA Buffer Boundary parameter, the program should ensure that the buffer is properly aligned so that it is continuous in the memory up to the specified buffer boundary.

Issuing CMD with Data Transfer (Using ADMA2)

The *Transaction Control with Data Transfer Using DAT Line (Using ADMA2)* figure and procedure describe transaction control using the DAT line for ADMA2 data transfers. For more information on ADMA2 operation and descriptor, see Advanced DMA 2 (ADMA2).

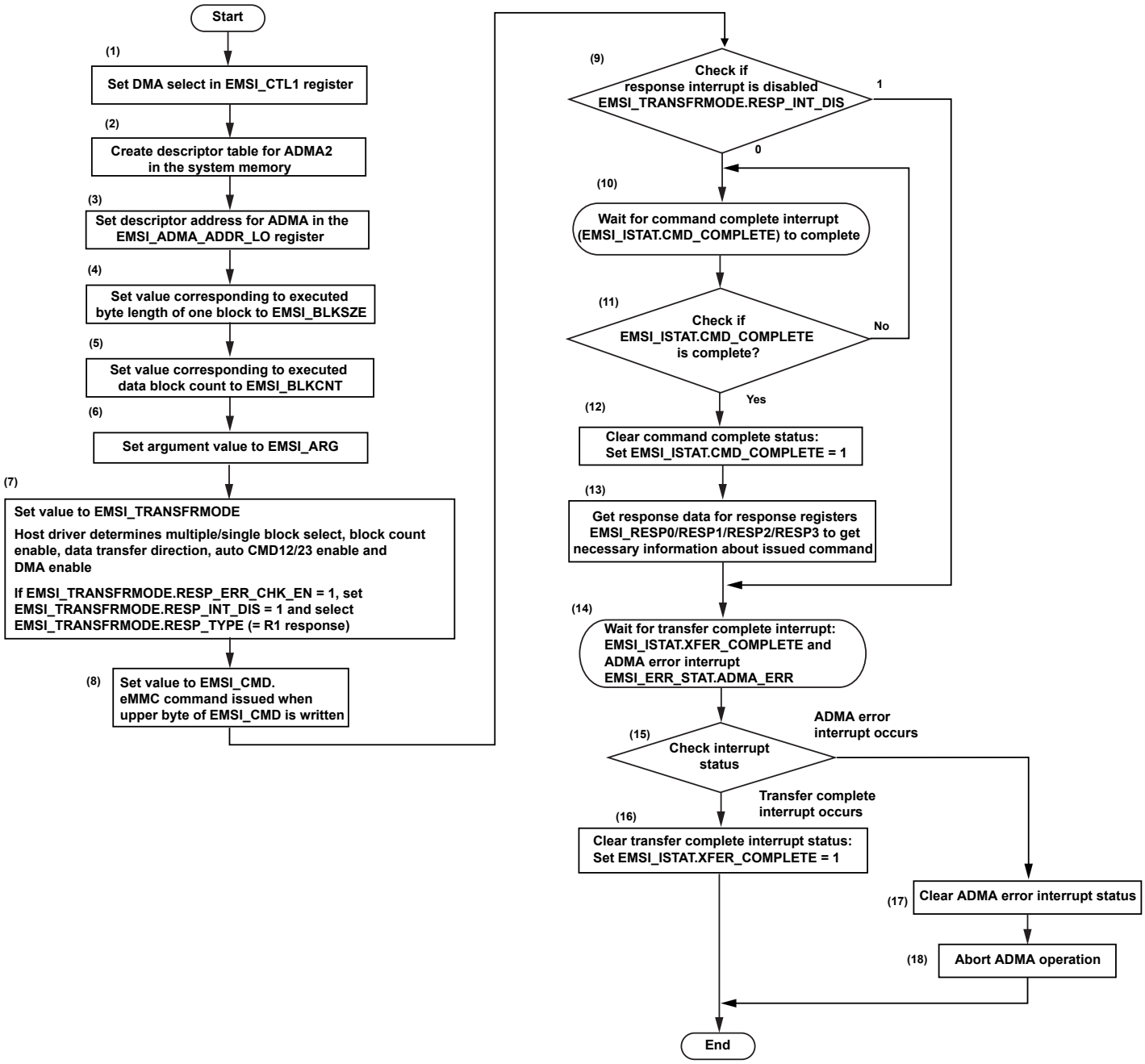


Figure 18-30: Transaction Control with Data Transfer Using DAT Line (Using ADMA2)

1. Configure DMA select in the `EMSI_CTL1.DMA_SEL` bit field.
2. Create a descriptor table for ADMA in the system memory.
3. Configure the descriptor address for ADMA in the `EMSI_ADMA_ADDR_LO` registers
4. Configure the value corresponding to the executed byte length of one block in the `EMSI_BLKSZ` register.

5. Configure the value corresponding to the executed data block count in the EMSI_BLKCNT register.
6. Configure the argument value in the EMSI_ARG register.
7. Configure the EMSI_TRNSFRMODE register (= R1 response).

   The host driver determines multiple/single block select, block count enable, data transfer direction, auto CMD12 enable and DMA enable. If EMSI_TRNSFRMODE.RESP_ERR_CHK_EN = 1, set EMSI_TRNSFRMODE.RESP_INT_DIS = 1 and select EMSI_TRNSFRMODE.RESP_TYPE.
8. Configure the argument value in the EMSI_CMD register. The eMMC command is issued when the upper byte of the EMSI_CMD register is written.
9. Check if the response interrupt is disabled in the EMSI_TRNSFRMODE.RESP_INT_DIS bit.
   a. If 1 - go to step 14.
   b. If 0 - go to step 10.
10. Wait for the command complete interrupt (EMSI_ISTAT.CMD_COMPLETE) to complete.
11. Check if EMSI_ISTAT.CMD_COMPLETE is complete.
   a. If no - go back to step 10.
   b. If yes - go to step 12.
12. Clear command complete status by setting EMSI_ISTAT.CMD_COMPLETE = 1
13. Get response data for response registers EMSI_RESP0/EMSI_RESP1/EMSI_RESP2/EMSI_RESP3 to get the necessary information about the issued command.
14. Wait for transfer complete interrupt in the EMSI_ISTAT.XFER_COMPLETE bit and the ADMA error interrupt in the EMSI_ERR_STAT.ADMA_ERR bit.
15. Check the interrupt status.
   a. Transfer complete interrupt occurs - go to step 19.
   b. DMA interrupt occurs - go to step 17
16. Clear transfer complete interrupt status by setting EMSI_ISTAT.XFER_COMPLETE = 1
17. Clear ADMA error interrupt status.
18. Abort ADMA operation.

NOTE: During card read/write DMA operations, it is recommended that the very last transfer descriptor also has the End bit set for marginal performance improvement. The End bit should always be set on a descriptor of type TRAN (not for link descriptor or other type descriptor). Descriptor types are provided in the following table for quick reference. The descriptor type recommendation in Table 18-16 Descriptor Types is applicable only to ADMA2 and ADMA3 transfers.

Table 18-16: Descriptor Types

Act2	Act1	Act0	Symbol	Comment	ADMA2 Descriptor Address
0	0	0	NOP	No operation	Do not execute current line and go to next line
0	1	0	rsv	Reserved	Same as NOP
1	0	0	Tran	Transfer data	Transfer data of one descriptor line
1	1	0	Link	Link descriptor	Link to another descriptor

Issuing CMD with Data Transfer (Using ADMA3)

The *Transaction Control with Data Transfer Using DAT Line (Using ADMA3)* figure and procedure describe transaction control using the DAT line for ADMA3 data transfers.

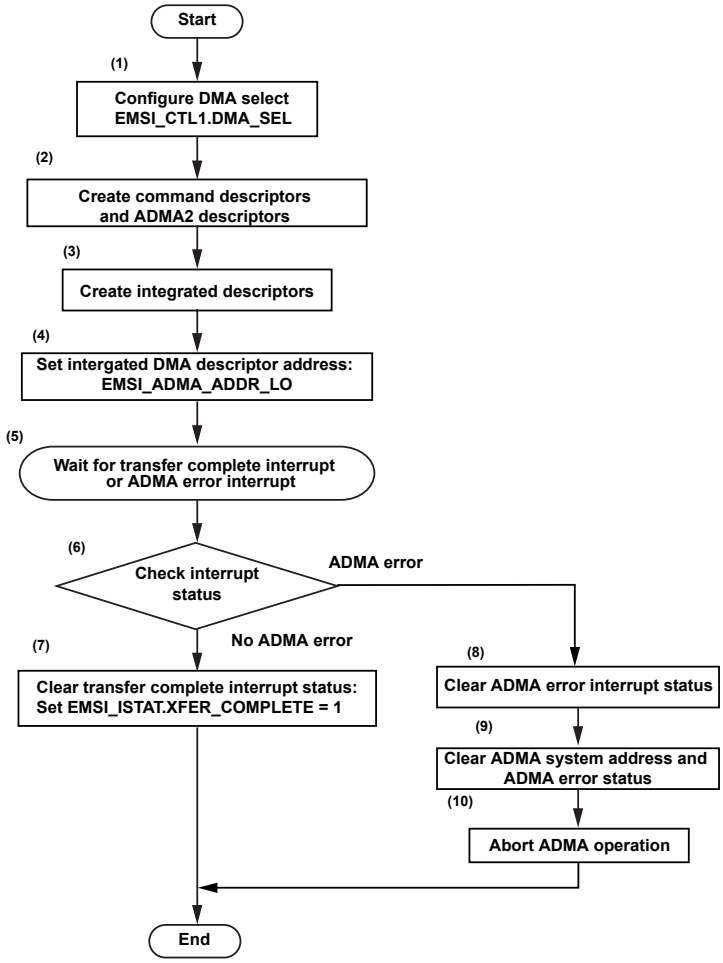


Figure 18-31: Transaction Control with Data Transfer Using DAT Line (Using ADMA3)

1. Configure DMA select.
2. Create command descriptors and ADMA2 descriptors.

3. Create integrated descriptors.
4. Configure the integrated DMA descriptor address in the EMSI_ADMA_ADDR_LO register.
5. Wait for transfer complete interrupt or ADMA error interrupt.
6. Check interrupt status.
    a. ADMA error - go to step 8.
    b. No ADMA error - go to step 7.
7. Clear transfer complete interrupt status by setting EMSI_ISTAT.XFER_COMPLETE = 1.
8. Clear the ADMA error interrupt status
9. Clear the ADMA system address and ADMA error status.
10. Abort the ADMA operation.

The following list describes the descriptor requirements for an ADMA3 CMD:

- Configuring infinite data transfer (EMSI_TRNSFRMODE.BLOCK_COUNT_EN = 0) is not allowed except for single block transfer (EMSI_TRNSFRMODE.MULTI_BLK_SEL = 0)
- As ADMA3 only supports 32-bit block count mode, EMSI_TRNSFRMODE.BLOCK_COUNT_EN is set to 1 and 16-bit block count (EMSI_BLKCNT.VALUE) is configured to 0x0000. Configuring stop count (32-bit block count = 0) is not allowed except for single block transfers (EMSI_TRNSFRMODE.MULTI_BLK_SEL = 0).
- Configuring no data transfer (Block Size = 0) is not allowed, except for no data transfer command in the command descriptor (CDEnd = 1).
- In cases where the memory data transfer command is used, the Auto CMD Auto Select option (0x3) in the EMSI_TRNSFRMODE.AUTO_CMD_EN bit field is used so that the command descriptor is independent of eMMC device support CMD23. For other commands, auto command disabled (0x0) is used. If EMSI_CTL2.CMD23_EN = 1 then Auto CMD23 is issued before the transfer commands and if EMSI_CTL2.CMD23_EN = 0, then Auto CMD12 is issued after the transfer complete. For SD cards that do not support CMD23 (card versions 2.00 and below) configure the EMSI_CTL2 register accordingly.

During card read/write DMA operations, it is recommended that the very last transfer descriptor also has the End bit set for marginal performance improvement. The End bit should always be set on a descriptor of type TRAN. The following table is applicable only to ADMA2 and ADMA3 transfers.

Table 18-17: Descriptor Types

Act2	Act1	Act0	Symbol	Comment	ADMA2 Descriptor Address
0	0	0	NOP	No Operation	Do not execute current line and go to the next line

Table 18-17: Descriptor Types (Continued)

Act2	Act1	Act0	Symbol	Comment	ADMA2 Descriptor Address
0	1	0	RSV	Reserved	Same as Nop. Do not execure current line and go to next line
1	0	0	TRAN	Transfer Data	Transfer data of one descriptor line
1	1	0	LINK	Link Descriptor	Link to another descriptor

Command Queueing

The following sections describe the processes for initializing the command queuing engine and submitting and completing a task in the Command Queuing engine.

Initializing the Command Queuing Engine

The *Initializing the Command Queuing Engine* figure shows the programming sequence to initialize the command queuing engine.

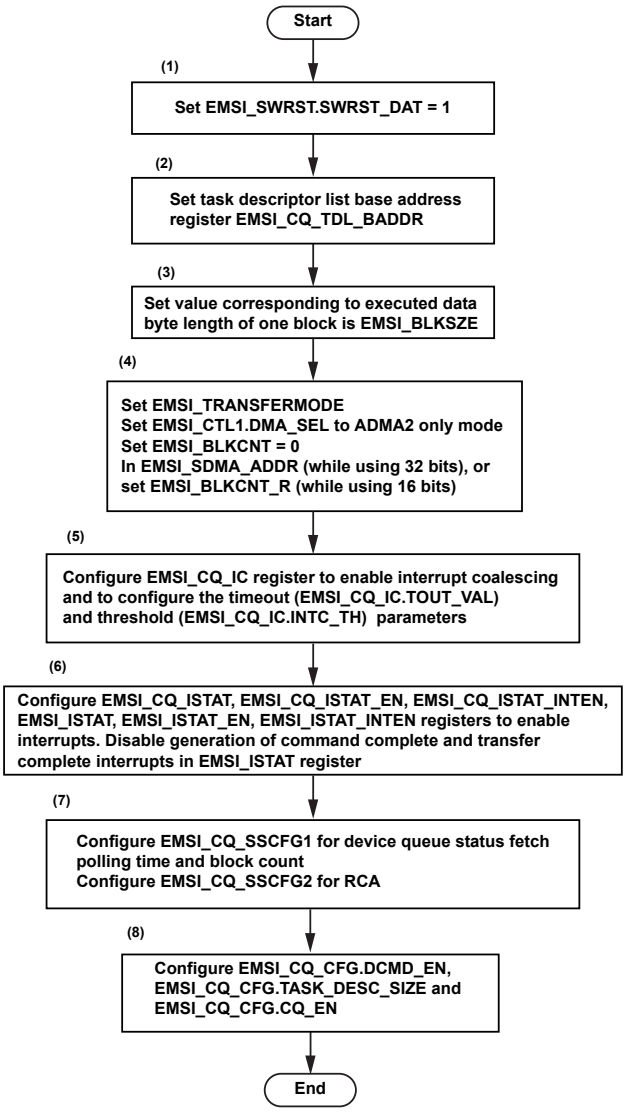


Figure 18-32: Initializing the Command Queuing Engine

1. Set `EMSI_SWRST.DAT` = 1.
2. Configure the task descriptor list base address register, `EMSI_CQ_TDL_BADDR`.
3. Configure a value corresponding to executed data byte length of one block in the `EMSI_BLKSZ` register.
4. Configure the following registers and bits:
    a. `EMSI_TRNSFRMODE` register.
    b. Configure `EMSI_CTL1.DMA_SEL` to ADMA2 only mode.
    c. Clear `EMSI_BLKCNT` = 0.

   d. In the EMSI_SDMA_ADDR register (while using 32 bits) or configure the EMSI_BLKCNT register (while using 16 bits).

5. Configure the command queuing interrupt coalescing register to enable interrupt coalescing and to configure the timeout (EMSI_CQ_IC.TOUT_VAL) and threshold (EMSI_CQ_IC.INTC_TH) parameters.
6. Configure the following interrupt registers to enable interrupts.
   a. EMSI_CQ_ISTAT, EMSI_CQ_ISTAT_EN, EMSI_CQ_ISTAT_INTEN.
   b. EMSI_ISTAT, EMSI_ISTAT_EN, EMSI_ISTAT_INTEN.
   c. Disable generation of command complete and transfer complete interrupts (EMSI_ISTAT.XFER_COMPLETE, EMSI_ISTAT.CMD_COMPLETE).
7. Configure the EMSI_CQ_SSCFG1 register for device queue status fetch polling time and block count. Configure the EMSI_CQ_SSCFG2 register for RCA.
8. Configure the EMSI_CQ_CFG.DCMD_EN, EMSI_CQ_CFG.TASK_DESC_SIZE and EMSI_CQ_CFG.CQ_EN bits.

Check the EXT_CSD register byte 308 for command queuing support from card side. Byte 307 of the EXT_CSD register denotes the command queuing depth that the device supports. The program must also enable command queuing from the eMMC device side before enabling from the eMSI controller side (CMDQ_MODE_EN [15] in the EXT_CSD register).

Submitting and Completing a Task in the Command Queuing Engine

The *Submitting and Completing a Task* figure and procedure shows the sequence to submit and complete a task in a command queuing engine.

NOTE: Task descriptors and transfer descriptors in the host memory are aligned to an 8-byte boundary.

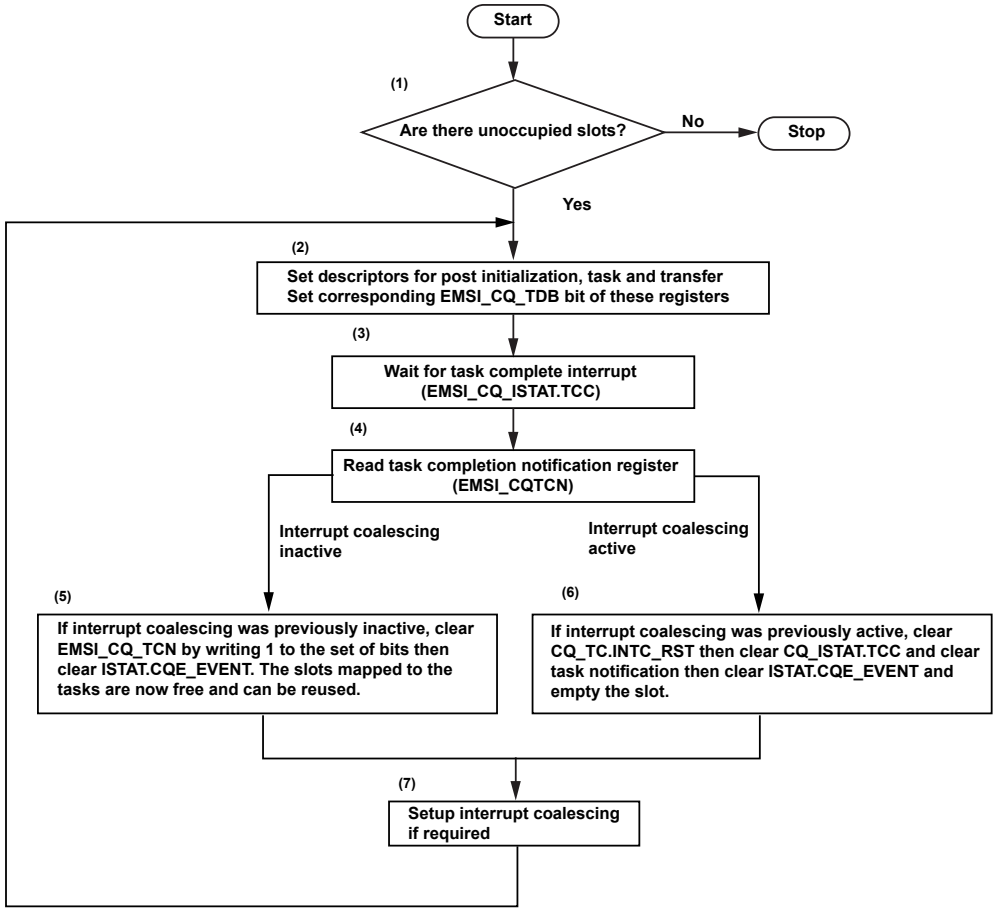


Figure 18-33: Submitting and Completing a Task

1. Are there unoccupied slots?
    a. If no - Stop.
    b. If yes - go to step 2.
2. Configure descriptors for post initialization, task, and transfer. Set the corresponding `EMSI_CQ_TDBR` bit of these registers.
3. Wait for the task complete interrupt (`EMSI_CQ_ISTAT.TCC`).
4. Read the task completion notification register (`EMSI_CQ_TCN`). If interrupt coalescing is inactive go to step 5. If interrupt coalescing is active go to step 6.
5. If interrupt coalescing was previously inactive, clear the `EMSI_CQ_TCN` register by writing 1 to the set of bits then clear the `EMSI_ISTAT.CQE_EVENT` bit. The slots mapped to the tasks are now free and can be reused.

6. If interrupt coalescing was previously active, clear the EMSI_CQ_IC.INTC_RST bit then clear the EMSI_CQ_ISTAT.TCC bit and clear task notification. Then clear the EMSI_ISTAT.CQE_EVENT bit and empty the slot.
7. Setup interrupt coalescing if required.

CQE Error Recovery

The *CQE Recovery Mode* figure and procedure shows the error recovery sequence for CQE mode.

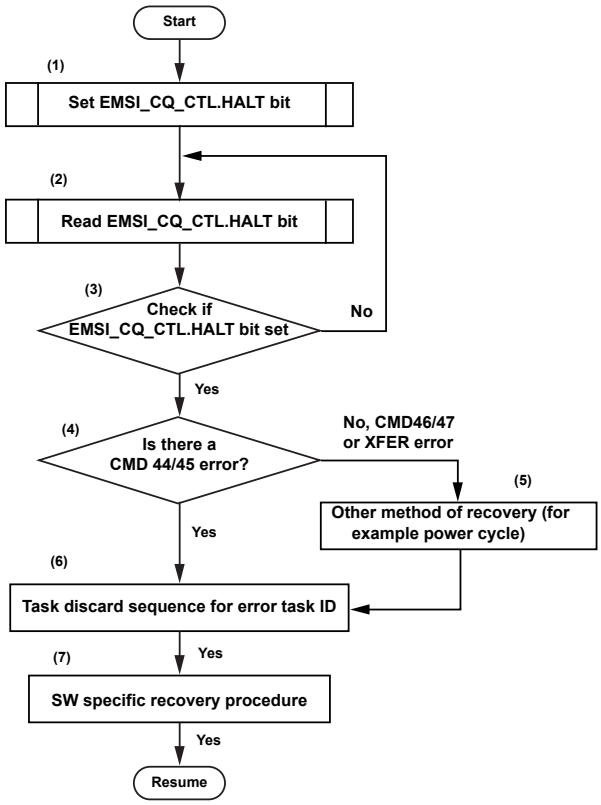


Figure 18-34: CQE Recovery Mode

1. Set the EMSI_CQ_CTL.HALT bit.
2. Read the EMSI_CQ_CTL.HALT bit.
3. Check if EMSI_CQ_CTL.HALT bit set.
    a. No - go to step 2.
    b. Yes - go to step 4.
4. Is there a CMD 44/45 error?
    a. No, CMD46/47 or XFER error - go to step 5.
    b. Yes - go to step 6.

5. Other method of recovery (for example power cycle).
6. Task discard sequence for error task ID.
7. SW specific recovery procedure.

NOTE: Writes to the Command Queuing DoorBell register (EMSI_CQ_TDBR) after requesting a HALT (by writing 1 to the EMSI_CQ_CTL.HALT bit field may be ignored by the eMSI controller. An ignored EMSI_CQ_TDBR write generates an error.

eMMC Device Boot Programming Sequences

Software must adhere to the following steps when working with boot operation. The partition that from the eMSI controller reads the boot data, can be selected in advance using EXT_CSD byte [179], bits [5:3]. The eMSI controller can choose to use the

- Single data rate mode with backward-compatible interface timing,
- Single data rate with high-speed interface timing

The EXT_CSD register byte [228], bit 2 tells the eMSI controller if the high-speed timing during boot is supported by the device.

Configure the eMMC device before configuring the eMSI controller for boot operation.

Please refer to *JEDEC eMMC 5.1* specification for more information.

NOTE: The eMSI controller does not support boot operation in DDR speed mode.

Preparing for a Boot

The *Programming Sequence to Prepare for a Boot* figure and procedure shows the programming sequence to prepare for a boot. This sequence is common for both the mandatory and alternate modes.

NOTE: The eMSI controller does not support boot in ADMA3 mode.

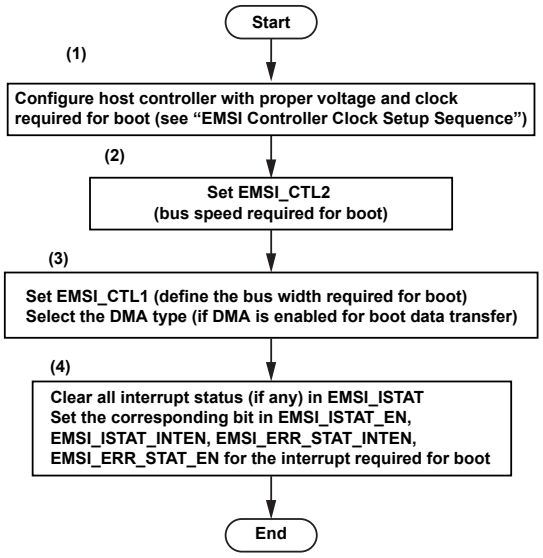


Figure 18-35: Programming Sequence to Prepare for a Boot

1. Configure the host controller with the proper voltage and clock required for booting (see eMSI Controller Clock Setup Sequence).
2. Configure the `EMSI_CTL2` register with the bus speed required for boot.
3. Configure the `EMSI_CTL1` register to define the bus width required for boot. Select the DMA type (if DMA is enabled for boot data transfer).
4. Clear all interrupt status (if any) in the `EMSI_ISTAT` register. Configure the corresponding bits in the `EMSI_ISTAT_EN`,`EMSI_ISTAT_INTEN`, `EMSI_ERR_STAT_INTEN` and `EMSI_ERR_STAT_EN` registers for the interrupt required for boot.

Initiating a Mandatory Boot in SDMA Mode

The *Programming Sequence to Initiate a Mandatory Boot in SDMA Mode* figure and procedure shows the programming sequence to initiate a mandatory boot in SDMA mode.

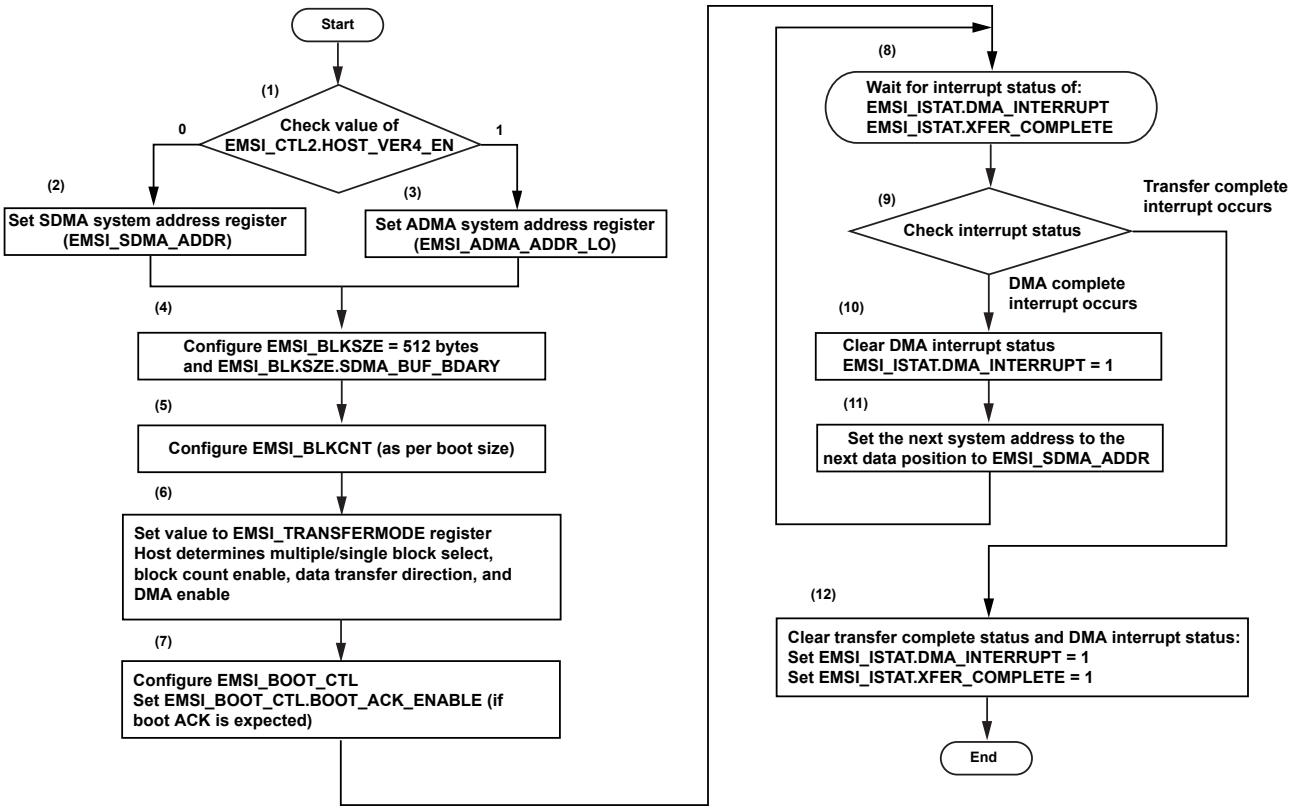


Figure 18-36: Programming Sequence to Initiate a Mandatory Boot in SDMA Mode

1. Check the value of the `EMSI_CTL2.HOST_VER4_EN` bit field.
   a. 0 - go to step 2.
   b. 1 - go to step 3
2. Configure the SDMA system address register (`EMSI_SDMA_ADDR`).
3. Configure the ADMA system address register (`EMSI_ADMA_ADDR_LO`).
4. Configure `EMSI_BLKSZ` = 512 bytes.
5. Configure `EMSI_BLKCNT` (as per boot size).
6. Configure the `EMSI_TRNSFRMODE` register. The host determines multiple/single block select, block count enable, data transfer direction, and DMA enable.
7. Configure the `EMSI_BOOT_CTL` register. Set `EMSI_BOOT_CTL.BOOT_ACK_ENABLE` if boot ACK is expected.
8. Wait for interrupt status of the `EMSI_ISTAT.DMA_INTERRUPT` and `EMSI_ISTAT.XFER_COMPLETE` bits.
9. Check interrupt status.

a. Transfer complete interrupt occurs - go to step 12.

b. DMA complete interrupt occurs - go to step 10.

10. Clear the DMA interrupt status (`EMSI_ISTAT.DMA_INTERRUPT` = 1).
11. Configure the next system address to the next data position to `EMSI_SDMA_ADDR`.
12. Clear the transfer complete status and DMA interrupt status bits. Set `EMSI_ISTAT.DMA_INTERRUPT` = 1 and `EMSI_ISTAT.XFER_COMPLETE` = 1.

Initiating a Mandatory Boot in ADMA2 Mode

The *Programming Sequence to Initiate a Mandatory Boot in ADMA2 Mode* figure shows the programming sequence for initiating a mandatory boot in ADMA2 mode.

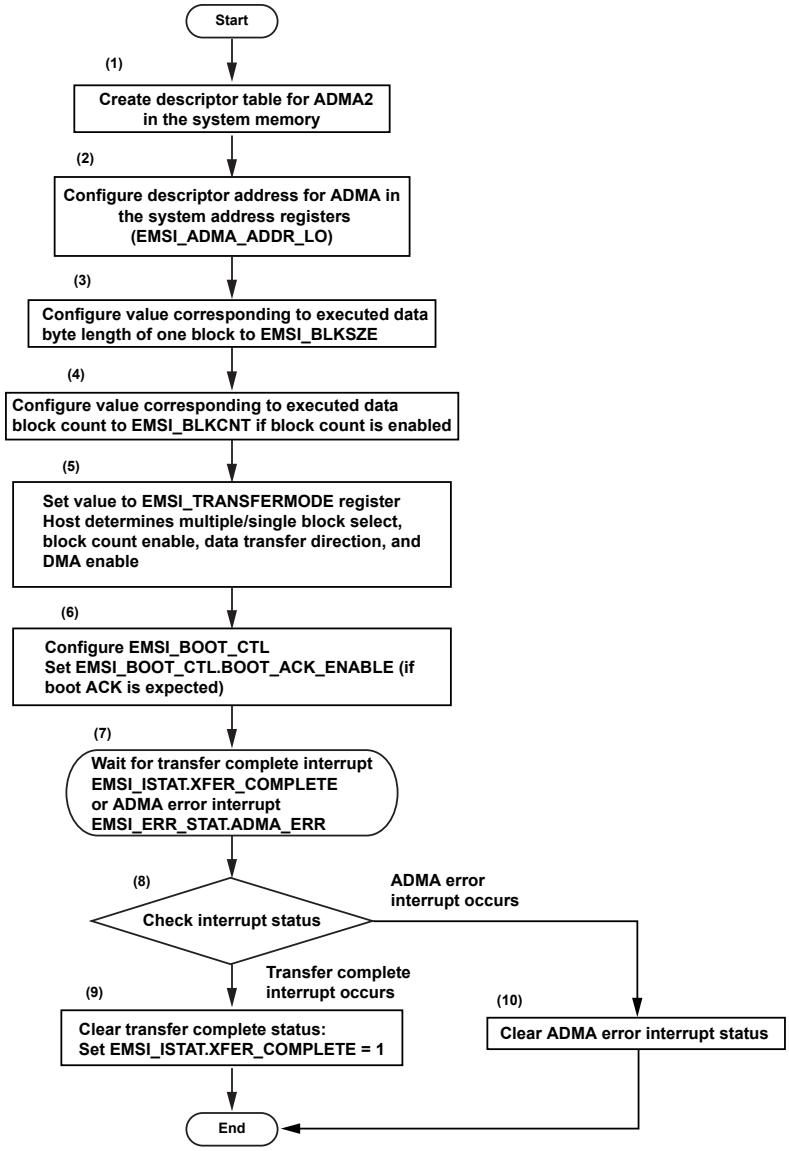


Figure 18-37: Programming Sequence to Initiate a Mandatory Boot in ADMA2 Mode

1. Create a descriptor table for ADMA in the system memory.
2. Configure the descriptor address for ADMA in the system address registers (EMSI_ADMA_ADDR_LO).
3. Configure the value corresponding to the executed data byte length of one block to the EMSI_BLKSZ register.
4. Configure the value corresponding to the executed data block count to the EMSI_BLKCNT register when block count is enabled.
5. Configure the EMSI_TRNSFRMODE register. The host determines multiple/single block select, block count enable, data transfer direction, and DMA enable.

6. Configure the EMSI_BOOT_CTL register. Set the EMSI_BOOT_CTL.BOOT_ACK_ENABLE bit (if boot ACK is expected).
7. Wait for the transfer complete interrupt EMSI_ISTAT.XFER_COMPLETE or ADMA error interrupt EMSI_ERR_STAT.ADMA_ERR.
8. Check interrupt status.
    a. Transfer complete interrupt occurs - go to step 9.
    b. ADMA error interrupt occurs - go to step 10.
9. Clear transfer complete status by setting EMSI_ISTAT.XFER_COMPLETE = 1.
10. Clear ADMA error interrupt status.

Initiating Alternate Boot

Alternate Boot is a command-based boot operation and can be initiated by issuing CMD0 with the 0xFFFFFFFA argument. For the programming sequence for initiating an alternate boot, see Issuing CMD with Data Transfer (Using SDMA) and Issuing CMD with Data Transfer (Using ADMA2) with the following exceptions:

- If boot acknowledgment is expected, the EMSI_BOOT_CTL.BOOT_TOUT_CNT is programmed to configure the timeout counter for boot acknowledgment before setting the EMSI_CMD register.
- Response check cannot be performed as CMD0 has no response.
- Issue CMD0 with 0xFFFFFFFA argument with EMSI_CMD.DATA_PRESENT_SEL set to 1.

When the boot is initiated, the device is assumed to be in pre-boot state and 74 clock cycles has expired after power is stable before issuing CMD0.

SD Transaction Mode

This section includes the sequences to generate and control different types of SD transactions that can be classified as transactions without data transfer using the DAT/CMD line and transactions with data transfer using the DAT/CMD line. The following procedures are included in this section.

- SD Card Initialization and Identification
- Changing Bus Width for an SD Card
- Changing Bus Speed Mode for an SD Card
- Issuing CMD Without Data Transfer
- Issuing CMD With Data Transfer

SD Card Initialization and Identification

The *SD Card Initialization and Identification* figure shows the sequence for SD card initialization and identification.

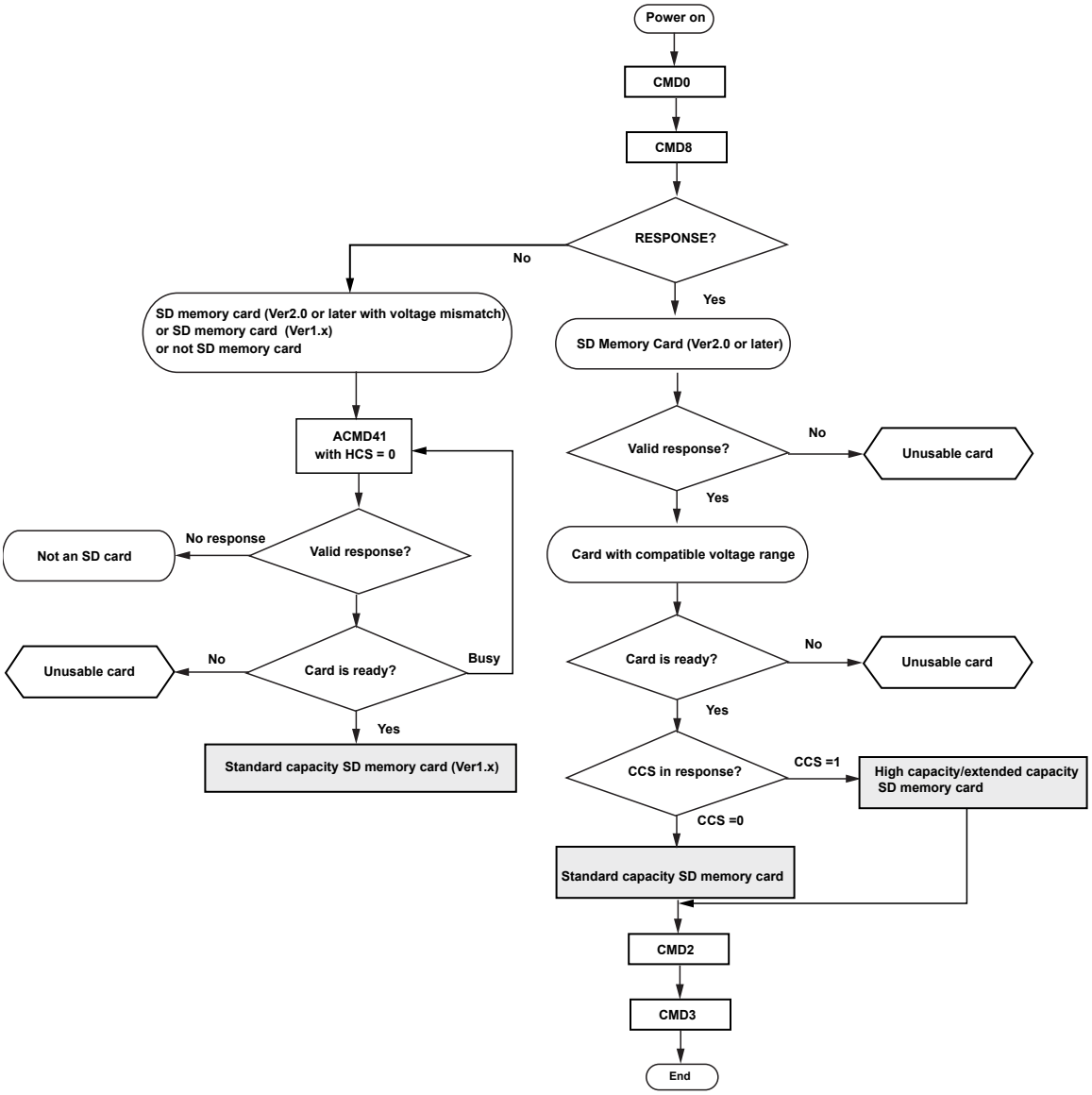


Figure 18-38: SD Card Initialization and Identification

Changing Bus Width for an SD Card

The *SD Card Change Bus Width* figure shows the sequence for changing the bit mode in a SD card.

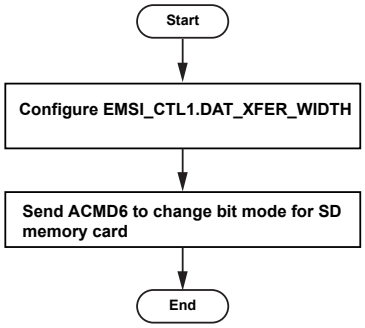


Figure 18-39: SD Card Change Bus Width

Changing Bus Speed Mode for an SD Card

The *Changing Bus Speed Mode* figure shows the process for changing bus speed mode in a SD card.

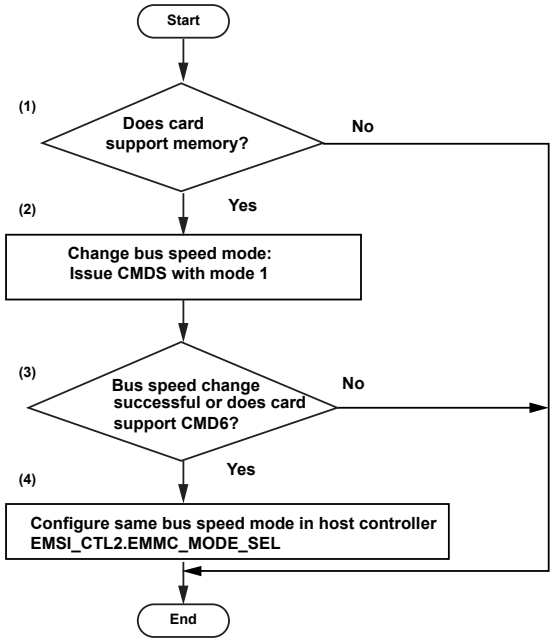


Figure 18-40: Changing Bus Speed Mode

Issuing CMD With or Without Data Transfers for an SD Card

Programs can issue commands with or without data transfers.

- The programming sequence for issuing a command without a data transfer for an SD card is like the sequence described in the Issuing CMD Without Data Transfer section.
- The programming sequence for issuing a command with a data transfer for an SD card is like the programming sequence described in the sections following Issuing CMD With Data Transfer.

  Data transfers can be performed using DMA. The supported DMA modes are SDMA, ADMA2, and ADMA3.

ADSP-SC59x EMSI Register Descriptions

The Enhanced Mobile Storage Interface (EMSI) contains the following registers.

Table 18-18: ADSP-SC59x EMSI Register List

Name	Description
EMSI_ADMA_ADDR_LO	ADMA System Address Register - Low
EMSI_ADMA_DESADDR_LO	ADMA3 Integrated Descriptor Address Register - Low
EMSI_ADMA_ERR_STAT	ADMA Error Status Register
EMSI_ARG	Argument Register
EMSI_AUTOCMD_STAT	Auto CMD Status Register
EMSI_BASEADDR0	Base Address 0 Register
EMSI_BASEADDR1	Base Address 1 Register
EMSI_BLKCNT	16-bit Block Count Register
EMSI_BLKSZ	Block Size Register
EMSI_BOOT_CTL	eMMC Boot Control Register
EMSI_CAP1	Capabilities 1 Register - 0 to 31
EMSI_CAP2	Capabilities Register - 32 to 63
EMSI_CLK_CTL	Clock Control Register
EMSI_CMD	Command Register
EMSI_CQVER	Command Queuing Version Register
EMSI_CQ_CAP	Command Queuing Capabilities Register
EMSI_CQ_CFG	Command Queuing Configuration Register
EMSI_CQ_CRARG	CQ Command Response Argument Register
EMSI_CQ_CRDCT	Command Response for Direct Command Register
EMSI_CQ_CRI	CQ Command Response Index
EMSI_CQ_CTL	Command Queuing Control Register
EMSI_CQ_DPT	Device Pending Tasks Register
EMSI_CQ_DQSTAT	Device Queue Status Register
EMSI_CQ_IC	Command Queuing Interrupt Coalescing Register
EMSI_CQ_ISTAT	Command Queuing Interrupt Status Register
EMSI_CQ_ISTAT_EN	Command Queuing Interrupt Status Enable Register
EMSI_CQ_ISTAT_INTEN	Command Queuing Interrupt Signal Enable Register
EMSI_CQ_RMEM	Command Response Mode Error Mask Register
EMSI_CQ_SSCFG1	CQ Send Status Configuration 1 Register

Table 18-18: ADSP-SC59x EMSI Register List (Continued)

Name	Description
EMSI_CQ_SSCFG2	CQ Send Status Configuration 2 Register
EMSI_CQ_TCLR	Command Queuing Task Clear Register
EMSI_CQ_TCN	Command Queuing Task Completion Notification Register
EMSI_CQ_TDBR	Command Queuing DoorBell Register
EMSI_CQ_TDL_BADDR	Command Queuing Task Descriptor List Base Address Register
EMSI_CQ_TERRINFO	CQ Task Error Information Register
EMSI_CTL1	EMSI Control 1 Register
EMSI_CTL2	EMSI Control 2 Register
EMSI_HOST_CNTRL_VERS	SD Host Controller Specification Version
EMSI_EMMC_CTL	eMMC Control Register
EMSI_ERR_STAT	Error Interrupt Status Register
EMSI_ERR_STAT_EN	Error Interrupt Status Enable Register
EMSI_ERR_STAT_INTEN	Error Interrupt Signal Enable Register
EMSI_FRC_AUTOCMDSTAT	Force Event Register for Auto CMD Error Status Register
EMSI_FRC_ERRSTAT	Force Event Register for Error Interrupt Status
EMSI_ISTAT	Interrupt Status Register
EMSI_ISTAT_EN	Interrupt Status Enable Register
EMSI_ISTAT_INTEN	Interrupt Signal Enable Register
EMSI_PRESET_DS	Preset Value for Default Speed
EMSI_PRESET_HSDDR	Preset Value for HSDDR
EMSI_PRESET_HSSDR	Preset Value for HSSDR
EMSI_PRESET_INIT	Preset Value for Initialization
EMSI_PRESET_LEGACY	Preset Value for Legacy Mode
EMSI_PSTATE	Present State Register
EMSI_RESP0	Response Register 0
EMSI_RESP1	Response Register 1
EMSI_RESP2	Response Register 2
EMSI_RESP3	Response Register 3
EMSI_SDMA_ADDR	SDMA System Address Register
EMSI_SWRST	Software Reset Register
EMSI_TO_CTL	Timeout Control Register

Table 18-18: ADSP-SC59x EMSI Register List (Continued)

Name	Description
EMSI_TRNSFRMODE	Transfer Mode Register
EMSI_VER_ID	EMSI Version
EMSI_VER_TYPE	EMSI Version Type
EMSI_WU_CTL	Wakeup Control Register

ADMA System Address Register - Low

The EMSI_ADMA_ADDR_LO register holds the 32-bit system address for DMA transfer. This register is applicable for eMMC mode.

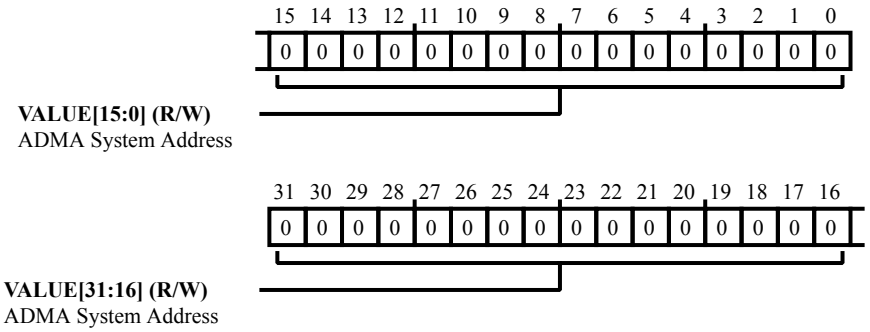


Figure 18-41: EMSI_ADMA_ADDR_LO Register Diagram

Table 18-19: EMSI_ADMA_ADDR_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	ADMA System Address. The EMSI_ADMA_ADDR_LO.VALUE bits indicate the 32 bits of the ADMA system address. -SDMA: If Host Version 4 Enable is set (=1), this register stores the system address of the data location -ADMA2: This register stores the byte address of the executing command of the descriptor table -ADMA3: This register is set by ADMA3. ADMA2 increments the address of this register that points to the next line, every time a descriptor line is fetched.

ADMA3 Integrated Descriptor Address Register - Low

The EMSI_ADMA_DESADDR_LO register holds the 32-bit Integrated Descriptor address.This register is applicable for eMMC mode.

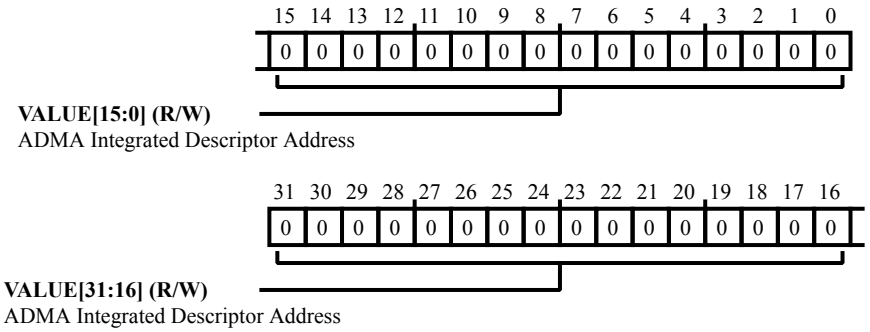


Figure 18-42: EMSI_ADMA_DESADDR_LO Register Diagram

Table 18-20: EMSI_ADMA_DESADDR_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	ADMA Integrated Descriptor Address. The EMSI_ADMA_DESADDR_LO.VALUE bits indicate the 32-bit of the ADMA integrated descriptor address. The start address of integrated descriptor is set to these register bits. The ADMA3 fetches one descriptor address and increments these bits to indicate the next descriptor address.

ADMA Error Status Register

The EMSI_ADMA_ERR_STAT register stores the ADMA state during an ADMA error. This register is applicable for an eMMC mode.

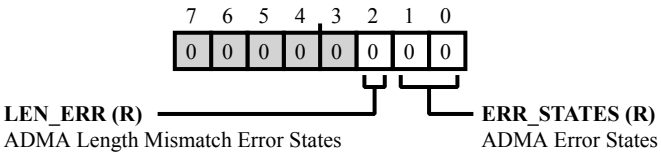


Figure 18-43: EMSI_ADMA_ERR_STAT Register Diagram

Table 18-21: EMSI_ADMA_ERR_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	LEN_ERR	ADMA Length Mismatch Error States. This error occurs in the following instances: - While the block count enable is being set, the total data length specified by the descriptor table is different from that specified by the block count and block length - When the total data length cannot be divided by the block length	
		0	No Error
		1	Error
1:0 (R/NW)	ERR_STATES	ADMA Error States. The EMSI_ADMA_ERR_STAT.ERR_STATES bits indicate the state of ADMA when an error occurs during ADMA data transfer.	
		0	Stop DMA - SYS_ADR Register Points to a Location Next to the Error Descriptor
		1	Fetch Descriptor - SYS_ADR Register Points to the Error Descriptor
		2	Reserved
		3	Transfer Data - SYS_ADR Register Points to a Location Next to the Error Descriptor

Argument Register

The EMSI_ARG register is used to configure the eMMC command argument.

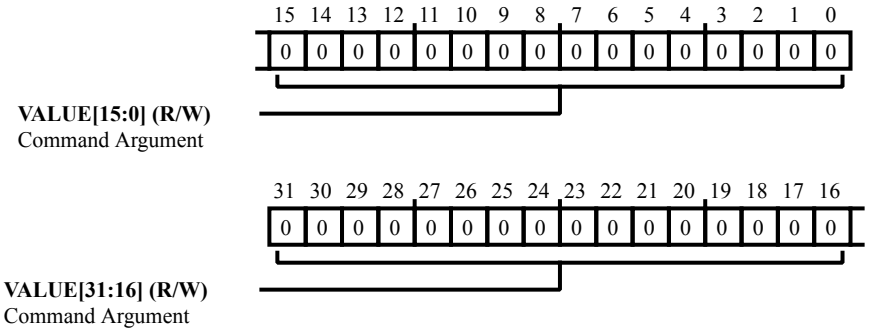


Figure 18-44: EMSI_ARG Register Diagram

Table 18-22: EMSI_ARG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command Argument. The EMSI_ARG.VALUE bits specify the eMMC command argument that is specified in bits 39-8 of the Command format.

Auto CMD Status Register

The EMSI_AUTOCMD_STAT register is used to indicate the CMD12 response error of Auto CMD12, and the CMD23 response error of Auto CMD23. The host driver can determine the kind of Auto CMD12/CMD23 errors that can occur in this register. Auto CMD23 errors are indicated in bit 04-01.

This register is valid only when Auto CMD Error is set in the EMSI_ERR_STAT register. This register is applicable for eMMC mode.

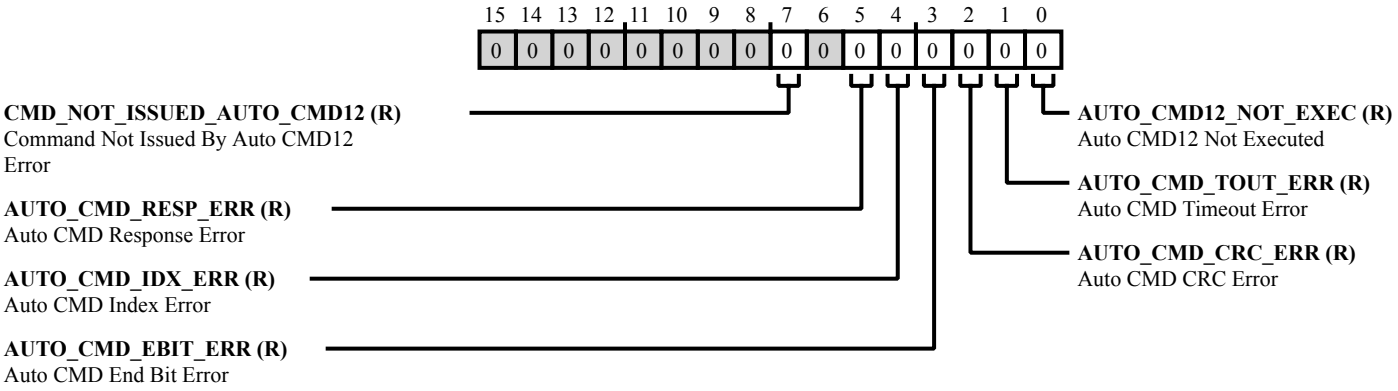


Figure 18-45: EMSI_AUTOCMD_STAT Register Diagram

Table 18-23: EMSI_AUTOCMD_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	CMD_NOT_ISSUED_AUTO_CMD12	Command Not Issued By Auto CMD12 Error. If the EMSI_AUTOCMD_STAT.CMD_NOT_ISSUED_AUTO_CMD12 bit is set (=1), CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register. The EMSI_AUTOCMD_STAT.CMD_NOT_ISSUED_AUTO_CMD12 bit is cleared (= 0) when Auto CMD Error is generated by Auto CMD23.	
		0	No Error
		1	Not Issued
5 (R/NW)	AUTO_CMD_RESP_ERR	Auto CMD Response Error. The EMSI_AUTOCMD_STAT.AUTO_CMD_RESP_ERR bit is set when Response Error Check Enable in the Transfer Mode register is set (=1) and an error is detected in R1 response of either Auto CMD12 or CMD13. This status is ignored if any bit between D00 to D04 is set (=1).	
		0	No Error
		1	Error

Table 18-23: EMSI_AUTOCMD_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	AUTO_CMD_IDX_ERR	Auto CMD Index Error. The EMSI_AUTOCMD_STAT.AUTO_CMD_IDX_ERR bit is set if the command index error occurs in response to a command.	
		0	No Error
		1	Error
3 (R/NW)	AUTO_CMD_EBIT_ERR	Auto CMD End Bit Error. The EMSI_AUTOCMD_STAT.AUTO_CMD_EBIT_ERR bit is set when detecting that the end bit of command response is 0.	
		0	No Error
		1	End Bit Error Generated
2 (R/NW)	AUTO_CMD_CRC_ERR	Auto CMD CRC Error. The EMSI_AUTOCMD_STAT.AUTO_CMD_CRC_ERR bit is set when detecting a CRC error in the command response.	
		0	No Error
		1	CRC Error Generated
1 (R/NW)	AUTO_CMD_TOUT_ERR	Auto CMD Timeout Error. The EMSI_AUTOCMD_STAT.AUTO_CMD_TOUT_ERR bit is set if no response is returned with 64 eMSI bus clock cycles from the end bit of the command. If the EMSI_AUTOCMD_STAT.AUTO_CMD_TOUT_ERR bit is set (=1), error status bits (D04-D01) are meaningless.	
		0	No Error
		1	Time Out
0 (R/NW)	AUTO_CMD12_NOT_EXEC	Auto CMD12 Not Executed. If multiple memory block data transfer is not started due to a command error, the EMSI_AUTOCMD_STAT.AUTO_CMD12_NOT_EXEC bit is not set because it is not necessary to issue an Auto CMD12. When the EMSI_AUTOCMD_STAT.AUTO_CMD12_NOT_EXEC bit is set (=1), the eMSI controller cannot issue Auto CMD12 to stop multiple memory block data transfer, due to some error. If the EMSI_AUTOCMD_STAT.AUTO_CMD12_NOT_EXEC bit is set (=1), error status bits (D04-D01) is meaningless. The EMSI_AUTOCMD_STAT.AUTO_CMD12_NOT_EXEC bit is cleared (=0) when Auto CMD Error is generated by Auto CMD23.	
		0	Executed
		1	Not Executed

Base Address 0 Register

The EMSI_BASEADDR0 register used as a pointer for the base address 0 register.

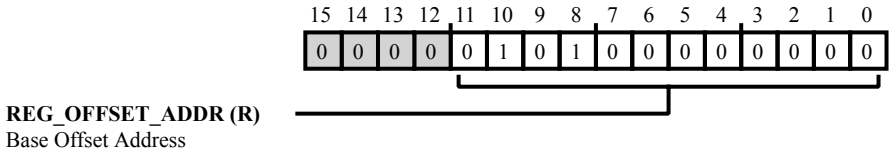


Figure 18-46: EMSI_BASEADDR0 Register Diagram

Table 18-24: EMSI_BASEADDR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/NW)	REG_OFFSET_ADDR	Base Offset Address.

Base Address 1 Register

The EMSI_BASEADDR1 register used as a pointer for the base address 1 register.

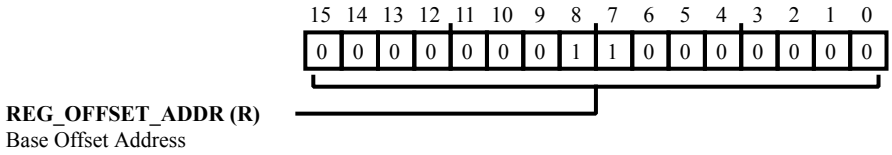


Figure 18-47: EMSI_BASEADDR1 Register Diagram

Table 18-25: EMSI_BASEADDR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	REG_OFFSET_ADDR	Base Offset Address.

16-bit Block Count Register

The EMSI_BLKCNT register is used to configure the number of data blocks. This register is applicable for eMMC mode.

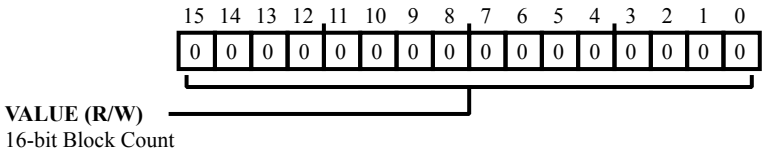


Figure 18-48: EMSI_BLKCNT Register Diagram

Table 18-26: EMSI_BLKCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Block Count. The EMSI_BLKCNT.VALUE bit field has the following settings: If the Host Version 4 Enable bit is set 0 or the 16-bit Block Count register is set to non-zero, the 16-bit Block Count register is selected. If the Host Version 4 Enable bit is set 1 and the 16-bit Block Count register is set to zero, the 32-bit Block Count register is selected. Following are the values for this bit field: 0x0: Stop Count 0x1: 1 Block 0x2: 2 Blocks ... - ... 0xFFFF: 65535 Blocks Note: For Host Version 4 Enable = 0, this register must be set to 0000h before programming the 32-bit block count register when Auto CMD23 is enabled for non-DMA and ADMA modes.

Block Size Register

The EMSI_BLKSZ register is used to configure an SDMA buffer boundary and the number of bytes in a data block. This register is applicable for eMMC mode.

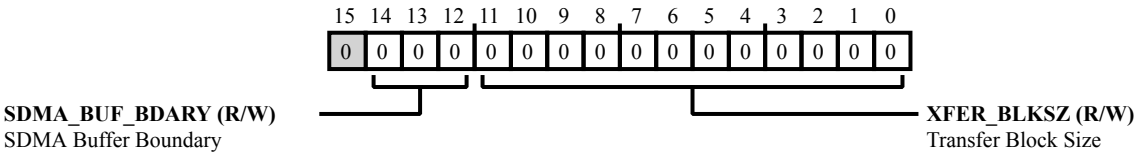


Figure 18-49: EMSI_BLKSZ Register Diagram

Table 18-27: EMSI_BLKSZ Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
14:12 (R/W)	SDMA_BUF_BDARY	SDMA Buffer Boundary. The EMSI_BLKSZ.SDMA_BUF_BDARY bits specify the size of contiguous buffer in system memory. The SDMA transfer waits at every boundary specified by these fields and the eMSI controller generates the DMA interrupt to request the host driver to update the SDMA system address register.	
		0	4K Bytes SDMA Buffer Boundary
		1	8K Bytes SDMA Buffer Boundary
		2	16K Bytes SDMA Buffer Boundary
		3	32K Bytes SDMA Buffer Boundary
		4	64K Bytes SDMA Buffer Boundary
		5	128K Bytes SDMA Buffer Boundary
		6	256K Bytes SDMA Buffer Boundary
		7	512K Bytes SDMA Buffer Boundary

Table 18-27: EMSI_BLKSZ Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	XFER_BLKSZ	Transfer Block Size. The `EMSI_BLKSZ.XFER_BLKSZ` bits specify the block size of data transfers. When using an eMMC device, this bit field is set to 512 bytes. The `EMSI_BLKSZ.XFER_BLKSZ` bits can be accessed only if no transaction is executing. Read operations to the bit field during transfers may return an invalid value, and write operations are ignored. The following are the value settings. - 0x1: 1 byte - 0x2: 2 bytes - 0x3: 3 bytes - ..... - 0x1FF: 511 byte - 0x200: 512 bytes - ..... - 0x800: 2048 bytes Note: This register must be programmed with a non-zero value for data transfer.

eMMC Boot Control Register

The EMSI_BOOT_CTL register is used to control the eMMC Boot operation.

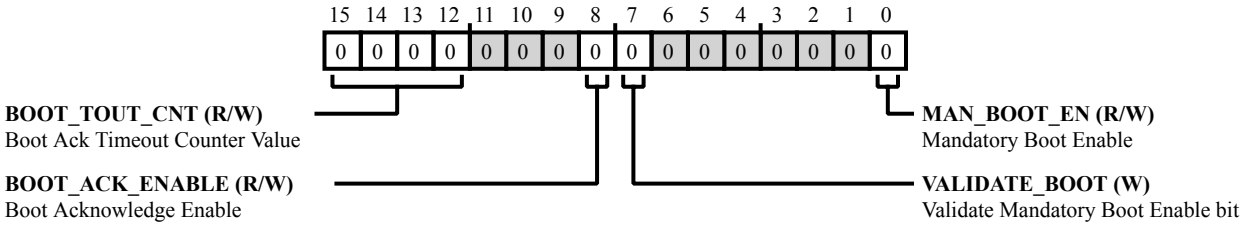


Figure 18-50: EMSI_BOOT_CTL Register Diagram

Table 18-28: EMSI_BOOT_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15:12 (R/W)	BOOT_TOUT_CNT	Boot Ack Timeout Counter Value. The EMSI_BOOT_CTL.BOOT_TOUT_CNT value determines the interval by which boot ack timeout (50 ms) is detected when boot ack is expected during boot operation. - 0xF: Reserved - 0xE: TMCLK x 2^27 -... - 0x1: TMCLK x 2^14 - 0x0: TMCLK x 2^13 NOTE: Please refer to EMSI_CAP1.TOUT_CLK_FREQ for TMCLK.	
8 (R/W)	BOOT_ACK_ENABLE	Boot Acknowledge Enable. When the EMSI_BOOT_CTL.BOOT_ACK_ENABLE bit set, eMSI controller checks for boot acknowledge start pattern of 0-1-0 during boot operation. The EMSI_BOOT_CTL.BOOT_ACK_ENABLE bit is applicable for both mandatory and alternate boot mode.	
		0	Boot Ack Disable
		1	Boot Ack Enable
7 (RX/W)	VALIDATE_BOOT	Validate Mandatory Boot Enable bit. The EMSI_BOOT_CTL.VALIDATE_BOOT bit is used to validate the MAN_BOOT_EN bit.	
		0	Ignore Mandatory Boot Enable Bit
		1	Validate Mandatory Boot Enable Bit

Table 18-28: EMSI_BOOT_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	MAN_BOOT_EN	Mandatory Boot Enable. The `EMSI_BOOT_CTL.MAN_BOOT_EN` bit is used to initiate the mandatory boot operation. The application sets the `EMSI_BOOT_CTL.MAN_BOOT_EN` bit along with VALIDATE_BOOT bit. Writing 0 is ignored. The eMSI controller clears the `EMSI_BOOT_CTL.MAN_BOOT_EN` bit after the boot transfer is completed or terminated.	
		0	Mandatory Boot Disable
		1	Mandatory Boot Enable

Capabilities 1 Register - 0 to 31

The EMSI_CAP1 register provides the host driver with information specific to the eMSI controller implementation.

The eMSI controller may implement these values as fixed or during power on initialization. This register is segregated into two 32-bit registers: EMSI_CAP1-2. The EMSI_CAP1 register is the lower part of capabilities register.

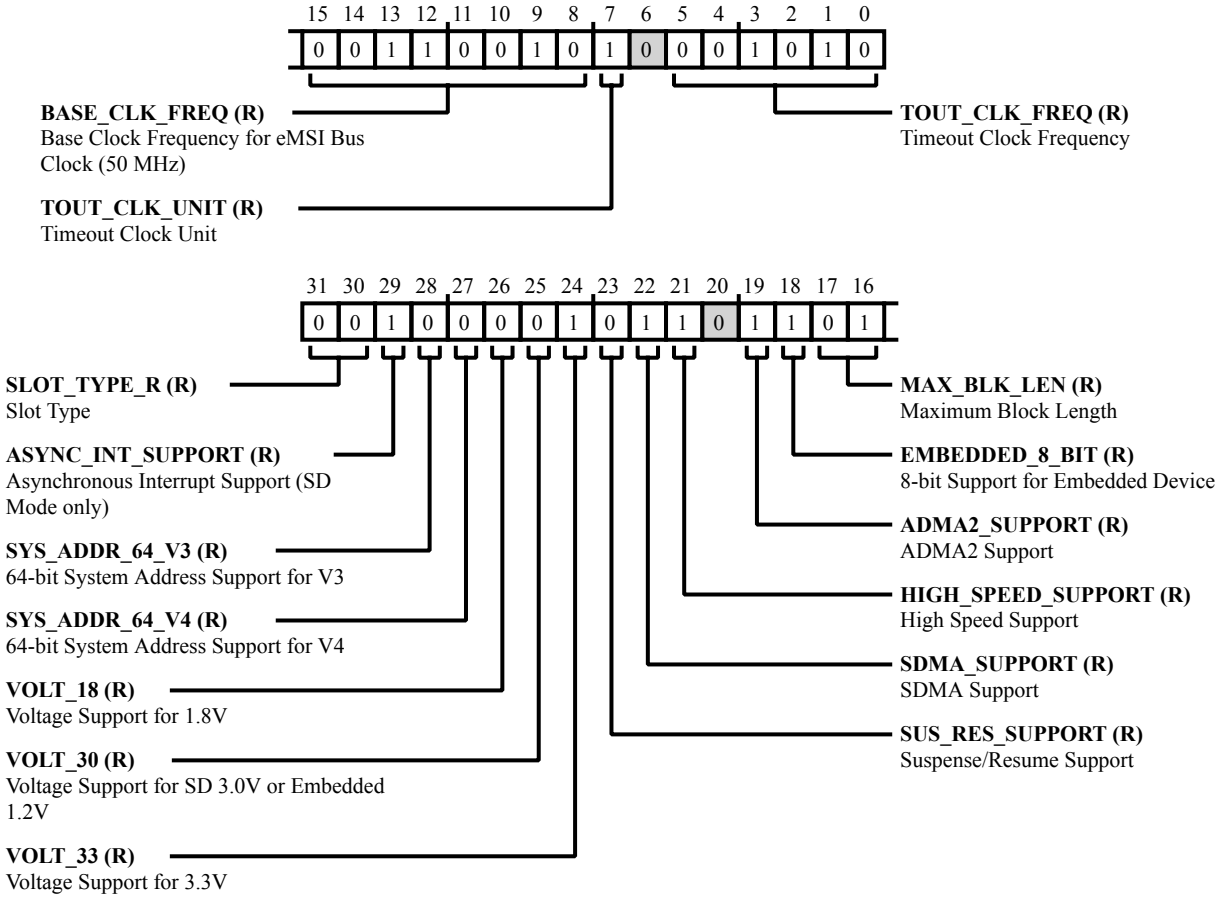


Figure 18-51: EMSI_CAP1 Register Diagram

Table 18-29: EMSI_CAP1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	SLOT_TYPE_R	Slot Type. The EMSI_CAP1.SLOT_TYPE_R bits indicate usage of a slot by a specific host system.	
		0	Removable Card Slot
		1	Embedded Slot for One Device
		2	Shared Bus Slot (SD Mode)
		3	UHS-II Multiple Embedded Devices
29 (R/NW)	ASYNC_INT_SUPPORT	Asynchronous Interrupt Support (SD Mode only).	
		0	Asynchronous Interrupt Not Supported
		1	Asynchronous Interrupt Supported
28 (R/NW)	SYS_ADDR_64_V3	64-bit System Address Support for V3. The EMSI_CAP1.SYS_ADDR_64_V3 bit sets the eMSI controller to support 64-bit system addressing of V3 mode. SDMA cannot be used in 64-bit addressing in Version 3 mode. If the EMSI_CAP1.SYS_ADDR_64_V3 bit is set (=1), to use 64-bit ADMA2 with a 96-bit descriptor is enabled by setting Host Version 4 Enable (EMSI_CTL2.HOST_VER4_EN= 0) and DMA select (EMSI_CTL1.DMA_SEL = 11b).	
		0	64-bit System Address for V3 is Not Supported
		1	64-bit System Address for V3 is Supported
27 (R/NW)	SYS_ADDR_64_V4	64-bit System Address Support for V4. The EMSI_CAP1.SYS_ADDR_64_V4 bit sets the eMSI controller to support 64-bit system addressing of V4 mode. When the EMSI_CAP1.SYS_ADDR_64_V4 bit is set (=1), full or part of 64-bit address must be used to decode the eMSI controller registers so that eMSI controller registers can be placed above system memory area. The 64-bit address decode of eMSI controller registers is effective regardless of setting to 64-bit addressing in the EMSI_CTL2 register. If the EMSI_CAP1.SYS_ADDR_64_V4 bit is set (=1), 64-bit DMA addressing for version 4 is enabled by setting Host Version 4 Enable (EMSI_CTL2.HOST_VER4_EN =1) and by setting 64-bit addressing (=1). SDMA can be used and ADMA2 uses a 128-bit descriptor.	
		0	64-bit System Address for V4 is Not Supported
		1	64-bit System Address for V4 is Supported

Table 18-29: EMSI_CAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/NW)	VOLT_18	Voltage Support for 1.8V.	
		0	1.8V Not Supported
		1	1.8V Supported
25 (R/NW)	VOLT_30	Voltage Support for SD 3.0V or Embedded 1.2V.	
		0	SD 3.0V or Embedded 1.2V Not Supported
		1	SD 3.0V or Embedded Supported
24 (R/NW)	VOLT_33	Voltage Support for 3.3V.	
		0	3.3V Not Supported
		1	3.3V Supported
23 (R/NW)	SUS_RES_SUPPORT	Suspense/Resume Support. The `EMSI_CAP1.SUS_RES_SUPPORT` bit indicates whether the eMSI controller supports suspend/resume functionality. If the `EMSI_CAP1.SUS_RES_SUPPORT` bit is 0, the host driver does not issue either suspend or resume commands because the suspend and resume mechanism is not supported.	
		0	Not Supported
		1	Supported
22 (R/NW)	SDMA_SUPPORT	SDMA Support. The `EMSI_CAP1.SDMA_SUPPORT` bit indicates whether the eMSI controller is capable of using SDMA to transfer data between the system memory and the host controller directly.	
		0	SDMA Not Supported
		1	SDMA Supported
21 (R/NW)	HIGH_SPEED_SUPPORT	High Speed Support. The `EMSI_CAP1.HIGH_SPEED_SUPPORT` bit indicates whether the eMSI controller and the host system supports high speed mode and they can supply the eMSI bus clock frequency from 25 MHz to 50 MHz.	
		0	High Speed Not Supported
		1	High Speed Supported
19 (R/NW)	ADMA2_SUPPORT	ADMA2 Support. The `EMSI_CAP1.ADMA2_SUPPORT` bit indicates whether the eMSI controller is capable of using ADMA2.	
		0	ADMA2 Not Supported
		1	ADMA2 Supported

Table 18-29: EMSI_CAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/NW)	EMBEDDED_8_BIT	8-bit Support for Embedded Device. The `EMSI_CAP1.EMBEDDED_8_BIT` bit indicates whether the eMSI controller is capable of using an 8-bit bus width mode. The `EMSI_CAP1.EMBEDDED_8_BIT` bit is not effective when the Slot Type is set to 10b.	
		0	8-bit Bus Width Not Supported
		1	8-bit Bus Width Supported
17:16 (R/NW)	MAX_BLK_LEN	Maximum Block Length. The `EMSI_CAP1.MAX_BLK_LEN` bit indicates the maximum block size that the host driver can read and write to the buffer in the eMSI controller. The buffer transfers this block size without wait cycles.	
		0	512 Byte
		1	1024 Byte
		2	2048 Byte
		3	Reserved

Table 18-29: EMSI_CAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:8 (R/NW)	BASE_CLK_FREQ	Base Clock Frequency for eMSI Bus Clock (50 MHz). The EMSI_CAP1.BASE_CLK_FREQ field indicate the base (maximum) clock frequency for the eMSI bus clock. The definition of these bits depend on the SD host controller version. - 6-Bit Base Clock Frequency: This mode is supported by the SD host controller version 1.00 and 2.00. The upper 2 bits are not effective and are always 0. The unit values are 1 MHz. The supported clock range is 10 MHz to 63 MHz. - 0x00: Get information through another method - 0x01: 1 MHz - 0x02: 2 MHz ..... - 0x3F: 63 MHz - 0x40-0xFF: Not Supported - 8-Bit Base Clock Frequency: This mode is supported by the SD host controller version 3.00. The unit values are 1 MHz. The supported clock range is 10 MHz to 255 MHz. - 0x00: Get information through another method - 0x01: 1 MHz - 0x02: 2 MHz -- ..... - 0xFF: 255 MHz If the frequency is 16.5 MHz, the larger value is set to 0001001b (17 MHz) because the host driver uses this value to calculate the clock divider value and it does not exceed the upper limit of the eMSI bus clock frequency. If the EMSI_CAP1.BASE_CLK_FREQ field are all 0, the host system has to get information using a different method.	
7 (R/NW)	TOUT_CLK_UNIT	Timeout Clock Unit. The EMSI_CAP1.TOUT_CLK_UNIT bit shows the unit of base clock frequency used to detect data timeout error.	
		0	KHz
		1	MHz

Table 18-29: EMSI_CAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/NW)	TOUT_CLK_FREQ	Timeout Clock Frequency. The `EMSI_CAP1.TOUT_CLK_FREQ` bit shows the base clock frequency used to detect Data Timeout Error. The timeout clock unit defines the unit of timeout clock frequency. It can be configured for kHz or MHz. - 0x00: Get information through another method - 0x01: 1KHz / 1MHz - 0x02: 2KHz / 2MHz - 0x03: 3KHz / 3MHz - ..... - 0x3F: 63KHz / 63MHz

Capabilities Register - 32 to 63

The EMSI_CAP2 register provides the host driver with information specific to the eMSI controller implementation.

The eMSI controller may implement these values as fixed or loaded from the flash memory during power on initialization. This register is segregated into two 32-bit registers: EMSI_CAP2-EMSI_CAP1 The EMSI_CAP2 register is the upper part of capabilities register.

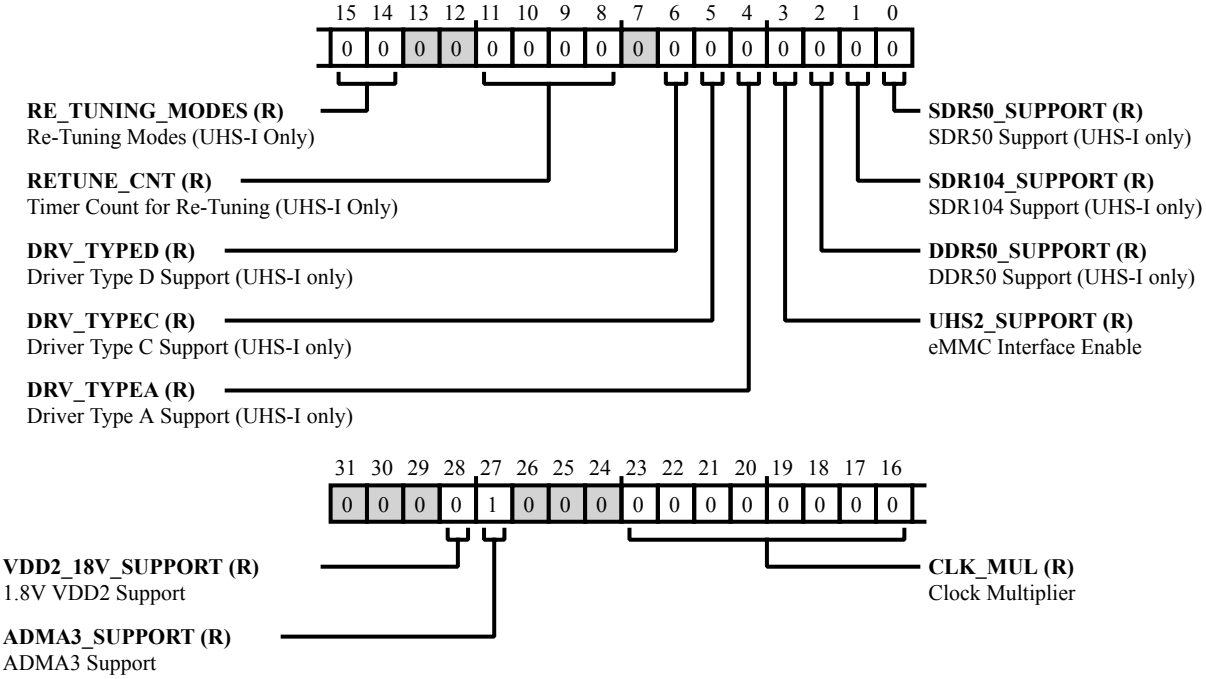


Figure 18-52: EMSI_CAP2 Register Diagram

Table 18-30: EMSI_CAP2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/NW)	VDD2_18V_SUPPORT	1.8V VDD2 Support. The EMSI_CAP2.VDD2_18V_SUPPORT bit indicates support of VDD2 for the Host System.	
		0	1.8V VDD2 is Not Supported
		1	1.8V VDD2 is Supported

Table 18-30: EMSI_CAP2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	ADMA3_SUPPORT	ADMA3 Support. The `EMSI_CAP2.ADMA3_SUPPORT` bit indicates whether the eMSI controller is capable of using ADMA3.	
		0	ADMA3 Not Supported
		1	ADMA3 Supported
23:16 (R/NW)	CLK_MUL	Clock Multiplier. The `EMSI_CAP2.CLK_MUL` field indicate the clock multiplier of the programmable clock generator. Clearing (=0) the `EMSI_CAP2.CLK_MUL` field indicates that the eMSI controller does not support a programmable clock generator. - 0x0: Clock Multiplier is not Supported - 0x1: Clock Multiplier M = 2 - 0x2: Clock Multiplier M = 3 - ..... - 0xFF: Clock Multiplier M = 256	
15:14 (R/NW)	RE_TUNING_MODES	Re-Tuning Modes (UHS-I Only). The `EMSI_CAP2.RE_TUNING_MODES` bits select the re-tuning method and limit the maximum data length.	
		0	Timer
		1	Timer and Re-Tuning Request (Not Supported)
		2	Auto Re-Tuning (for Transfer)
		3	Reserved
11:8 (R/NW)	RETUNE_CNT	Timer Count for Re-Tuning (UHS-I Only). - 0x0: Re-Tuning Timer disabled - 0x1: 1 seconds - 0x2: 2 seconds - 0x3: 4 seconds - ..... - 0xB: 1024 seconds - 0xC: Reserved - 0xD: Reserved - 0xE: Reserved - 0xF: Get information from other source	

Table 18-30: EMSI_CAP2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/NW)	DRV_TYPED	Driver Type D Support (UHS-I only). The `EMSI_CAP2.DRV_TYPED` bit indicates support of driver type D for 1.8 signaling.	
		0	Driver Type D is Not Supported
		1	Driver Type D is Supported
5 (R/NW)	DRV_TYPEC	Driver Type C Support (UHS-I only). The `EMSI_CAP2.DRV_TYPEC` bit indicates support of driver type C for 1.8 signaling.	
		0	Driver Type C is Not Supported
		1	Driver Type C is Supported
4 (R/NW)	DRV_TYPEA	Driver Type A Support (UHS-I only). The `EMSI_CAP2.DRV_TYPEA` bit indicates support of driver type A for 1.8 signaling.	
		0	Driver Type A is Not Supported
		1	Driver Type A is Supported
3 (R/NW)	UHS2_SUPPORT	eMMC Interface Enable. The `EMSI_CAP2.UHS2_SUPPORT` bit is used to enable the eMMC interface.	
		0	UHS-II is Not Supported
		1	Reserved
2 (R/NW)	DDR50_SUPPORT	DDR50 Support (UHS-I only).	
		0	DDR50 is Not Supported
		1	DDR50 is Supported
1 (R/NW)	SDR104_SUPPORT	SDR104 Support (UHS-I only). The `EMSI_CAP2.SDR104_SUPPORT` bit reports that SDR104 requires tuning.	
		0	SDR104 is Not Supported
		1	SDR104 is Supported
0 (R/NW)	SDR50_SUPPORT	SDR50 Support (UHS-I only). The `EMSI_CAP2.SDR50_SUPPORT` bit indicates that SDR50 is supported. The bit 13 (USE_TUNING_SDR50) indicates whether SDR50 requires tuning or not.	
		0	SDR50 is Not Supported
		1	SDR50 is Supported

Clock Control Register

The EMSI_CLK_CTL register controls the eMSI bus clock to an eMMC device. This register is applicable for eMMC mode.

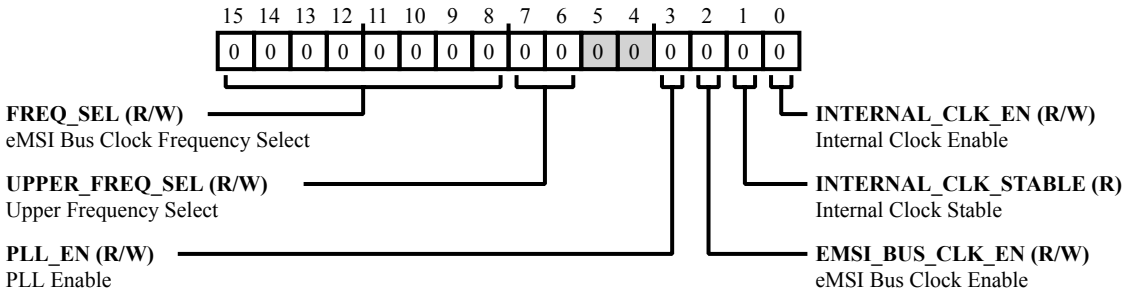


Figure 18-53: EMSI_CLK_CTL Register Diagram

Table 18-31: EMSI_CLK_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	FREQ_SEL	eMSI Bus Clock Frequency Select. The EMSI_CLK_CTL.FREQ_SEL bits are used to select the frequency of the eMSI bus clock signal. These bits depend on setting of (EMSI_CTL2.PRESET_VAL_EN). If preset value enable = 0, these bits are set by the host driver. If preset value enable = 1, these bits are automatically set to a value specified in one of the preset value register. The value is reflected on the lower 8-bit of the frequency divider signal. 10-bit Divided Clock Mode: - 0x3FF: 1/2046 divided clock - ..... - N : 1/2N divided clock - ..... - 0x002: 1/4 divided clock - 0x001: 1/2 divided clock - 0x000: base clock (0 to 50MHz)
7:6 (R/W)	UPPER_FREQ_SEL	Upper Frequency Select. The EMSI_CLK_CTL.UPPER_FREQ_SEL bits specify the upper 2 bits of 10-bit eMSI bus clock frequency select control. The value is reflected on the upper 2 bits of the frequency divider.

Table 18-31: EMSI_CLK_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	PLL_EN	PLL Enable. The EMSI_CLK_CTL.PLL_EN bit is used to activate the PLL (applicable when Host Version 4 Enable = 1). When Host Version 4 Enable = 0, INTERNAL_CLK_EN bit may be used to activate PLL. Note: If the EMSI_CLK_CTL.PLL_EN bit is not used to active the PLL when EMSI_CTL2.HOST_VER4_EN = 1, it is recommended to set the EMSI_CLK_CTL.PLL_EN bit to 1.	
		0	PLL is in Low Power Mode
		1	PLL is Enabled
2 (R/W)	EMSI_BUS_CLK_EN	eMSI Bus Clock Enable. The EMSI_CLK_CTL.EMSI_BUS_CLK_EN bit stops the SDCLK when cleared (=0). The SDCLK Frequency Select bit can be changed when the EMSI_CLK_CTL.EMSI_BUS_CLK_EN bit is cleared (=0).	
		0	Disable Providing eMSI Bus Clock
		1	Enable Providing eMSI Bus Clock
1 (R/NW)	INTERNAL_CLK_STABLE	Internal Clock Stable. The EMSI_CLK_CTL.INTERNAL_CLK_STABLE bit enables the host driver to check the clock stability twice after the Internal Clock Enable bit is set and after the PLL Enable bit is set.	
		0	Not Ready
		1	Ready
0 (R/W)	INTERNAL_CLK_EN	Internal Clock Enable. The EMSI_CLK_CTL.INTERNAL_CLK_EN bit is cleared (=0) when the host driver is not using the eMSI controller. The eMSI controller must stop its internal clock to enter a very low power state. However, registers can still be read and written to. If the EMSI_CLK_CTL.INTERNAL_CLK_EN bit is not used to control the internal clock, it is recommended to set the EMSI_CLK_CTL.INTERNAL_CLK_EN bit to 1.	
		0	Stop
		1	Oscillate

Command Register

The EMSI_CMD register is used to provide the information related to a command and a response packet. This register is applicable for eMMC mode. Writing to the upper byte of this register triggers eMMC command generation.

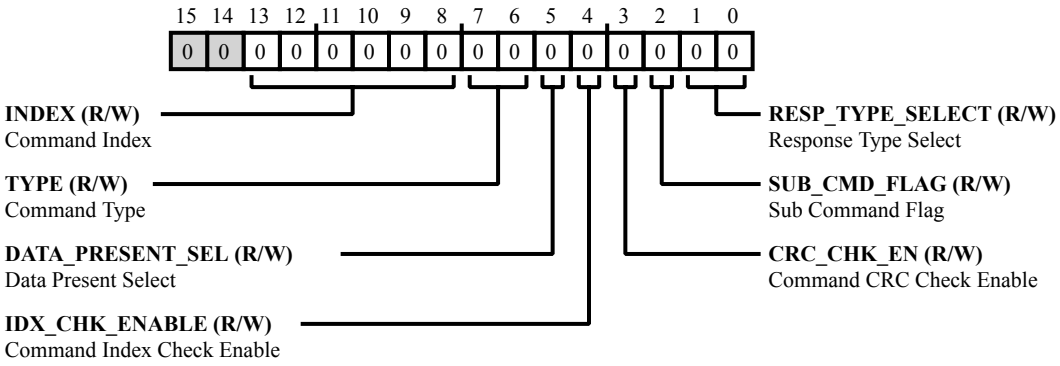


Figure 18-54: EMSI_CMD Register Diagram

Table 18-32: EMSI_CMD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
13:8 (R/W)	INDEX	Command Index. The EMSI_CMD.INDEX bits are set to the command number that is specified in bits 45-40 of the Command Format.	
7:6 (R/W)	TYPE	Command Type. The EMSI_CMD.TYPE bits indicate the command type. Note: While issuing Abort CMD (CMD12) this field is set to 0x3.	
		0	Normal
		1	Reserved
		2	Reserved
		3	Abort
5 (R/W)	DATA_PRESENT_SEL	Data Present Select. The EMSI_CMD.DATA_PRESENT_SEL bit is set (=1) to indicate that data is present and that the data is transferred using the DAT line. The EMSI_CMD.DATA_PRESENT_SEL bit is cleared (=0) in the following instances: - Command using the CMD line - Command with no data transfer but using busy signal on the DAT[0] line	
		0	No Data Present
		1	Data Present

Table 18-32: EMSI_CMD Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	IDX_CHK_ENABLE	Command Index Check Enable. The `EMSI_CMD.IDX_CHK_ENABLE` bit enables the eMSI controller to check the index field in the response to verify if it has the same value as the command index. If the value is not the same, it is reported as a Command Index error. Note: `EMSI_CMD.IDX_CHK_ENABLE` must be cleared (=0) for the command with no response, R2 response, R3 response and R4 response.	
		0	Disable
		1	Enable
3 (R/W)	CRC_CHK_EN	Command CRC Check Enable. The `EMSI_CMD.CRC_CHK_EN` bit enables the eMSI controller to check the CRC field in the response. If an error is detected, it is reported as a Command CRC error. Note: CRC Check enable must be cleared (=0) for the command with no response, R3 response, and R4 response.	
		0	Disable
		1	Enable
2 (R/W)	SUB_CMD_FLAG	Sub Command Flag. The `EMSI_CMD.SUB_CMD_FLAG` bit distinguishes between a main command and a sub command. The Sub Command flag is set =1 while issuing an abort command (CMD12) while using asynchronous abort.	
		0	Main Command
		1	Sub Command
1:0 (R/W)	RESP_TYPE_SELECT	Response Type Select. The `EMSI_CMD.RESP_TYPE_SELECT` bit indicates the type of response expected from the card.	
		0	No Response
		1	Response Length 136
		2	Response Length 48
		3	Response Length 48; Check Busy After Response

Command Queuing Version Register

The EMSI_CQVER register provides information about the version of the eMMC Command Queueing standard, which is implemented by the CQE in BCD format.

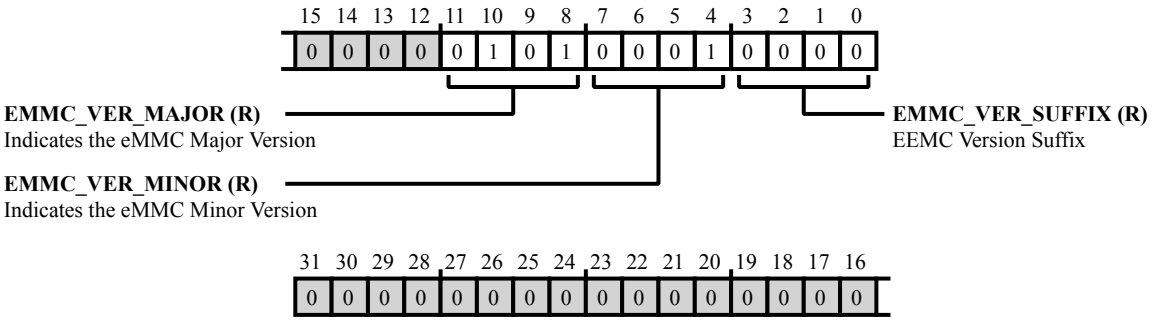


Figure 18-55: EMSI_CQVER Register Diagram

Table 18-33: EMSI_CQVER Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:8 (R/NW)	EMMC_VER_MAJOR	Indicates the eMMC Major Version. The EMSI_CQVER.EMMC_VER_MAJOR bit indicates the eMMC major version (1st digit left of decimal point) in BCD format
7:4 (R/NW)	EMMC_VER_MINOR	Indicates the eMMC Minor Version. The EMSI_CQVER.EMMC_VER_MINOR bit indicates the eMMC minor version (1st digit right of decimal point) in BCD format
3:0 (R/NW)	EMMC_VER_SUFFIX	EEMC Version Suffix. The EMSI_CQVER.EMMC_VER_SUFFIX bit indicates the eMMC version suffix (2nd digit right of decimal point) in BCD format

Command Queuing Capabilities Register

The EMSI_CQ_CAP register indicates the capabilities of the command queuing engine.

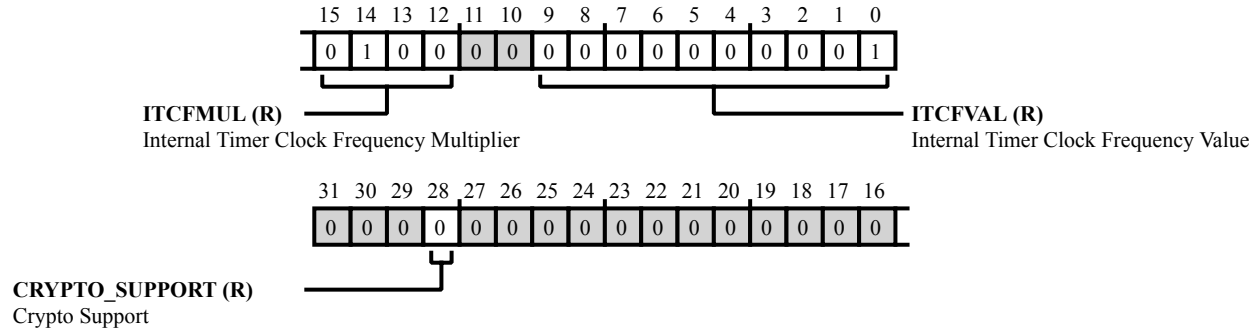


Figure 18-56: EMSI_CQ_CAP Register Diagram

Table 18-34: EMSI_CQ_CAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/NW)	CRYPTO_SUPPORT	Crypto Support. The EMSI_CQ_CAP.CRYPTO_SUPPORT bit indicates whether the Host Controller supports cryptographic operations.	
		0	Crypto Not Supported
		1	Crypto Supported
15:12 (R/NW)	ITCFMUL	Internal Timer Clock Frequency Multiplier. The EMSI_CQ_CAP.ITCFMUL bit field indicates the frequency of the clock used for interrupt coalescing timer and for determining the SQS polling period. See the EMSI_CQ_CAP.ITCFVAL definition for details. Values 0x5 to 0xF are reserved.	
		0	1KHz Clock
		1	10KHz Clock
		2	100KHz Clock
		3	1MHz Clock
		4	10MHz Clock
9:0 (R/NW)	ITCFVAL	Internal Timer Clock Frequency Value. The EMSI_CQ_CAP.ITCFVAL bit field scales the frequency of the timer clock provided by EMSI_CQ_CAP.ITCFMUL. The final clock frequency of actual timer clock is calculated as ITCFVAL x ITCFMUL. The default value is 10MHz.	

Command Queuing Configuration Register

The EMSI_CQ_CFG register controls CQE behavior affecting the general operation of command queuing engine.

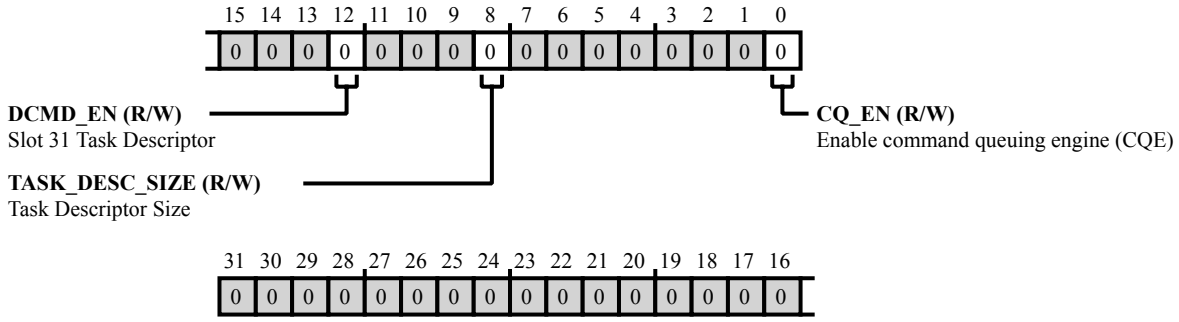


Figure 18-57: EMSI_CQ_CFG Register Diagram

Table 18-35: EMSI_CQ_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	DCMD_EN	Slot 31 Task Descriptor. The EMSI_CQ_CFG.DCMD_EN bit indicates to the hardware whether the task descriptor in slot #31 of the TDL is a data transfer descriptor or a direct-command descriptor. The CQE uses the EMSI_CQ_CFG.DCMD_EN bit when a task is issued in slot #31, to determine how to decode the Task Descriptor.	
		0	Task Descriptor in Slot #31 is a Data Transfer Task Descriptor
		1	Task Descriptor in Slot #31 is a DCMD Task Descriptor
8 (R/W)	TASK_DESC_SIZE	Task Descriptor Size. The EMSI_CQ_CFG.TASK_DESC_SIZE bit indicates the size of task descriptor used in host memory. The EMSI_CQ_CFG.TASK_DESC_SIZE bit can only be configured when Command Queuing Enable bit is 0 (command queuing is disabled).	
		0	Task Descriptor Size is 64 Bits
		1	Task Descriptor Size is 128 Bits

Table 18-35: EMSI_CQ_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	CQ_EN	Enable command queuing engine (CQE). When the CQE is disable, the software controls the eMSI bus using the registers between EMSI_SDMA_ADDR (offset 0x000) to EMSI_HOST_CNTRL_VERS (offset 0x0FE). Before the software writes to the EMSI_CQ_CFG.CQ_EN bit, the software verifies that the eMSI controller is in idle state and there are no ongoing commands or data transfers. When software wants to exit command queuing mode, it clears all previous tasks (if any) before clearing (=0) the EMSI_CQ_CFG.CQ_EN bit.	
		0	Disable Command Queuing
		1	Enable Command Queuing

CQ Command Response Argument Register

The EMSI_CQ_CRARG register stores the argument of the last received command response.

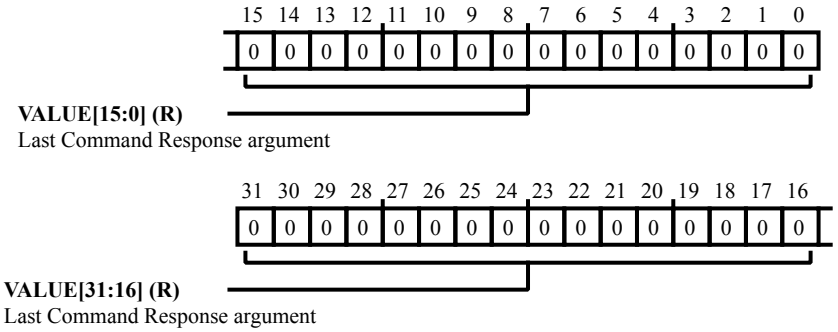


Figure 18-58: EMSI_CQ_CRARG Register Diagram

Table 18-36: EMSI_CQ_CRARG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Last Command Response argument. The EMSI_CQ_CRARG.VALUE bit field stores the argument of the last received command response. The controller updates the value every time a command response is received.

Command Response for Direct Command Register

The EMSI_CQ_CRDCT register stores the response of last executed DCMD.

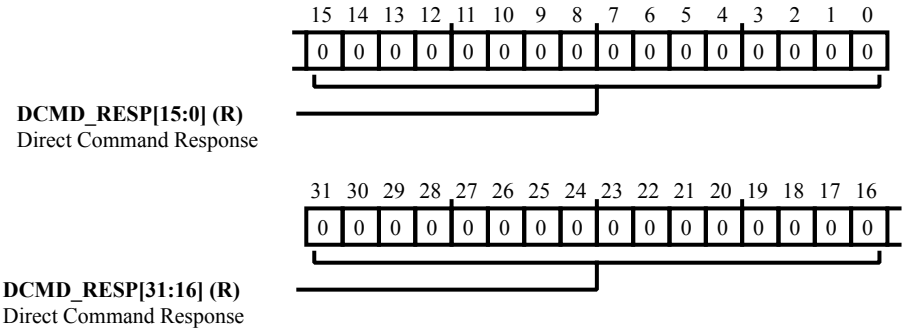


Figure 18-59: EMSI_CQ_CRDCT Register Diagram

Table 18-37: EMSI_CQ_CRDCT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DCMD_RESP	Direct Command Response. The EMSI_CQ_CRDCT.DCMD_RESP bit field contains the response of the command generated by the last direct command (DCMD) task that was sent. The contents of the EMSI_CQ_CRDCT.DCMD_RESP bit field are valid only after bit 31 of the register is cleared by the controller.

CQ Command Response Index

The EMSI_CQ_CRI register stores the index of the last received command response.

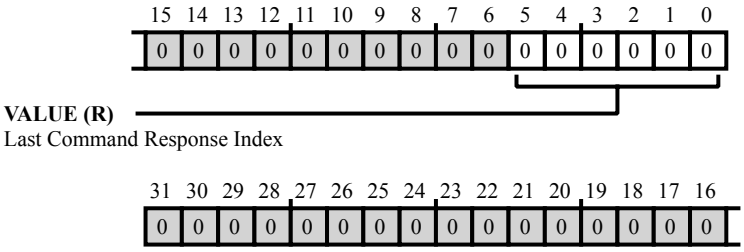


Figure 18-60: EMSI_CQ_CRI Register Diagram

Table 18-38: EMSI_CQ_CRI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/NW)	VALUE	Last Command Response Index. The EMSI_CQ_CRI.VALUE bit field stores the index of the last received command response. The controller updates the value every time a command response is received.

Command Queuing Control Register

The EMSI_CQ_CTL register controls CQE behavior affecting the general operation of command queuing module or simultaneous operation of multiple tasks.

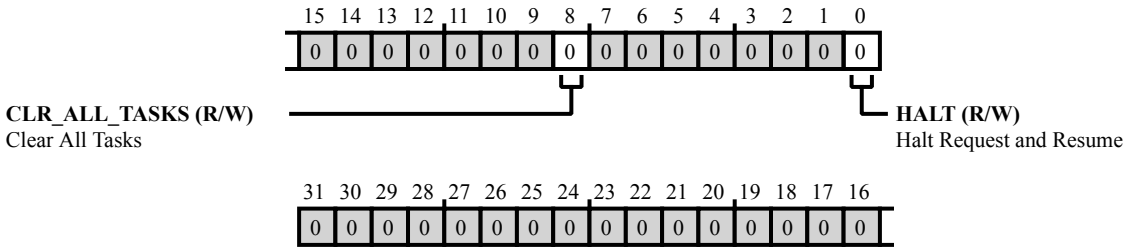


Figure 18-61: EMSI_CQ_CTL Register Diagram

Table 18-39: EMSI_CQ_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	CLR_ALL_TASKS	Clear All Tasks. The EMSI_CQ_CTL.CLR_ALL_TASKS bit can only be written when the controller is halted. The EMSI_CQ_CTL.CLR_ALL_TASKS bit does not clear tasks in the eMMC device. The software has to use the CMDQ_TASK_MGMT command (CMD48) separately to clear device's queue.	
		0	Programming 0 Has No Effect
		1	Clears All the Tasks in the Controller
0 (R/W)	HALT	Halt Request and Resume. The EMSI_CQ_CTL.HALT bit provides a halt request and resume.	
		0	Software writes 0 to this bit to exit from the halt state and resume CQE activity.
		1	Software writes 1 to the EMSI_CQ_CTL.HALT bit when it wants to acquire software control over the eMMC bus and to disable CQE from issuing command on the bus. For example, issuing a Discard Task command (CMDQ_TASK_MGMT). When the software writes 1, CQE completes the ongoing task (if any in progress). After the task is completed and the CQE is in idle state, CQE does not issue new commands and indicates to the software by setting the EMSI_CQ_CTL.HALT bit to 1. The software can poll on the EMSI_CQ_CTL.HALT bit until it is set (=1) and only then send commands on the eMMC bus.

Device Pending Tasks Register

The EMSI_CQ_DPT register maintains the list of tasks that are queued into device and are awaiting execution completion.

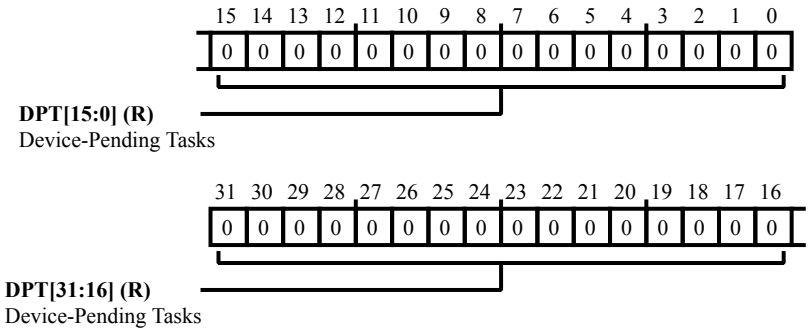


Figure 18-62: EMSI_CQ_DPT Register Diagram

Table 18-40: EMSI_CQ_DPT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DPT	Device-Pending Tasks. Each of the 32 bits are bit mapped to the 32 tasks. - Bit-N(1): Task-N has been successfully queued into the device and is awaiting execution - Bit-N(0): Task-N is not yet queued Bit n of this register is set if and only if QUEUED_TASK_PARAMS (CMD44) and QUEUED_TASK_ADDRESS (CMD45) were sent for this specific task and if this task has not been executed. The controller sets the EMSI_CQ_DPT.DPT bit after receiving a successful response for CMD45. CQE clears the EMSI_CQ_DPT.DPT bit after the task has completed execution. Software reads this register in the task-discard procedure to determine if the task is queued in the device.

Device Queue Status Register

The EMSI_CQ_DQSTAT register stores the most recent value of the device's queue status.

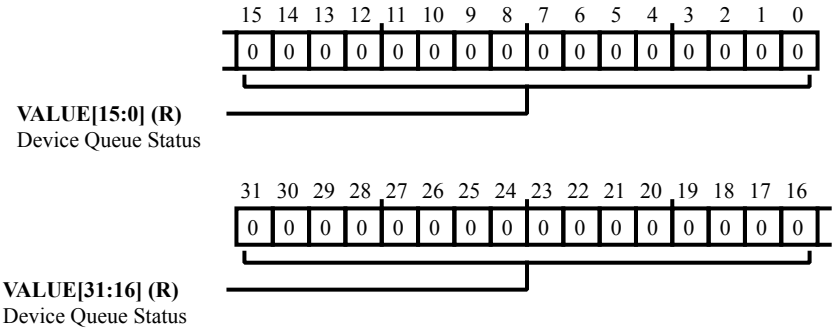


Figure 18-63: EMSI_CQ_DQSTAT Register Diagram

Table 18-41: EMSI_CQ_DQSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Device Queue Status. Each of the EMSI_CQ_DQSTAT.VALUE bits are bit mapped to the 32 tasks as follows: - Bit-N(1): Device has marked task N as ready for execution - Bit-N(0): Task-N is not ready for execution. This task may be pending in the device or not submitted. The eMSI controller updates this register with response of the device queue status command.

Command Queuing Interrupt Coalescing Register

The EMSI_CQ_IC register controls and configures the interrupt coalescing feature.

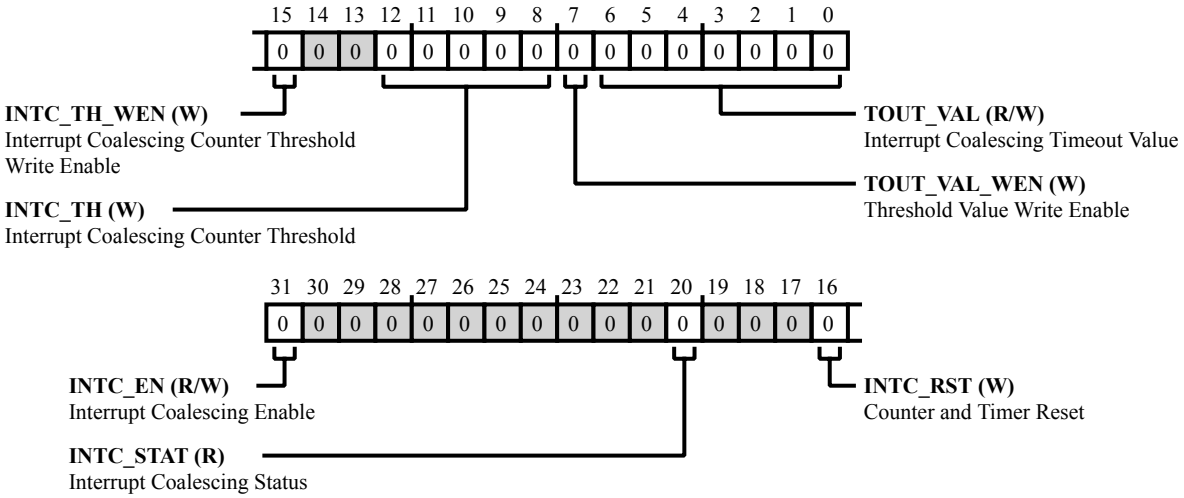


Figure 18-64: EMSI_CQ_IC Register Diagram

Table 18-42: EMSI_CQ_IC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	INTC_EN	Interrupt Coalescing Enable.	
		0	Interrupt Coalescing Mechanism is Disabled (Default)
		1	Interrupt Coalescing Mechanism is Active. Interrupts are Counted and Timed, and Coalesced Interrupts are Generated
20 (R/NW)	INTC_STAT	Interrupt Coalescing Status. The EMSI_CQ_IC.INTC_STAT bit indicates to the software whether any tasks (with INT=0) have completed and counted towards interrupt coalescing (that is, this is set if and only if INTC counter > 0).	
		0	INT0 Task Completions Have Not Occurred Since Last Counter Reset (INTC Counter == 0)
		1	At Least One INT0 Task Completion Has Been Counted (INTC Counter > 0)
16 (RX/W)	INTC_RST	Counter and Timer Reset. When the host driver writes 1 to the EMSI_CQ_IC.INTC_RST, the interrupt coalescing timer and counter are reset.	
		0	No Effect
		1	Interrupt Coalescing Timer and Counter are Reset

Table 18-42: EMSI_CQ_IC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (RX/W)	INTC_TH_WEN	Interrupt Coalescing Counter Threshold Write Enable. When software writes 1 to the `EMSI_CQ_IC.INTC_TH_WEN` bit, the value `EMSI_CQ_IC.INTC_TH` is updated with the contents written on the same cycle.	
		0	Clears INTC_TH_WEN
		1	Sets INTC_TH_WEN
12:8 (RX/W)	INTC_TH	Interrupt Coalescing Counter Threshold. Software uses the `EMSI_CQ_IC.INTC_TH` field to configure the number of task completions (only tasks with INT=0 in the task descriptor), which are required in order to generate an interrupt. Counter Operation: As data transfer tasks with INT=0 complete, they are counted by CQE. The counter is reset by software during the interrupt service routine. The counter stops counting when it reaches the value configured in `EMSI_CQ_IC.INTC_TH`, and generates interrupt. - 0x0: Interrupt coalescing feature disabled - 0x1: Interrupt coalescing interrupt generated after 1 task when INT=0 completes - 0x2: Interrupt coalescing interrupt generated after 2 tasks when INT=0 completes - ..... - 0x1f: Interrupt coalescing interrupt generated after 31 tasks when INT=0 completes To write to this field, the `EMSI_CQ_IC.INTC_TH_WEN` bit must be set during the same write operation.	
7 (RX/W)	TOUT_VAL_WEN	Threshold Value Write Enable. When software writes 1 to the `EMSI_CQ_IC.TOUT_VAL_WEN` bit, the value in the `EMSI_CQ_IC.TOUT_VAL` bit field is updated with the contents written on the same cycle.	
		0	Clears TOUT_VAL_WEN
		1	Sets TOUT_VAL_WEN

Table 18-42: EMSI_CQ_IC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	TOUT_VAL	Interrupt Coalescing Timeout Value. Software uses the `EMSI_CQ_IC.TOUT_VAL` field to configure the maximum time allowed between the completion of a task on the bus and the generation of an interrupt. Timer Operation: The timer is reset by software during the interrupt service routine. It starts running when the first data transfer task with INT=0 is completed, after the timer was reset. When the timer reaches the value configured in ICTOVAL field, it generates an interrupt and stops. The timer's unit is equal to 1024 clock periods of the clock whose frequency is specified in the `EMSI_CQ_CAP.ITCFVAL` bit field. - 0x0: Timer is disabled. Timeout-based interrupt is not generated - 0x1: Timeout on 01x1024 cycles of timer clock frequency - 0x2: Timeout on 02x1024 cycles of timer clock frequency - ..... - 0x7f: Timeout on 127x1024 cycles of timer clock frequency In order to write to this field, the `EMSI_CQ_IC.TOUT_VAL_WEN` bit must be set at the same write operation.

Command Queuing Interrupt Status Register

The EMSI_CQ_ISTAT register indicates pending interrupts that require service. Each bit in this register is asserted in response to a specific event, only if the respective bit is set in the EMSI_CQ_ISTAT_EN register.

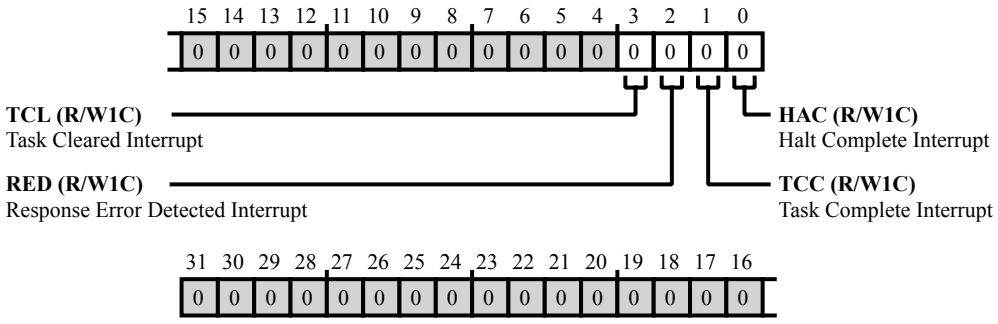


Figure 18-65: EMSI_CQ_ISTAT Register Diagram

Table 18-43: EMSI_CQ_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W1C)	TCL	Task Cleared Interrupt. The EMSI_CQ_ISTAT.TCL bit is asserted (if EMSI_CQ_ISTAT_EN.TCL =1) when a task clear operation is completed by the CQE. The completed task clear operation is either an individual task clear (by writing EMSI_CQ_TCLR) or clearing of all tasks (by writing EMSI_CQ_CTL). A value of 1 clears this status bit.	
		0	TCL Interrupt is Not Set
		1	TCL Interrupt is Set
2 (R/W1C)	RED	Response Error Detected Interrupt. The EMSI_CQ_ISTAT.RED status bit is asserted (if EMSI_CQ_ISTAT_EN.RED =1) when a response is received with an error bit set in the device status field. Configure the EMSI_CQ_RMEM register to identify the device status bit fields that may trigger an interrupt and that are masked. A value of 1 clears this status bit.	
		0	RED Interrupt is Not Set
		1	RED Interrupt is Set

Table 18-43: EMSI_CQ_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	TCC	Task Complete Interrupt. The EMSI_CQ_ISTAT.TCC status bit is asserted (if EMSI_CQ_ISTAT_EN.TCC =1) when at least one of the following conditions are met: - A task is completed and the INT bit is set in its Task Descriptor - Interrupt caused by Interrupt Coalescing logic due to timeout - Interrupt Coalescing logic reached the configured threshold A value of 1 clears this status bit.	
		0	TCC Interrupt is Not Set
		1	TCC Interrupt is Set
0 (R/W1C)	HAC	Halt Complete Interrupt. The EMSI_CQ_ISTAT.HAC status bit is asserted (only if EMSI_CQ_ISTAT_EN.HAC =1) when the EMSI_CQ_CTL.HALT bit transitions from 0 to 1 indicating that the eMSI controller has completed its current ongoing task and has entered the halt state. A value of 1 clears this status bit.	
		0	HAC Interrupt is Not Set
		1	HAC Interrupt is Set

Command Queuing Interrupt Status Enable Register

The EMSI_CQ_ISTAT_EN register enables and disables the reporting of the corresponding interrupt status to the host software in the EMSI_CQ_ISTAT register. When a bit is set (1) and the corresponding interrupt condition is active, then the bit in the EMSI_CQ_ISTAT register is asserted. Interrupt sources that are disabled (when 0) are not indicated in the EMSI_CQ_ISTAT register. This register is bit-index matched to the EMSI_CQ_ISTAT register.

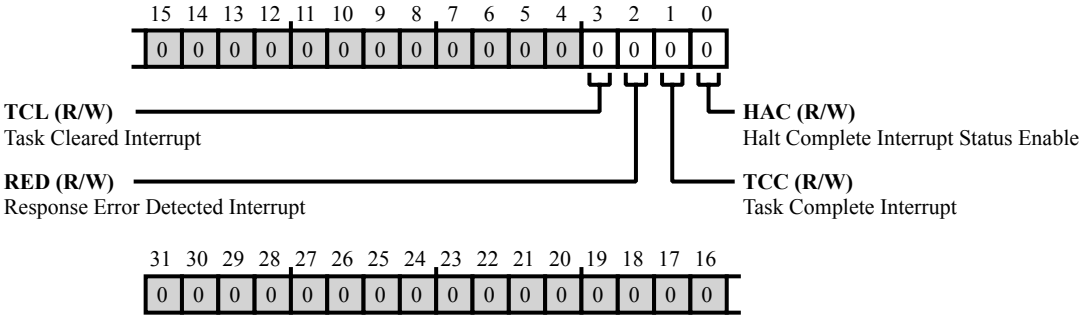


Figure 18-66: EMSI_CQ_ISTAT_EN Register Diagram

Table 18-44: EMSI_CQ_ISTAT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	TCL	Task Cleared Interrupt. Task cleared interrupt status enable.	
		0	CQIS.TCL is Disabled
		1	CQIS.TCL is Set When Its Interrupt Condition is Active
2 (R/W)	RED	Response Error Detected Interrupt. Response error detected interrupt status enable.	
		0	CQIS.RED is Disabled
		1	CQIS.RED is Set When Its Interrupt Condition is Active
1 (R/W)	TCC	Task Complete Interrupt. Task complete interrupt status enable.	
		0	CQIS.TCC is Disabled
		1	CQIS.TCC is Set When Its Interrupt Condition is Active

Table 18-44: EMSI_CQ_ISTAT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	HAC	Halt Complete Interrupt Status Enable.	
		0	CQIS.HAC is Disabled
		1	CQIS.HAC is Set When Its Interrupt Condition is Active

Command Queuing Interrupt Signal Enable Register

The EMSI_CQ_ISTAT_INTEN register enables and disables the generation of interrupts to host software. When a bit is set and the corresponding bit in EMSI_CQ_ISTAT is set, then an interrupt is generated. This register is bit-index matched to the EMSI_CQ_ISTAT register.

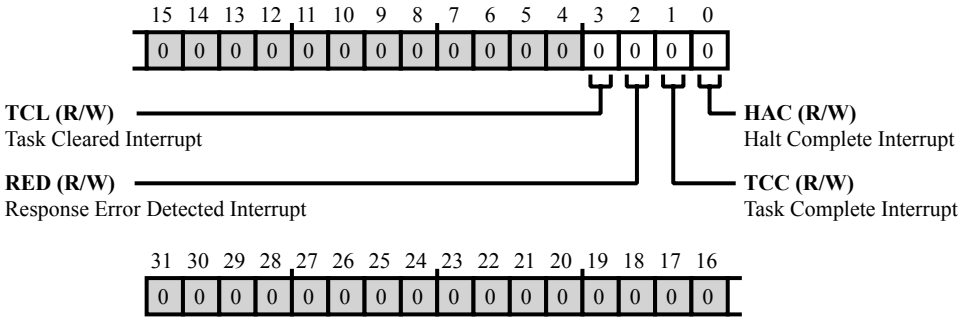


Figure 18-67: EMSI_CQ_ISTAT_INTEN Register Diagram

Table 18-45: EMSI_CQ_ISTAT_INTEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	TCL	Task Cleared Interrupt. Task cleared interrupt signal enable.	
		0	CQIS.TCL Interrupt Signal Generation is Disabled
		1	CQIS.TCL Interrupt Signal Generation is Active
2 (R/W)	RED	Response Error Detected Interrupt. Response error detected interrupt signal enable.	
		0	CQIS.RED Interrupt Signal Generation is Disabled
		1	CQIS.RED Interrupt Signal Generation is Active
1 (R/W)	TCC	Task Complete Interrupt. Task complete interrupt signal enable.	
		0	CQIS.TCC Interrupt Signal Generation is Disabled
		1	CQIS.TCC Interrupt Signal Generation is Active
0 (R/W)	HAC	Halt Complete Interrupt. Halt complete interrupt signal enable.	
		0	CQIS.HAC Interrupt Signal Generation is Disabled
		1	CQIS.HAC Interrupt Signal Generation is Active

Command Response Mode Error Mask Register

The EMSI_CQ_RMEM register controls the generation of the response error detect (RED) interrupt. Only the bits enabled here can contribute to RED.

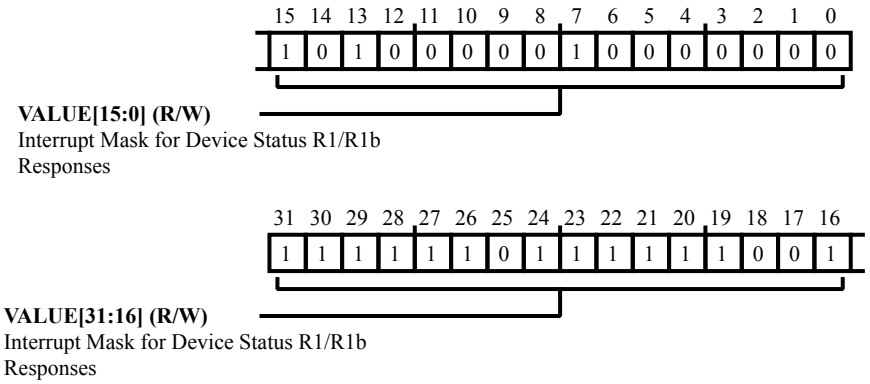


Figure 18-68: EMSI_CQ_RMEM Register Diagram

Table 18-46: EMSI_CQ_RMEM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Interrupt Mask for Device Status R1/R1b Responses. The EMSI_CQ_RMEM.VALUE bits are bit-mapped to the device response. The EMSI_CQ_RMEM.VALUE bits are used as an interrupt mask on the device status field that is received in R1/R1b responses. - 1: When a R1/R1b response is received, with a bit i in the device status set, a RED interrupt is generated. - 0: When a R1/R1b response is received, bit i in the device status is ignored. The reset value of this register is set to trigger an interrupt on all "Error" type bits in the device status. Note: Responses to CMD13 (SQS) encode the QSR so that they are ignored by this logic.

CQ Send Status Configuration 1 Register

The EMSI_CQ_SSCFG1 register is used for removing an outstanding task in the CQE. The register controls when SEND_QUEUE_STATUS commands are sent.

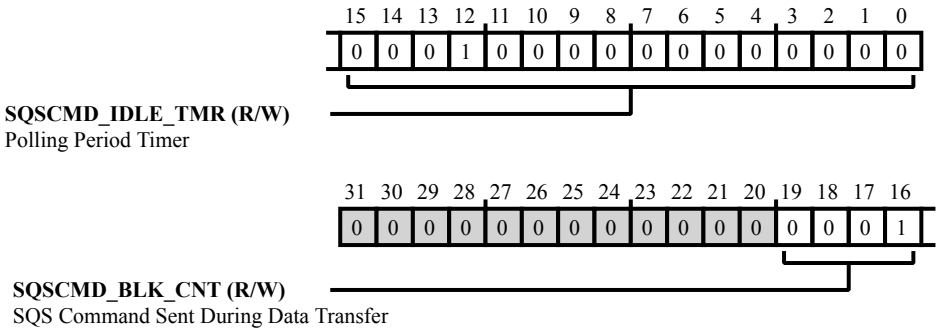


Figure 18-69: EMSI_CQ_SSCFG1 Register Diagram

Table 18-47: EMSI_CQ_SSCFG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:16 (R/W)	SQSCMD_BLK_CNT	SQS Command Sent During Data Transfer. The EMSI_CQ_SSCFG1.SQSCMD_BLK_CNT field indicates when SQS CMD is sent while data transfer is in progress. A value of 'n' indicates that CQE sends status command on the CMD line, during the transfer of data block BLOCK_CNT-n, on the data lines, where BLOCK_CNT is the number of blocks in the current transaction. - 0x0: SEND_QUEUE_STATUS (CMD13) command is not sent during the transaction. Instead, it is sent only when the data lines are idle. - 0x1: SEND_QUEUE_STATUS command is to be sent during the last block of the transaction. - 0x2: SEND_QUEUE_STATUS command when last 2 blocks are pending. - 0x3: SEND_QUEUE_STATUS command when last 3 blocks are pending. - ..... - 0xf: SEND_QUEUE_STATUS command when last 15 blocks are pending. The EMSI_CQ_SSCFG1.SQSCMD_BLK_CNT bits should be programmed only when EMSI_CQ_CFG.CQ_EN is = 0.

Table 18-47: EMSI_CQ_SSCFG1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	SQSCMD_IDLE_TMR	Polling Period Timer. The `EMSI_CQ_SSCFG1.SQSCMD_IDLE_TMR` bit field configures the polling period to be used when using periodic SEND_QUEUE_STATUS (CMD13) polling. Periodic polling is used when tasks are pending in the device, but no data transfer is in progress. When a SEND_QUEUE_STATUS response indicates that no task is ready for execution, the CQE counts the configured time until it issues the next SEND_QUEUE_STATUS. Timer units are clock periods of the clock whose frequency is specified in the `EMSI_CQ_CAP.ITCFVAL` bits. The minimum value is 0001h (1 clock period) and the maximum value is FFFFh (65535 clock periods). For example, a `EMSI_CQ_CAP.ITCFVAL` field value of 0 indicates a 19.2 MHz clock frequency (period = 52.08 ns). If the setting in `EMSI_CQ_SSCFG1.SQSCMD_IDLE_TMR` is 1000h, the calculated polling period is 4096*52.08 ns= 213.33 ns. The value in this field should be programmed only when `EMSI_CQ_CFG.CQ_EN` = 0.

CQ Send Status Configuration 2 Register

The EMSI_CQ_SSCFG2 register is used for configuring the RCA field in SEND_QUEUE_STATUS command argument.

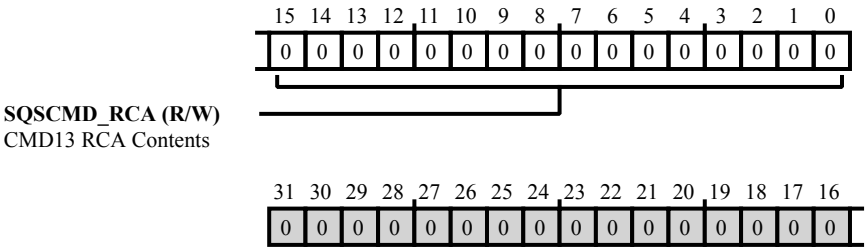


Figure 18-70: EMSI_CQ_SSCFG2 Register Diagram

Table 18-48: EMSI_CQ_SSCFG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	SQSCMD_RCA	CMD13 RCA Contents. The EMSI_CQ_SSCFG2.SQSCMD_RCA bit field provides CQE with the contents of the 16-bit RCA field in SEND_QUEUE_STATUS (CMD13) command argument. The CQE copies this field to bits 31:16 of the argument when transmitting the SEND_ QUEUE_STATUS (CMD13) command.

Command Queuing Task Clear Register

The EMSI_CQ_TCLR register is used for removing an outstanding task in the CQE. The register must be used only when the CQE is in a halt state.

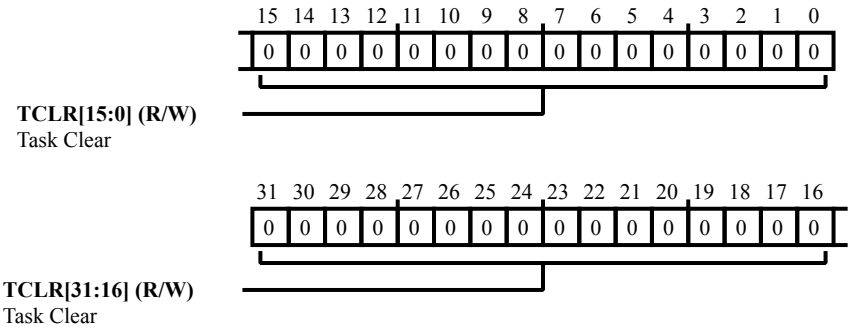


Figure 18-71: EMSI_CQ_TCLR Register Diagram

Table 18-49: EMSI_CQ_TCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TCLR	Task Clear. Writing 1 to bit n of the EMSI_CQ_TCLR.TCLR bit field orders the CQE to clear a task that the software has previously issued. This bit can only be written when the CQE is in the halt state as indicated in the EMSI_CQ_CTL.HALT bit. When software writes 1 to a bit in this register, CQE updates the value to 1, and starts clearing the data structures related to the task. The CQE clears the bit fields (sets a value of 0) in the EMSI_CQ_TCLR register and in the EMSI_CQ_TDBR register once the clear operation is complete. Software must poll on this register until it is cleared to verify that a clear operation was done.

Command Queuing Task Completion Notification Register

The EMSI_CQ_TCN register is used by the CQE to notify software about completed tasks.

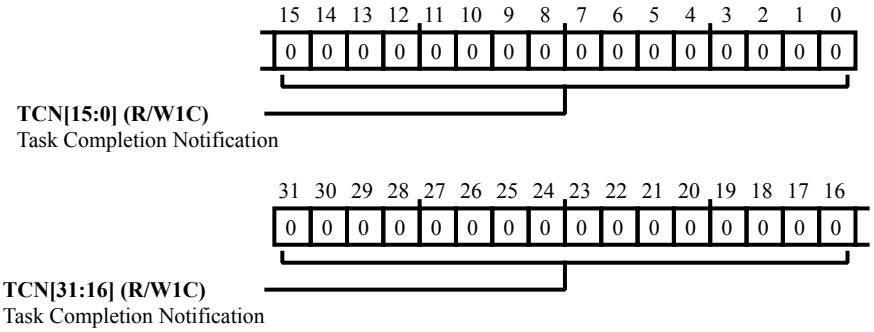


Figure 18-72: EMSI_CQ_TCN Register Diagram

Table 18-50: EMSI_CQ_TCN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W1C)	TCN	Task Completion Notification. Each of the EMSI_CQ_TCN.TCN bits are bit mapped to the 32 tasks. - Bit-N(1): Task-N has completed execution (with success or errors) - Bit-N(0): Task-N has not completed, could be pending or not submitted. On task completion, software may read this bit field to know tasks that have completed. After reading this register, software may clear the relevant bit fields by writing 1 to the corresponding bits.

Command Queuing DoorBell Register

The EMSI_CQ_TDBR register, causes software to trigger the CQE to process a new task.

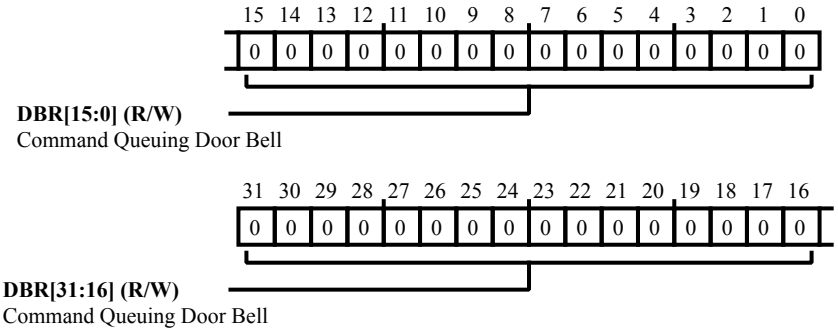


Figure 18-73: EMSI_CQ_TDBR Register Diagram

Table 18-51: EMSI_CQ_TDBR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DBR	Command Queuing Door Bell. The software configures EMSI_CQ_TDL_BADDR and enables EMSI_CQ_TDBR EMSI_CQ_CFG CQE in CQCFG before using this register. Writing 1 to bit n of this register triggers the CQE to start processing the task encoded in slot n of the TDL. Writing 0 by the software does not have any impact on the hardware, and does not change the value of the register bit. The CQE always processes tasks according to the order submitted to the list by write transactions. The CQE processes data transfer tasks by reading the task descriptor and sending QUEUED_TASK_PARAMS (CMD44) and QUEUED_TASK_ADDRESS (CMD45) commands to the device. CQE processes DCMD tasks (in slot #31, when enabled) by reading the task descriptor, and generating the command encoded by its index and argument. The corresponding bit is cleared to 0 by CQE in one of the following events: - A task execution is completed (with success or error). - The task is cleared using EMSI_CQ_TCLR register. - All tasks are cleared using EMSI_CQ_CTL register. - CQE is disabled using EMSI_CQ_CFG register. Software may initiate multiple tasks at the same time (batch submission) by writing 1 to multiple bits of this register in the same transaction. In the case of batch submission, CQE processes the tasks in order of the task index, starting with the lowest index. If one or more tasks in the batch are marked with QBR, the ordering of execution is based on said processing order.

Command Queuing Task Descriptor List Base Address Register

The EMSI_CQ_TDL_BADDR register is used for configuring the lower 32 bits of the byte address of the head of the Task Descriptor List in the host memory.

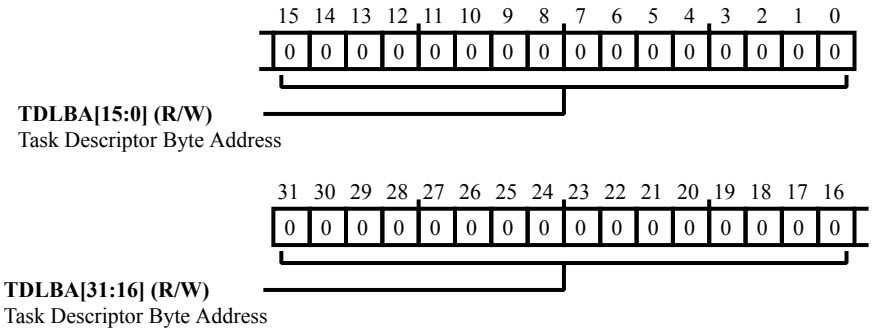


Figure 18-74: EMSI_CQ_TDL_BADDR Register Diagram

Table 18-52: EMSI_CQ_TDL_BADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TDLBA	Task Descriptor Byte Address. The EMSI_CQ_TDL_BADDR.TDLBA bit field stores the LSB bits (31:0) of the byte address of the head of the task descriptor List in system memory. The size of the task descriptor list is 32 x (task descriptor size + transfer descriptor size) as configured by the host driver. This address is set on 1 KB boundary. The lower 10 bits of this register are cleared (=0) by the software and are ignored by CQE.

CQ Task Error Information Register

The EMSI_CQ_TERRINFO register is updated by the CQE when an error occurs on data or a command related to a task activity. When such an error is detected by the CQE or indicated by the eMSI controller, the CQE stores task IDs and indices of commands that were executed on the command line and data lines when the error occurred in this register.

Software must use this information in the error recovery procedure.

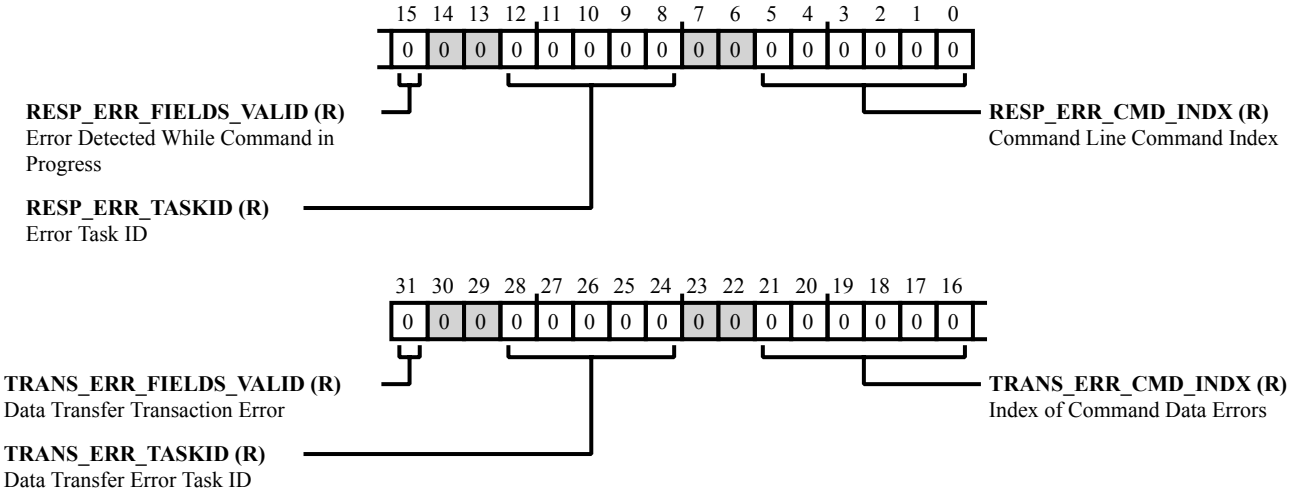


Figure 18-75: EMSI_CQ_TERRINFO Register Diagram

Table 18-53: EMSI_CQ_TERRINFO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	TRANS_ERR_FIELDS_VALID	Data Transfer Transaction Error. The EMSI_CQ_TERRINFO.TRANS_ERR_FIELDS_VALID bit is updated when an error is detected while a data transfer transaction was in progress.	
		0	Ignore Contents of TRANS_ERR_TASKID and TRANS_ERR_CMD_INDX
		1	Data Transfer Related Error Detected. Check Contents of TRANS_ERR_TASKID and TRANS_ERR_CMD_INDX Fields
28:24 (R/NW)	TRANS_ERR_TASKID	Data Transfer Error Task ID. The EMSI_CQ_TERRINFO.TRANS_ERR_TASKID bit field captures the ID of the task that was executed and whose data transfer has errors.	
21:16 (R/NW)	TRANS_ERR_CMD_INDX	Index of Command Data Errors. The EMSI_CQ_TERRINFO.TRANS_ERR_CMD_INDX bit field captures the index of the command that was executed and whose data transfer has errors.	

Table 18-53: EMSI_CQ_TERRINFO Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	RESP_ERR_FIELDS_VALID	Error Detected While Command in Progress. The `EMSI_CQ_TERRINFO.RESP_ERR_FIELDS_VALID` bit is updated when an error is detected while a command transaction was in progress.	
		0	Ignore Contents of RESP_ERR_TASKID and RESP_ERR_CMD_INDX
		1	Response-related Error is Detected. Check Contents of RESP_ERR_TASKID and RESP_ERR_CMD_INDX Fields
12:8 (R/NW)	RESP_ERR_TASKID	Error Task ID. The `EMSI_CQ_TERRINFO.RESP_ERR_TASKID` bit field captures the ID of the task which was executed on the command line when the error occurred.	
5:0 (R/NW)	RESP_ERR_CMD_INDX	Command Line Command Index. The `EMSI_CQ_TERRINFO.RESP_ERR_CMD_INDX` bits capture the index of the command that was executed on the command line when the error occurred.	

EMSI Control 1 Register

The EMSI_CTL1 register is used to control the operation of the eMSI controller. This register is applicable for eMMC mode.

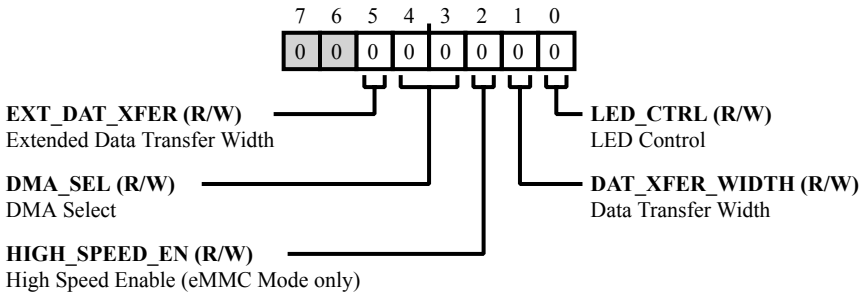


Figure 18-76: EMSI_CTL1 Register Diagram

Table 18-54: EMSI_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	EXT_DAT_XFER	Extended Data Transfer Width. The EMSI_CTL1.EXT_DAT_XFER bit controls 8-bit bus width mode of an eMMC device.	
		0	Bus Width is Selected by the Data Transfer Width
		1	8-bit Bus Width
4:3 (R/W)	DMA_SEL	DMA Select. The EMSI_CTL1.DMA_SEL bit field is used to select the DMA type. When EMSI_CTL2.HOST_VER4_EN =1: - 0x0: SDMA is selected - 0x1: Reserved - 0x2: ADMA2 is selected - 0x3: ADMA2 or ADMA3 is selected When EMSI_CTL2.HOST_VER4_EN =0: - 0x0: SDMA is selected - 0x1: Reserved - 0x2: 32-bit Address ADMA2 is selected	
		0	SDMA is Selected
		1	Reserved
		2	ADMA2 is Selected
		3	ADMA2 or ADMA3 is Selected

Table 18-54: EMSI_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	HIGH_SPEED_EN	High Speed Enable (eMMC Mode only). In eMMC mode, the `EMSI_CTL1.HIGH_SPEED_EN` bit is used to determine the selection of the preset value for high speed mode. This bit is set = 1 when the eMSI bus clock is 0-50MHz. This bit is cleared = 0 when the eMSI bus clock frequency is 0-25MHz.	
		0	Normal Speed Mode (eMSI bus clock frequency 0-25 MHz)
		1	High Speed Mode (eMSI bus clock frequency 0-50 MHz)
1 (R/W)	DAT_XFER_WIDTH	Data Transfer Width. For eMMC mode,the `EMSI_CTL1.DAT_XFER_WIDTH` bit selects the data transfer width of the eMSI controller. The host driver sets it to match the data width of the eMMC device.	
		0	1-bit Mode
		1	4-bit Mode
0 (R/W)	LED_CTRL	LED Control. The `EMSI_CTL1.LED_CTRL` bit is used to caution the user not to remove the card while the eMMC card is being accessed. The value is reflected on the EMSI_LED_CONTROL signal. The `EMSI_CTL1.LED_CTRL` bit can be set to 1 by host driver at the start of data transfer and cleared when data transfer is complete.	
		0	LED Off
		1	LED On

EMSI Control 2 Register

The EMSI_CTL2 register is used to control how the eMSI controller operates. This register is applicable for eMMC mode.

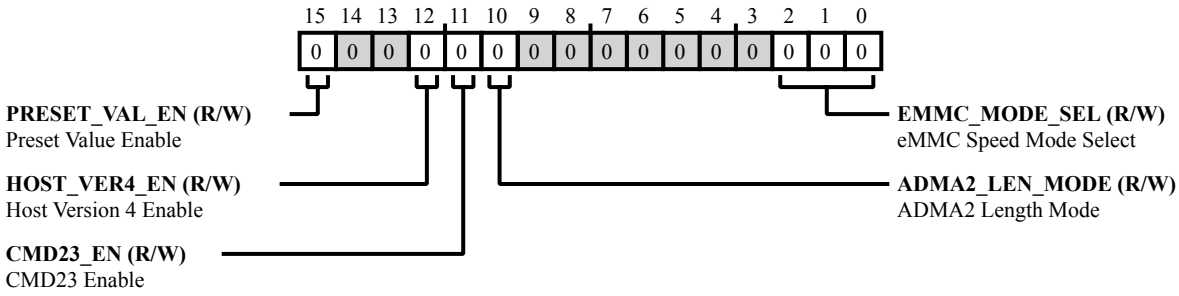


Figure 18-77: EMSI_CTL2 Register Diagram

Table 18-55: EMSI_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	PRESET_VAL_EN	Preset Value Enable. The EMSI_CTL2.PRESET_VAL_EN bit enables automatic selection of the eMSI bus clock frequency using the preset value registers. When the EMSI_CTL2.PRESET_VAL_EN bit is set, the eMSI bus clock frequency generation (frequency select) is performed by the controller. These values are selected from the set of preset value registers based on the selected speed mode.	
		0	eMSI bus clock controlled by Host Driver
		1	Automatic Selection by Preset Value are Enabled
12 (R/W)	HOST_VER4_EN	Host Version 4 Enable. The EMSI_CTL2.HOST_VER4_EN bit selects either Version 3.00 compatible mode or Version 4 mode. The functions of the following fields are modified for Host Version 4 mode. - SDMA Address: SDMA uses the EMSI_ADMA_ADDR_LO register instead of the EMSI_SDMA_ADDR register - ADMA2/ADMA3 selection: ADMA3 is selected by DMA select in the EMSI_CTL1 register - 32-bit Block Count: the EMSI_SDMA_ADDR register is modified to the 32-bit Block Count register Note: It is recommended not to program ADMA3 integrated descriptor address registers, and command queuing registers (if applicable) while operating in host version less than 4 mode (EMSI_CTL2.HOST_VER4_EN = 0).	
		0	Version 3.00 Compatible Mode
		1	Version 4 Mode

Table 18-55: EMSI_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	CMD23_EN	CMD23 Enable. If the device supports CMD23, the `EMSI_CTL2.CMD23_EN` bit is set (=1). The `EMSI_CTL2.CMD23_EN` bit is used to select Auto CMD23 or Auto CMD12 for ADMA3 data transfer.	
		0	Auto CMD23 is Disabled
		1	Auto CMD23 is Enabled
10 (R/W)	ADMA2_LEN_MODE	ADMA2 Length Mode. The `EMSI_CTL2.ADMA2_LEN_MODE` bit selects ADMA2 Length mode to be either 16-bit or 26-bit.	
		0	16-bit Data Length Mode
		1	26-bit Data Length Mode
2:0 (R/W)	EMMC_MODE_SEL	eMMC Speed Mode Select. In eMMC mode, the `EMSI_CTL2.EMMC_MODE_SEL` bits are used to select the speed mode.	
		0	Legacy
		1	High Speed SDR
		2	Reserved
		3	Reserved
		4	High Speed DDR
		5	Reserved
		6	Reserved
		7	Reserved

SD Host Controller Specification Version

The EMSI_HOST_CNTRL_VERS register is used to indicate the supported SD host controller specification version number.

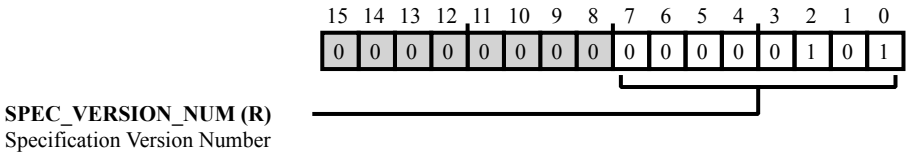


Figure 18-78: EMSI_HOST_CNTRL_VERS Register Diagram

Table 18-56: EMSI_HOST_CNTRL_VERS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:0 (R/NW)	SPEC_VERSION_NUM	Specification Version Number. The EMSI_HOST_CNTRL_VERS.SPEC_VERSION_NUM bits indicate the SD host controller specification version. The upper and lower 4-bits indicate the version. Values 0x06-0xFF are reserved.	
		0	SD Host Controller Specification Version 1.00
		1	SD Host Controller Specification Version 2.00
		2	SD Host Controller Specification Version 3.00
		3	SD Host Controller Specification Version 4.00
		4	SD Host Controller Specification Version 4.10
		5	SD Host Controller Specification Version 4.20

eMMC Control Register

The EMSI_EMMC_CTL register is used to control the eMMC operation.

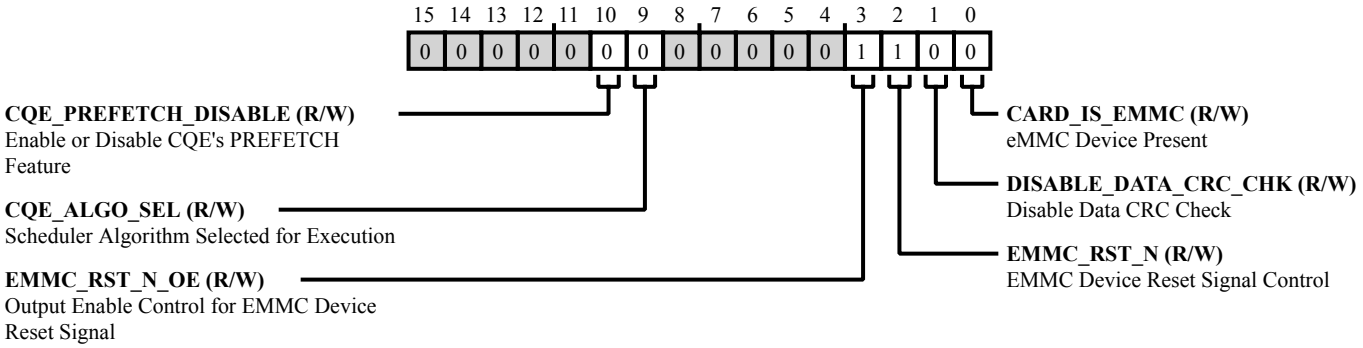


Figure 18-79: EMSI_EMMC_CTL Register Diagram

Table 18-57: EMSI_EMMC_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	CQE_PREFETCH_DISABLE	Enable or Disable CQE's PREFETCH Feature. The EMSI_EMMC_CTL.CQE_PREFETCH_DISABLE bit field allows software to disable the CQE's data prefetch feature when set (=1).	
		0	CQE Can Prefetch Data for Successive WRITE Transfers and Pipeline Successive READ Transfers
		1	Prefetch for WRITE and Pipeline for READ are Disabled
9 (R/W)	CQE_ALGO_SEL	Scheduler Algorithm Selected for Execution. The EMSI_EMMC_CTL.CQE_ALGO_SEL bit selects the algorithm used for selecting one of the many ready tasks for execution.	
		0	Priority Based Reordering with FCFS to Resolve Equal Priority Tasks
		1	First Come First Serve, in the Order of DBR Rings
3 (R/W)	EMMC_RST_N_OE	Output Enable Control for EMMC Device Reset Signal. The EMSI_EMMC_CTL.EMMC_RST_N_OE bit field controls the devise reset for EMMC devices.	
		0	eMMC_rst_n_oe is 0
		1	eMMC_rst_n_oe is 1

Table 18-57: EMSI_EMMC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	EMMC_RST_N	EMMC Device Reset Signal Control. The `EMSI_EMMC_CTL.EMMC_RST_N` register field controls the RST signal of the eMSI controller.	
		0	Reset to eMMC device asserted (active Low)
		1	Reset to eMMC device is deasserted
1 (R/W)	DISABLE_DATA_CRC_CHK	Disable Data CRC Check. The `EMSI_EMMC_CTL.DISABLE_DATA_CRC_CHK` bit controls masking of CRC16 error for card write in eMMC mode. This is useful in bus testing (CMD19) for an eMMC device. In bus testing, an eMMC card does not send CRC status for a block, which may generate CRC error. This CRC error can be masked using the `EMSI_EMMC_CTL.DISABLE_DATA_CRC_CHK` bit during bus testing.	
		0	DATA CRC Check is Enabled
		1	DATA CRC Check is Disabled
0 (R/W)	CARD_IS_EMMC	eMMC Device Present. The `EMSI_EMMC_CTL.CARD_IS_EMMC` bit indicates the type of device connected. An application programs the `EMSI_EMMC_CTL.CARD_IS_EMMC` bit based on the device connected to the eMSI controller.	
		0	Card Connected to the eMSI controller is a Non-eMMC Card
		1	Card Connected to the eMSI controller is an eMMC card

Error Interrupt Status Register

The EMSI_ERR_STAT register indicates error occurred during eMMC transactions. Writing a 1 clears the bit and writing a 0 retains the bit unchanged. Signals defined in this register can be enabled by the Error Interrupt Status Enable register, but not by the EMSI_ERR_STAT_INTEN register. More than one status can be cleared with a single register write. This register is applicable for eMMC mode.

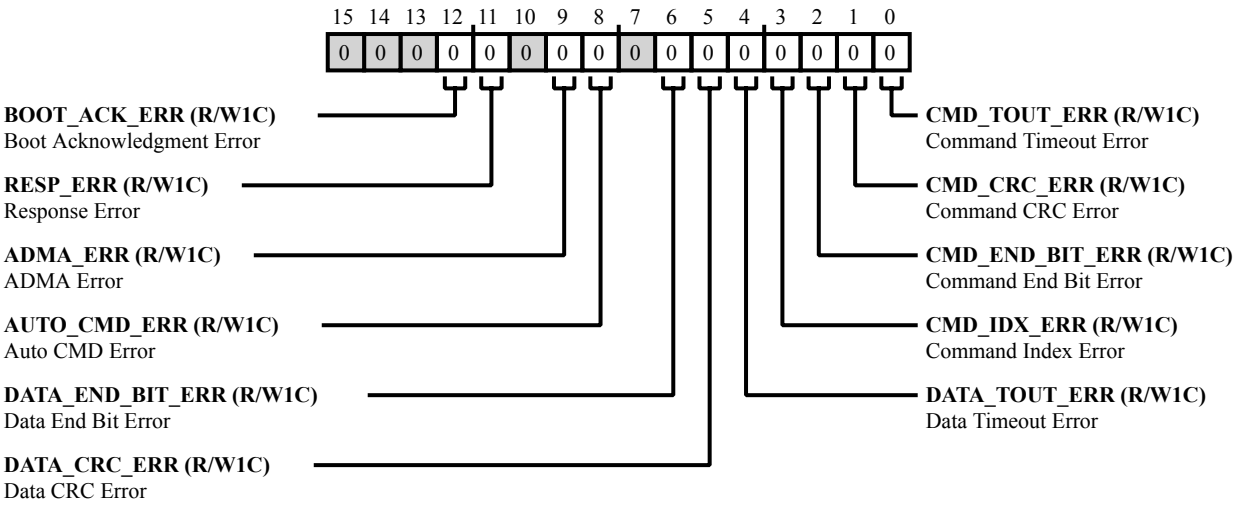


Figure 18-80: EMSI_ERR_STAT Register Diagram

Table 18-58: EMSI_ERR_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W1C)	BOOT_ACK_ERR	Boot Acknowledgment Error. The EMSI_ERR_STAT.BOOT_ACK_ERR bit is set when there is a timeout for boot acknowledgment or when detecting boot ack status having a value other than 010. This is applicable only when boot acknowledgment is expected in eMMC mode.	
		0	No Error
		1	Error
11 (R/W1C)	RESP_ERR	Response Error. Host Version 4.00 supports a response error check function to avoid overhead of response error check by the host driver during DMA execution. If the EMSI_TRNSFRMODE.RESP_ERR_CHK_EN =1, the eMSI controller checks R1 response. If an error is detected in a response, the EMSI_ERR_STAT.RESP_ERR bit is set (=1).This is applicable in eMMC mode.	
		0	No Error
		1	Error

Table 18-58: EMSI_ERR_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W1C)	ADMA_ERR	ADMA Error. The EMSI_ERR_STAT.ADMA_ERR bit is set when the eMSI controller detects error during ADMA-based data transfer. The error could be due to following reasons: - Error response received from System bus - ADMA3, ADMA2 Descriptors invalid - CQE Task or Transfer descriptors invalid When the error occurs, the state of the ADMA is saved in the ADMA Error Status register. In eMMC CQE mode: The eMSI controller generates this Interrupt when it detects an invalid descriptor data (Valid=0) at the ST_FDS state. An ADMA error state in the ADMA error status indicates that an error has occurred in ST_FDS state. The host driver may find that Valid bit is not set at the error descriptor.	
		0	No Error
		1	Error
8 (R/W1C)	AUTO_CMD_ERR	Auto CMD Error. This error status is used by Auto CMD12 and Auto CMD23 in eMMC mode. The EMSI_ERR_STAT.AUTO_CMD_ERR bit is set when detecting that any of the bits D00 to D05 in Auto CMD Error Status register has changed from 0 to 1. D07 is effective in case of Auto CMD12. The Auto CMD error status register is valid while the EMSI_ERR_STAT.AUTO_CMD_ERR bit is set (=1) and may be cleared by clearing the EMSI_ERR_STAT.AUTO_CMD_ERR bit.	
		0	No Error
		1	Error
6 (R/W1C)	DATA_END_BIT_ERR	Data End Bit Error. The EMSI_ERR_STAT.DATA_END_BIT_ERR bit is set in eMMC mode either when detecting a 0 at the end bit position of read data that uses the DAT line or at the end bit position of the CRC status.	
		0	No Error
		1	Error
5 (R/W1C)	DATA_CRC_ERR	Data CRC Error. The EMSI_ERR_STAT.DATA_CRC_ERR bit is set in eMMC mode when detecting a CRC error when transferring read data which uses the DAT line, when detecting the write CRC status having a value of other than 010, or on a write CRC status timeout.	
		0	No Error
		1	Error

Table 18-58: EMSI_ERR_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W1C)	DATA_TOUT_ERR	Data Timeout Error. The EMSI_ERR_STAT.DATA_TOUT_ERR bit is set in eMMC mode when detecting one of the following timeout conditions: - Busy timeout for R1b type - Busy timeout after Write CRC status - Write CRC Status timeout - Read Data timeout	
		0	No Error
		1	Time Out
3 (R/W1C)	CMD_IDX_ERR	Command Index Error. The EMSI_ERR_STAT.CMD_IDX_ERR bit is set if a command index error occurs in the command response in eMMC mode.	
		0	No Error
		1	Error
2 (R/W1C)	CMD_END_BIT_ERR	Command End Bit Error. The EMSI_ERR_STAT.CMD_END_BIT_ERR bit is set when detecting that the end bit of a command response is 0 in eMMC mode.	
		0	No Error
		1	End Bit Error Generated
1 (R/W1C)	CMD_CRC_ERR	Command CRC Error. Command CRC Error is generated in eMMC mode for following two cases. - The eMSI controller detects a CMD line conflict by monitoring the CMD line when a command is issued. If the eMSI controller drives the CMD line to 1 level, but detects 0 level on the CMD line at the next eMSI bus clock edge, then the eMSI controller aborts the command (stop driving CMD line) and set the EMSI_ERR_STAT.CMD_CRC_ERR bit to 1. The command timeout error is also set (=1) to distinguish a CMD line conflict. - If a response is returned and the command timeout error is clear (=0) (indicating no timeout), the EMSI_ERR_STAT.CMD_CRC_ERR bit is set (=1) when detecting a CRC error in the command response.	
		0	No Error
		1	CRC Error Generated

Table 18-58: EMSI_ERR_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W1C)	CMD_TOUT_ERR	Command Timeout Error. In eMMC mode,the `EMSI_ERR_STAT.CMD_TOUT_ERR` bit is set only if no response is returned within 64 eMSI bus clock cycles from the end bit of the command. If the eMSI controller detects a CMD line conflict, along with Command CRC Error bit, the `EMSI_ERR_STAT.CMD_TOUT_ERR` bit is set (=1), without waiting for 64 eMSI bus clock cycles.	
		0	No Error
		1	Time Out

Error Interrupt Status Enable Register

The EMSI_ERR_STAT_EN register sets the interrupt status for error interrupt status register (EMSI_ERR_STAT) when the corresponding bit in EMSI_ERR_STAT_EN is set (=1). This register is applicable for eMMC mode.

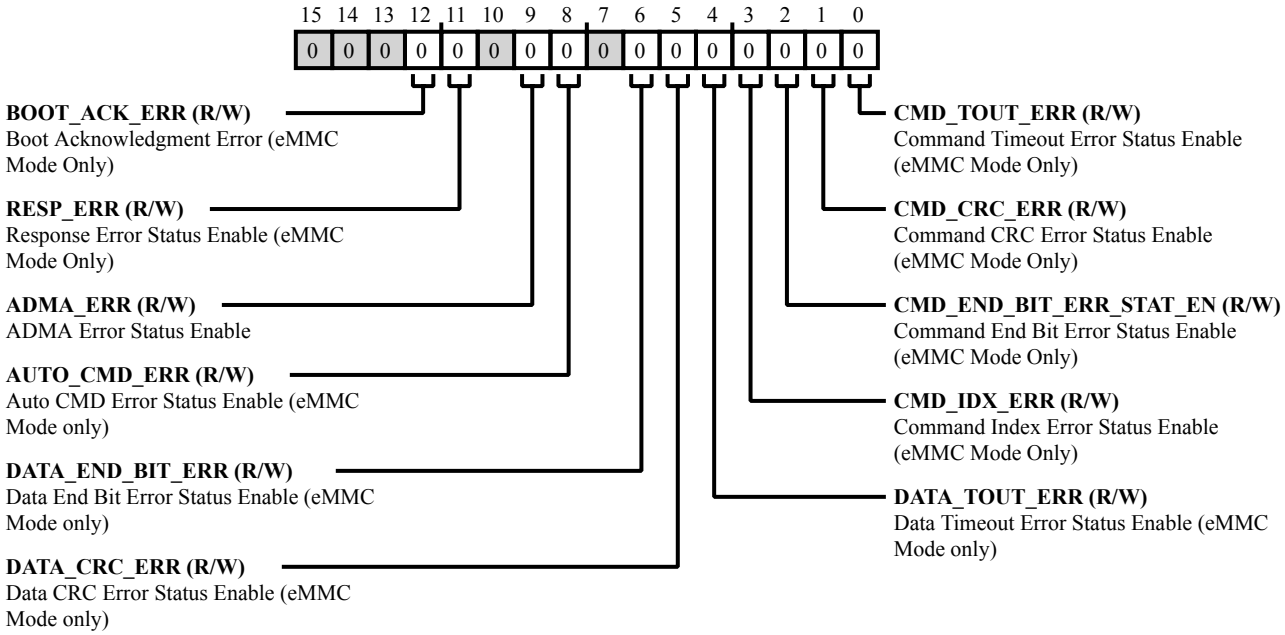


Figure 18-81: EMSI_ERR_STAT_EN Register Diagram

Table 18-59: EMSI_ERR_STAT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	BOOT_ACK_ERR	Boot Acknowledgment Error (eMMC Mode Only). When the EMSI_ERR_STAT_EN.BOOT_ACK_ERR bit is set (=1), it enables setting the corresponding error in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
11 (R/W)	RESP_ERR	Response Error Status Enable (eMMC Mode Only). When the EMSI_ERR_STAT_EN.RESP_ERR bit is set (=1), it enables setting the corresponding error in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled

Table 18-59: EMSI_ERR_STAT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	ADMA_ERR	ADMA Error Status Enable. When the EMSI_ERR_STAT_EN.ADMA_ERR bit is set (=1), it enables setting the corresponding error in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
8 (R/W)	AUTO_CMD_ERR	Auto CMD Error Status Enable (eMMC Mode only). When the EMSI_ERR_STAT_EN.AUTO_CMD_ERR bit is set (=1), it enables setting the corresponding error in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
6 (R/W)	DATA_END_BIT_ERR	Data End Bit Error Status Enable (eMMC Mode only). When the EMSI_ERR_STAT_EN.DATA_END_BIT_ERR bit is set (=1), it enables setting the corresponding error in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
5 (R/W)	DATA_CRC_ERR	Data CRC Error Status Enable (eMMC Mode only). When the EMSI_ERR_STAT_EN.DATA_CRC_ERR bit is set (=1), it enables setting the corresponding error in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
4 (R/W)	DATA_TOUT_ERR	Data Timeout Error Status Enable (eMMC Mode only). When the EMSI_ERR_STAT_EN.DATA_TOUT_ERR bit is set (=1), it enables setting the corresponding error in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
3 (R/W)	CMD_IDX_ERR	Command Index Error Status Enable (eMMC Mode Only). When the EMSI_ERR_STAT_EN.CMD_IDX_ERR bit is set (=1), it enables setting the corresponding error in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled

Table 18-59: EMSI_ERR_STAT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	CMD_END_BIT_ERR_STAT_EN	Command End Bit Error Status Enable (eMMC Mode Only). When the `EMSI_ERR_STAT_EN.CMD_END_BIT_ERR_STAT_EN` bit is set (=1), it enables setting the corresponding error in the `EMSI_ERR_STAT` register.	
		0	Masked
		1	Enabled
1 (R/W)	CMD_CRC_ERR	Command CRC Error Status Enable (eMMC Mode Only). When the `EMSI_ERR_STAT_EN.CMD_CRC_ERR` bit is set (=1), it enables setting the corresponding error in the `EMSI_ERR_STAT` register.	
		0	Masked
		1	Enabled
0 (R/W)	CMD_TOUT_ERR	Command Timeout Error Status Enable (eMMC Mode Only). When the `EMSI_ERR_STAT_EN.CMD_TOUT_ERR` bit is set (=1), it enables setting the corresponding error in the `EMSI_ERR_STAT` register.	
		0	Masked
		1	Enabled

Error Interrupt Signal Enable Register

The EMSI_ERR_STAT_INTEN register is used to select the interrupt status that is notified to the core as an interrupt. All status bits share the same 1-bit interrupt line. Setting any of these bits to 1 enables the interrupt. This register is applicable for eMMC mode.

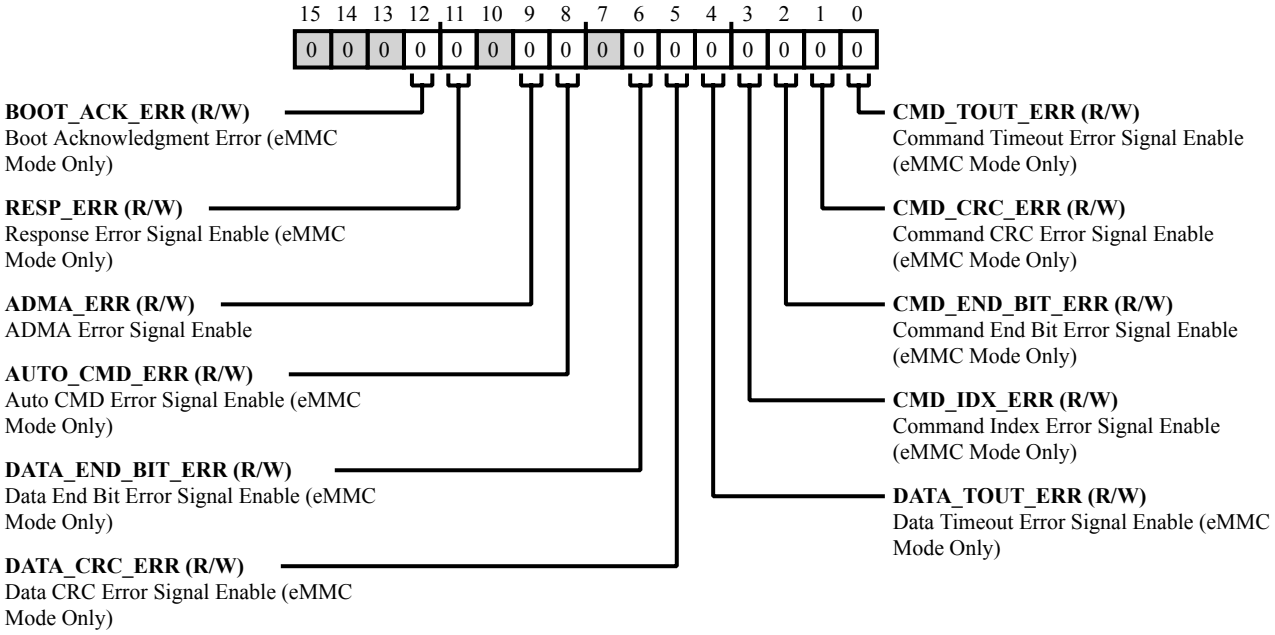


Figure 18-82: EMSI_ERR_STAT_INTEN Register Diagram

Table 18-60: EMSI_ERR_STAT_INTEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	BOOT_ACK_ERR	Boot Acknowledgment Error (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.BOOT_ACK_ERR bit is set (=1), it enables generating an interrupt signal when the boot acknowledgment error in the EMSI_ERR_STAT register is set.	
		0	Masked
		1	Enabled
11 (R/W)	RESP_ERR	Response Error Signal Enable (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.RESP_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled

Table 18-60: EMSI_ERR_STAT_INTEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	ADMA_ERR	ADMA Error Signal Enable. When the EMSI_ERR_STAT_INTEN.ADMA_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
8 (R/W)	AUTO_CMD_ERR	Auto CMD Error Signal Enable (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.AUTO_CMD_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
6 (R/W)	DATA_END_BIT_ERR	Data End Bit Error Signal Enable (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.DATA_END_BIT_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
5 (R/W)	DATA_CRC_ERR	Data CRC Error Signal Enable (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.DATA_CRC_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
4 (R/W)	DATA_TOUT_ERR	Data Timeout Error Signal Enable (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.DATA_TOUT_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled

Table 18-60: EMSI_ERR_STAT_INTEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	CMD_IDX_ERR	Command Index Error Signal Enable (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.CMD_IDX_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	No Error
		1	Error
2 (R/W)	CMD_END_BIT_ERR	Command End Bit Error Signal Enable (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.CMD_END_BIT_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
1 (R/W)	CMD_CRC_ERR	Command CRC Error Signal Enable (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.CMD_CRC_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled
0 (R/W)	CMD_TOUT_ERR	Command Timeout Error Signal Enable (eMMC Mode Only). When the EMSI_ERR_STAT_INTEN.CMD_TOUT_ERR bit is set (=1), it enables generating an interrupt signal error in the corresponding bit in the EMSI_ERR_STAT register.	
		0	Masked
		1	Enabled

Force Event Register for Auto CMD Error Status Register

The EMSI_FRC_AUTOCMDSTAT register is not a physically implemented but is an address at which the Auto CMD Error Status register (EMSI_AUTOCMD_STAT) can be written.This register is applicable for an eMMC mode. Setting a bit in this register sets the corresponding bit in the EMSI_AUTOCMD_STAT register.

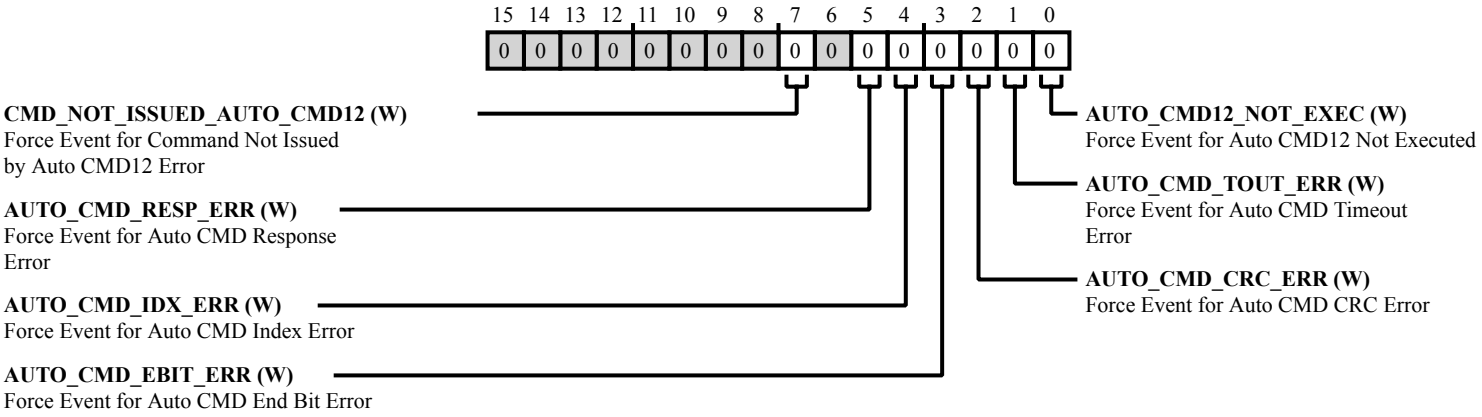


Figure 18-83: EMSI_FRC_AUTOCMDSTAT Register Diagram

Table 18-61: EMSI_FRC_AUTOCMDSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (RX/W)	CMD_NOT_ISSUED_AUTO_CMD12	Force Event for Command Not Issued by Auto CMD12 Error. Setting the EMSI_FRC_AUTOCMDSTAT.CMD_NOT_ISSUED_AUTO_CMD12 bit sets the corresponding bit in the EMSI_AUTOCMD_STAT register.	
		0	Not Affected
		1	Command Not Issued by Auto CMD12 Error Status is Set
5 (RX/W)	AUTO_CMD_RESP_ERR	Force Event for Auto CMD Response Error. Setting the EMSI_FRC_AUTOCMDSTAT.AUTO_CMD_RESP_ERR bit sets the corresponding bit in the EMSI_AUTOCMD_STAT register.	
		0	Not Affected
		1	Auto CMD Response Error Status is Set
4 (RX/W)	AUTO_CMD_IDX_ERR	Force Event for Auto CMD Index Error. Setting the EMSI_FRC_AUTOCMDSTAT.AUTO_CMD_IDX_ERR bit sets the corresponding bit in the EMSI_AUTOCMD_STAT register.	
		0	Not Affected
		1	Auto CMD Index Error Status is Set

Table 18-61: EMSI_FRC_AUTOCMDSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (RX/W)	AUTO_CMD_EBIT_ERR	Force Event for Auto CMD End Bit Error. Setting the EMSI_FRC_AUTOCMDSTAT.AUTO_CMD_EBIT_ERR bit sets the corresponding bit in the EMSI_AUTOCMD_STAT register.	
		0	Not Affected
		1	Auto CMD End Bit Error Status is Set
2 (RX/W)	AUTO_CMD_CRC_ERR	Force Event for Auto CMD CRC Error. Setting the EMSI_FRC_AUTOCMDSTAT.AUTO_CMD_CRC_ERR bit sets the corresponding bit in the EMSI_AUTOCMD_STAT register.	
		0	Not Affected
		1	Auto CMD CRC Error Status is Set
1 (RX/W)	AUTO_CMD_TOUT_ERR	Force Event for Auto CMD Timeout Error. Setting the EMSI_FRC_AUTOCMDSTAT.AUTO_CMD_TOUT_ERR bit sets the corresponding bit in the EMSI_AUTOCMD_STAT register.	
		0	Not Affected
		1	Auto CMD Timeout Error Status is Set
0 (RX/W)	AUTO_CMD12_NOT_EXEC	Force Event for Auto CMD12 Not Executed. Setting the EMSI_FRC_AUTOCMDSTAT.AUTO_CMD12_NOT_EXEC bit sets the corresponding bit in the EMSI_AUTOCMD_STAT register.	
		0	Not Affected
		1	Auto CMD12 Not Executed Status is Set

Force Event Register for Error Interrupt Status

The EMSI_FRC_ERRSTAT register is not physically implemented but is an address at which the error interrupt status register (EMSI_ERR_STAT) can be written. The effect of a write to this address is reflected in the (EMSI_ERR_STAT) register if the corresponding bit of the error interrupt status enable register (EMSI_ERR_STAT_EN) is set. This register is applicable for an eMMC mode.

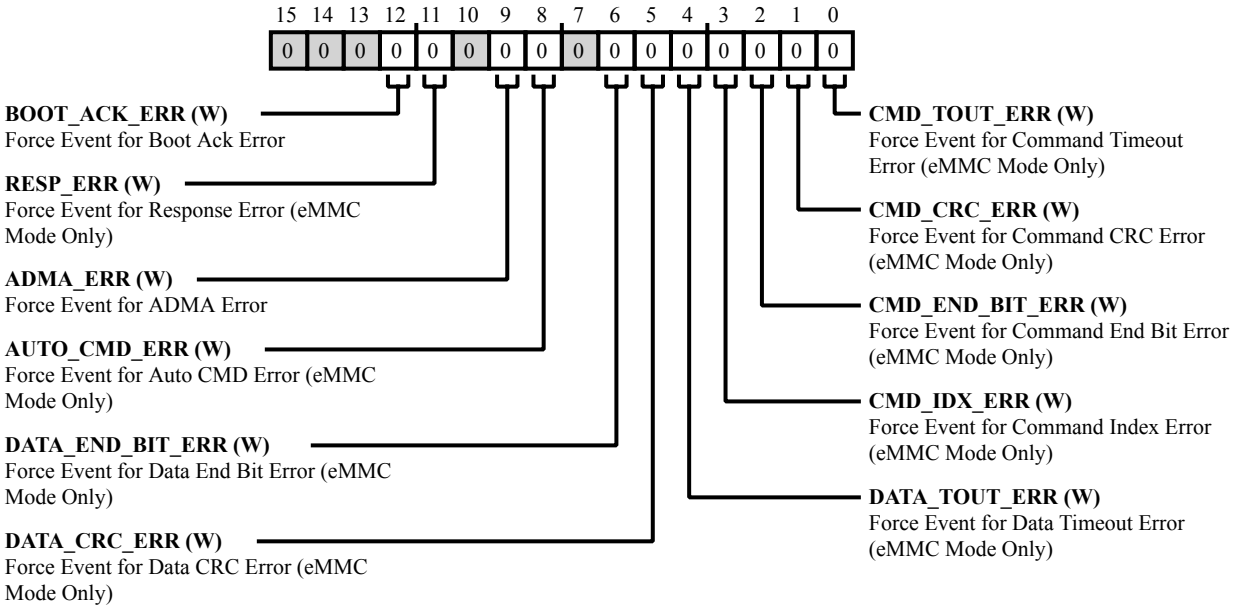


Figure 18-84: EMSI_FRC_ERRSTAT Register Diagram

Table 18-62: EMSI_FRC_ERRSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (RX/W)	BOOT_ACK_ERR	Force Event for Boot Ack Error. A write to the EMSI_FRC_ERRSTAT.BOOT_ACK_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Boot Ack Error Status is Set
11 (RX/W)	RESP_ERR	Force Event for Response Error (eMMC Mode Only). A write to the EMSI_FRC_ERRSTAT.RESP_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Response Error Status is Set

Table 18-62: EMSI_FRC_ERRSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (RX/W)	ADMA_ERR	Force Event for ADMA Error. A write to the EMSI_FRC_ERRSTAT.ADMA_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	ADMA Error Status is Set
8 (RX/W)	AUTO_CMD_ERR	Force Event for Auto CMD Error (eMMC Mode Only). A write to the EMSI_FRC_ERRSTAT.AUTO_CMD_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Auto CMD Error Status is Set
6 (RX/W)	DATA_END_BIT_ERR	Force Event for Data End Bit Error (eMMC Mode Only). A write to the EMSI_FRC_ERRSTAT.DATA_END_BIT_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Data End Bit Error Status is Set
5 (RX/W)	DATA_CRC_ERR	Force Event for Data CRC Error (eMMC Mode Only). A write to the EMSI_FRC_ERRSTAT.DATA_CRC_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Data CRC Error Status is Set
4 (RX/W)	DATA_TOUT_ERR	Force Event for Data Timeout Error (eMMC Mode Only). A write to the EMSI_FRC_ERRSTAT.DATA_TOUT_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Data Timeout Error Status is Set

Table 18-62: EMSI_FRC_ERRSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (RX/W)	CMD_IDX_ERR	Force Event for Command Index Error (eMMC Mode Only). A write to the EMSI_FRC_ERRSTAT.CMD_IDX_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Command Index Error Status is Set
2 (RX/W)	CMD_END_BIT_ERR	Force Event for Command End Bit Error (eMMC Mode Only). A write to the EMSI_FRC_ERRSTAT.CMD_END_BIT_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Command End Bit Error Status is Set
1 (RX/W)	CMD_CRC_ERR	Force Event for Command CRC Error (eMMC Mode Only). A write to the EMSI_FRC_ERRSTAT.CMD_CRC_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Command CRC Error Status is Set
0 (RX/W)	CMD_TOUT_ERR	Force Event for Command Timeout Error (eMMC Mode Only). A write to the EMSI_FRC_ERRSTAT.CMD_TOUT_ERR address is reflected in the EMSI_ERR_STAT register if the corresponding bit of the EMSI_ERR_STAT_EN register is set.	
		0	Not Affected
		1	Command Timeout Error Status is Set

Interrupt Status Register

The EMSI_ISTAT register reflects the interrupt status. This register is applicable for eMMC mode.

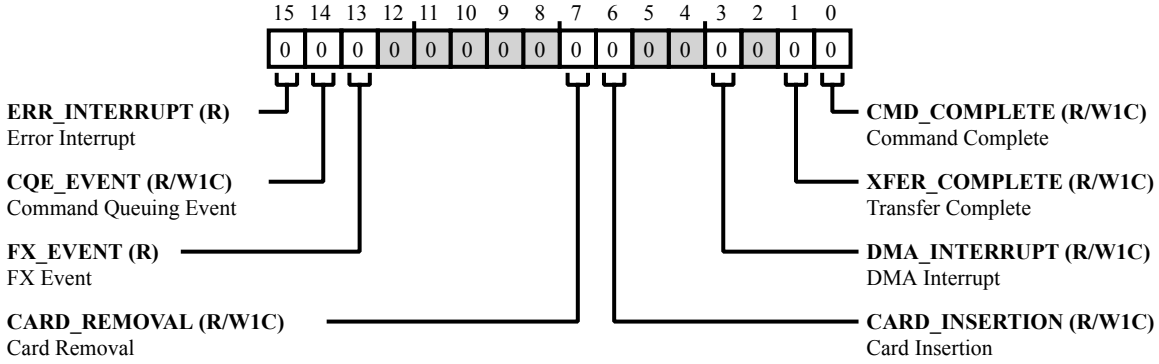


Figure 18-85: EMSI_ISTAT Register Diagram

Table 18-63: EMSI_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	ERR_INTERRUPT	Error Interrupt. The EMSI_ISTAT.ERR_INTERRUPT bit is set if any bit in the EMSI_ERR_STAT register are set.	
		0	No Error
		1	Error
14 (R/W1C)	CQE_EVENT	Command Queuing Event. The EMSI_ISTAT.CQE_EVENT status bit is set if command queuing related event has occurred in eMMC mode. Read the EMSI_CQ_ISTAT/EMSI_CQ_TERRINFO registers for more details.	
		0	No Event
		1	Command Queuing Event is Detected
13 (R/NW)	FX_EVENT	FX Event. The EMSI_ISTAT.FX_EVENT status bit is set when R[14] of the response register is set (=1) and response type R1 is cleared (=0) in the EMSI_TRNSFRMODE register. This interrupt is used with the response check function.	
		0	No Event
		1	FX Event is Detected

Table 18-63: EMSI_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	CARD_REMOVAL	Card Removal. The EMSI_ISTAT.CARD_REMOVAL bit is set if the card inserted in the present state register changes from 1 to 0.	
		0	Card State Stable or Debouncing
		1	Card Removed
6 (R/W1C)	CARD_INSERTION	Card Insertion. The EMSI_ISTAT.CARD_INSERTION bit is set if the card inserted in the present state register changes from 0 to 1.	
		0	Card State Stable or Debouncing
		1	Card Inserted
3 (R/W1C)	DMA_INTERRUPT	DMA Interrupt. The EMSI_ISTAT.DMA_INTERRUPT bit is set if the eMSI controller detects the SDMA Buffer Boundary during transfer. In case of ADMA, by setting the Int field in the descriptor table, the eMSI controller generates this interrupt. This interrupt is not generated after a transfer complete.	
		0	No DMA Interrupt
		1	DMA Interrupt is Generated
1 (R/W1C)	XFER_COMPLETE	Transfer Complete. The EMSI_ISTAT.XFER_COMPLETE bit is set when a read/write transfer and a command with status busy is completed.	
		0	Not Complete
		1	Command Execution is Completed
0 (R/W1C)	CMD_COMPLETE	Command Complete. In an eMMC Mode, the EMSI_ISTAT.CMD_COMPLETE bit is set when the end bit of a response except for Auto CMD12 and Auto CMD23. This interrupt is not generated when the EMSI_TRNSFRMODE.RESP_INT_DIS bit is set (=1).	
		0	No Command Complete
		1	Command Complete

Interrupt Status Enable Register

The EMSI_ISTAT_EN register enables/masks the interrupt status for the corresponding bit in the interrupt status register (EMSI_ISTAT) register. This register is applicable for eMMC mode.

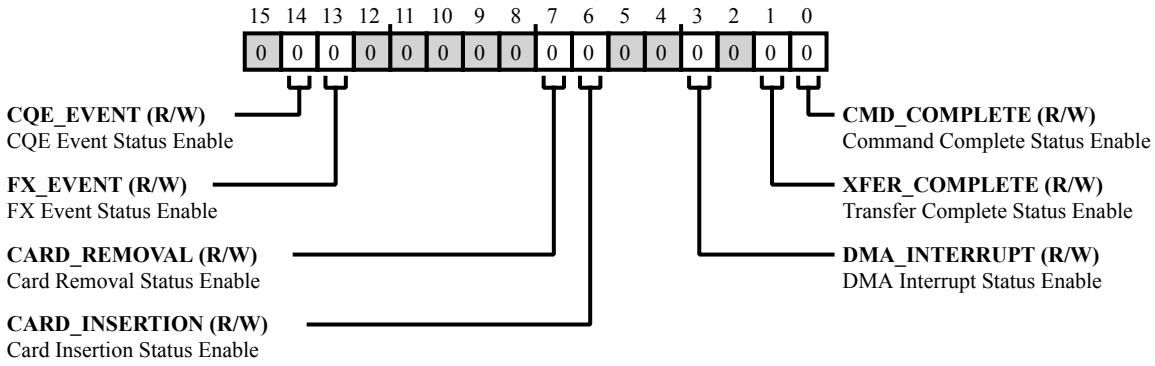


Figure 18-86: EMSI_ISTAT_EN Register Diagram

Table 18-64: EMSI_ISTAT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	CQE_EVENT	CQE Event Status Enable. The EMSI_ISTAT_EN.CQE_EVENT bit, when set (=1), enables the interrupt status for the corresponding bit in the EMSI_ISTAT register.	
		0	Masked
		1	Enabled
13 (R/W)	FX_EVENT	FX Event Status Enable. The EMSI_ISTAT_EN.FX_EVENT bit is added from Version 4.10.	
		0	Masked
		1	Enabled
7 (R/W)	CARD_REMOVAL	Card Removal Status Enable. The EMSI_ISTAT_EN.CARD_REMOVAL bit, when set (=1), enables the interrupt status for the corresponding bit in the EMSI_ISTAT register.	
		0	Masked
		1	Enabled
6 (R/W)	CARD_INSERTION	Card Insertion Status Enable. The EMSI_ISTAT_EN.CARD_INSERTION bit, when set (=1), enables the interrupt status for the corresponding bit in the EMSI_ISTAT register.	
		0	Masked
		1	Enabled

Table 18-64: EMSI_ISTAT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	DMA_INTERRUPT	DMA Interrupt Status Enable. The `EMSI_ISTAT_EN.DMA_INTERRUPT` bit, when set (=1), enables the interrupt status for the corresponding bit in the `EMSI_ISTAT` register.	
		0	Masked
		1	Enabled
1 (R/W)	XFER_COMPLETE	Transfer Complete Status Enable. The `EMSI_ISTAT_EN.XFER_COMPLETE` bit, when set (=1), enables the interrupt status for the corresponding bit in the `EMSI_ISTAT` register.	
		0	Masked
		1	Enabled
0 (R/W)	CMD_COMPLETE	Command Complete Status Enable. The `EMSI_ISTAT_EN.CMD_COMPLETE` bit, when set (=1), enables the interrupt status for the corresponding bit in the `EMSI_ISTAT` register.	
		0	Masked
		1	Enabled

Interrupt Signal Enable Register

The EMSI_ISTAT_INTEN register is used to select the interrupt status that is indicated to the core as the interrupt. All these status bits share the same 1-bit interrupt line. Setting any of these bits to 1 enables interrupt generation. This register is applicable for eMMC mode.

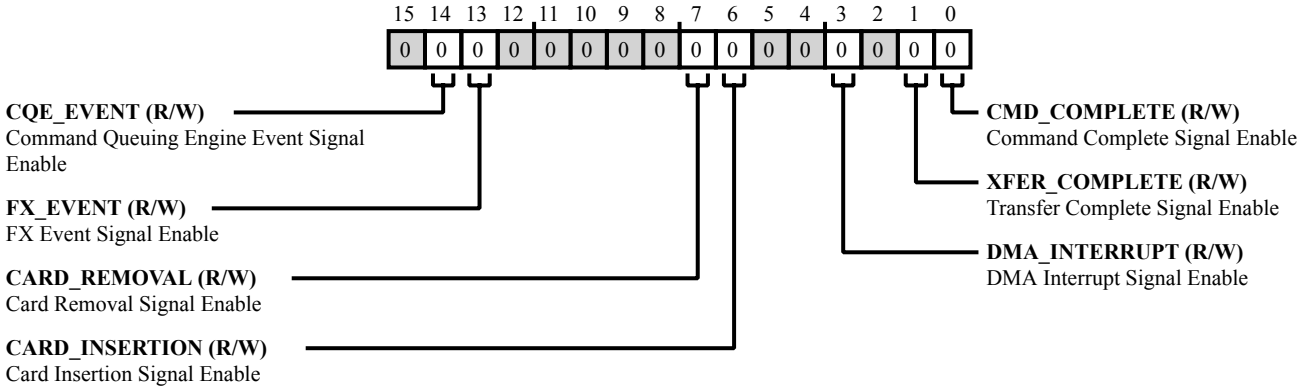


Figure 18-87: EMSI_ISTAT_INTEN Register Diagram

Table 18-65: EMSI_ISTAT_INTEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	CQE_EVENT	Command Queuing Engine Event Signal Enable. When the EMSI_ISTAT_INTEN.CQE_EVENT bit is set (=1) it selects the interrupt status that is indicated to the core.	
		0	Masked
		1	Enabled
13 (R/W)	FX_EVENT	FX Event Signal Enable. When the EMSI_ISTAT_INTEN.FX_EVENT bit is set (=1) it selects the interrupt status that is indicated to the core.	
		0	Masked
		1	Enabled
7 (R/W)	CARD_REMOVAL	Card Removal Signal Enable. When the EMSI_ISTAT_INTEN.CARD_REMOVAL bit is set (=1) it selects the interrupt status that is indicated to the core.	
		0	Masked
		1	Enabled

Table 18-65: EMSI_ISTAT_INTEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	CARD_INSERTION	Card Insertion Signal Enable. When the `EMSI_ISTAT_INTEN.CARD_INSERTION` bit is set (=1) it selects the interrupt status that is indicated to the core.	
		0	Masked
		1	Enabled
3 (R/W)	DMA_INTERRUPT	DMA Interrupt Signal Enable. When the `EMSI_ISTAT_INTEN.DMA_INTERRUPT` bit is set (=1) it selects the interrupt status that is indicated to the core.	
		0	Masked
		1	Enabled
1 (R/W)	XFER_COMPLETE	Transfer Complete Signal Enable. When the `EMSI_ISTAT_INTEN.XFER_COMPLETE` bit is set (=1) it selects the interrupt status that is indicated to the core.	
		0	Masked
		1	Enabled
0 (R/W)	CMD_COMPLETE	Command Complete Signal Enable. When the `EMSI_ISTAT_INTEN.CMD_COMPLETE` bit is set (=1) it selects the interrupt status that is indicated to the core.	
		0	Masked
		1	Enabled

Preset Value for Default Speed

The EMSI_PRESET_DS register defines preset value for default speed mode in SD mode.

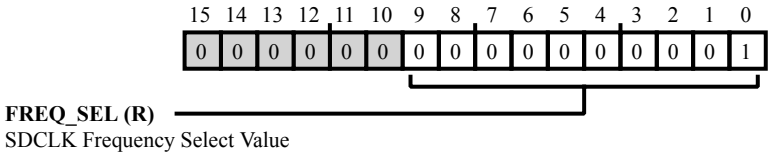


Figure 18-88: EMSI_PRESET_DS Register Diagram

Table 18-66: EMSI_PRESET_DS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	FREQ_SEL	SDCLK Frequency Select Value. The EMSI_PRESET_DS.FREQ_SEL bits configure the 10-bit preset value that is set in the EMSI_CLK_CTL.FREQ_SEL bit field.

Preset Value for HSDDR

The EMSI_PRESET_HSDDR register defines the preset value for high speed DDR speed mode in eMMC mode.

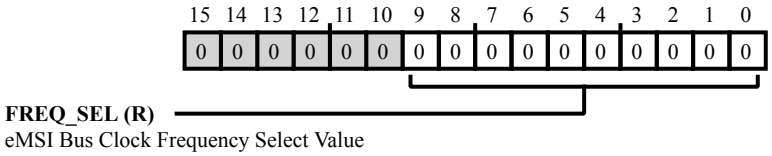


Figure 18-89: EMSI_PRESET_HSDDR Register Diagram

Table 18-67: EMSI_PRESET_HSDDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	FREQ_SEL	eMSI Bus Clock Frequency Select Value. The EMSI_PRESET_HSDDR.FREQ_SEL bits specify a 10-bit preset value that must be set in the EMSI_CLK_CTL.FREQ_SEL bit field, as described by an eMSI system.

Preset Value for HSSDR

The EMSI_PRESET_HSSDR register defines the preset value for high speed SDR speed mode in eMMC mode.

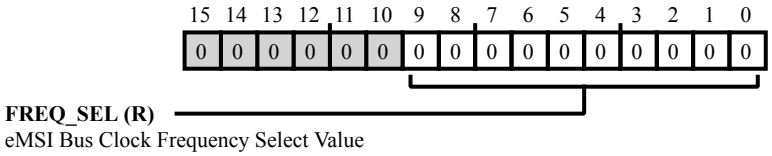


Figure 18-90: EMSI_PRESET_HSSDR Register Diagram

Table 18-68: EMSI_PRESET_HSSDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	FREQ_SEL	eMSI Bus Clock Frequency Select Value. The EMSI_PRESET_HSSDR.FREQ_SEL bits specify a 10-bit preset value that must be set in the EMSI_CLK_CTL.FREQ_SEL bit field, as described by a eMSI system.

Preset Value for Initialization

The EMSI_PRESET_INIT register defines Preset Value for Initialization in eMMC mode.

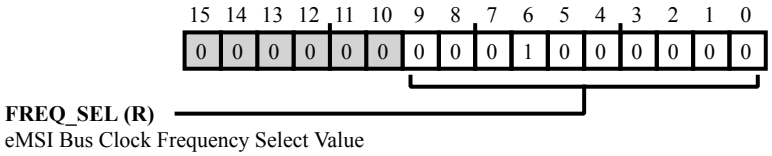


Figure 18-91: EMSI_PRESET_INIT Register Diagram

Table 18-69: EMSI_PRESET_INIT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	FREQ_SEL	eMSI Bus Clock Frequency Select Value. The EMSI_PRESET_INIT.FREQ_SEL bits configure the 10-bit preset value that is set in the EMSI_CLK_CTL.FREQ_SEL bit field. The value should be written manually in the EMSI_CLK_CTL.FREQ_SEL bit field for eMSI bus clock initialization.

Preset Value for Legacy Mode

The EMSI_PRESET_LEGACY register defines the preset value for legacy speed mode in eMMC mode.

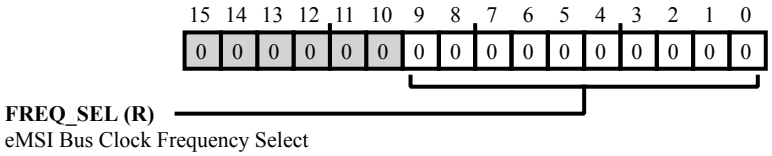


Figure 18-92: EMSI_PRESET_LEGACY Register Diagram

Table 18-70: EMSI_PRESET_LEGACY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	FREQ_SEL	eMSI Bus Clock Frequency Select. The EMSI_PRESET_LEGACY.FREQ_SEL bits specify a 10-bit preset value that must be set in the EMSI_CLK_CTL.FREQ_SEL bit field, as described by a eMSI system.

Present State Register

The EMSI_PSTATE register indicates the present status of the eMSI controller. This register is applicable for eMMC mode.

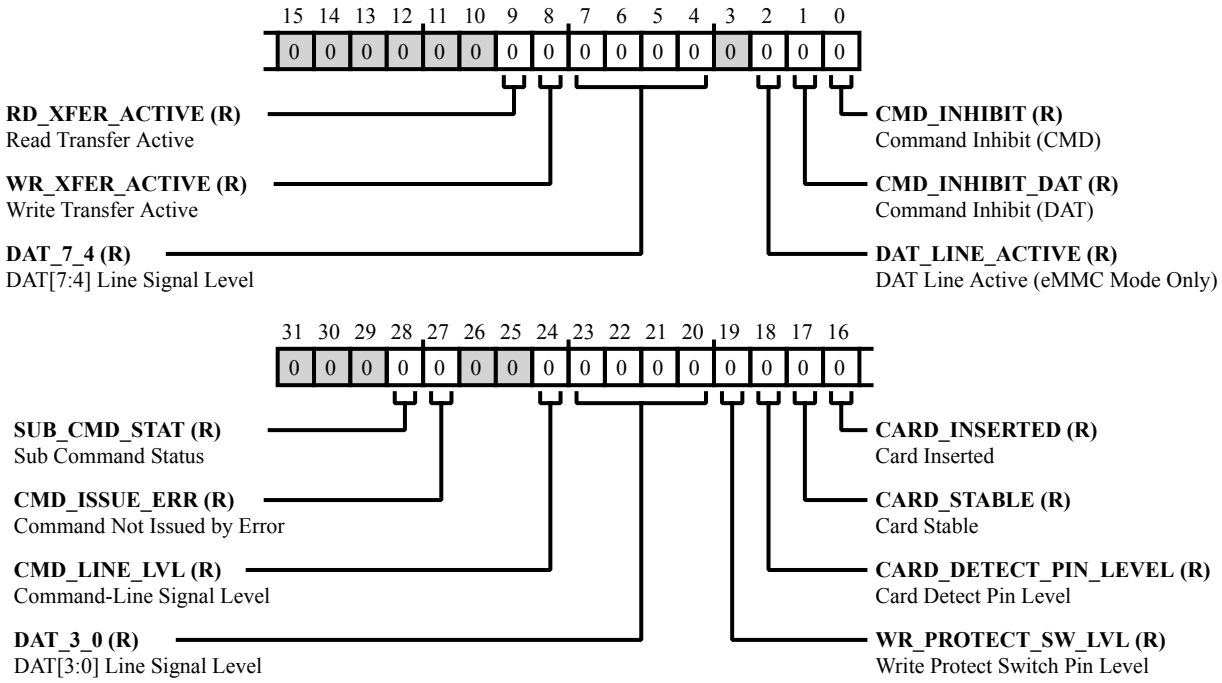


Figure 18-93: EMSI_PSTATE Register Diagram

Table 18-71: EMSI_PSTATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/NW)	SUB_CMD_STAT	Sub Command Status. The EMSI_PSTATE.SUB_CMD_STAT bit is used to distinguish between a main command and a sub command status.	
		0	Main Command Status
		1	Sub Command Status
27 (R/NW)	CMD_ISSUE_ERR	Command Not Issued by Error. The EMSI_PSTATE.CMD_ISSUE_ERR bit is set if a command cannot be issued after setting the command register due to an error except the Auto CMD12 error.	
		0	No Error for Issuing a Command
		1	Command Cannot Be Issued

Table 18-71: EMSI_PSTATE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/NW)	CMD_LINE_LVL	Command-Line Signal Level. The EMSI_PSTATE.CMD_LINE_LVL bit is used to check the CMD line level to recover from errors and for debugging. The EMSI_PSTATE.CMD_LINE_LVL field reflect the value of the CMD signal.	
23:20 (R/NW)	DAT_3_0	DAT[3:0] Line Signal Level. The EMSI_PSTATE.DAT_3_0 bit is used to check the DAT line level to recover from errors and for debugging.	
19 (R/NW)	WR_PROTECT_SW_LVL	Write Protect Switch Pin Level. The EMSI_PSTATE.WR_PROTECT_SW_LVL bit is supported only for memory and combo cards. The EMSI_PSTATE.WR_PROTECT_SW_LVL bit reflects the synchronized value of the WP signal.	
		0	Write Protected
		1	Write Enabled
18 (R/NW)	CARD_DE-TECT_PIN_LEVEL	Card Detect Pin Level. The EMSI_PSTATE.CARD_DETECT_PIN_LEVEL bit reflects the inverse synchronized value of the CD signal.	
		0	No Card Present
		1	Card Present
17 (R/NW)	CARD_STABLE	Card Stable. The EMSI_PSTATE.CARD_STABLE bit indicates the stability of the Card Detect Pin Level. A card is not detected if the EMSI_PSTATE.CARD_STABLE bit is set (=1) and the value of the EMSI_PSTATE.CARD_INSERTED bit is 0.	
		0	Reset or Debouncing
		1	No Card or Inserted
16 (R/NW)	CARD_INSERTED	Card Inserted. The EMSI_PSTATE.CARD_INSERTED bit indicates whether a card has been inserted. The eMSI controller debounces this signal so that the host driver need not wait for it to stabilize.	
		0	Reset, Debouncing, or No Card
		1	Card Inserted
9 (R/NW)	RD_XFER_ACTIVE	Read Transfer Active. The EMSI_PSTATE.RD_XFER_ACTIVE bit indicates whether a read transfer is active for eMMC mode.	
		0	No Valid Data
		1	Transferring Data

Table 18-71: EMSI_PSTATE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	WR_XFER_ACTIVE	Write Transfer Active. The EMSI_PSTATE.WR_XFER_ACTIVE status indicates whether a write transfer is active for eMMC mode.	
		0	No Valid Data
		1	Transferring Data
7:4 (R/NW)	DAT_7_4	DAT[7:4] Line Signal Level. The EMSI_PSTATE.DAT_7_4 bit is used to check the DAT line level to recover from errors and for debugging.	
2 (R/NW)	DAT_LINE_ACTIVE	DAT Line Active (eMMC Mode Only). The EMSI_PSTATE.DAT_LINE_ACTIVE bit indicates whether one of the DAT lines on the eMMC bus is in use. In the case of read transactions, the EMSI_PSTATE.DAT_LINE_ACTIVE bit indicates whether a read transfer is executing on the eMMC bus. In the case of write transactions, the EMSI_PSTATE.DAT_LINE_ACTIVE bit indicates whether a write transfer is executing on the eMMC bus. For a command with busy, this status indicates whether the command executing busy is executing on an eMMC bus.	
		0	DAT Line Inactive
		1	DAT Line Active
1 (R/NW)	CMD_INHIBIT_DAT	Command Inhibit (DAT). The EMSI_PSTATE.CMD_INHIBIT_DAT bit is applicable for eMMC mode and is generated if either DAT line active or read transfer active is set (=1). If the EMSI_PSTATE.CMD_INHIBIT_DAT bit is cleared (=0) it indicates that the eMSI controller can issue subsequent eMMC commands.	
		0	Can Issue Command Which Used DAT Line
		1	Cannot Issue Command Which Used DAT Line

Table 18-71: EMSI_PSTATE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	CMD_INHIBIT	Command Inhibit (CMD). The `EMSI_PSTATE.CMD_INHIBIT` bit indicates the following : For eMMC mode, if the `EMSI_PSTATE.CMD_INHIBIT` bit is cleared (=0) it indicates that the CMD line is not in use and the eMSI controller can issue an eMMC command using the CMD line. The `EMSI_PSTATE.CMD_INHIBIT` bit is set when the command register is written. The `EMSI_PSTATE.CMD_INHIBIT` bit is cleared when the command response is received. The `EMSI_PSTATE.CMD_INHIBIT` bit is not cleared by the response of auto CMD12/23 but cleared by the response of read/write command.	
		0	eMSI Controller is Ready to Issue a Command
		1	eMSI Controller is Not Ready to Issue a Command

Response Register 0

The EMSI_RESP0 register stores 39-08 bits of the Response Field for an eMMC mode. The response for an eMMC command can be a maximum of 128 bits. These 128 bits are segregated into four 32-bit registers, EMSI_RESP0-3.

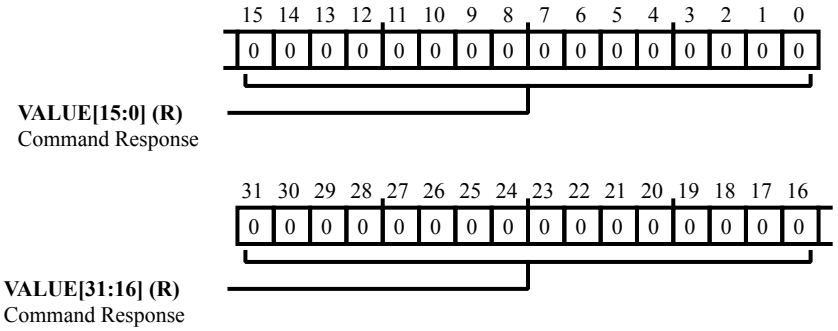


Figure 18-94: EMSI_RESP0 Register Diagram

Table 18-72: EMSI_RESP0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Command Response. The EMSI_RESP0.VALUE bits reflect 39-8 bits of eMMC Response Field. Note: For Auto CMD, the 32-bit response (bits 39-8 of the Response Field) is updated in the EMSI_RESP3 register.

Response Register 1

The EMSI_RESP1 register stores bits 71-40 of the Response Field for eMMC mode. This register is used to store the response from the eMMC device. The response can be a maximum of 128 bits. These 128 bits are segregated into four 32-bit registers EMSI_RESP0-3.

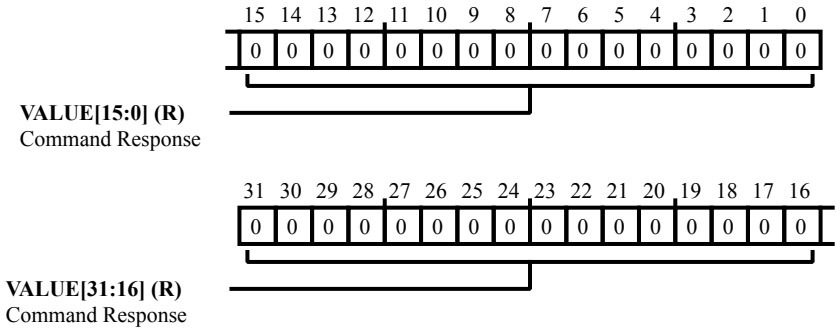


Figure 18-95: EMSI_RESP1 Register Diagram

Table 18-73: EMSI_RESP1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Command Response. The EMSI_RESP1.VALUE bits reflect bits 71-40 of the eMMC Response Field.

Response Register 2

The EMSI_RESP2 register stores 103-72 bits of the Response Field for an eMMC mode. The response for eMMC command can be a maximum of 128 bits. These 128 bits are segregated into four 32-bit registers: EMSI_RESP0-3

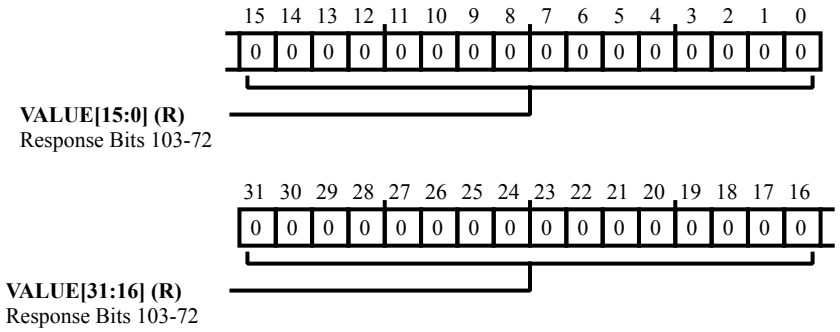


Figure 18-96: EMSI_RESP2 Register Diagram

Table 18-74: EMSI_RESP2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Response Bits 103-72. The EMSI_RESP2.VALUE bit field stores bits 103-72 of the Response Field for eMMC mode. The response for eMMC command can be a maximum of 128 bits.

Response Register 3

The EMSI_RESP3 register stores bits 135-104 of the Response Field for an eMMC mode. The eMMC response can be a maximum of 128 bits. These 128 bits are segregated into four 32-bit registers: EMSI_RESP0-3.

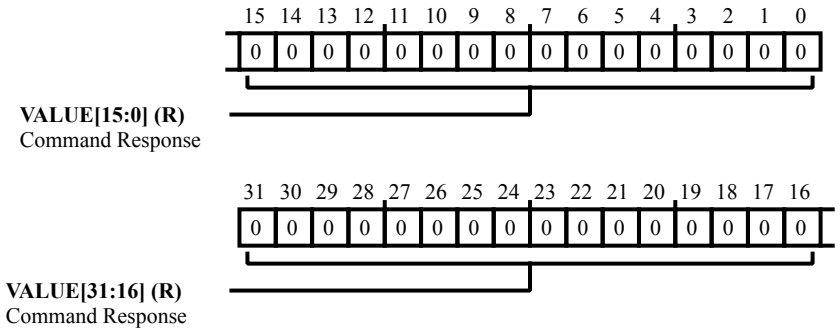


Figure 18-97: EMSI_RESP3 Register Diagram

Table 18-75: EMSI_RESP3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Command Response. These bits reflect bits 135-104 of eMMC Response Field. Note: For Auto CMD, this register also reflects the 32-bit response (bits 39-8 of the Response Field).

SDMA System Address Register

The EMSI_SDMA_ADDR register is used to configure a 32-bit block count or a SDMA system address based on the Host Version 4 Enable bit in the EMSI_CTL2 register. This register is applicable for eMMC mode.

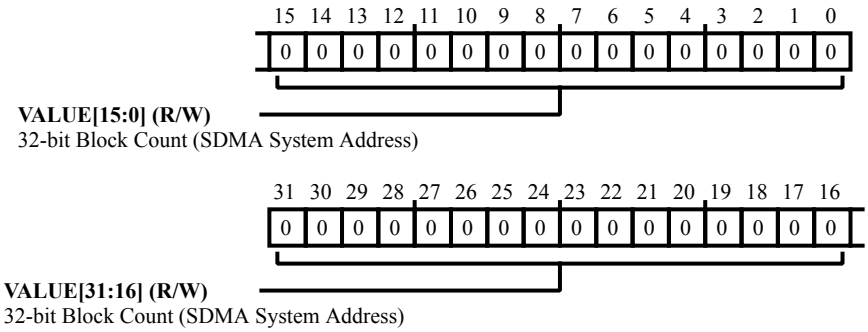


Figure 18-98: EMSI_SDMA_ADDR Register Diagram

Table 18-76: EMSI_SDMA_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	32-bit Block Count (SDMA System Address). The `EMSI_SDMA_ADDR.VALUE` bit field contains the system memory address for an SDMA transfer in the 32-bit addressing mode (Host Version 4 Enable = 0). When the eMSI controller stops a SDMA transfer, this bit field points to the system address of the next contiguous data position. It can be accessed only if no transaction is executing. Reading this register during data transfers may return an invalid value. 32-bit Block Count (Host Version 4 Enable = 1): From the SD Host Controller Version 4.10 specification, this register is redefined as 32-bit block count. The eMSI controller decrements the block count of this register for every block transfer and the data transfer stops when the count reaches zero. Following are the values for BLOCKCNT_SDMASA: - 0xFFFF_FFFF: 4G - 1 Block - ... - 0x0000_0002: 2 Blocks - 0x0000_0001: 1 Block - 0x0000_0000: Stop Count Note: - For Host Version 4 Enable = 0, the host driver does not program the system address in this register while operating in ADMA mode. The system address must be programmed in the ADMA System Address register. - For Host Version 4 Enable = 0, the host driver programs a non-zero 32-bit block count value in this register when Auto CMD23 is enabled for ADMA modes. Auto CMD23 cannot be used with SDMA. - This register must be programmed with a non-zero value for data transfer if the 32-bit block count register is used instead of the 16-bit block count register.

Software Reset Register

The EMSI_SWRST register is used to generate a reset pulse by writing 1 to each bit of this register. After completing the reset, the eMSI controller clears each bit. As it takes some time to complete a software reset, the host driver confirms that these bits are 0. This register is applicable for eMMC mode.

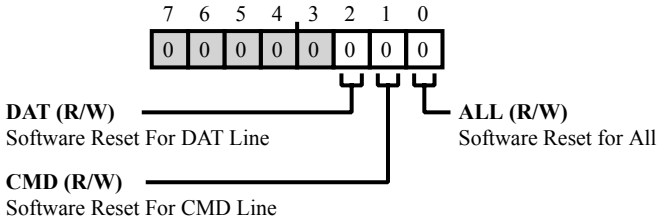


Figure 18-99: EMSI_SWRST Register Diagram

Table 18-77: EMSI_SWRST Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	DAT	Software Reset For DAT Line. The EMSI_SWRST.DAT bit is used in eMMC mode and it resets only a part of the data circuit and the DMA circuit is also reset. The following registers and bits are cleared by the EMSI_SWRST.DAT bit: Present state register • Read transfer active • Write transfer active • DAT line active • Command Inhibit (DAT) Normal Interrupt status register • DMA interrupt • Transfer complete.	
		0	Work
		1	Reset

Table 18-77: EMSI_SWRST Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	CMD	Software Reset For CMD Line. The EMSI_SWRST.CMD bit resets only a part of the command circuit to be able to issue a command. This reset is effective only for a command issuing circuit (including response error statuses related to Command Inhibit (CMD) control) and does not affect the data transfer circuit. The eMSI controller can continue data transfer even after this reset is executed while handling subcommand-response errors. The following registers and bits are cleared by the EMSI_SWRST.CMD bit: - Present State register: Command Inhibit (CMD) bit - Interrupt Status register: Command Complete bit - Error Interrupt Status: Response error statuses related to Command Inhibit (CMD) bit	
		0	Work
		1	Reset
0 (R/W)	ALL	Software Reset for All. This reset affects the entire eMSI controller. During its initialization, the host driver sets (=1) the EMSI_SWRST.ALL bit to reset the eMSI controller. All registers are reset except the capabilities register. If the EMSI_SWRST.ALL bit is set (=1), the host driver must issue reset command and reinitialize the card.	
		0	Work
		1	Reset

Timeout Control Register

The EMSI_TO_CTL register is used to set the Data Timeout Counter value for an eMMC mode according to the timer clock defined by the capabilities register, while initializing the eMSI controller.

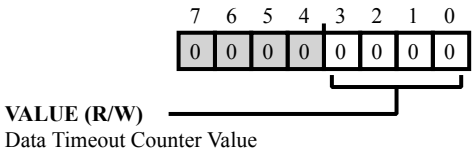


Figure 18-100: EMSI_TO_CTL Register Diagram

Table 18-78: EMSI_TO_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	VALUE	Data Timeout Counter Value. The EMSI_TO_CTL.VALUE value determines the interval by which DAT line time-outs are detected. The timeout clock frequency is generated by dividing the base clock TMCLK value by this value. When setting this register, prevent inadvertent timeout events by clearing the EMSI_ERR_STAT_EN.DATA_TOUT_ERR bit. The values for these bits are: - 0xF: Reserved - 0xE: TMCLK x 2^{27} - ..... - 0x1: TMCLK x 2^{14} - 0x0: TMCLK x 2^{13} Note: During a boot operating in an eMMC mode, an application must configure the boot data timeout value (approximately 1 sec) in the EMSI_TO_CTL.VALUE bit.

Transfer Mode Register

The EMSI_TRNSFRMODE register is used to control the operation of data transfers for an eMMC mode. The host driver sets this register before issuing a command that transfers data.

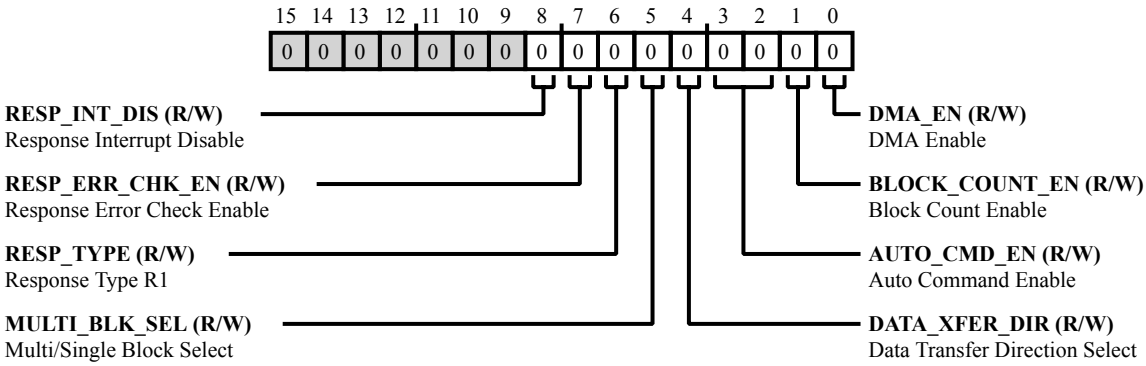


Figure 18-101: EMSI_TRNSFRMODE Register Diagram

Table 18-79: EMSI_TRNSFRMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	RESP_INT_DIS	Response Interrupt Disable. The eMSI controller supports the response check function to avoid overhead of response error check by the host driver. Only a response type of R1 can be checked by the controller. If the host driver checks the response error, set the EMSI_TRNSFRMODE.RESP_INT_DIS bit to 0 and wait for command complete interrupt and then check the response register. If the eMSI controller checks the response error, set the EMSI_TRNSFRMODE.RESP_INT_DIS bit = 1 and set the EMSI_TRNSFRMODE.RESP_ERR_CHK_EN bit = 1. The command complete interrupt is disabled by the EMSI_TRNSFRMODE.RESP_INT_DIS bit regardless of the command complete signal enable.	
		0	Response Interrupt is Enabled
		1	Response Interrupt is Disabled

Table 18-79: EMSI_TRNSFRMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RESP_ERR_CHK_EN	Response Error Check Enable. The eMSI controller supports the response check function to avoid overhead of response error checking by host driver. Response types of only R1 can be checked by the controller. If the eMSI controller checks the response error, set the `EMSI_TRNSFRMODE.RESP_ERR_CHK_EN` bit = 1 and set the `EMSI_TRNSFRMODE.RESP_INT_DIS` bit = 1. If an error is detected, the Response Error interrupt is generated in the `EMSI_ERR_STAT` register. Note: Response error check must not be enabled for any response type other than R1.	
		0	Response Error Check is Disabled
		1	Response Error Check is Enabled
6 (R/W)	RESP_TYPE	Response Type R1. The `EMSI_TRNSFRMODE.RESP_TYPE` bit selects R1 as a response type when the Response Error Check is selected. Error statuses checked in R1: - OUT_OF_RANGE - ADDRESS_ERROR - BLOCK_LEN_ERROR - WP_VIOLATION - CARD_IS_LOCKED - COM_CRC_ERROR - CARD_ECC_FAILED - CC_ERROR - ERROR	
		0	R1 (eMMC)
		1	Reserved
5 (R/W)	MULTI_BLK_SEL	Multi/Single Block Select. The `EMSI_TRNSFRMODE.MULTI_BLK_SEL` bit is set when issuing multiple-block transfer commands using the DAT line. If the `EMSI_TRNSFRMODE.MULTI_BLK_SEL` bit is cleared (=0) it is not necessary to set the block count register.	
		0	Single Block
		1	Multiple Block

Table 18-79: EMSI_TRNSFRMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	DATA_XFER_DIR	Data Transfer Direction Select. The EMSI_TRNSFRMODE.DATA_XFER_DIR bit defines the direction of DAT line data transfers. The EMSI_TRNSFRMODE.DATA_XFER_DIR bit is set (=1) by the host driver to transfer data from the eMMC device to the eMSI controller and it is cleared (=0) for all other commands.	
		0	Write (Host to Card)
		1	Read (Card to Host)
3:2 (R/W)	AUTO_CMD_EN	Auto Command Enable. The EMSI_TRNSFRMODE.AUTO_CMD_EN bit field determines the use of Auto Command functions.	
		0	Auto Command Disabled
		1	Auto CMD12 Enable
		2	Auto CMD23 Enable
		3	Auto CMD Auto Select
1 (R/W)	BLOCK_COUNT_EN	Block Count Enable. The EMSI_TRNSFRMODE.BLOCK_COUNT_EN bit is used to enable the block count register, which is relevant for multiple block transfers. If the EMSI_TRNSFRMODE.BLOCK_COUNT_EN bit is cleared (=0), the block count register is disabled, which is useful in executing an infinite transfer.	
		0	Disable
		1	Enable
0 (R/W)	DMA_EN	DMA Enable. The EMSI_TRNSFRMODE.DMA_EN bit enables the DMA functionality. If the EMSI_TRNSFRMODE.DMA_EN bit is set (=1), a DMA operation begins when the host driver writes to the Command register (or EMSI_ADMA_DESADDR_LO for ADMA3). The program can select one of the DMA modes by using DMA Select in the EMSI_CTL1 register.	
		0	No Data Transfer or Non-DMA Data Transfer
		1	DMA Data Transfer

EMSI Version

The EMSI_VER_ID register reflects the current release number of the eMSI controller.

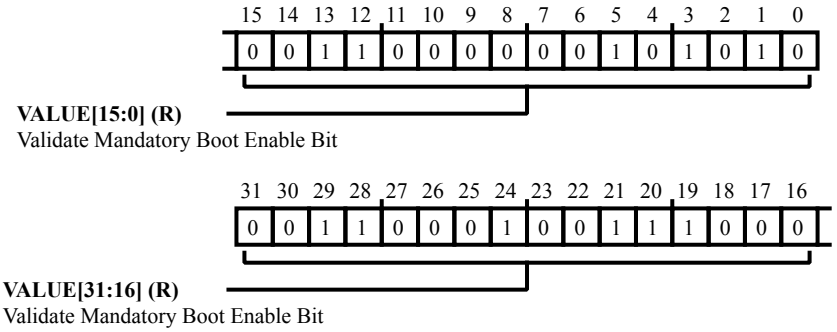


Figure 18-102: EMSI_VER_ID Register Diagram

Table 18-80: EMSI_VER_ID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Validate Mandatory Boot Enable Bit. The EMSI_VER_ID.VALUE bit field indicates the eMSI controller current release number that is read by an application. An application reading this register in conjunction with the register, gathers details of the current release.

EMSI Version Type

The EMSI_VER_TYPE register reflects the current release type of the eMSI controller.

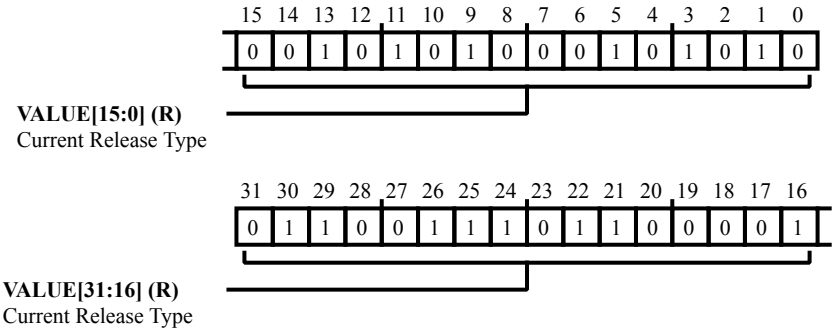


Figure 18-103: EMSI_VER_TYPE Register Diagram

Table 18-81: EMSI_VER_TYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Current Release Type. This field indicates the eMSI controller current release type that is read by an application. An application reading this register in conjunction with the register, gathers details of the current release.

Wakeup Control Register

The EMSI_WU_CTL register is mandatory for the Host Controller, but the wakeup functionality depends on the Host Controller system hardware and software. The Host Driver maintains voltage on the SD Bus by setting the SD Bus Power to 1 in the Power Control Register, while a wakeup event through the Card Interrupt is desired.

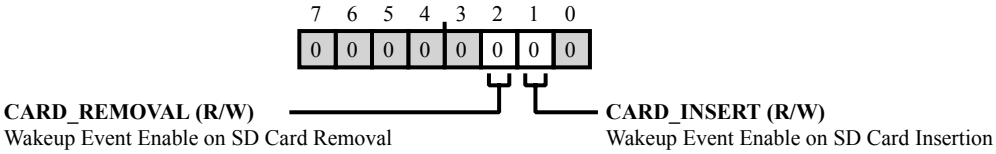


Figure 18-104: EMSI_WU_CTL Register Diagram

Table 18-82: EMSI_WU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	CARD_REMOVAL	Wakeup Event Enable on SD Card Removal. The EMSI_WU_CTL.CARD_REMOVAL bit enables wakeup event through Card Removal assertion in the Normal Interrupt Status register. For the SDIO card, Wake Up Support (FN_WUS) in the Card Information Structure (CIS) register does not affect the EMSI_WU_CTL.CARD_REMOVAL bit.	
		0	Disable
		1	Enable
1 (R/W)	CARD_INSERT	Wakeup Event Enable on SD Card Insertion. The EMSI_WU_CTL.CARD_INSERT bit enables wakeup event through Card Insertion assertion in the Normal Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not affect the EMSI_WU_CTL.CARD_INSERT bit.	
		0	Disable
		1	Enable

19 Link Port (LP)

Link ports allow the processor to connect to other processors or peripheral link ports using a simple communication protocol for high-speed parallel data transfer. This peripheral allows various I/O peripheral interconnection schemes to I/O peripheral devices as well as co-processing and multiprocessing schemes.

The link ports of the processor support 8-bit wide data transfers. The link port pins are multiplexed in the GPIO ports. For information on processor multiplexing, see the data sheet for the specific processor.

Link ports can operate independently and simultaneously, allowing glueless high-speed connectivity of up to four external processors.

NOTE: Reference to CLKO8 in this chapter can be considered as OCLK_0.

LP Features

All link ports are identical in their design and have the following common features.

- Bidirectional ports with eight data signals (`LP_D0` - `LP_D7`), an acknowledge signal (`LP_ACK`), and a clock signal (`LP_CLK`).
- Provide high-speed, point-to-point data transfers to other processors, allowing different types of interconnections between multiple processors.
- Pack data into 32-bit words. The processor can directly read this data or transfer it through DMA to or from on-chip memory.
- Support for data buffering through a 2-deep FIFO for transmit and a 4-deep FIFO for receive.
- Programmable clock and acknowledge based handshake mechanism for efficient communication.
- A dedicated DMA channel.
- Support for data transfer in 2-bit mode (DDR/non-DDR only) and 4-bit mode (DDR/non-DDR)

LP Functional Description

This section provides a description of the link port, including a list of its registers and a functional block diagram.

ADSP-SC59x LP Register List

The Link Port LP is an 8-bit wide parallel port that can connect to another processor's LP or another LP-compatible device. This port allows a variety of interconnection schemes to I/O peripheral devices as well as co-processing and multiprocessing schemes. A set of registers governs LP operations. For more information on LP functionality, see the LP register descriptions.

Table 19-1: ADSP-SC59x LP Register List

Name	Description
LP_CTL	Control Register
LP_DIV	Clock Divider Value Register
LP_RX	Receive Buffer Register
LP_STAT	Status Register
LP_TX	Transmit Buffer Register
LP_TXIN_SHDW	Shadow Input Transmit Buffer Register
LP_TXOUT_SHDW	Shadow Output Transmit Buffer Register

ADSP-SC59x LP Interrupt List

Table 19-2: ADSP-SC59x LP Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
117	LP0_DMA	LP0 DMA Channel		30
118	LP0_STAT	LP0 Status		
119	LP1_DMA	LP1 DMA Channel		36
120	LP1_STAT	LP1 Status		
277	LP0_DMA_ERR	LP0 DMA Data Error		
278	LP1_DMA_ERR	LP1 DMA Data Error		

ADSP-SC59x LP Trigger List

Table 19-3: ADSP-SC59x LP Trigger List Generators

Trigger ID	Name	Description	Sensitivity
43	LP0_DMA	LP0 DMA Channel	
44	LP1_DMA	LP1 DMA Channel	

Table 19-4: ADSP-SC59x LP Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
31	LP0_DMA	LP0 DMA Channel	Pulse
32	LP1_DMA	LP1 DMA Channel	Pulse

ADSP-SC59x LP DMA Channel List

Table 19-5: ADSP-SC59x LP DMA Channel List

DMA ID	DMA Channel Name	Description
DMA30	LP0_DMA	LP0 DMA Channel
DMA36	LP1_DMA	LP1 DMA Channel

Block Diagram

The ***Link Port Block Diagram*** shows the block diagram of a link port.

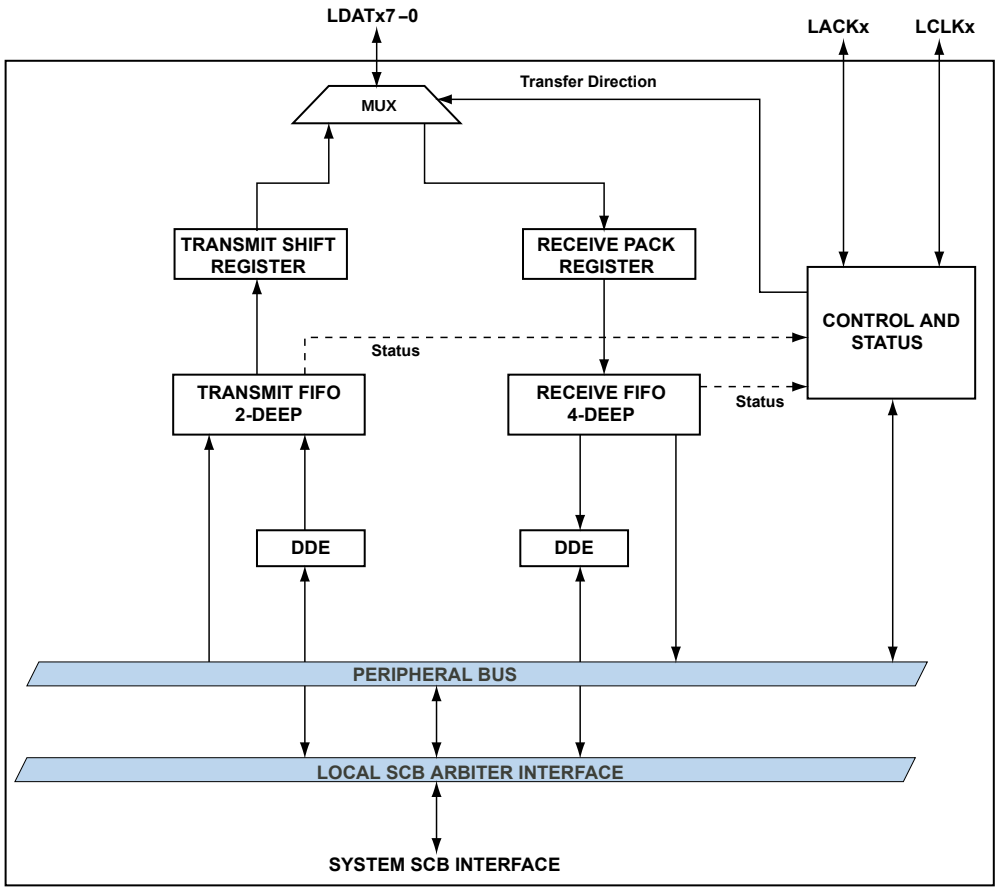


Figure 19-1: Link Port Block Diagram

External Connections

As shown in the ***Link Port Pin Connections*** figure, a link port has eight data lines (LP_D0 – LP_D7), a clock line (LP_CLK), and an acknowledge line (LP_ACK). A link port can act as either a transmitter or a receiver but not both at the same time.

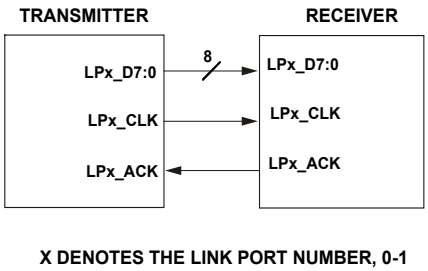


Figure 19-2: Link Port Pin Connections

Use external pull-downs for the LP_CLK and LP_ACK pins so that the link port can enable the transmitter and receiver, irrespective of the state of the other.

Internal Blocks

As shown in the block diagram, the link ports have independent modules for transmit and receive. If enabled as transmitter, the link port uses a 2 deep 32-bit FIFO. If enabled as a receiver, the port uses a 4 deep 32-bit FIFO. The core MMR access bus interfaces with these FIFOs. The distributed DMA engines (DDE) use the system cross bar (SCB) interface to access the FIFO. The link port uses the LP_CTL.TRAN bit to determine whether the module is enabled for transmit or receive operation.

Architectural Concepts

The following sections describe the architectural concepts of the link port.

- Link Port Protocol
- FIFO Buffers
- Handshake for Link Port Enable Process
- Clocking
- Multi-Processor Connectivity

Link Port Protocol (SC59x)

A link port transmitted word consists of 4 bytes and the communication proceeds as follows.

1. The transmitter asserts the link port clock (LP_CLK) with each byte of data. The receiver uses the falling edge of LP_CLK driven by the transmitter to latch the byte.
2. When the receiver is ready to accept another word in the receive buffer it asserts the acknowledge signal, LP_ACK.

3. The transmitter samples LP_ACK driven by the receiver at the beginning of each word transmission. If LP_ACK is deasserted, then the transmitter does not transmit the next word.
4. The transmitter leaves LP_CLK high and continues to drive the first byte of the next word until LP_ACK is asserted.
5. When this assertion occurs, the transmitter drives LP_CLK low. The transmission of the next word starts. If the transmit buffer is empty, LP_CLK remains low until the buffer refills, regardless of the state of LP_ACK.

The LP_ACK signal can deassert when it anticipates that the buffer could fill. The receiver reasserts the LP_ACK signal as soon as the internal DMA grant signal has occurred or the core reads the receive buffer. Either of these actions frees a buffer location.

NOTE: The LP_ACK signal inhibits transmission of the next word and not of the current byte.

The LP_ACK signal provides a handshake between the receiver and transmitter in the following configurations.

- When configured as a transmitter, the port drives both the data and the clock while LP_ACK is three-stated. In this mode, LP_CLK is always synchronous with OCLK_0.
- When configured as a receiver, the link port drives the acknowledge signal and the data and clock lines are three-stated. In this case, the external LP_CLK signal can either be synchronous or asynchronous with OCLK_0.
- When the link port is disabled, the data, clock, and acknowledge signals are three-stated.

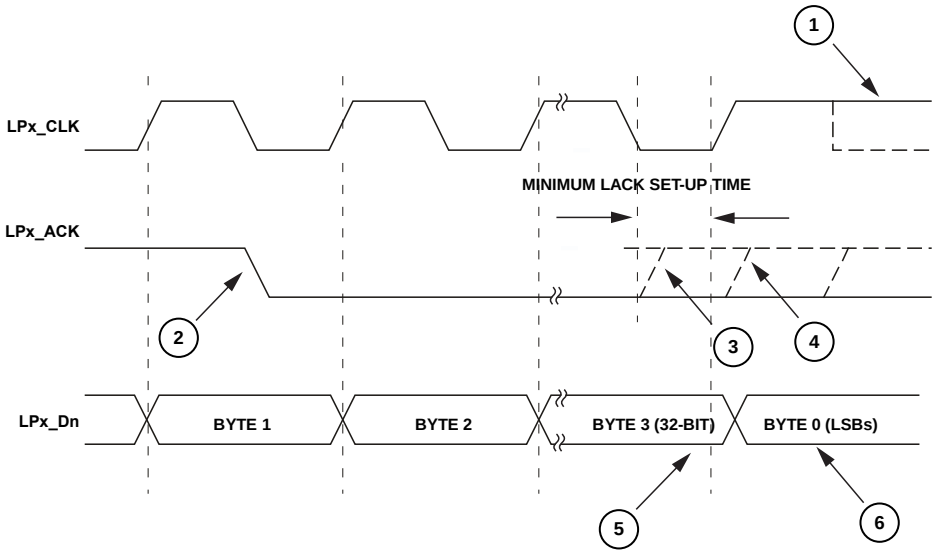


Figure 19-3: Link Port Communication and Handshake Waveform

The following list describes the stages shown in the ***Link Port Communication and Handshake Waveform*** figure.

1. LP_CLK stays high at byte 0 when LP_ACK is sampled low on the previous LP_CLK rising edge. LP_CLK high indicates a stall.
2. The LP_ACK signal can deassert after byte 0.

3. The LP_ACK signal reasserts as soon as the link buffer is not full (depending on Rx FIFO conditions).
4. The transmitter samples LP_ACK to determine whether to transmit the next word.
5. The receiver accepts the remaining word even if LP_ACK is deasserted. The transmitter does not send the following word.
6. Transmission of data for next word is held until LP_ACK is asserted.

The transmitter samples the LP_ACK signal. If the signal is high, the transmitter gives out the falling edges of LP_CLK for data sampling. The LP_ACK signal is first sampled at the rising edge of OCLK_0. One more OCLK_0 stage synchronizes the signal further. This synchronized signal is given to the subsequent logic. The LP_CLK falling edge is aligned with OCLK_0 falling edge in a 1:1 clock ratio mode and with the OCLK_0 rising edge for the rest of the clock ratios. The following figures explain how the synchronization is maintained between the LP_ACK and LP_CLK signals.

In the following figure, synchronizing time is guaranteed to be 1.5 OCLK_0 cycles.

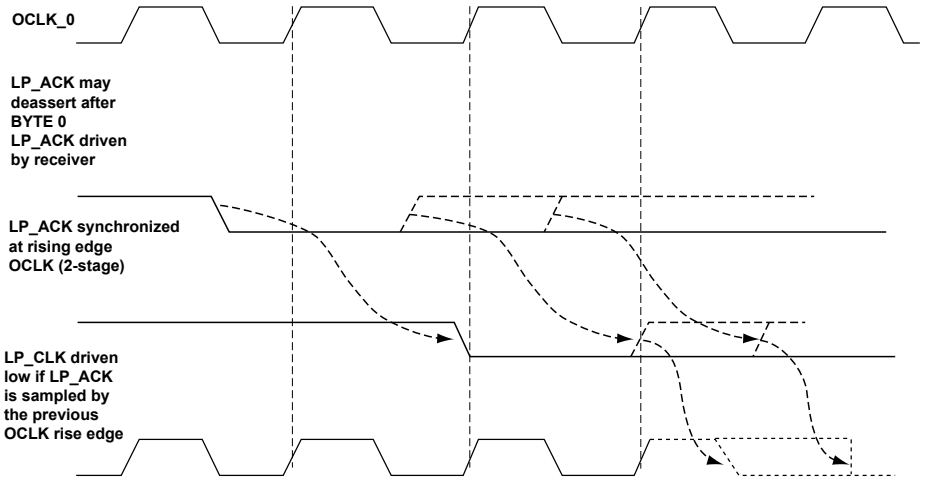


Figure 19-4: LP_ACK Synchronization for OCLK_0:LP_CLK=1:1

In the following figure, synchronizing time is guaranteed to be 2 OCLK_0 cycles.

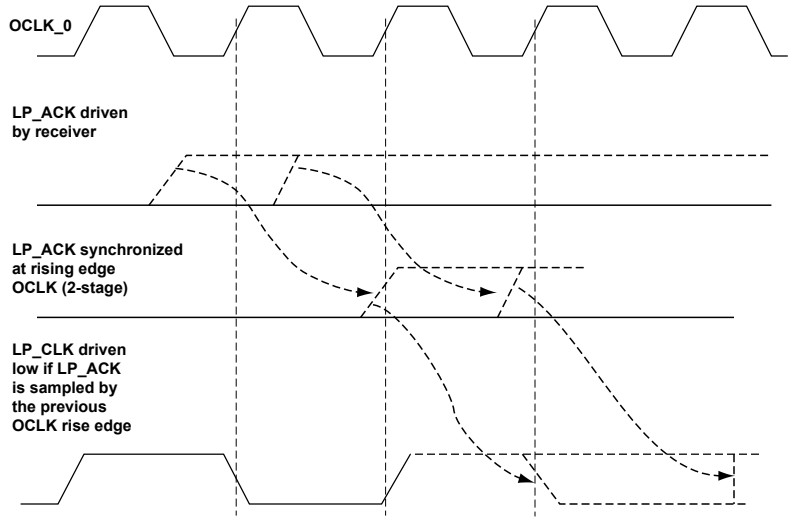


Figure 19-5: LP_ACK Synchronization for OCLK_0 LP_CLK=1:2, 1:4 and Up

The link port uses the value programmed in the LP_DIV register at the transmitter to determine the frequency of the link port clock (LP_CLK). However, the signal appearing on the LP_CLK pin is also dependent on the status of the LP_ACK pin driven by the receiver. The ***Relationship Between Internal Link Port Clock and Link Port Clock at the Pins*** figure shows this relationship.

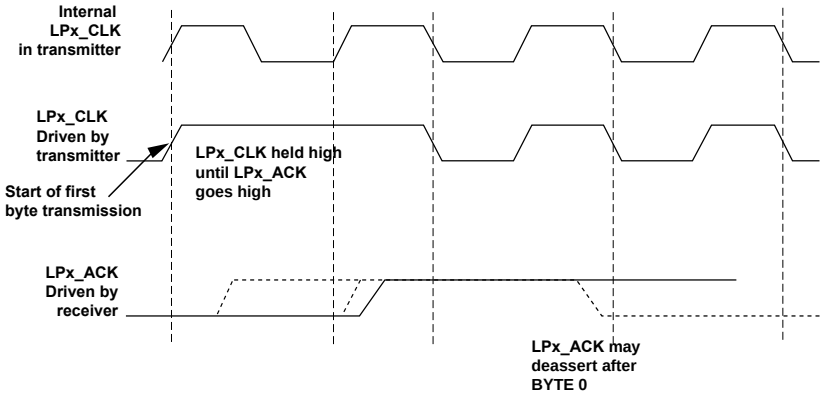


Figure 19-6: Relationship Between Internal Link Port Clock and Link Port Clock at the Pins

FIFO Buffers

When a link port is configured as transmitter, the link port uses a 2-deep FIFO buffer. A shift register unpacks the single 32-bit word to four 8-bit data bytes. As the FIFO has space for more data, the link port makes a new DMA request. If the FIFO becomes empty, the LP_CLK signal is deasserted. The core can access FIFO through the LP_TX register.

The core or DMA makes three writes (2-stage FIFO and 1 shift register) to the transmit buffer before it signals a full condition. The link port uses the LP_STAT.FFST bit field to reflect the status of the FIFO but not the shift register full or empty condition. However, the program can poll the LP_STAT.LPBS bit to discover whether the link

port is driving data from the shift register to the pins. The LP_STAT.LPBS bit is also set when receiver has held off transmission by driving LP_ACK low.

NOTE: When the 2-deep FIFO and the output shift-register overflow, any further write to the link port buffer overwrites the input stage of the FIFO.

NOTE: The core can also read the transmit FIFO through the data LP_TX register.

NOTE: If the transmitter is disabled while performing writes to the transmit FIFO, a FIFO full condition is signaled after two writes.

The transmit buffer registers have shadow registers. Using these shadow registers, both stages of the 2-deep FIFO can be read without updating the status registers. The LP_TXIN_SHDW register corresponds to the input stage of the FIFO. The LP_TXOUT_SHDW register corresponds to the output stage of the FIFO as shown in the ***Transmit FIFO path*** figure.

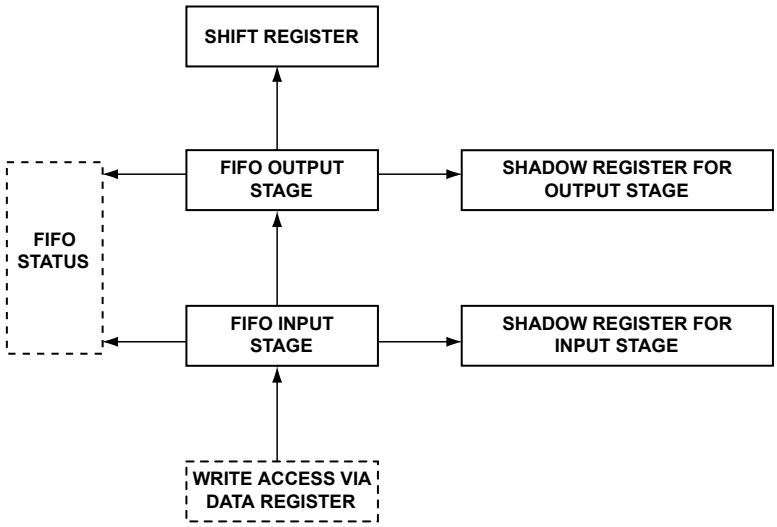


Figure 19-7: Transmit FIFO path

When a link port is configured as receiver, data transfers to the core or DMA from the full 4-deep receive FIFO. An internal packing register packs data to 32 bits. Four reads can occur from the receive buffer by the core or DMA before it signals an empty condition. The link port uses the LP_STAT.FFST bits to reflect the status of the 4-deep read buffer FIFO. The core can access this FIFO through the LP_RX register.

NOTE: When receive FIFO overflows (LP_STAT.ROVF bit=1), any further data from the transmitter is lost. Only the data retained in the receive FIFO can be retrieved further.

The receiver drives the LP_ACK output signal low, after the first byte of data for the next-to-last empty slot (in the 4-deep FIFO) is received. This functionality prevents data loss due to the transmitter starting transmission of the next word before the LP_ACK signal reaches the transmitter. (The timing is due to the larger delay in synchronization.) This functionality guarantees that even after allowing for the extra synchronization cycle in the transmitter and receiver, there is no overflow in the receive FIFO. The ***LACK Generation Based on Receive FIFO Status*** figure

shows how FIFO slots influence the acknowledge signal generation. The grayed sections show received data. The white sections show empty locations where the decision to pull LP_ACK high is taken.

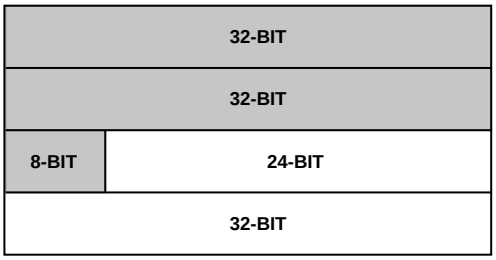


Figure 19-8: LACK Generation Based on Receive FIFO Status

NOTE: The link port uses a 4-deep receive FIFO only under a worst case situation, as mentioned. In all other cases, respond as if the FIFO has only a 3-deep stage. The LP_ACK signal is pulled high before the last stage of the FIFO.

The link port has memory-mapped buffers for both receive and transmit operations. A JTAG-based emulator can read the FIFO which can cause unexpected problems in data transfers. This activity can only happen during an emulation event (typically hitting a breakpoint or single-stepping). The emulator issues core reads through JTAG. To work around this issue, see the tools documentation for more information.

Handshake for Link Port Enable Process

In a link port-based system, the transmitter and the receiver can be enabled at different times. Use external pull-downs for the LP_CLK and LP_ACK signals.

If the receiver is enabled before the transmitter, the external pull-down holds the LP_CLK signal of the transmitter low. The receiver is held off. The receiver can wait for a rising edge on the LP_CLK signal to assert its receive service request interrupt. This rising edge occurs only when transmitter starts driving the first data on to the bus, after the application enables it.

If the transmitter is enabled before the receiver, the external pull-down holds the LP_ACK signal of the receiver low. Transmission is held off. Refer to the ***Enable the Transmitter Before the Receiver*** figure. The transmitter can wait for a rising edge on the LP_ACK signal to assert its transmit service request interrupt. This rising edge is asserted as soon as the receiver is enabled after the hardware drives the LP_ACK high.

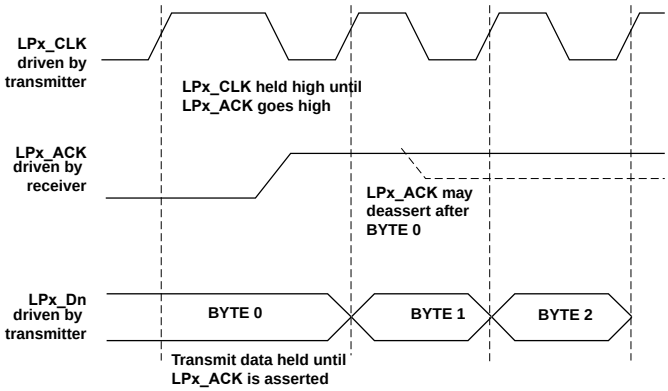


Figure 19-9: Enable the Transmitter Before the Receiver

NOTE: Service request interrupts or status are asserted only when the link port (receiver or transmitter) is disabled.

Clocking

The link port clock (LP_CLK) is derived from the internal system clock (OCLK_0). The link port clock to system clock ratio can be configured in the LP_DIV register. This value applies to the transmitter only. The receiver can operate at any asynchronous frequency up to the maximum frequency, independent of the ratio programmed. The following formula describes the relationship between the frequency of the link port clock, the OCLK_0 frequency, and the LP_DIV value.

- $f_{LP_CLK} = f_{OCLK_0}$< or = $f_{LP_CLK\text{-}MAX}$ if DIV = 0
- $f_{LP_CLK} = f_{OCLK_0}/(2 \times DIV)$ if DIV > 0

Where: f_{LP_CLK} = link clock frequency, $f_{LP_CLK\text{-}MAX}$ = link clock maximum frequency, and f_{OCLK0_0} = system clock frequency.

While programming the LP_DIV register to select the clock ratio, ensure that the LP_CLK frequency does not exceed the maximum frequency supported for the device. For supported frequencies, see the product-specific data sheet.

NOTE: The Drive Strength (DS) value must be changed to b(010) for the LPCLK pin to operate the link port at the maximum frequency of 125 MHz. Refer the PADS_PORTB0_DS and PADS_PORTC0_DS register bit descriptions for details.

Multi-Processor Connectivity

Link ports can operate independently, allowing glueless connection with external processors. Link ports have dedicated DMA channels, allowing independent data transfers. The following group of figures shows some examples of different bus connection topology that can be used in multi-processor system design. The inter-connection methods are not limited to these examples.

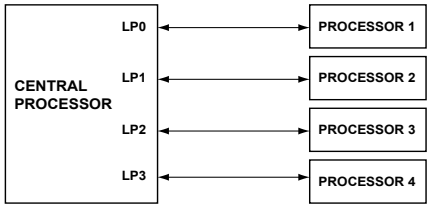


Figure 19-10: Central Processor-Based Model

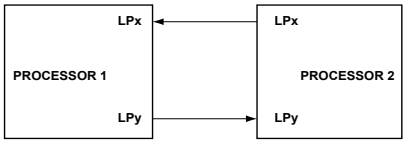


Figure 19-11: Link Port Full-Duplex Transfer Model

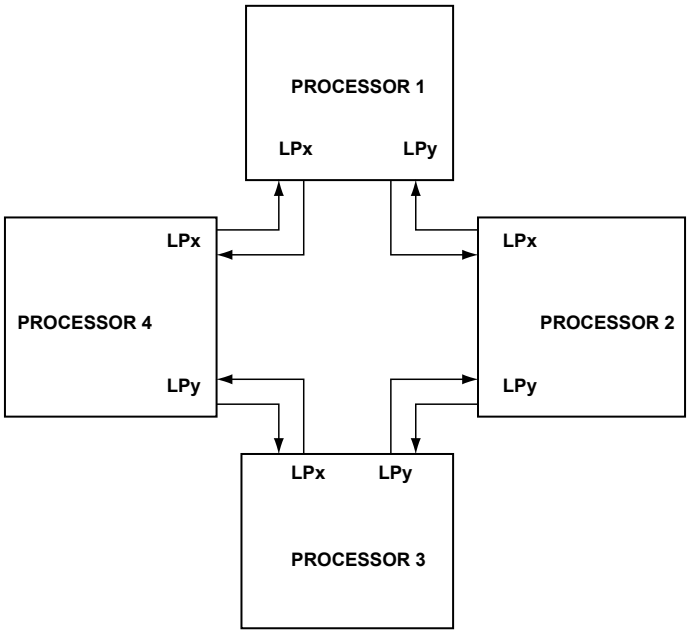


Figure 19-12: Link Port Ring Model

The link port protocol does not include built-in support for multiple requesters. However, there can be situations where multiple devices try to become the bus requester at the same time. Multi-requester conflicts can be resolved using token passing. In token passing, the token is a software flag that passes between processors.

At reset, the token is set to reside in the link port of one device, making it the requester and the transmitter. When a receiver (completer) wants to become the requester, it can assert its `LP_ACK` signal to get the attention of the requester. The requester knows, through the software protocol, whether to respond with actual data or whether the token is requested. If the requester wishes to give up the token, it can send back a user-defined token release word and thereafter clear its token flag.

Simultaneously, the completer sets its token and can thereafter transmit. The token release word can be any user-defined value. Because the transmitter and receiver expect a code word, this word does not need to be exclusive of

normal data transmission. If the requester wishes to give up the token, it can send back a user-defined token release word and thereafter clear its token flag. Simultaneously, the completer examines the data sent back and if it is the token release word, the completer sets its token, and can thereafter transmit.

The link port protocol includes handshake mechanism to inform the other end of transfer (transmit or receive) of an enable instance. However, it does not support handshakes to inform a disabled instance, while a chunk of data transfers. The application must assume the disabled state of the other end and take appropriate action.

For example, in a multi-processing environment, a receiver did not read its full FIFO for an extended time due to internal bus arbitrations. The transmitter can require software or a peripheral timer-based timeout to inform the application that the `LP_ACK` signal is low for an extended period.

LP Operating Modes

The link port does not have particular modes of operation, as the peripheral is based on a simple protocol. The following sections explain the data transfer modes, using the core and using DMA.

- Core Data Transfers
- DMA Data Transfers

LP Data Transfer Modes

This section describes link port DMA and core data transfers.

Core Data Transfers

If DMA is disabled for a link port buffer, the processor core can write or read internal FIFO buffers as a memory-mapped register through the MMR access bus. To avoid FIFO overflow or underflow, the core can access the FIFO registers in one of the following ways.

- Access link port registers using an interrupt service routine (ISR) mapped to the data request interrupt of the link port. The interrupt request remains high only when the FIFO is accessible (when the FIFO is not full in transmit mode and not empty in receive mode).
- Poll the FIFO status bits of the `LP_STAT` register. Write to the transmit FIFO if not full or read from the receive FIFO if not empty.

DMA Data Transfers

Dedicated DMA channels are available for each link port. DMA-related activity is explained in the following steps.

1. Data Receive – Once the DMA channel and link port module are configured and enabled, the external device begins writing data to the FIFO through the data pins of the link port. The FIFO detects this activity and in turn sends a DMA request. After the request is granted, the DMA transfer progresses until the FIFO is empty.
2. Data Transmit – Once the DMA channel and link port module are configured and enabled, setting the `LP_CTL.EN` bit automatically asserts a DMA request when the transmit FIFO is empty. After the request is granted, DMA fills the FIFO. The external device begins reading data from the FIFO through the data pins of

the link port. The FIFO detects that there is room in the buffer and asserts another DMA request, continuing the process.

LP Event Control

This section describes how the link port uses interrupts and status signals.

Interrupt Signals

Each link port has two dedicated interrupt lines registered with the system event controller—a data request interrupt and a status interrupt. Data request interrupts are asserted based on FIFO conditions for data transfer. Status interrupts are asserted when a service request status or an overflow status is set. The following list explains each of these interrupts.

- ***Data Request Interrupt.*** Asserted if the FIFO is not full in transmission mode and the FIFO is not empty in reception mode. This functionality serves as a core triggered interrupt in non-DMA mode and as the DMA interrupt request in DMA mode. Generation of this interrupt is based on the `LP_STAT.FFST` (status bit of the link port buffer).
- ***Link Port Transmit Service Request Interrupt (LTRQ)***. Allow a disabled link port to generate an interrupt when an external access is attempted. When a link port is configured as transmitter, the transmit service request interrupt is enabled by setting the `LP_CTL.TRQMSK` bit. When set, an external receiver can indicate to the disabled transmitter that it must receive data through the connected link port. The receiver does so by driving a high level on the `LP_ACK` line. When the `LP_ACK` of the disabled transmitter link port is detected high, a `LP_STAT.LTRQ` interrupt is generated. The transmitter can enable itself for data transfer with the receiver. The link port needs a pull-down on `LP_ACK` for this feature to function properly.
- ***Link Port Receive Service Request Interrupt (LRRQ)***. When a link port is configured as receiver, this interrupt is enabled by setting the `LP_CTL.RRQMSK` bit. When set, an external transmitter can indicate to the disabled receiver that it must receive data through the connected link port. The transmitter does so by driving out the first data. When the `LP_CLK` of the disabled receiver link port is detected high, a `LP_STAT.LRRQ` interrupt is generated. The receiver can further enable itself for data transfer with the transmitter. The link port needs a pull-down on the `LP_CLK` signal for this feature to function properly.
- ***Link Port Receive Overflow Interrupt (LPOVF)***. Generated when the receiver FIFO overflows and is enabled by setting the `LP_CTL.ROVFMSK` bit. This interrupt can happen if the transmitter continues to transmit data even though the receiver has deasserted `LP_ACK` signal causing the receive FIFO to overflow.

Enabling Link Port Interrupts

A data request interrupt is fed to the system event controller directly and can be controlled separately from the application.

To mask the interrupt, set the mask bits in `LP_CTL` register corresponding to service interrupts and the overflow interrupt. These interrupts are OR'ed and fed to the SIC as a single `LP_STAT` interrupt. These interrupts are latched and stored in the associated bits of `LP_STAT` register. If an `LP_STAT` interrupt occurs, in the ISR, programs

can read the LP_STAT register bits to determine the type of interrupt. These bits are write-one-to-clear (W1C); writing one to the bit resets the bit and disables the corresponding interrupt.

Status and Error Signals

This section explains the various status signals in the LP_STAT register.

- ***Transfer Status signals***. The link port uses the bus status bit (LP_STAT.LPBS) to give the status of the bus condition (busy or idle) when the link port is configured as transmitter. The LP_STAT.LPBS is high if the link port drives data into the link port pins. Programs can poll this bit after polling the LP_STAT.FFST bit to disable the link port safely.

  The link buffer status (LP_STAT.FFST) field directly indicates the status of the FIFO (including empty or full conditions) during data transfer. Software can poll this field in the LP_STAT register before writing to the FIFO (in case of transmission) or reading from the FIFO (in case of reception). The LP_STAT.FFST bit is automatically cleared when the link port is disabled.

- ***Transfer Request Status signals***. The link port uses the receive request status (LP_STAT.LRRQ) bit to indicate that an external receiver wants to receive data (in case the link port is a disabled transmitter). The link port uses the transmit request status (LP_STAT.LTRQ) bit to indicate that an external transmitter wants to send data (in case the link port is a disabled receiver). Software can poll these bits to enable the transmitter or receiver accordingly.

- ***Error Status signals***. In receive mode 32-bit data is received in four chunks of 8-bit data. This data is then packed to a single 32-bit data before loading the FIFO. The link buffer error status (LP_STAT.LPACK) bit is high during this packing process and goes low after packing.

  The link port overflow status (LP_STAT.ROVF) bit is set when the receive FIFO overflows. This event can occur if the transmitter continues to transmit data even though the receiver has deasserted LP_ACK causing the receiver FIFO to overflow.

LP Programming Model

The following sections provide information on configuring the operating mode and enabling the link ports.

- Setting Up a DMA Transmit Operation
- Setting Up a DMA Receive Operation
- Setting Up a Core Transmit Operation
- Setting Up a Core Receive Operation

Setting Up a DMA Transmit Operation

This following procedure describes the typical steps for configuring the link ports in DMA transmit mode.

1. Enable the link port pins in the GPIO port mux using the appropriate PORT_FER and PORT_MUX registers.

2. Configure the drive strength value to b(010) for the link port clock pins to operate the link port at maximum frequency (PADS_PORTB0_DS and PADS_PORTC0_DS registers).
3. Install interrupt handlers for DMA and for transfer status (service request interrupt).
4. Configure the link port to transmit by setting the LP_CTL bit and enable the transmit request interrupt mask by setting the LP_CTL.TRQMSK bit.
5. Program the link port clock divider by writing a value to the LP_DIV register.
6. If using DMA stop mode or auto buffer mode, program the appropriate DMA registers.

   ADDITIONAL INFORMATION: An example configuration is: DMA_ADDRSTART, DMA_XCNT, DMA_XMOD, and DMA_CFG registers (Stop/Auto, DMA_CFG.PSIZE =1, DMA_CFG.MSIZE =4, interrupt generation and memory read).
7. Wait for the link port receiver (connected externally) to be enabled. The application can wait for the transmit service request interrupt to assert.
8. Clear the transmit service request interrupt status by writing 1 to the LP_STAT.LTRQ bit.
9. Enable DMA by setting the DMA_CFG.EN bit.
10. Enable the link port by setting the LP_CTL.EN bit.
11. Wait for DMA to assert a transfer completion interrupt.
12. Clear the DMA interrupt source by writing 1 to the DMA_STAT.IRQDONE bit.

Setting Up a DMA Receive Operation

This section describes the typical steps for using the link ports in DMA receive mode.

1. Enable the link port pins in GPIO port mux using the appropriate PORT_FER and PORT_MUX registers.
2. Install interrupt handlers for DMA and for transfer status (service request interrupt).
3. Configure the link port for reception (clear the LP_CTL.TRAN bit) and enable the receive request interrupt mask by setting the LP_CTL.RRQMSK bit.
4. If using DMA stop mode or auto buffer mode, program the DMA registers.

   ADDITIONAL INFORMATION: An example configuration is: DMA_ADDRSTART, DMA_XCNT, DMA_XMOD, and DMA_CFG registers (Stop/Auto, DMA_CFG.PSIZE=1, DMA_CFG.MSIZE=4, interrupt generation and memory write).
5. If using DMA array mode or list mode, create DMA configuration data structures filled with components.

   ADDITIONAL INFORMATION: An example configuration is: DMA_ADDRSTART, DMA_XCNT, DMA_XMOD, and DMA_CFG registers (Array/List, DMA_CFG.PSIZE =1, DMA_CFG.MSIZE =4, interrupt generation, memory write and fetch =4/5) and DMA_DSCPTR_NXT register (if list mode). Further, program

DMA configuration register (Array/List, DMA_CFG.PSIZE =1, DMA_CFG.MSIZE =4, Memory Write and Fetch =4/5) and program the DMA_DSCPTR_NXT register (if list mode).

6. Wait for the link port transmitter (connected externally) to be enabled with subsequent data transmission. The application can wait for the receive service request interrupt to assert.
7. Clear the receive service request interrupt status by writing 1 to the LP_STAT.LRRQ bit.
8. Enable DMA by setting the DMA_CFG.EN bit.
9. Enable the link port by setting the LP_CTL.EN bit.
10. Wait for DMA to assert the transfer complete interrupt.
11. Clear the DMA interrupt source by writing 1 to the DMA_STAT.IRQDONE bit of the DMA status register.

Setting Up a Core Transmit Operation

This section describes the typical steps for using the link ports in processor core based transmission.

1. Enable the link port pins in the GPIO port mux using the appropriate PORT_FER and PORT_MUX registers.
2. Configure the drive strength value to b(010) for the link port clock pins to operate the link port at maximum frequency (PADS_PORTB0_DS and PADS_PORTC0_DS registers).
3. Install interrupt handlers for data transfer and for transfer status (service request interrupt). The interrupt handlers for data transfer are the same source or ID as the DMA interrupt line in the SEC.
4. Configure the link port for transmission by setting the LP_CTL.TRAN bit) and enable the transmit request interrupt mask by setting the LP_CTL.TRQMSK bit).
5. Program the link port clock divider by writing a value in to the LP_DIV register.
6. Wait for the link port receiver (connected externally) to be enabled. The application can wait for a transmit service request interrupt to assert.
7. Clear the transmit service request interrupt status by writing 1 to the LP_STAT.LTRQ bit.
8. Enable the link port by setting the LP_CTL.EN bit.
9. The data request interrupt is asserted whenever there is free space in the FIFO. The application can write to the LP_TX register based on the FIFO conditions (half or empty) reflected in the LP_STAT.FFST bit field.

Setting Up a Core Receive Operation

This section describes the typical steps for using the link ports in processor core-based reception.

1. Enable the link port pins in the GPIO port mux using the appropriate PORT_FER and PORT_MUX registers.
2. Install interrupt handlers for data transfer and for transfer status (service request interrupt). The interrupt handlers for data transfer are the same source or ID as the DMA interrupt line in the SEC).

3. Configure link port for reception (clear LP_CTL.TRAN bit). Enable the receive request interrupt mask bit (set LP_CTL.RRQMSK).
4. Wait for the link port transmit (connected externally) to be enabled with subsequent transmission of data. The application can wait for receive service request interrupt to be asserted.
5. Clear the receive service request interrupt status by writing 1 to the LP_STAT.LRRQ bit.
6. Enable the link port by setting the LP_CTL.EN bit.
7. The data request interrupt is asserted whenever there is free space in the FIFO. The application can read from the LP_RX register based on the FIFO conditions (1 or 2 or 3 data available) which is reflected in the LP_STAT.FFST bit field.

ADSP-SC59x LP Register Descriptions

Link Port (LP) contains the following registers.

Table 19-6: ADSP-SC59x LP Register List

Name	Description
LP_CTL	Control Register
LP_DIV	Clock Divider Value Register
LP_RX	Receive Buffer Register
LP_STAT	Status Register
LP_TX	Transmit Buffer Register
LP_TXIN_SHDW	Shadow Input Transmit Buffer Register
LP_TXOUT_SHDW	Shadow Output Transmit Buffer Register

Control Register

The LP_CTL register provides LP interrupt masking, selection of transfer direction, and link port enable.

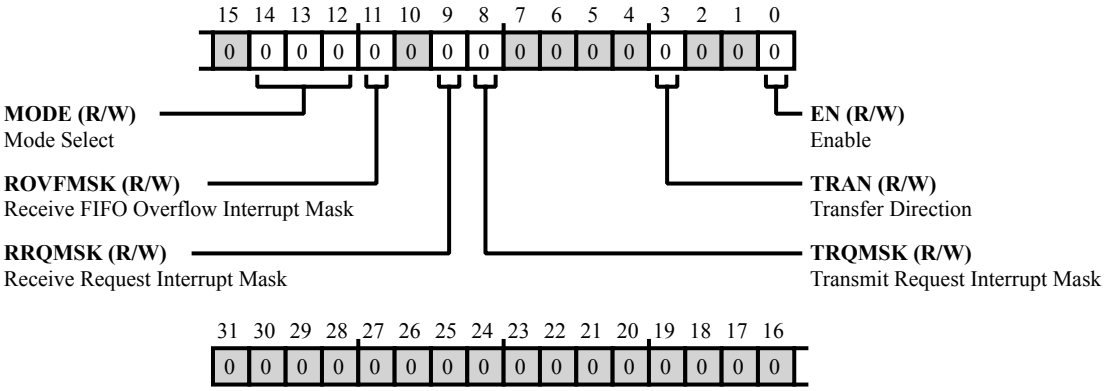


Figure 19-13: LP_CTL Register Diagram

Table 19-7: LP_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
14:12 (R/W)	MODE	Mode Select. The LP_CTL.MODE field indicates the mode of operation: Single Data Rates (SDR) of 2 bit and 4 bit, Double Data Rates (DDR) of 2 bit, 4 bit, and 8 bit. Note that the LRRQ interrupt is not supported in DDR mode. The LP receiver must be enabled by the user before starting the LP transmitter. For LP DDR mode, the value of LP DDR clock should be twice that of LP clock. CDU CLKO9 is used for the LP DDR clock.	
		0	8 bit SDR. The default mode. (D0, D1, D2, D3, D4, D5, D6 and D7 data pins used)
		1	4 bit SDR. (D0, D1, D2 and D3 data pins are used)
		2	2 bit SDR. (D0 and D1 data pins are used)
		4	8 bit DDR. (D0, D1, D2, D3, D4, D5, D6 and D7 data pins are used)
		5	4 bit DDR. (D0, D1, D2 and D3 data pins are used)
		6	2 bit DDR. (D0 and D1 data pins are used)
11 (R/W)	ROVFMSK	Receive FIFO Overflow Interrupt Mask.	
		0	Mask Disable Receive FIFO Overflow Interrupt
		1	Unmask Enable Receive FIFO Overflow Interrupt

Table 19-7: LP_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	RRQMSK	Receive Request Interrupt Mask. Link Port Receive Request Mask	
		0	Mask Disable Receive Request interrupt.
		1	Unmask Enable Receive Request interrupt.
8 (R/W)	TRQMSK	Transmit Request Interrupt Mask. Link Port Transmit Request Mask	
		0	Mask Disable Transmit Request interrupt.
		1	Unmask Enable Transmit Request interrupt.
3 (R/W)	TRAN	Transfer Direction. The `LP_CTL.TRAN` bit selects the transfer direction as transmit (if set) or receive (if cleared) for link buffer.	
		0	Receive Direction transfer is receive
		1	Transmit Direction transfer is transmit
0 (R/W)	EN	Enable. The `LP_CTL.EN` enables or disables the link port. When the processor disables the port (`LP_CTL.EN` transitions from high to low), the processor clears the corresponding `LP_STAT` bits.	
		0	Disable Disable linkport
		1	Enable linkport Enable linkport

Clock Divider Value Register

The LP_DIV register selects the divisor for ratio between the internal LP clock (LCLK) and system clock (CLKO8). This programming is applicable only for the transmitter. The receiver can operate at any asynchronous frequency up to the maximum frequency independent of the ratio programmed.

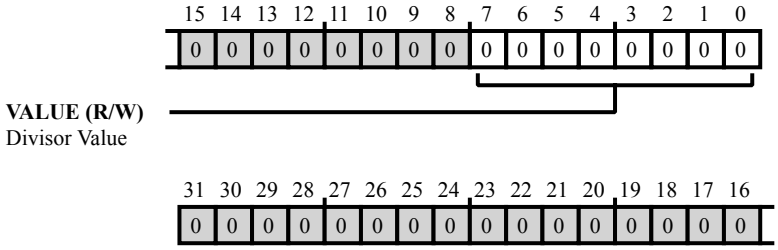


Figure 19-14: LP_DIV Register Diagram

Table 19-8: LP_DIV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	Divisor Value. The LP_DIV.VALUE bits select the clock divider (relating the LP' internally generated clock (LCLK) to the system clock (CLKO8). The LP_DIV.VALUE should be programmed prior to LP enable. For LP_DIV.VALUE = 0, LCLK = CLKO8 For LP_DIV.VALUE = xxxxxxxx, LCLK = CLKO8 / (2 x DIV)

Receive Buffer Register

The LP_RX register buffers the receive data flow through the LP. The receive buffer is a four-location deep FIFO. In the receive buffer, data is transferred to the core or DMA from the receive FIFO where an internal register does the packing. This packing register is not software accessible. For more information on LP buffer features and operations, see the LP functional description.

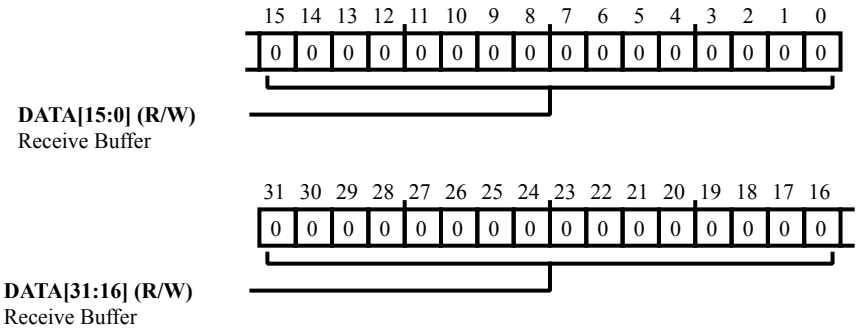


Figure 19-15: LP_RX Register Diagram

Table 19-9: LP_RX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	Receive Buffer.

Status Register

The LP_STAT register provides status information on link port interrupts, FIFO, buses, and receive/transmit requests.

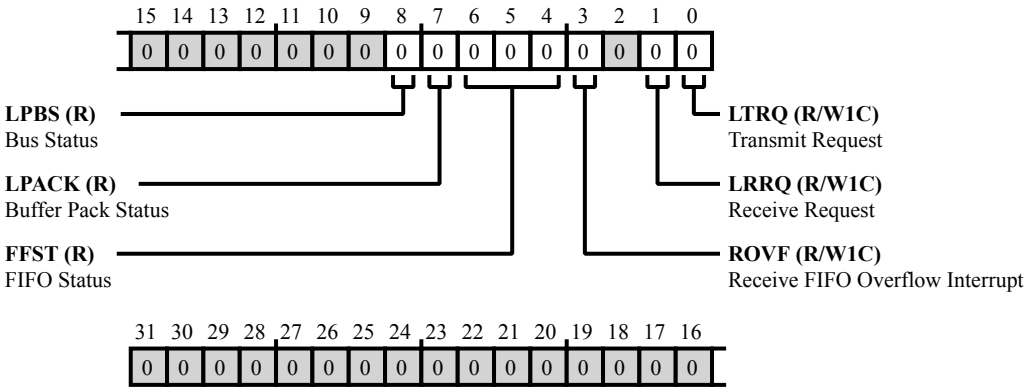


Figure 19-16: LP_STAT Register Diagram

Table 19-10: LP_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	LPBS	Bus Status. The LP_STAT.LPBS bit indicates the LPDAT bus status. LP_STAT.LPBS is kept high if data is being driven by the link port into the LP_D[n] pins.	
		0	Link port bus is idle
		1	Link port bus is busy
7 (R/NW)	LPACK	Buffer Pack Status. The LP_STAT.LPACK bit indicates packing status. In receive mode, 32-bit data is received in 4 blocks of 8-bit data. Then, the data is packed to get a single 32-bit data before loading the FIFO. The LP_STAT.LPACK bit is high during this packing process and goes low after packing. In transmit mode, 32-bit data in the FIFO is unpacked to 4 blocks of 8-bit data before sending. The LP_STAT.LPACK is high during unpacking.	
		0	Packing Complete Packing done
		1	Packing Incomplete Packing is in progress
6:4 (R/NW)	FFST	FIFO Status. The LP_STAT.FFST bits indicate the FIFO status. These bits are cleared when the LP is disabled.	
		0	TX - Empty; RX - Empty Link buffer (TX OR RX) empty

Table 19-10: LP_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		1	TX - Reserved ; RX - Has 1 data word RX has 1 word of data. TX reserved
		2	TX - Reserved; RX - Has 2 data words RX has 2 word of data. TX reserved.
		3	TX - Reserved; RX - Has 3 data words RX has 3 word of data. TX reserved.
		4	TX - One Word; RX -Has 4 data words RX has 4 word of data. TX 1 word of data.
		5	Reserved
		6	TX - FIFO Full; RX - Reserved RX reserved. TX reserved.
		7	Reserved
3 (R/W1C)	ROVF	Receive FIFO Overflow Interrupt. This interrupt is generated when the receiver FIFO overflows. This overflow may happen if the transmitter continues to transmit data even though the receiver has de-asserted the LP_ACK pin.	
1 (R/W1C)	LRRQ	Receive Request. The LP generates this interrupt when the LP_CLK pin of a disabled link port (the receiver) is forced high by another link port (the transmitter). Note that the LRRQ interrupt is not supported in DDR mode.	
0 (R/W1C)	LTRQ	Transmit Request. The LP generates this interrupt when the LP_ACK pin of a disabled link port (the transmitter) is forced high by another link port (the receiver).	

Transmit Buffer Register

The LP_TX register buffers the transmit data flow through the LP. The transmit buffer is two words deep. In the transmit buffer, the input stage of the FIFO is used to accept core data or DMA data from internal memory, and the data is transferred to the link port interface from the output stage of the FIFO. The output stage performs the unpacking in the transmit buffer. The least significant byte is transmitted first. As each word is unpacked and transmitted, the next location in FIFO becomes available and a new DMA request is made if DMA is enabled. If the register becomes empty, the LP asserts the LP_CLK signal. For more information on LP buffer features and operations, see the LP functional description.

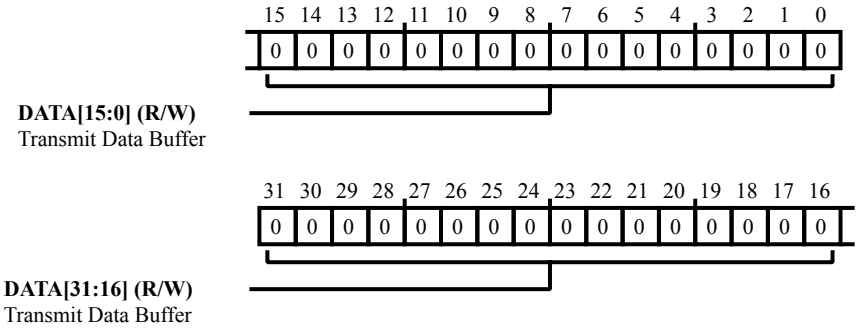


Figure 19-17: LP_TX Register Diagram

Table 19-11: LP_TX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	Transmit Data Buffer.

Shadow Input Transmit Buffer Register

The LP_TXIN_SHDW register contains the same data as the input stage of the transmit buffer. Read of this shadow transmit buffer does not update the LP_STAT register.

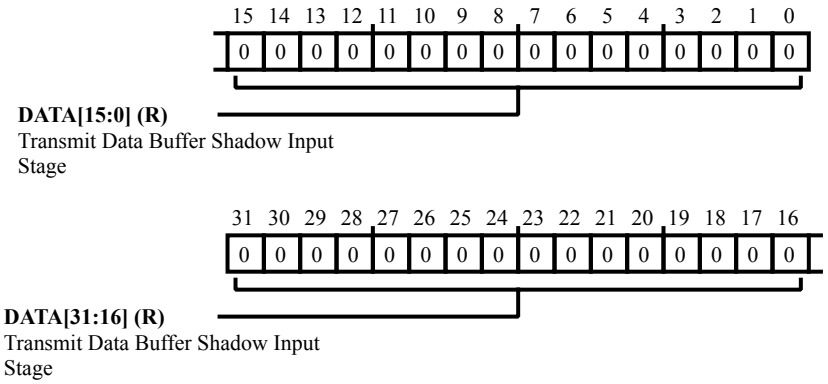


Figure 19-18: LP_TXIN_SHDW Register Diagram

Table 19-12: LP_TXIN_SHDW Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Transmit Data Buffer Shadow Input Stage.

Shadow Output Transmit Buffer Register

The LP_TXOUT_SHDW register contains the same data as the output stage of the transmit buffer. Read of this shadow transmit buffer does not update the LP_STAT register.

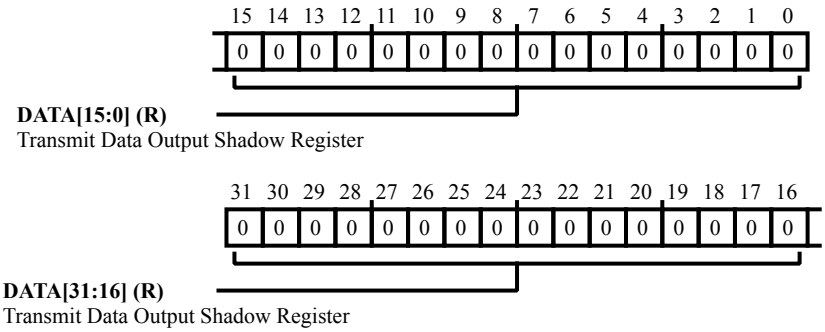


Figure 19-19: LP_TXOUT_SHDW Register Diagram

Table 19-13: LP_TXOUT_SHDW Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Transmit Data Output Shadow Register.

20 Pulse Density Modulation (PDM) Microphone Interface

The pulse density modulation (PDM) microphone interface converts digital PDM microphone data to I^2S/TDM format. The microphone data in I^2S/TDM format is then routed internally to the SPORT/ASRC or externally via the DAI pins. Each DAI unit has one PDM interface.

PDM Features

The PDM interface includes the following features:

- Four channels of PDM audio inputs from digital microphones
- 16×, 32×, or 64× decimation ratio of PDM to pulse code modulation (PCM) audio data
- 24-bit resolution to support high sound pressure level (SPL) microphones
- 126 dB A-weighted SNR
- 4 kHz to 192 kHz output sampling rate
- Bit clock rates of 64×, 128×, 192×, 256×, 384×, or 512× the output sampling rate

  NOTE: Refer to the product data sheet for the maximum BCLK frequency.
- Automatic PDM clock generation
- Target I^2S or TDM output interface
- Supports up to 16 TDM slots
- Configurable TDM slot routing and sizes

PDM Functional Description

The following are the typical connections for the PDM module:

- PDM interface–PDM clock (PDM_CLK0) and PDM data (PDM_DAT0 and PDM_DAT1) are typically routed to DAI pins via the Signal Routing Unit (SRU) which is in turn connected to a PDM data source, for example, PDM microphones.
- Target I^2S/TDM interface –Bit clock (BCLK), frame sync (LRCLK), and serial data (SDATA) signals are typically routed internally to the SPORT using the SRU for the microphone data to reach memory in PCM format.
- Control registers–All PDM related control registers are included in the DAI MMR system.

ADSP-SC59x PDM Register List

Pulse Density Modulation interface block

Table 20-1: ADSP-SC59x PDM Register List

Name	Description
PDM_CTL0	PDM Control Register
PDM_HPF_CTL	High Pass Filter Control Register
PDM_RESET	Software Reset Register
PDM_SP_CTL0	Serial Port Control0 Register
PDM_SP_CTL1	Serial Port Control1 Register

PDM Block Diagram

The *PDM Interface Block Diagram* figure illustrates the interconnection of PDM block in the DAI subsystem.

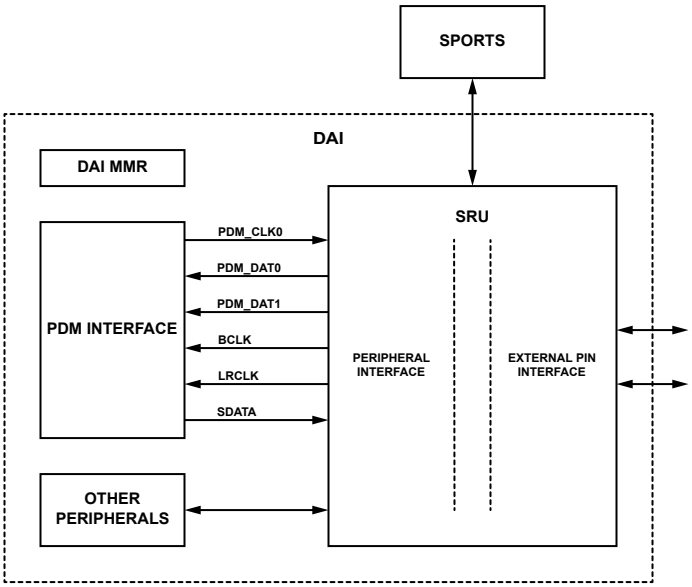


Figure 20-1: PDM Interface Block Diagram

PDM Architectural Concepts

The PDM interface provides up to four channels of decimation from a 1-bit PDM source to a 24-bit PCM audio stream. The down sampling ratio is typically 64 × f_S, where f_S is the PCM output sampling rate. The down sampling ratio can also be set at 32 × f_S or 16 × f_S, to facilitate higher output sampling rates. All channels decimate at the same ratio. The 24-bit down-sampled PCM audio data is output via standard stereo (I^2S, left justified, right justified) or TDM format. The input sources for the PDM interface can be any device that has a target PDM output, such as a digital microphone.

Internally, there are four channels. The *PDM Channel Mapping* table details the mapping of the PDM_DATx input pins to the internal channels. If a channel is not in use, the associated PDM_DATx pin can be disabled to save power.

Table 20-2: PDM Channel Mapping

Input Pin	PDM_CLK Edge	Internal Channel
PDM_DAT0	Falling	0
PDM_DAT0	Rising	1
PDM_DAT1	Falling	2
PDM_DAT1	Rising	3

PDM Initialization and Clocking

After I^2S clocks are applied, there are 16 full frame sync cycles of initialization time before the PDM clocks start. Once the PDM clocks start running, 48 additional frame sync cycles elapse before the PDM data is driven on the SDATA pin.

During normal operation, if the bit clock or frame sync is removed then the PDM clock output stops immediately and the PDM interface enters a lower power state. If the clocks resume, the PDM interface relocks to the bit clock and frame sync signals and adjusts the PDM clock output accordingly. The length of time before the PDM clock outputs resume is 4 frames ± 1 frame to lock to the incoming signal. If the format of the clock signals change, the PDM interface detects this change at the end of the frame and stops the PDM clock outputs. Then, the device reconfigures and resumes sending PDM clocks with no user intervention. Again, the PDM clock outputs resume after 4 frames ±1 frame to lock to the incoming signal.

The PDM interface requires a clock rate that is a minimum of 64× the frame sync sample rate. Bit clock rates of 128×, 192×, 256×, 384×, and 512× the frame sync rate are also supported. The PDM interface automatically detects the ratio between the bit clock and frame sync signals and generates a PDM clock output at 64 × the frame synce rate by default. If lower decimation ratios are selected in the `PDM_CTL0` register, the PDM output clock rate corresponds to the configuration set in the `PDM_CTL0.DEC_RATIO` bit field. The minimum sampling rate is 4 kHz and the maximum sampling rate is 192 kHz. The PDM clock range is 256 kHz to 6.144 MHz. Internally, all processing is done at the PDM clock rate.

SPORT Interface

The PDM interface supports I^{2}S and TDM serial output formats. It supports up to 16 TDM slots, with slot widths of 16, 24, or 32-bits. Internal channels are connected to output slots by configuring the appropriate channel slot bit fields in the PDM_SP_CTL1 register. By default, each channel is routed to the slot with the same number. For example, the reset value of the PDM_SP_CTL1.CH3_SLOT bit field is 0x0011, connecting channel 3 to slot 3.

Each channel has a drive select bit: PDM_SP_CTL1.CH0_DRV, PDM_SP_CTL1.CH1_DRV, PDM_SP_CTL1.CH2_DRV, or PDM_SP_CTL1.CH3_DRV. If the drive select bit is set to one, the channel is driven on the serial port in the appropriate output slot; and if it is cleared to zero, the channel does not drive (tristate high impedance mode). Note that this feature is useful only when the SPORT data is driven out of the processor via a DAI pin buffer and not when the data is routed internally to the SPORT or ASRC. To use this feature, tie the PDMx_SDATA_OE_O signal to the DAIx_PBENxx_I signal of the DAI pin buffer.

It is possible to erroneously configure more than one channel to drive on a single TDM slot. There is no cross-checking of register settings to prevent this configuration, but it will not damage the device. In this case, only the data from the lowest channel number is driven into the slot and the data from the higher channels will not be driven.

The PDM_SP_CTL0.SAI_MODE bit configures the SPORT interface mode. The two modes are stereo and TDM. The primary difference between these two modes is the format of the frame sync clock that is expected and the polarity of the active edge of the clock.

With the PDM_SP_CTL0.SAI_MODE bit and the PDM_SP_CTL0.LRCLK_POL bit set to zero, the serial port is in stereo mode with the clock polarity set to normal. In this mode, there must be only two channels of data sent. The frame starts with the falling edge of the frame sync, with an expected cycle of 50% high and 50% low. Channel 0 sends out data when the clock is low. As soon as the frame sync goes high the data from channel 0 stops and channel 1 begins sending data. Both edges of the frame sync clock are used. If the duty cycle is not 50/50, there may be errors in the resulting data. In this mode of operation, the PDM interface does not expect 32-bit clock transitions for each channel. All bit clock to frame sync ratios are supported.

With the PDM_SP_CTL0.SAI_MODE bit set to one and the PDM_SP_CTL0.LRCLK_POL bit set to zero, the serial port is in TDM mode with the clock polarity set to normal. In this mode, it can transmit anywhere from one channel to as many as four channels spread out across 16 data slots in TDM-16 format.

The PDM interface supports bit clock rates of 64×, 128×, 192×, 256×, 384×, or 512× the output sampling rate. The BCLK frequency however, must not exceed the maximum value specified in the data sheet. These bit clock rates are combined with the three different TDM slot sizes of 16-bit, 24-bit, or 32-bit slots, selected by the PDM_SP_CTL0.CH0_SLOT_WIDTH bit field. Note that as soon as the next frame sync edge is detected, the PDM interface restarts from slot 0 and any unreceived data from the previous frame is lost. This is how to achieve unusual TDM formats like TDM-5 or TDM-10. In addition, there is only support for TDM-16 or less for placing data into a TDM slot. Data cannot be placed into slots above 16, however it is possible to configure the PDM interface to tristate all unused TDM slots, which includes all the slots above the first 16 slots for modes that have more than 16 slots.

In TDM mode, the frame sync is expected to be a positive going pulse that is at least one-bit clock period wide. A falling edge is not necessary, it is only important that the signal is low long enough to meet the timing specification for a read before transitioning from low to high. The frame starts with the rising edge of this pulse. The data is clocked out according to the slot width and using the data format specified in the PDM_SP_CTL0 register. The PDM interface continues to send data until all active channels are sent and then the device waits for the next frame sync clock edge to start sending the next set of frame samples.

In TDM-8 format, when the PDM interface is set to output channel 0 through channel 3 into slot 0 to slot 3, the PDM interface can tristate for the remainder of the frame. This allows another PDM to output four channels onto slot 4 through slot 7. These slots do not have to be consecutive, and the two devices can be configured to interleave their respective data. The serial port can be set up to drive only when there is data to drive into a data slot. If all eight channels are not in use, configuring the drive select bit in the PDM_SP_CTL1 for the unused channels will assign those slots to drive or tristate in the TDM data stream. Note that this feature is useful only when the SPORT data is driven out of the chip via a DAI pin buffer and not when the data is routed internally to the SPORT or ASRC. To use this feature, tie the PDMx_SDATA_OE_O signal to the DAIx_PBENxx_I signal of the DAI pin buffer.

To invert the IRCLK polarity, set the PDM_SP_CTL0.LRCLK_POL bit to one. With inverted clock polarity in stereo mode, channel 0 is sent out when the frame sync is high, so the start of the frame is a low to high transition. With inverted clock polarity in TDM mode, the frame sync pulse is negative, so the frame starts with the high to low transition.

Configure the PDM_SP_CTL0.DATA_FORMAT bit field to align data within the 32-bit data slot. There is support for left justified mode, delayed by one-bit clock period; and right justified modes for 24-bit, 20-bit, and 16-bit data word sizes.

SPORT Timing

The *Stereo Mode* figure is a timing diagram for I²S mode with 24-bit data.

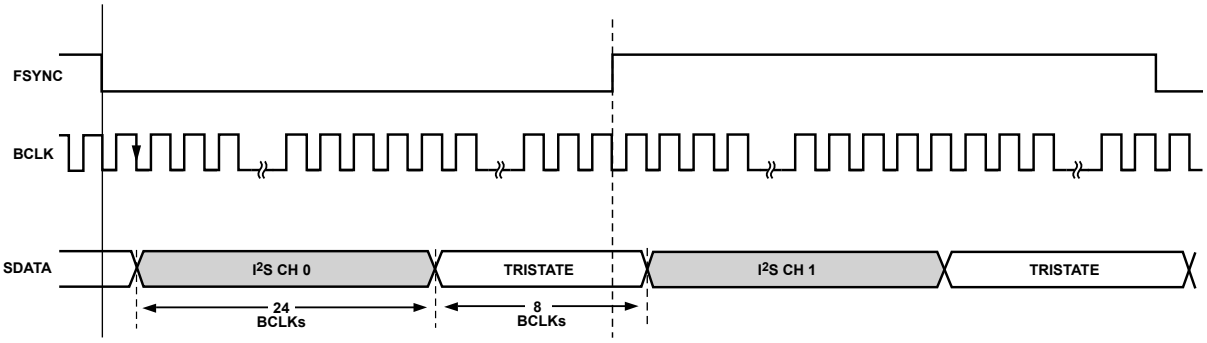


Figure 20-2: Stereo Mode

The *TDM4 Mode Four Channel* figure is a timing diagram for TDM4 mode with default channel assignments, 24-bit data, and 32-bit slots, and normal polarity clocks.

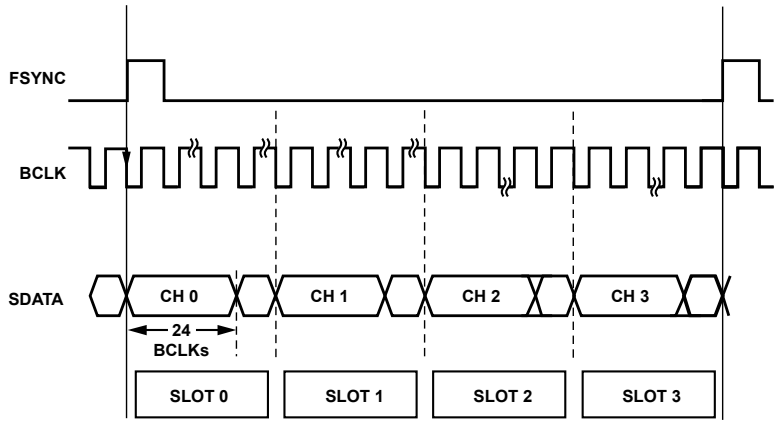


Figure 20-3: TDM4 Four Channel

The ***TDM4 Mode Two Channel*** figure is a timing diagram that shows TDM4 mode with only two channels enabled, 24-bit data delayed by zero BCLK, 32-bit slots, and normal polarity clocks.

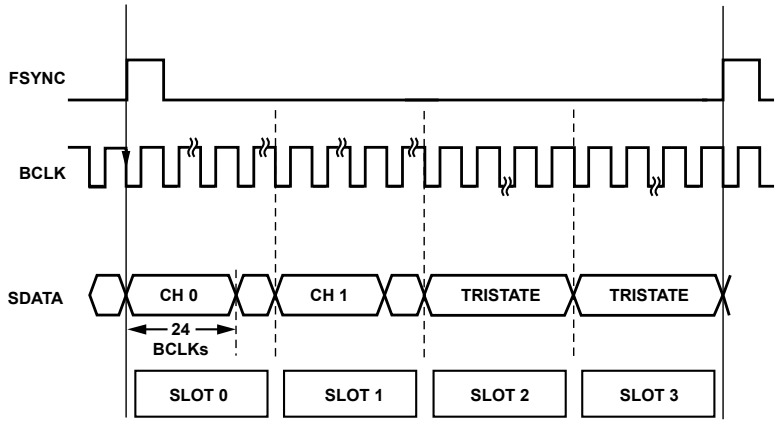


Figure 20-4: TDM4 Two Channel

The ***TDM8 Mode Four Channel*** figure is a timing diagram that shows TDM8 mode with all the four channels enabled, 24-bit data delayed by zero BCLK, 32-bit slots, and normal polarity clocks.

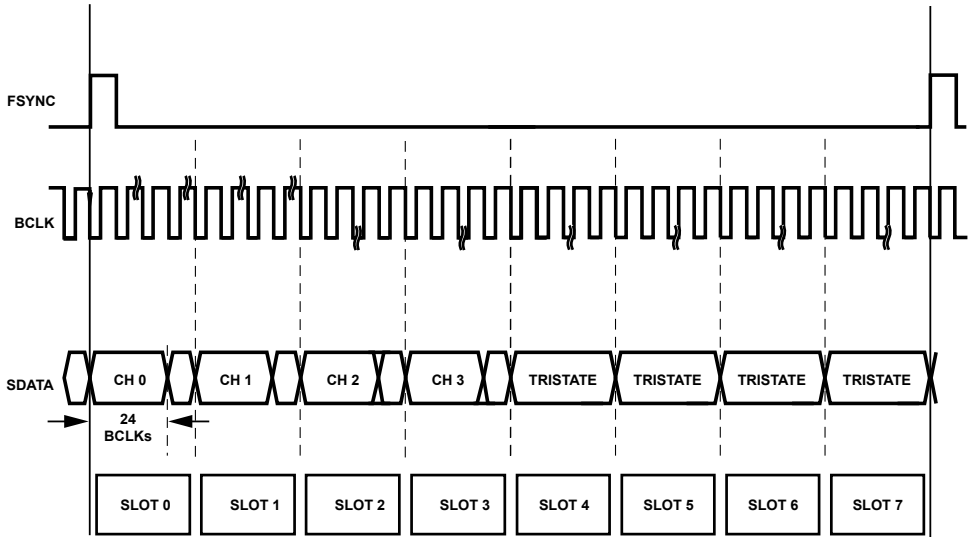


Figure 20-5: TDM8 Four Channels

The *TDM2 Mode Two Channel* figure is a timing diagram that shows TDM2 mode with two channels enabled, 24-bit data delayed by one BCLK, 32-bit slots, and normal polarity clocks.

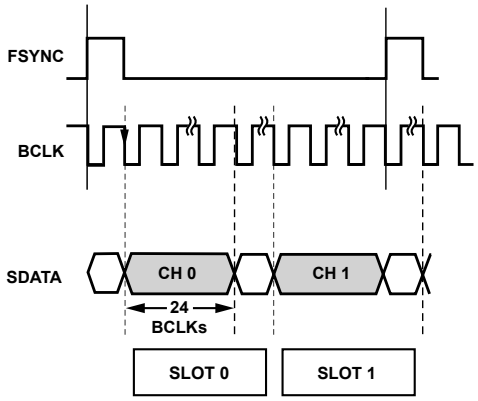


Figure 20-6: TDM2 Two Channels

High Pass Filter

There is a first order high-pass filter in the signal path. It is disabled by default and can be enabled by setting the `PDM_HPF_CTL.EN` bit. To adjust the cutoff frequency, configure the `PDM_HPF_CTL.FC` bit field. The settings are relative to the output sampling rate.

The *HPF Cutoff Frequency Selection* table shows the setting and the cutoff frequency for common sampling rates.

Table 20-3: HPF Cutoff Frequency Selections

PDM_HPF_CTL.HPF_FC	Multiplication Factor	48 kHz Sampling Rate Cutoff Frequency (Hz)	32 kHz Sampling Rate Cutoff Frequency (Hz)
0101	0.00505	242.4	161.6
1001	0.00251	120.48	80.32

Table 20-3: HPF Cutoff Frequency Selections (Continued)

PDM_HPF_CTL.HPF_FC	Multiplication Factor	48 kHz Sampling Rate Cutoff Frequency (Hz)	32 kHz Sampling Rate Cutoff Frequency (Hz)
0111	0.00125	60	40
1000	0.000623	29.904	19.936
1001	0.000311	14.928	9.952
1010	0.000155	7.44	4.96
1011	0.0000777	3.7296	2.4864
1100	0.0000389	1.8672	1.2448
1101	0.0000194	0.9312	0.6208
1110	0.00000971	0.46608	0.31072
1111	0.00000468	0.23328	0.15552

PDM Programming Model

The following section provides information on configuring the PDM interface.

Configuring the PDM Interface

Use the following steps to configure the PDM interface:

Optional high pass filter configuration:

- Configure the `PDM_HPF_CTL` register to enable the high pass filter using the `PDM_HPF_CTL.EN` bit and set the cutoff frequency by programming the `PDM_HPF_CTL.FC` bit field.

1. Configure `PDM_SP_CTL0` register to set the following parameters:
    a. SPORT mode (Stereo/TDM) with the `PDM_SP_CTL0.SAI_MODE` bit
    b. SPORT data format (BCLK delay) with `PDM_SP_CTL0.DATA_FORMAT` bit field
    c. SPORT slot width (16, 24, or 32 bits) with `PDM_SP_CTL0.CH0_SLOT_WIDTH` bit field – applicable only for the TDM mode
    d. BCLK capture edge (rising/falling) with the `PDM_SP_CTL0.BCLK_POL` bit
    e. LRCLK polarity (normal/inverted) with the `PDM_SP_CTL0.LRCLK_POL` bit
2. Configure the `PDM_SP_CTL1` register to set the following parameters:
    a. PDM to SPORT channel slot mapping with serial port channel slot bit fields (CHx_SLOT)
    b. To select if the PDM channel will get driven to the associated SPORT channel slot with the serial port channel drive select bit (CHx_DRV).

3. Configure the PDM_CTL0 register to configure the decimation ratio with the PDM_CTL0.DEC_RATIO bit field and finally enable the PDM clock with the PDM_CTL0.CLK0_EN bit.

Software Reset

The PDM_RESET.SOFT bit is set to perform a soft reset of the PDM block. Setting this bit resets internal bit clock counters and audio data, but it does not reset the PDM register settings. This bit is self-clearing.

ADSP-SC59x PDM Register Descriptions

Pulse Density Modulation interface block (PDM) contains the following registers.

Table 20-4: ADSP-SC59x PDM Register List

Name	Description
PDM_CTL0	PDM Control Register
PDM_HPF_CTL	High Pass Filter Control Register
PDM_RESET	Software Reset Register
PDM_SP_CTL0	Serial Port Control0 Register
PDM_SP_CTL1	Serial Port Control1 Register

PDM Control Register

The PDM_CTL0 register (read/write) controls the channel activation, clock enable and Decimation ratio.

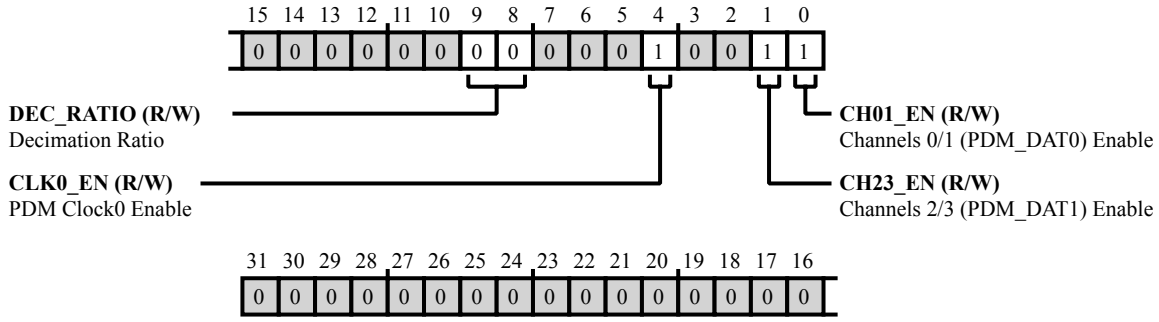


Figure 20-7: PDM_CTL0 Register Diagram

Table 20-5: PDM_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	DEC_RATIO	Decimation Ratio.	
		0	64 times decimation
		1	32 times decimation
		2	16 times decimation
4 (R/W)	CLK0_EN	PDM Clock0 Enable.	
		0	PDM_CLK0 output disabled
		1	PDM_CLK0 output enabled
1 (R/W)	CH23_EN	Channels 2/3 (PDM_DAT1) Enable.	
		0	PDM channels 2/3 disabled
		1	PDM channels 2/3 enabled
0 (R/W)	CH01_EN	Channels 0/1 (PDM_DAT0) Enable.	
		0	PDM channels 0/1 disabled
		1	PDM channels 0/1 enabled

High Pass Filter Control Register

The PDM_HPF_CTL register (read/write) controls the High Pass Filter.

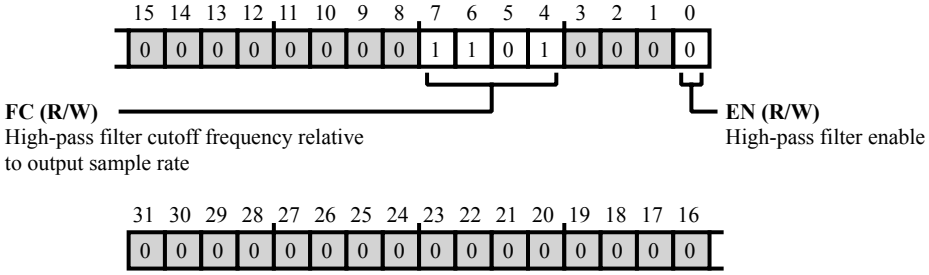


Figure 20-8: PDM_HPF_CTL Register Diagram

Table 20-6: PDM_HPF_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/W)	FC	High-pass filter cutoff frequency relative to output sample rate.	
		0	Reserved
		1	Reserved
		2	Reserved
		3	Reserved
		4	Reserved
		5	0.00505
		6	0.00251
		7	0.00125
		8	0.000623
		9	0.000311
		10	0.000155
		11	0.0000777
		12	0.0000389
		13	0.0000194
		14	0.00000971
		15	0.00000486
0 (R/W)	EN	High-pass filter enable.	
		0	High Pass Filter Off
		1	High Pass Filter On

Software Reset Register

The PDM_RESET register (read/write) controls the software reset operations.

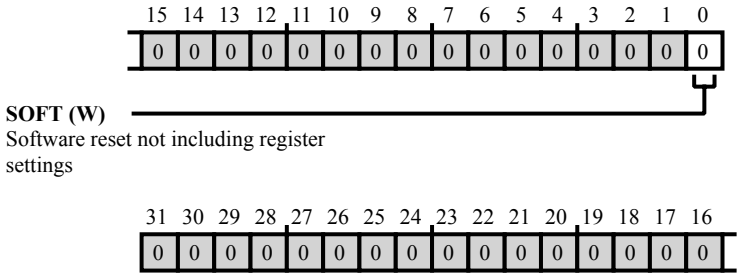


Figure 20-9: PDM_RESET Register Diagram

Table 20-7: PDM_RESET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (RX/W)	SOFT	Software reset not including register settings.	
		0	N/A
		1	Write once to soft reset

Serial Port Control0 Register

The PDM_SP_CTL0 register (read/write) controls serial port mode, data format and slot width.

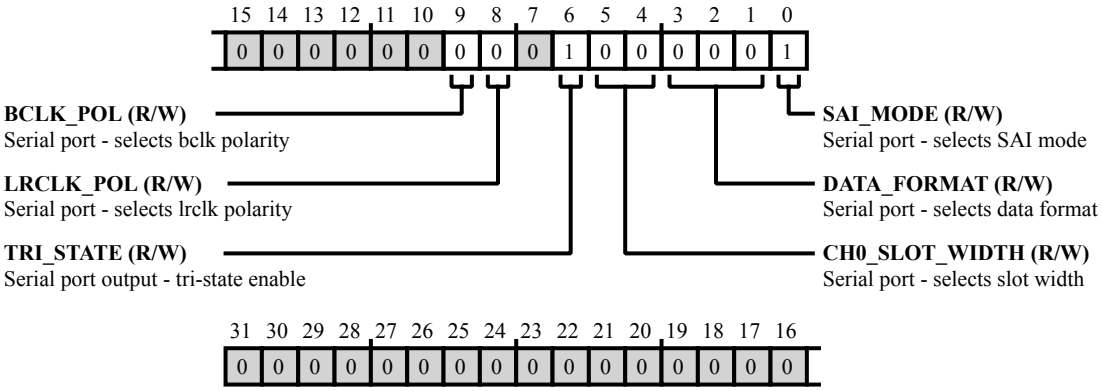


Figure 20-10: PDM_SP_CTL0 Register Diagram

Table 20-8: PDM_SP_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	BCLK_POL	Serial port - selects bclk polarity.	
		0	Capture on rising edge
		1	Capture on falling egde
8 (R/W)	LRCLK_POL	Serial port - selects lrclk polarity.	
		0	Normal Polarity
		1	Inverted Polarity
6 (R/W)	TRI_STATE	Serial port output - tri-state enable.	
		0	tri-state disabled
		1	tri-state enabled
5:4 (R/W)	CH0_SLOT_WIDTH	Serial port - selects slot width.	
		0	32 BCLK's per slot
		1	16 BCLK's per slot
		2	24 BCLK's per slot
3:1 (R/W)	DATA_FORMAT	Serial port - selects data format.	
		0	Typical I2S mode, delay by 1
		1	Left Justified,delay by 0
		2	Delay by 8
		3	Delay by 12

Table 20-8: PDM_SP_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4	Delay by 16
0 (R/W)	SAI_MODE	Serial port - selects SAI mode.	
		0	STEREO (I2S, LJ or RJ)
		1	TDM

Serial Port Control1 Register

The PDM_SP_CTL1 register (read/write) controls the mapping of channel 0/1/2/3 to TDM slots.

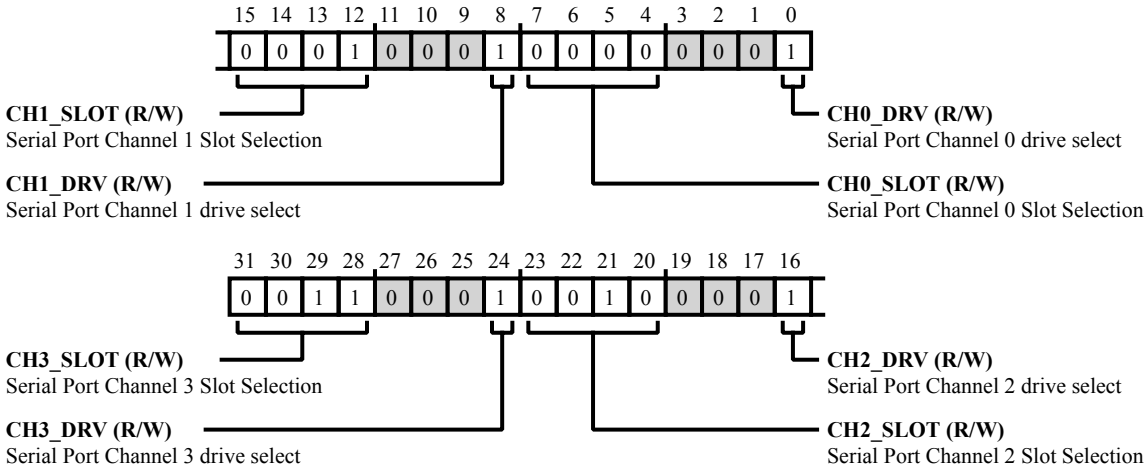


Figure 20-11: PDM_SP_CTL1 Register Diagram

Table 20-9: PDM_SP_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/W)	CH3_SLOT	Serial Port Channel 3 Slot Selection.	
		0	Map channel to TDM slot 0/I2S Left
		1	Map channel to TDM slot 1/I2S Right
		2	Map channel to TDM slot 2
		3	Map channel to TDM slot 3
		4	Map channel to TDM slot 4
		5	Map channel to TDM slot 5
		6	Map channel to TDM slot 6
		7	Map channel to TDM slot 7
		8	Map channel to TDM slot 8
		9	Map channel to TDM slot 9
		10	Map channel to TDM slot 10
		11	Map channel to TDM slot 11
		12	Map channel to TDM slot 12
		13	Map channel to TDM slot 13
		14	Map channel to TDM slot 14

Table 20-9: PDM_SP_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		15	Map channel to TDM slot 15
24 (R/W)	CH3_DRV	Serial Port Channel 3 drive select. This determines whether the associated channel gets driven in its assigned slot or not.	
		0	Channel will not be driven on serial port
		1	Channel will be driven on serial port in selected output slot.
23:20 (R/W)	CH2_SLOT	Serial Port Channel 2 Slot Selection.	
		0	Map channel to TDM slot 0/I2S Left
		1	Map channel to TDM slot 1/I2S Right
		2	Map channel to TDM slot 2
		3	Map channel to TDM slot 3
		4	Map channel to TDM slot 4
		5	Map channel to TDM slot 5
		6	Map channel to TDM slot 6
		7	Map channel to TDM slot 7
		8	Map channel to TDM slot 8
		9	Map channel to TDM slot 9
		10	Map channel to TDM slot 10
		11	Map channel to TDM slot 11
		12	Map channel to TDM slot 12
		13	Map channel to TDM slot 13
		14	Map channel to TDM slot 14
		15	Map channel to TDM slot 15
16 (R/W)	CH2_DRV	Serial Port Channel 2 drive select. This determines whether the associated channel gets driven in its assigned slot or not.	
		0	Channel will not be driven on serial port
		1	Channel will be driven on serial port in selected output slot.
15:12 (R/W)	CH1_SLOT	Serial Port Channel 1 Slot Selection.	
		0	Map channel to TDM slot 0/I2S Left
		1	Map channel to TDM slot 1/I2S Right

Table 20-9: PDM_SP_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	Map channel to TDM slot 2
		3	Map channel to TDM slot 3
		4	Map channel to TDM slot 4
		5	Map channel to TDM slot 5
		6	Map channel to TDM slot 6
		7	Map channel to TDM slot 7
		8	Map channel to TDM slot 8
		9	Map channel to TDM slot 9
		10	Map channel to TDM slot 10
		11	Map channel to TDM slot 11
		12	Map channel to TDM slot 12
		13	Map channel to TDM slot 13
		14	Map channel to TDM slot 14
		15	Map channel to TDM slot 15
8 (R/W)	CH1_DRV	Serial Port Channel 1 drive select. This determines whether the associated channel gets driven in its assigned slot or not.	
		0	Channel will not be driven on serial port
		1	Channel will be driven on serial port in selected output slot.
7:4 (R/W)	CH0_SLOT	Serial Port Channel 0 Slot Selection.	
		0	Map channel to TDM slot 0/I2S Left
		1	Map channel to TDM slot 1/I2S Right
		2	Map channel to TDM slot 2
		3	Map channel to TDM slot 3
		4	Map channel to TDM slot 4
		5	Map channel to TDM slot 5
		6	Map channel to TDM slot 6
		7	Map channel to TDM slot 7
		8	Map channel to TDM slot 8
		9	Map channel to TDM slot 9
		10	Map channel to TDM slot 10

Table 20-9: PDM_SP_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		11	Map channel to TDM slot 11
		12	Map channel to TDM slot 12
		13	Map channel to TDM slot 13
		14	Map channel to TDM slot 14
		15	Map channel to TDM slot 15
0 (R/W)	CH0_DRV	Serial Port Channel 0 drive select. This determines whether the associated channel gets driven in its assigned slot or not.	
		0	Channel will not be driven on serial port
		1	Channel will be driven on serial port in selected output slot.

21 Universal Serial Bus Controller (USBC)

The USB OTG controller provides a low-cost connectivity solution for consumer mobile devices such as cell phones, digital still cameras, and MP3 players. It allows these devices to transfer data using a point-to-point USB connection without the need for a personal computer host.

The USB controller can operate in a traditional USB device-only mode as well as the host mode presented in the On-The-Go (OTG) supplement to the USB 2.0 specification.

USBC Features

The USB OTG controller provides the following features:

- Complies with the On-The-Go Supplement to the USB 2.0 Specification (revision 2.0)
- Supports high speed (HS, 480 Mbps), full speed (FS, 12 Mbps) and low speed (LS, 1.5 Mbps) rates
- Provides scatter/gather DMA mode access support on the application side
- Provides ULPI interface support on the MAC PHY
- Supports up to 12 bidirectional endpoints including control endpoint 0
- Supports Session Request Protocol (SRP)
- Supports Host Negotiation Protocol (HNP)
- Supports up to 16 host channels. In host mode, when the number of device endpoints to be supported is more than the number of host channels, software can reprogram the channels to support up to 127 devices, each having 32 endpoints (IN + OUT), for a maximum of 4,064 endpoints.
- Includes automatic ping capabilities
- Supports the keep-alive in low speed mode and SOFs in high and full speed modes

The application interface uses the bus to support the following:

- Bus completer interface for accessing Control and Status Registers (CSRs). (Access to FIFO and queues is not supported).
    - Bus requester interface for data FIFO access

  - Supports 32-bit data on the AHB bus
  - Supports little endian mode
  - Supports INCR4, INCR8, INCR16, INCR, and single transfers on the bus completer interface
  - Supports error and okay bus responses on the bus requester interface. The bus completer interface supports only an okay response. An error response is generated on the bus completer interface when the transfer size (HSIZE) is not equal to 32 bits.
  - Software-selectable bus burst type on bus requester interface in DMA mode
- When INCR4 is chosen, the core uses INCR/INCR4 or single
- When INCR8 is chosen, the core normally uses INCR8. However, at the beginning and at the end of a transfer, it can use INCR or single, depending on the size of the transfer.
- When INCR16 is chosen, the core normally uses INCR16. However, at the beginning and at the end of a transfer, it can use INCR or single, depending on the size of the transfer.
- Handles the fixed burst address alignment. For example, INCR16 is used only when lower addresses [5:0] are all 0.
- Generates bus busy cycles on the bus requester interface
- Supports the 1KB boundary breakup

The MAC PHY interface supports the ULPI interface and 8-bit SDR or 4-bit DDR

The system memory architecture has the following features:

- Supports internal DMA mode for the system memory architecture
- Provides descriptor-based scatter/gather DMA operation

Supports Non-DWORD alignment:

- In host mode, supports scatter/gather DMA mode, IN and OUT transfers - Non-DWORD alignment of buffer addresses
- In device mode, supports scatter/gather DMA mode, IN and OUT transfers - Non-DWORD alignment of buffer addresses

NOTE: Non-DWORD alignment support is available only for buffer addresses and not for descriptors.

The internal memory features:

- Support for a dedicated transmit FIFO for each of the device IN endpoints. Each FIFO can hold multiple packets.
- An interface for remote memory support used to signal the core of a DMA write complete event on the system
- Support for packet-based, dynamic FIFO memory allocation for endpoints for small FIFOs, and flexible, efficient use of RAM.

- Support to change the FIFO memory size of an endpoint during transfers
- Support for endpoint FIFO sizes that are not powers of 2, to allow the use of contiguous memory locations.
- Optional support for transmit and receive thresholding in DMA mode (in device mode). Thresholding and threshold length are selectable through global registers.

USBC Functional Description

The USB OTG controller includes the following interfaces:

- *Bus completer and data RAM* – the bus completer provides the microcontroller with read and write access to the CSRs of the core. The data RAM interface connects to single port RAM (SPRAM) for transaction data storage which is 4096 deep.
- *USB 2.0 PHYs and serial transceiver* – The USB OTG core supports the ULPI PHY interface (Revision 1.1). The ULPI PHY is unidirectional, with 8-bit SDR (single data rate).
- Internal DMA controller (bus requester) – enables the core to act as a requester on the bus to transfer data to and from the bus.
- PHY control interfaces to ULPI PHY registers.
- Descriptor-based scatter/gather DMA controller for device and host mode. (Descriptor-based congruent-sequential DMA is not supported).
- The core supports the scatter/gather DMA operation in both device and host mode.

NOTE: Hubs (split transfers) are not supported in the host scatter/gather DMA mode of operation. Split transfers are supported only in host buffer DMA (internal DMA) mode of operation.

ADSP-SC59x USBC Register List

The Universal Serial Bus Controller (USBC) is a multi-point high-speed dual-role USB 2.0-compliant controller. The USB controller can operate in a traditional USB peripheral-only mode as well as the host mode presented in the On- The-Go (OTG) supplement to the USB 2.0 Specification. A set of registers governs USBC operations. For more information on USBC functionality, see the USB controller register descriptions.

Table 21-1: ADSP-SC59x USBC Register List

Name	Description
USBC_ISTAT_D	Device All Endpoint Interrupt Status Register
USBC_IMSK_D	Device All Endpoint Interrupt Mask Register
USBC_CFG_D	Device Configuration Register
USBC_CTL_D	Device Control Register
USBC_CTL_IEP0_D	Device Control IN Endpoint 0 Control Register

Table 21-1: ADSP-SC59x USBC Register List (Continued)

Name	Description
USBC_CTL_IEP[n]_D	Device Control IN Endpoint n Control Register
USBC_DMA_ADDR_IEP0_D	Device Control IN Endpoint 0 DMA Address Register
USBC_DMA_BADDR_IEP0_D	Device IN Endpoint 0 Buffer Address Register
USBC_DMA_BADDR_IEP[n]_D	Device Control IN Endpoint n DMA Buffer Address Register
USBC_DMA_ADDR_IEP[n]_D	Device Control IN Endpoint n DMA Address Register
USBC_IMSK_IEP_FEMPT_D	Device IN Endpoint FIFO Empty Interrupt Mask Register
USBC_ISTAT_IEP0_D	Device Control IN Endpoint 0 Interrupt Control Register
USBC_ISTAT_IEP[n]_D	Device Control IN Endpoint n Interrupt Control Register
USBC_IMASK_IEP_D	Device IN Endpoint Common Interrupt Mask Register
USBC_TSIZ_IEP0_D	Device Control IN Endpoint 0 Transfer Size Register
USBC_TSIZ_IEP[n]_D	Device Control IN Endpoint n Transfer Size Register
USBC_TXFIFOSZ1_IEP_D	Device IN Endpoint 1 Transmit FIFO Size Register
USBC_TXFIFOSZ2_IEP_D	Device IN Endpoint 2 Transmit FIFO Size Register
USBC_TXFIFOSZ3_IEP_D	Device IN Endpoint 3 Transmit FIFO Size Register
USBC_CTL_OEP0_D	Device OUT Endpoint 0 Control Register
USBC_CTL_OEP[n]_D	Device OUT Endpoint n Control Register
USBC_DMA_ADDR_OEP0_D	Device OUT Endpoint 0 DMA Address Register
USBC_DMA_BADDR_OEP0_D	Device OUT Endpoint 0 Buffer Address Register
USBC_DMA_BADDR_OEP[n]_D	Device OUT Endpoint n Buffer Address Register
USBC_DMA_ADDR_OEP[n]_D	Device OUT Endpoint n DMA Address Register
USBC_ISTAT_OEP0_D	Device OUT Endpoint 0 Interrupt Register
USBC_ISTAT_OEP[n]_D	Device OUT Endpoint n Interrupt Register
USBC_IMASK_OEP_D	Device OUT Endpoint Common Interrupt Mask Register
USBC_TSIZ_OEP0_D	Device OUT Endpoint 0 Transfer Size Register
USBC_TSIZ_OEP[n]_D	Device OUT Endpoint n Transfer Size Register
USBC_STAT_D	Device Status Register
USBC_THR_CTL_D	Device Threshold Control Register
USBC_TXFSTAT_IEP0_D	Device Control IN Endpoint Transmit FIFO Status Register
USBC_TXFSTAT_IEP[n]_D	Device Control IN Endpoint Transmit FIFO Status Register
USBC_VBUSDIS_D	Device VBUS Discharge Time Register
USBC_VBUSPULSE_D	Device VBUS Pulsing Time Register

Table 21-1: ADSP-SC59x USBC Register List (Continued)

Name	Description
USBC_AHB_CFG	Bus Configuration Register
USBC_DFIFO_CFG	DFIFO Configuration Register
USBC_HWCFG1	User Hardware Configuration 1 Register
USBC_HWCFG2	User Hardware Configuration 2 Register
USBC_HWCFG3	User Hardware Configuration 3 Register
USBC_HWCFG4	User Hardware Configuration 4 Register
USBC_IMSK	Interrupt Mask Register
USBC_ISTAT	Interrupt Status Register
USBC_TXFIFOSZ_NP	Non-periodic Transmit FIFO Size Register
USBC_TXFIFO_STAT_NP	Non-periodic Transmit FIFO/Queue Status Register
USBC_OTG_CTL	OTG Control and Status Register
USBC_OTG_IRQ	OTG Interrupt Register
USBC_PHYIF_CTL	PHY Interface Control Register
USBC_RST_CTL	Reset Register
USBC_RXFIFOSZ	Receive FIFO Size Register
USBC_RXDATA_STAT	Receive Status Read/Pop Register
USBC_RXDBG_STAT	Receive Status Debug Read Register
USBC_MODID	Module ID Register
USBC_CFG	USB Configuration Register
USBC_ISTAT_H	Host All Channels Interrupt Register
USBC_IMSK_H	Host All Channels Interrupt Mask Register
USBC_CHAR[n]_H	Host Channel n Characteristics Register
USBC_DMA_BADDR[n]_H	Host Channel n DMA Buffer Address Register
USBC_DMA_ADDR[n]_H	Host Channel n DMA Address Register
USBC_CFG_H	Host Configuration Register
USBC_IMSK[n]_H	Host Channel n Interrupt Mask Register
USBC_ISTAT[n]_H	Host Channel n Interrupt Status Register
USBC_SPLT_CTL[n]_H	Host Channel n Split Control Register
USBC_TSIZ[n]_H	Host Channel n Transfer Size Register
USBC_FIR_H	Host Frame Interval Register
USBC_FL_BADDR_H	Host Frame List Base Address Register

Table 21-1: ADSP-SC59x USBC Register List (Continued)

Name	Description
USBC_FNUM_H	Host Frame Number/Frame Time Remaining Register
USBC_PORT_CTL_H	Host Port Control and Status Register
USBC_TXFIFOSZ_PER_H	Host Periodic Transmit FIFO Size Register
USBC_PWR_CTL	Power and Clock Gating Control Register

ADSP-SC59x USBULPI Interrupt List

Table 21-2: ADSP-SC59x USBULPI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
241	USBC0_INT	USBC0 USB0 Interrupt	None	

USBC Block Diagram

The ***USBC Block Diagram*** shows the USB OTG controller in a typical system.

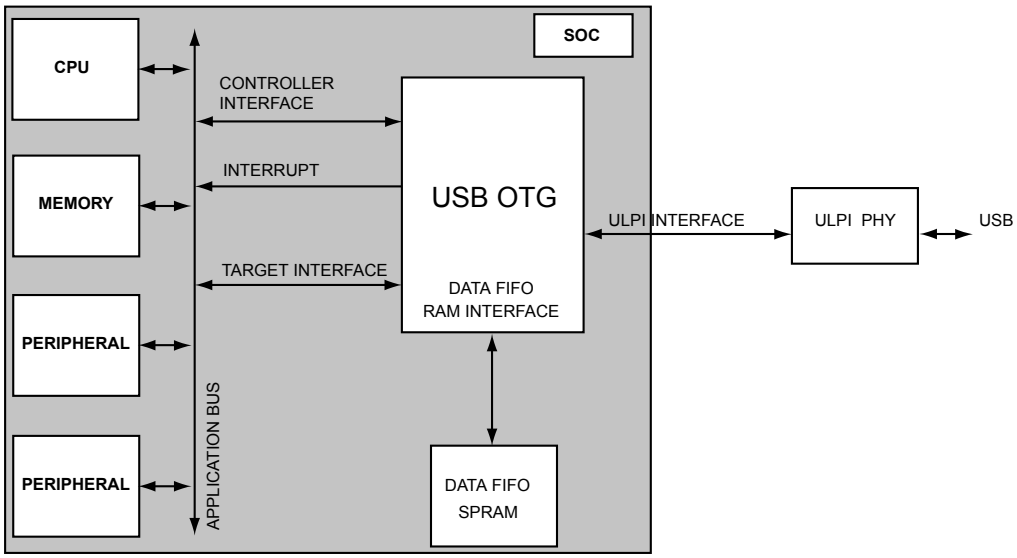


Figure 21-1: USBC Block Diagram

USBC Definitions

A list of common USB terms and their definitions as used in this specification and based on the USB controller follows:

'A' Device

The USB device with a mini-A plug inserted into its receptacle. The A device always supplies power to VBUS.

'B' Device

The USB device with a standard-B or mini-B plug inserted into its receptacle. The B device starts a session as the peripheral.

Bidirectional endpoint

An endpoint that can concurrently support both receive and transfer packets.

Control endpoint

An endpoint used only for transfer of USB control packets for setup and configuration. In all USB devices, the control endpoint refers to the bidirectional endpoint 0.

Dual role device

A USB device that can operate either as the USB host in an OTG session or as a traditional USB peripheral.

Endpoint

A single physical communication channel for USB, implemented as a FIFO and control logic for that endpoint. Each endpoint has an associated USB transfer type, maximum packet size, bandwidth requirement, endpoint number, and (often) a fixed transfer direction.

Frame

A regular, fixed 1 ms (LS and FS) / 125 us timeslot that can contain several transactions. The transfer type determines the permissible transactions for a given endpoint.

HNP

Host negotiation protocol. Part of the USB OTG supplement that allows the host function to be transferred between two connected dual role devices.

Packet

The lowest level of data exchange on USB. The transfer type and buffer size of the USB peripheral determine the size.

PHY

The PHY is a transceiver circuit that implements the physical layer of USB. For full speed USB OTG, this circuit includes line drivers and receivers, pull-up, or pull-down resistors as well as device ID and VBUS level detection.

Session

A period during which USB transfers take place within an OTG connection. The A device (drives VBUS) or B device (initiates SRP) can initiate this period. VBUS is powered during a session.

SRP

Session request protocol. Part of the USB OTG supplement that allows a B device to turn on VBUS and initiate a USB session.

Transaction

Collection of one or more packets in sequence

Transfer

Collection of one or more transactions in sequence

Unidirectional endpoint

Endpoint with its direction fixed in a single direction (for example, it can only receive packets from the USB) in both host and peripheral modes.

Transmit and Receive FIFOs

A transmit and receive FIFO interface on the USB OTG core is used to move data in and out of the data FIFO RAM. These movements are classified as periodic (for isochronous and interrupt transfers) or non-periodic (for bulk and control transfers). The interface has

- A single receive FIFO for all host IN and device OUT transfers
- Dedicated Tx FIFO for each IN endpoint for periodic /non-periodic device IN transfers

DDR Mode

The ULPI PHY can be unidirectional or bidirectional, either with 8-bit SDR or 4-bit DDR mode. In the USBC module, the selection of SDR and DDR modes is software configurable. In SDR mode, data is transferred on D0-D7 of the ULPI bus. In DDR mode, data is transferred only on D0-D3 (4-bit wide) of ULPI bus. In SDR mode, DATA[3:0] pins are synchronized to the rising edge of the clock. In DDR mode, DATA[3:0] are synchronized to both the rising and falling edges of the clock; DATA[7:4] pins can be left unconnected. DDR mode helps to reduce the board traces.

All 4-bit DDR mode timing and operations adhere to the ULPI specification for 4-bit data mode.

To configure the USB controller into DDR mode, software must set (=1) the USBC_CFG.DDRSEL bit before starting any ULPI transmission. All USB/ULPI features, operations, and register settings remain the same for SDR and DDR modes.

NOTE: Ensure that the USB PHY is also configured to DDR mode before configuring the USB controller to DDR mode.

Restrictions

The USBC modules has some restrictions for response time and thresholding.

Response Time Restriction

For the controller as a device, there is a worst-case response time for any tokens that follow an isochronous OUT transaction. This worst-case response time depends on the bus clock frequency. The controller registers are in the bus domain; the controller does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT. For an isochronous transaction, there is no handshake, and the next token can come sooner than expected. When the bus clock is the same as the PHY clock, this worst-case value is seven PHY clocks. When the bus clock is faster, this value is smaller.

If this worst-case condition occurs:

- For bulk/interrupt tokens, the controller responds with a NAK and drops the isochronous and setup tokens. The host interprets this event as a time-out condition for SETUP and retries the SETUP packet.
- For isochronous transfers, the controller sets the USBC_ISTAT.INCOMPLP interrupt to inform the application that isochronous IN/OUT packets were dropped.

At the system level, this limitation can reduce the performance of the device controller because the controller drops or sends NAK to the token which is sent by the host too close to end of the preceding isochronous OUT token. For bulk, interrupt, and control transactions, the system recovers as the host retires the token; the transactions will eventually complete. For isochronous transactions, the host proceeds to the next transaction scheduled for the next interval.

A significant performance degradation occurs in the following cases:

- The device is connected to a host which is capable of consistently sending a token within 88-96 bit times, that is, six to seven PHY clocks from the end of the preceding isochronous OUT token.
- The device is connected to a host through multiple layers of hubs such that the interpacket gap between an isochronous OUT token and the following token shrinks to less than seven PHY clocks.

However, the system does not stop responding or enter an unrecoverable state. It has been observed that the USB host implementation is unable to send tokens with the worst-case inter-packet gap of 88-96 bit times.

Thresholding

When enabling thresholding, the bus must run at 60 MHz or faster. Do not enable the `USBC_THR_CTL_D.RXTHREN` bit because it may cause issues in the RxFIFO (especially during error conditions such as Rx error and babble).

USBC Architectural Concepts

The USB controller operates in either of two USB operation modes (device or host mode) at a given time.

In device mode, the USB controller encodes, decodes, checks, and directs all USB packets sent and received, responding appropriately to host requests. Data is transferred from the processor core memory into the Tx FIFOs of the device onto USB as IN packets. In the other direction, USB OUT packets are received into the Rx FIFOs (having been sent from the host) and transferred to system memory for processing or storage. In peripheral mode, the USB controller acts as a completer device to another USB host; either a personal computer or another OTG host controller.

When operating in host mode, the USB controller uses simple hosting capabilities to requester point-to-point connections with another USB peripheral, initiating transfers on the bus for the peripheral to respond. USB IN packets are received into the Rx FIFOs for transfer into the processor core memory. Data written into Tx FIFOs is transmitted onto the bus as USB OUT packets. In this mode, the USB controller encodes, decodes, and checks USB packets sent and received. The controller automatically schedules isochronous and interrupt transfers from the endpoint buffers. It performs one transaction every *n* frames, where n represents the polling interval programmed for the endpoint.

Any of the endpoints can be programmed to be written to or read from using the DMA requester channels. This configuration provides the most efficient means of transferring data between the controller and on-chip memory.

The USB provides two top-level maskable interrupts, each of which can be sourced from any or all transmit endpoint status, receive endpoint status, or global USB status.

The RAM interface of the USB controller supports a single block of synchronous single-port RAM used to buffer the USB packets.

ATTENTION: Check the processor data sheet for requirements regarding minimum system clock frequency needed for proper USB operation.

System Level Architecture

The following sections describe the system level architecture.

Bus Interface Unit (BIU)

The bus interface unit (BIU) includes the completer bus interface unit (BIUS) and the requester bus interface unit (BIUM).

Requester Bus Interface Unit (BIUM)

The internal DMA controller translates internal DMA requests into bus requester requests. The DMA address, transfer count, and packet count registers reside in the CSR block. The address of the selected channel is given as an input to the internal DMA controller. The BIUM converts the internal DMA request cycles into bus cycles. Characteristics of this interface include the following:

- Fully AMBA 2.0-compliant bus requester —No restrictions
- SINGLE, INCR, INC4, INC8, or INC8 burst types. When INC4, INC8, or INC16 is chosen, SINGLE and INCR cycles can happen due to a split, retry, or early termination.

  The USBC core uses INCR in the following scenarios:

  - Remaining data transfers (when the data length is not a direct multiple of the burst length, the remaining data is sent using INCR burst type).
- Handles bus ERROR condition
- Inserts busy cycle when needed by the application
- Handles bus 1 KB boundary breaking
- Data bus and address bus (of the bus requester) are 32 bits wide. The USBC core generates transfer with HSIZE = 2 (DWORD) and 0 (BYTE).

Completer Bus Interface Unit (BIUS)

The bus completer interface unit converts bus cycles to CSR write/read, data FIFO read/write, and DFIFO push/pop signals. DFIFO read and write access is made available only for testing purposes. The read, write, push, and pop signals are active high for one clock.

The wait states for different accesses are:

- Zero wait states on writes to the CSR, except when the state machine is busy with a previous push to DFIFO or write data RAM
- Zero wait states on pushes to DFIFO, one additional wait state when the USB accesses the FIFO RAM
- Zero wait states on debug writes to DFIFO, one additional wait state when the USB accesses the FIFO RAM
- One wait state on reads to CSR registers
- One initial wait state on NSEQ pop access and no subsequent wait states on NSEQ/SEQ back-to-back pops to DFIFO; one additional wait state when the USB accesses the FIFO RAM.
- Two wait states on software debug reads to DFIFO, one additional wait state when the USB accesses the FIFO RAM (debug access to DFIFO is not optimized to save area)

The BIUS is:

- Fully AMBA 2.0 compliant bus completer with no restrictions
- Supports INCR4, INCR8, INCR16, INCR and SINGLE transfers

- Supports busy and early terminations
- CSR and DFIFO reads/writes must always be 32-bit; 8- and 16-bit write accesses commit unknown values to the DFIFO.
- Generates OKAY responses only
- Does not generate SPLIT, RETRY, or ERROR responses

To make the software transparent to the hardware implementation, one 4 KB region is mapped to each device endpoint or host channel. The controller has:

- One common Rx FIFO, used in host and device modes
- Separate IN endpoint transmit FIFO for each device mode IN endpoints in a dedicated transmit FIFO operation
- The FIFO SPRAM is also used for storing some register values to save gates. In scatter/gather DMA mode, four SPRAM locations (four 35-bit words) are reserved for this reason.
- In DMA mode, one SPRAM location (one 35-bit word) is used for storing the DMA address

  Within the controller:

- Writes to any non-periodic IN endpoint or OUT channel are mapped to the non-periodic Tx FIFO

In device mode, a write access to an IN endpoint is mapped to the corresponding endpoint Tx FIFO. The `USBC_CTL_IEP0_D`[30:27] field maps an endpoint to a specific device periodic FIFO).

When `USBC_CTL_IEP[n]_D.TXFNUM` = 0, the endpoint is mapped to a common non-periodic Tx FIFO. Otherwise, it is mapped to the FIFO number selected by these fields. Hardware maintains the non-periodic Tx queue and host mode Tx periodic queues for internal operation. For debugging, software can read the top of the queue information. Because the read domain of these queues is in the PHY domain, no debug pop access is provided to these queues, saving area; access is provided only to the top of the queue.

NOTE: In device mode, each periodic IN endpoint has a separate buffer allocated, holding only one packet at a time. Therefore, there is no device mode periodic queue. Limiting to one packet shrinks the individual FIFO controller, saving area.

The ***Host FIFO Address Mapping*** figure shows FIFO mapping in host mode.

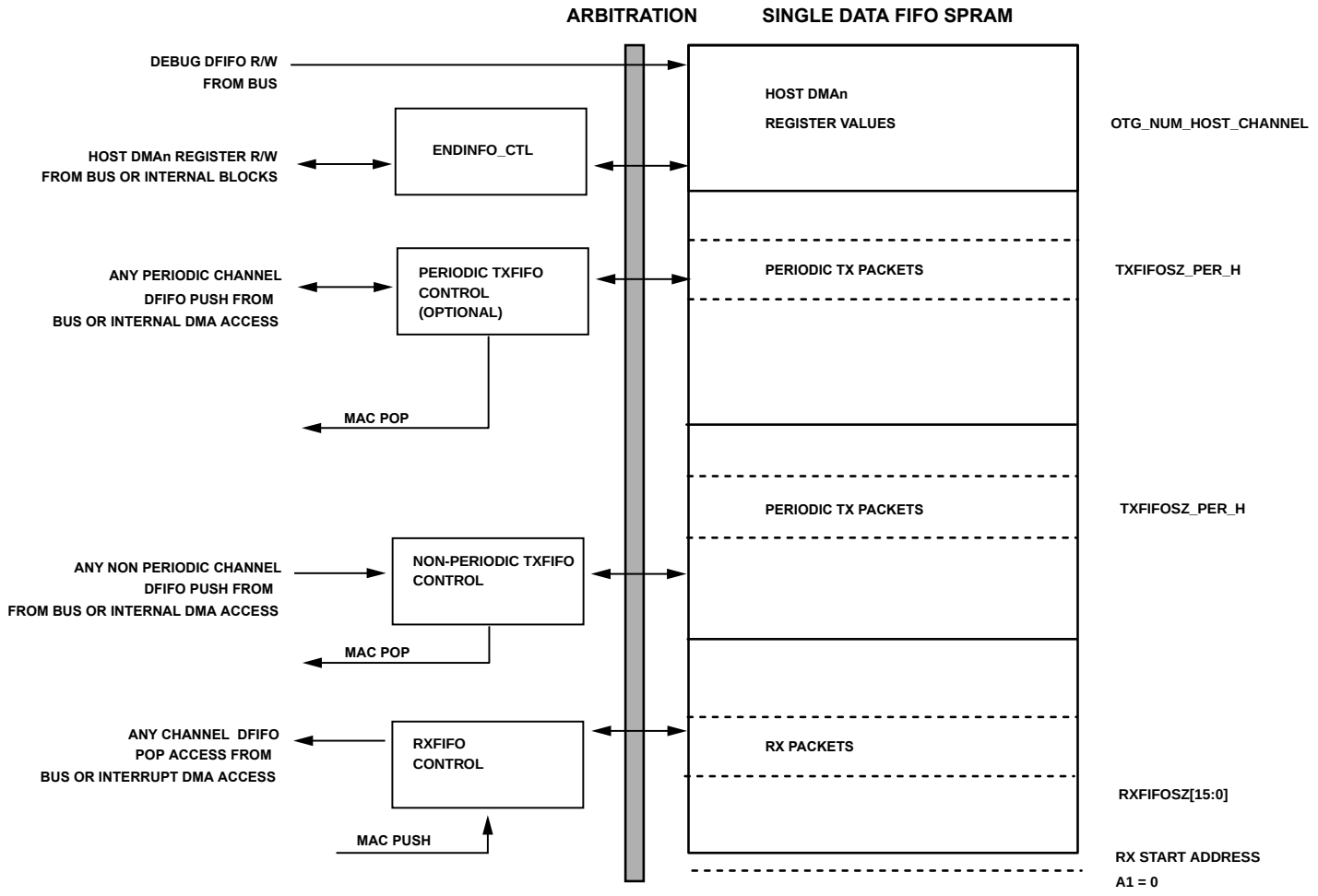


Figure 21-2: Host FIFO Address Mapping

The ***Device FIFO Address Mapping*** figure shows FIFO mapping in device mode.

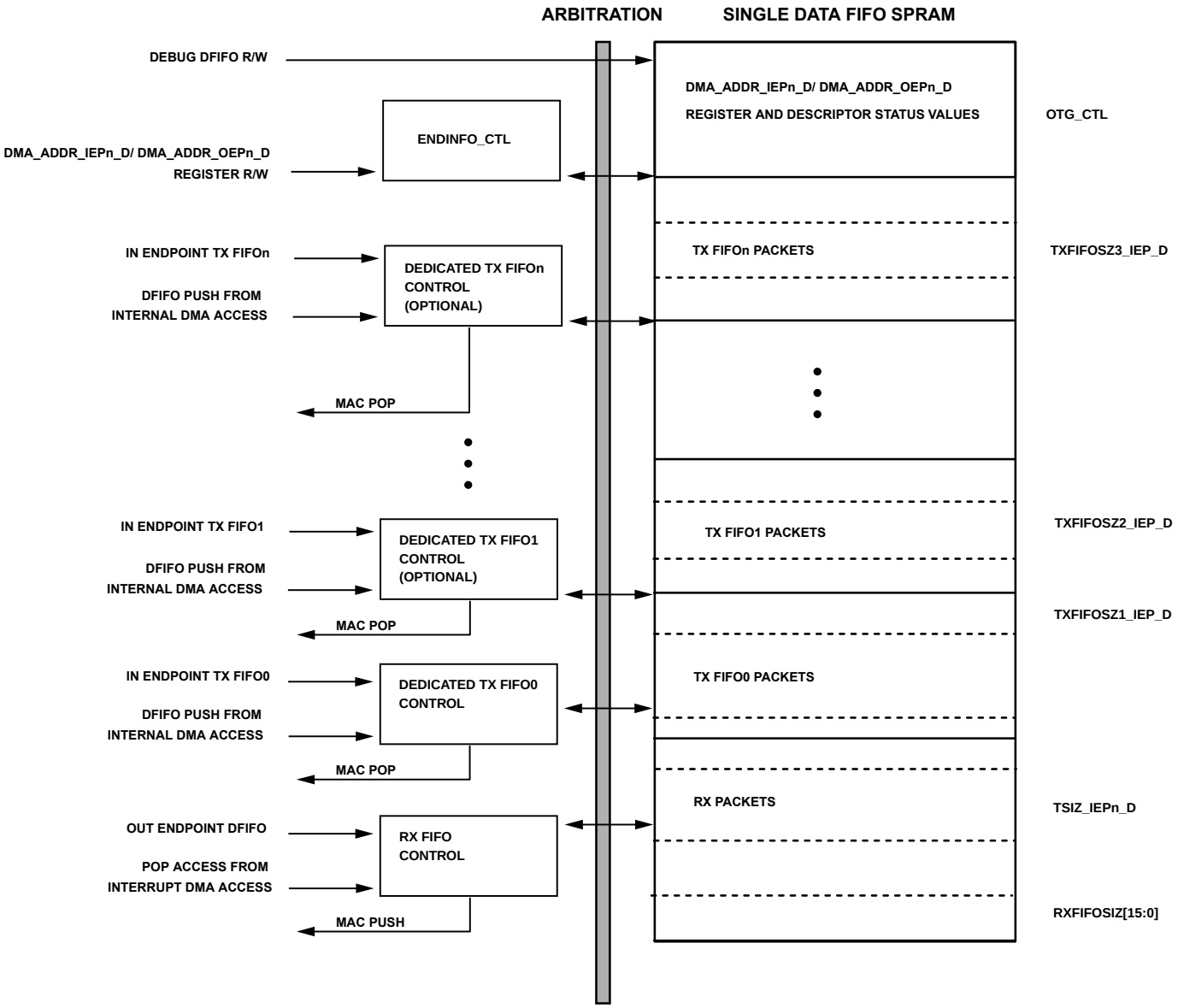


Figure 21-3: Device FIFO Address Mapping

Internal DMA Controller

The USB OTG provides an internal DMA mode of operation. The internal DMA controller has two modes of operation: buffer DMA mode and scatter/gather DMA mode.

The *USBC Internal DMA Control Block Diagram* shows the internal DMA controller mode.

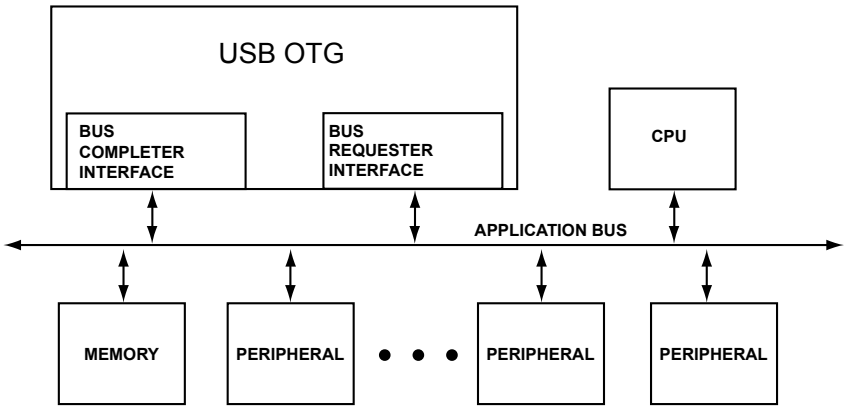


Figure 21-4: USBC Internal DMA Control Block Diagram

Host Architecture

The host uses one transmit FIFO for all non-periodic OUT transactions. The transmit FIFO is used as a transmit buffer to hold the data (payload of the transmit packet) to be transmitted over the USB port. The host pipes the USB transactions through request queues. Each entry in the request queue holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the requests are written into the queue determines the sequence of transactions on the USB. The host processes the non-periodic request queue, at the beginning of each (micro) frame.

The host uses one receive FIFO for all non-periodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to system memory. The status of each packet received also goes into the FIFO. The status entry holds the IN channel number (along with other information such as received byte count and validity status) to perform a transaction on the bus.

The ***System Level Host BIU Block Diagram*** shows the bus interface architecture of the USB OTG controller in host mode.

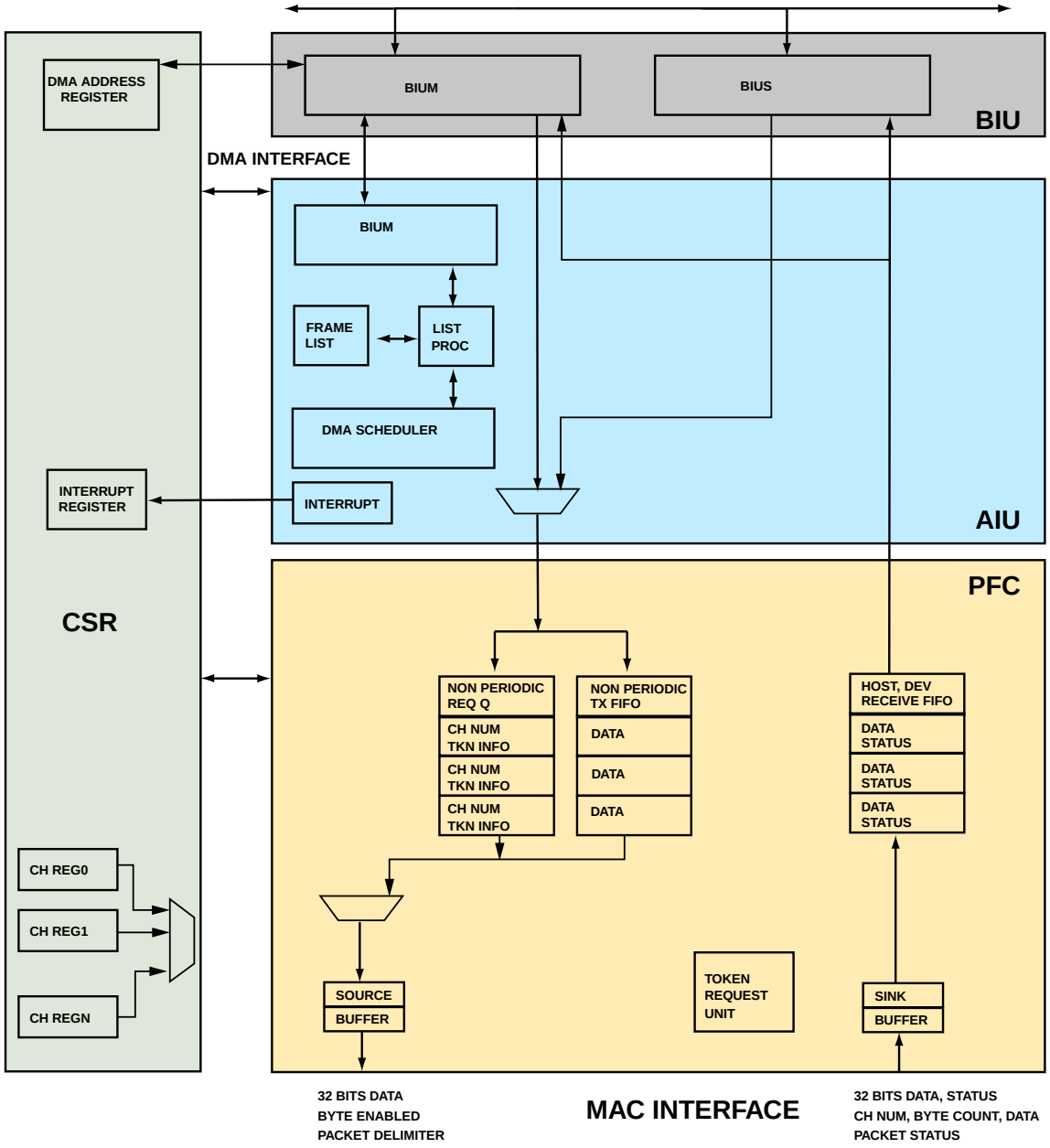


Figure 21-5: System Level Host BIU Block Diagram

Device Architecture

In device mode, the USB OTG core uses a dedicated transmit FIFO operation or a single receive FIFO. The ***System Level Device BIU Block Diagram*** shows the bus interface architecture of the USB OTG controller in device mode.

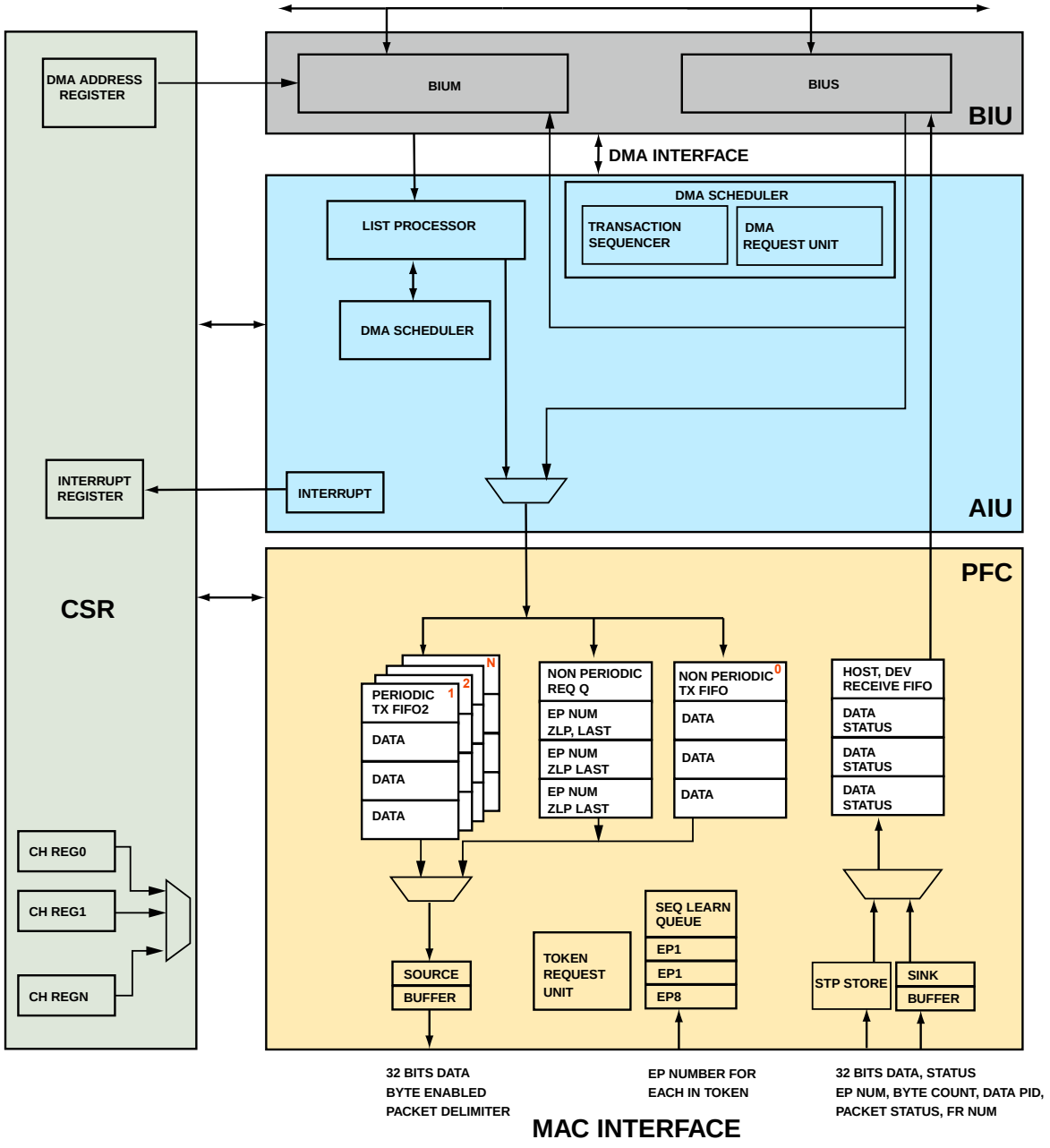


Figure 21-6: System Level Device BIU Block Diagram

Dedicated Transmit FIFO Operation

In a dedicated transmit FIFO operation, the USB OTG core uses individual transmit FIFOs for each IN endpoint. The core also supports thresholding in the transmit and receive directions when DMA mode is selected. For transmit mode, there are separate controls to enable thresholding for isochronous and non-isochronous transfers. Support is also provided for two different threshold values for the bus and the MAC on the transmit side.

When thresholding is enabled, the core can be configured to have a less-than-one packet sized FIFO. The core internally handles an underrun condition during transmit and corrupts the packet (inverts the CRC) on the USB.

During a receive operation with thresholding, when a packet ends up in a FIFO overflow condition, the core NAKs the OUT packet and internally rewinds the pointers. When thresholding is enabled, the best practice is to have a FIFO size two times the threshold value. If packet transmission results in an underrun condition (eventually resulting in packet corruption on the USB), the host can time out the endpoint after three consecutive errors.

Single Receive FIFO Operation

The OTG device uses a single receive FIFO to receive data for all the OUT endpoints. The receive FIFO holds the status of the received data packet, such as byte count, data PID, and the validity of the received data. The DMA or the application reads the data out of the receive FIFO as it is received.

Control and Status Registers (CSR)

The CSR block resides in the bus clock domain and contains all registers except the power and clock gating control register (`USBC_PWR_CTL`) and the core interrupt register (`USBC_ISTAT`[31:29]). These register bits must be active in power down mode, and hence are implemented in the BIUS module. The bus completer interface unit can write and read registers, and the core can set or reset the interrupt and status bits. The interrupts have priority over bus completer interface updates. For example, when a bus completer interface access tries to clear an interrupt bit when the core is setting the same interrupt bit, the interrupt bit is set, and the bus write to that bit is ignored.

Because the core does not check byte enables, all register accesses are 32 bits.

Even though bus completer writes have only the granularity of byte writes, each interrupt bit in an interrupt register is controlled separately. Thus, the bus completer interface can clear an interrupt bit when the core is setting a different interrupt bit, but no read modify byte write (which could cause erroneous interrupt behaviors) is performed.

All implemented registers are 32 bits wide, and only those registers required by the configuration are implemented. For example, when the core is configured for only two endpoints, then registers corresponding to endpoints 3–15 are not implemented. In addition, because the core can be only be either in host or device mode at a given time, the host and device registers use the same physical registers to save area. Writes to unimplemented registers are ignored and reads from them return an unknown value. Cycles complete gracefully with an OK response. When device mode registers are accessed during host mode and conversely, the mode mismatch interrupt (`USBC_ISTAT.MODEMIS`) is set, write data is ignored, and read data returns an unknown value.

Application Interface Unit (AIU)

The application Interface Unit (AIU) consists of the following interfaces:

- Bus requester
- Bus completer
- Packet FIFO controller
- Control and status registers

The AIU is responsible for the following functions.

- Generating and writing the delimiter (byte enables and last DWORD indicator) into the transmit FIFOs (based on FIFO number) for transmit packets in device (IN) and host (OUT) modes
- Writing a token into the request queue (periodic/non-periodic) for transmit transactions in device and host modes
- Writing a token into the request queue for receive (IN) transactions (only in host mode)
- Selecting the periodic or non-periodic request queue based on the type of transaction. When the channel or endpoint type indicates a periodic transaction, the periodic queue is selected.
- Generating interrupts for both transmit and receiving operations in device and host modes
- Address decoding for selecting the appropriate transmit FIFOs and queues
- Generating DMA interface signals to access an internal DMA controller for data transfer
- Round-robin arbitration among periodic and non-periodic channels and endpoints. Higher priority is always given to periodic channels or endpoints .
- Transmit/receive threshold handling (when transmit thresholding is enabled)
- Rewinding logic for DMA address and transfer size during an underrun or overflow condition (when thresholding is enabled)

DMA Scheduler (DSCH)

The DSCH is responsible for interfacing to an internal DMA. It controls the transfer of data packets between the system memory and the OTG core.

The DMA scheduler includes the following:

- *Arbiter* – This logic provides the sequence in which the channels/endpoints are to be processed in DMA mode. In host mode, the arbiter provides round-robin arbitration among periodic and non-periodic channels. Periodic channels are processed with higher priority.

  In device mode, when a threshold is enabled, the priority is as follows:

  1. Any transmit endpoint which is active on the USB
  2. Any receive data in a receive FIFO
  3. Round-robin arbitration on periodic transmit endpoints
  4. Round-robin arbitration on non-periodic transmit endpoints. In device mode when thresholding is not enabled, round robin arbitration is used for periodic and non-periodic IN endpoints with priority given to periodic IN endpoints.

- *DMA Request State Machine* – The state machine is responsible for the following:
  - Requesting the internal DMA for data fetch (from system memory to transmit FIFO, one maximum packet size or last packet size at a time)

- Writing the OUT request token into the request queue at the end of data fetch in host mode.
- Writing the IN request token into the request queue in host mode
- Requesting the internal DMA for data update (from the receive FIFO to system memory, one maximum packet size or last packet size at a time)
- Writing the request queue for ping, complete split, zero-length packet, or disable channel requests for the host

Periodic FIFO Control (PFC)

The PFC includes periodic and non-periodic transmit FIFOs and an endpoint information controller.

Periodic Transmit Data FIFO

The periodic transmit data FIFO stores periodic transmit data. There are different periodic FIFO structures in device and host modes.

Device Mode

There are no periodic or non-periodic FIFOs. FIFOs are implemented based on the number of IN endpoints and the application chooses a specific FIFO for an IN endpoint when enabling an endpoint. The following periodic IN endpoint characteristics remain valid in a dedicated transmit FIFO operation:

- The BIUS or the DMA interface unit writes data into a dedicated FIFO; the MAC reads the data from it.
- Each FIFO holds data for a single periodic IN endpoint. For high-bandwidth endpoints, this FIFO can hold in a single micro frame only the data to be transmitted, which can be no more than three maximum-packet-size packets.
- A whole packet must be in the FIFO before it can be transmitted on the USB, when thresholding is not enabled.
- When thresholding is enabled, the core starts transmitting on the USB in response to an IN token, when there is at least one MAC threshold amount of data available in the FIFO. For high-bandwidth periodic IN endpoints, the core can transmit one packet that is already available in the FIFO, while a second packet is being pushed into the FIFO.

Host Mode

Host mode periodic OUT transfers are not supported.

Non-Periodic Transmit Data FIFO

The non-periodic transmit data FIFO stores the data to be transmitted on all non-periodic IN endpoints in device mode and OUT channels in host mode. A non-periodic request queue associated with this FIFO stores endpoint or channel-related information.

- The BIUS or the DMA interface unit writes data to this FIFO; the MAC reads this data from it.

- In device mode, for each data packet in the data FIFO, there is a corresponding endpoint number and IN data packet type information entry in the request queue.
- In host mode, for each data packet in the data FIFO, there is a corresponding channel number and token-related control information entry in the request queue.
- The request queue also holds IN token entries in host mode (even though there is no associated data in the data FIFO).
- The depth of the non-periodic request queue is eight packets; it is implemented using flip-flops.
- Since dynamic FIFO sizing is enabled, the data FIFO RAM size and start address must be programmed in the CSRs.
- The data FIFO is 35 bits wide:
    - Bits [34:32]: A 4-bit byte enable, and a 1-bit packet delimiter are encoded into 3 bits and stored in external RAM. This reduces the data width in the external data RAM.
    - Encoding is as follows:
        - 3'b100: Last DWORD of the packet, all 4 bytes are valid
        - 3'b011: Last DWORD of the packet, bytes 0, 1, and 2 are valid
        - 3'b010: Last DWORD of the packet, bytes 0, and 1 are valid
        - 3'b001: Last DWORD of the packet, only byte 0 is valid
        - 3'b111: Non-last DWORD of a packet, all 4 bytes are valid
- Bits [31:0]: Data
- The request queue is 7 bits wide:
    - Host Mode:
        - Bits [6:3]: Channel number
        - Bits [2:0]: Token-related control information
    - Device Mode:
        - Bits [6:3]: IN endpoint number
        - Bit [2]: Not used
        - Bit [1]: Indicates a zero-length data packet
        - Bit [0]: Not used

Dedicated Transmit FIFO Operation

In host mode, the FIFO architecture is the same as in non-periodic transmit FIFO operation.

In device mode, there are no periodic or non-periodic FIFOs. FIFOs are implemented based on the number of IN endpoints (OTG_NUM_IN_EPS) (`USBC_HWCFG2.NDEVEPS`), and the application chooses a specific FIFO for an IN endpoint when it enables an endpoint. The following non-periodic IN endpoint characteristics remain valid in dedicated transmit FIFO mode:

- There are no queues associated with device mode FIFOs.
- If dynamic FIFO sizing is enabled, the data FIFO RAM size and the start address must be programmed in the CSRs
- The data FIFO is 35 bits wide. Bits 31:0 are data, and bits 34:32 are byte enables and packet delimiter encoding.

  Bits 34:32 are decoded below:

  - 3'b100 - Last DWORD of the packet, all 4 bytes valid
  - 3'b011 - Last DWORD of the packet, bytes 0, 1, 2 valid
  - 3'b010 - Last DWORD of the packet, bytes 0, 1 are valid
  - 3'b001 - Last DWORD of the packet, only byte 0 is valid
  - 3'b111 - Non last DWORD of a packet, all 4 bytes valid. 4-bit byte enables and 1-bit packet delimiter are encoded into 3 bits before storing into the external RAM. This reduces the data width in the external data RAM .

Receive Data FIFO

This FIFO stores data and status received on all OUT endpoints in device mode and all IN endpoints in host mode.

- The MAC writes data into this FIFO, and the BIUS or DMA interface unit reads this data from it.
- This FIFO can hold multiple OUT packets belonging to different endpoints or channels
- For each data OUT packet received on the USB, there is an associated status that indicates the byte count, packet status, and endpoint or channel number of the received packet.
- Data packets received with errors are flushed, so the BIUS and DMA interface unit never receive a corrupted data packet.
- Setup data packets received by the MAC in device mode are also written to the receive FIFO.
- Space is reserved in the receive data FIFO to store setup packets and related status information, based on the control endpoint supported by the device. The core never uses this reserved space to store any other type of data. This ensures that there is always enough space in the receive data FIFO to store setup data packets.
- If the device supports n endpoints, (4 * n + 6) DWORDs of space are allocated for receiving setup data packets. The extra 6 DWORDs are allocated to accommodate any back-to-back setup data packets sent by the USB host.
- The data FIFO is 35 bits wide:

- Bits [34:32]: Unused
- Bits [31:0]: data word
- Device mode status:
  - Bits [24:21]: 4 LSBs of the frame number in which the packet was received
  - Bits [20:17]: Packet status
  - Bits [16:15]: Data PID
  - Bits [14:4]: Byte count
  - Bits [3:0]: Endpoint number
- Host mode status:
  - Bits [20:17]: Packet status
  - Bits [16:15]: Data PID
  - Bits [14:4]: Byte count
  - Bits [3:0]: Channel number

Endpoint Information Controller (EPINFO_CTL)

The EPINFO_CTL manages the stored values in the last few locations of the SPRAM. The SPRAM Allocation table shows the space allocated for scatter/gather DMA.

Table 21-3: PRAM Allocation

Mode	SPRAM Space Allocation (EP_LOC_CNT)
Host	Four locations per channel are used in SPRAM to store the base descriptor address, current descriptor address, current buffer pointer, and the status quadlet. For example, if there are ten channels, the last forty locations are reserved for storing these values.
Device	Four locations per endpoint direction are used in SPRAM to store the base descriptor address, current descriptor address, current buffer pointer and the status quadlet. The application writes data to the base descriptor address. When the application reads the location where it wrote the base descriptor address, it receives the current descriptor address. For example, if there are ten bidirectional endpoints, the last 80 locations are reserved for storing these values.

Media Access Controller (MAC)

The Media Access Controller (MAC) module handles USB transactions, and device, host, and OTG protocols. This section discusses the MAC components, transaction and protocol handling.

MAC Components

The *MAC Block Diagram* shows the major MAC module components.

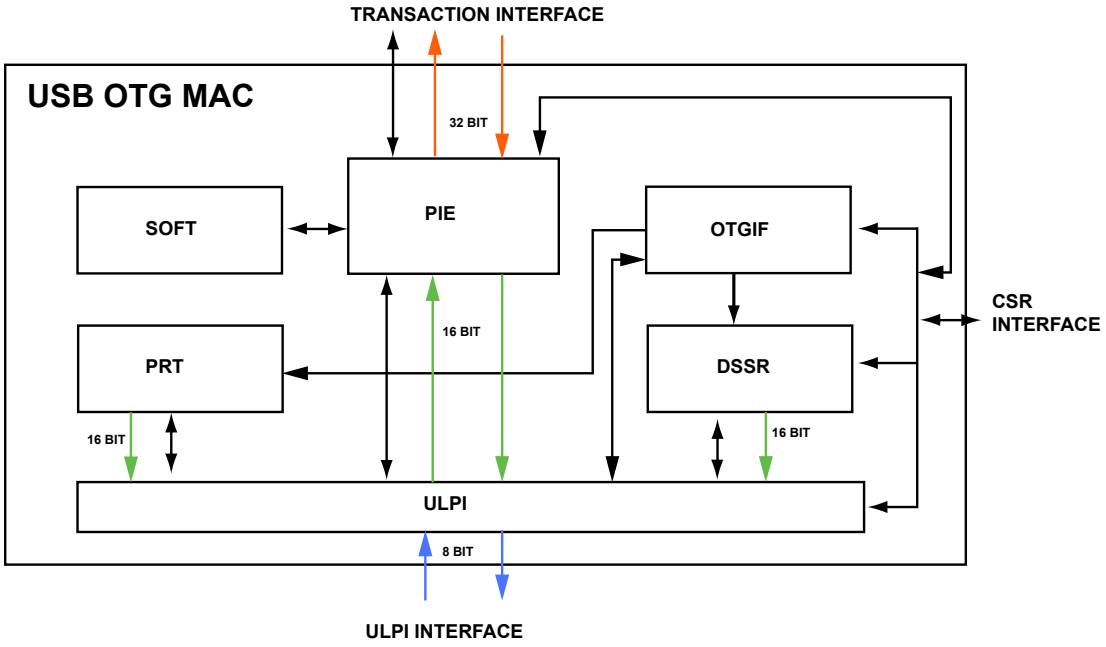


Figure 21-7: MAC Block Diagram

The MAC major blocks include

- *Device Speed enumeration, Suspend, and Resume block (DSSR)* – This block performs the speed enumeration, suspend, resume, and remote wakeup functions. The DSSR block is only active in device mode.
- *Parallel Interface Engine (PIE)* –This block is responsible for token, data, and handshake packet generation and reception, and PID and CRC checking and generation. It generates handshake and data packets based on data integrity and on CSR control and FIFO status information. The PIE also handles the data transfer to and from the FIFO, and the status update to the PFC and AIU.
- *SOF tracker (SOFT)* – This block tracks SOF packets and generates SOF interrupts in device mode. It handles missing SOFs and delayed SOFs to keep the frame number synchronization between the host and the device.
- *Port (PRT)* – The port block is only active in host mode. It is responsible for connect and disconnect detection, USB reset and speed enumeration, suspend and resume generation, remote wakeup detection, SOF generation, and high-speed test mode handling.
- *OTG Interface (OTGIF)* – The OTG interface block handles SRP and HNP. These OTG protocols are implemented either through the regular ULPI interface or the I^2C interface (for USB 1.1 serial transceivers only).
- *ULPI Interface* – The block converts data widths for the 8-bit PHY interface and multiplexes output signals to the PHY from multiple blocks.

USB Transaction Handling

The USB handles transactions differently in device and host modes.

Device Mode

In device mode, the MAC decodes and checks the integrity of token packets as it receives them from the host. If the received token is a valid OUT or setup token, the MAC waits and checks the PID of the following data packet. It then writes the data to the Rx FIFO (when it is available). After the packet is completed, the MAC checks the data integrity, sends the appropriate handshake when required to the host, and writes the transaction status to the receiving status queue. When the OUT token is received and the Rx FIFO is not available, the MAC sends the host a NAK handshake. If the received token is a valid ping token, the MAC sends the appropriate handshake, based on the FIFO status and CSR control information. When an IN token is received and the data is available in the FIFO, the MAC reads the data, builds the data packet, and sends it. It waits for a handshake packet, if any, from the host, and then updates the transaction status to the PFC. When an IN token is received and the data is not available in the FIFO, the MAC sends the host a NAK handshake.

Host Mode

In host mode, the MAC receives a token request from the AIU to start a USB transaction. After receiving a token request, the MAC builds ,and sends the requested token packet. For OUT or setup transactions, the MAC reads data from the Tx FIFO, builds and sends a data packet. It waits for a handshake packet, if any, from the device, and updates the transaction status to the AIU. For an IN or ping transaction, the MAC waits for a data or handshake packet from the device. If it receives a handshake packet, the MAC updates the status to the AIU. When it receives a data packet with the correct PID, the MAC writes the data into the Rx FIFO, checks the integrity of the data, sends a handshake packet, when required, to the device, then updates the status to the AIU.

Host Protocol Handling

In host mode, the MAC detects the device connect and disconnect, handles the USB reset and speed enumeration process, initiates USB suspend and resume, detects remote wakeup, generates SOF packets, and handles high-speed test modes.

Device Protocol Handling

In device mode, the MAC handles the USB reset sequence and speed enumeration process to determine the USB operating speed. The MAC detects USB suspend and resume signaling from the host, initiates remote wakeup, handles soft connect and disconnect, decodes and tracks SOF packets, plus handles high-speed test modes.

OTP Protocol Handling

The MAC handles Host Negotiation Protocol (HNP) for host and peripheral role swapping. It handles the Session Request Protocol (SRP), which allows an A-device to turn off VBUS to save power when the USB bus is used. It also provides a means for a B-device to request the A-device to activate VBUS.

USBC Programming Model

Programming the USB includes initializing and connecting the host and device cores and initializing the endpoint.

Program the Host Core

The following sections describe how to program the host core.

Initialize the Host

To initialize the USBC core as host, the application must perform the following steps:

1. Program the USBC_IMSK.PRTINT bit to unmask the PDM_HPF_CTL.FC bits.
2. Program the USBC_CFG_H register to select the full-speed host or high-speed host.
3. Program USBC_PORT_CTL_H.PWR =1. This configuration drives VBUS on the USB.
4. Wait for the USBC_PORT_CTL_H.CONNDET interrupt. This interrupt indicates that a device is connected to the port.
5. Program USBC_PORT_CTL_H.RST=1. This configuration starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program USBC_PORT_CTL_H.RST=0.
8. Wait for the USBC_PORT_CTL_H.ENCHNG interrupt.
9. Read the USBC_PORT_CTL_H.SPD field to get the enumerated speed.
10. Program the USBC_FIR_H register with a value corresponding to the selected PHY clock.
11. Program the USBC_RXFIFOSZ register to select the size of the RxFIFO.
12. Program the USBC_TXFIFOSZ_NP register to select the size and the start address of the non-periodic FIFO.

    ADDITIONAL INFORMATION: To communicate with devices, the application must initialize and enable at least one channel. At this point, the host is up and running and the port register begins to report device disconnects, etc. The port is active with the start of frame (SOF) packets occurring downstream of the enabled port.

Connect the Host

Complete the following steps to connect the host:

1. Connect the USB cable the host port; the core triggers the USBC_ISTAT.CONIDSTSCHNG interrupt.
2. When the host application detects the interrupt, the application can perform one of the following actions:
   a. Turn on VBUS by setting USBC_PORT_CTL_H.PWR = 1, or
   b. Wait for SRP signaling from the device to turn on VBUS

3. The PHY indicates VBUS power-on by asserting the UTMI_VBUSVALID signal.
4. The host core detects the device connection and triggers the host port interrupt (USBC_ISTAT.PRTINT) to the application.
5. The application reads the USBC_PORT_CTL_H register to check whether the USBC_PORT_CTL_H.CONNDET bit is set (=1).

Disconnect the Host

The host disconnection process depends on the configuration. Complete the following steps to disconnect the host:

1. When the device is disconnected from the USB cable (but the cable is still connected to the USB host), the core triggers the USBC_ISTAT.DISCONINT (disconnect detected) interrupt.
2. The host application can either to turn off the VBUS by programming USBC_PORT_CTL_H.PWR = 0, or, if the VBUS is controlled externally, the core does not switch off the VBUS.

Program the Device Core

The following sections describe how to initialize and connect the device core.

Initialize the Device

The application must meet the following conditions to set up the device core to handle traffic:

- In completer mode, the USBC_IMSK.NPTXFEMP and USBC_IMSK.RXFLVL bits must be cleared (=0)
- In DMA mode, the USBC_IMSK.NPTXFEMP and USBC_IMSK.RXFLVL interrupts must be masked.

The application must perform the following steps to initialize the core at device on, power on, or after a mode change from host to device.

1. Program the following fields in USBC_CFG_D register:
    a. USBC_CFG_D.DESCDMA–(applicable only when USBC_AHB_CFG.DMAEN parameter is set (=1))
    b. USBC_CFG_D.SPD–device speed
    c. USBC_CFG_D.NZSTSOUTHSHK–non-zero length status OUT handshake
    d. USBC_CFG_D.PERFRINT–periodic frame interval (when periodic endpoints are supported)
2. Program the device threshold control register. This operation is required only when using DMA mode and thresholding.
3. Clear (=0) the USBC_CTL_D.SFTDISCON bit. The core issues a connect after this bit is cleared.
4. Program the USBC_IMSK register to unmask the following interrupts.
    a. USB reset

b. Enumeration done

c. Early suspend

d. USB suspend

e. SOF

5. Wait for the USBC_ISTAT.USBRST interrupt; it indicates that a reset has been detected on the USB and lasts for about 10 ms. On receiving this interrupt, the application must perform the steps described in Initialize upon USB Reset.

6. Wait for the USBC_ISTAT.ENUMDONE interrupt. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the USBC_ISTAT_D register to determine the enumeration speed and perform the steps described in Initialize upon Enumeration Completion.

   At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0. To communicate with devices, the application must initialize and enable at least one channel. The host is up and running and the port register begins to report device disconnects, etc. The port is active with SOFs occurring downstream of the enabled port.

Connect the Device

The device connection process depends on an active VBUS and a connection to the USB cable.

VBUS is Active and the Device is Connected

When VBUS is active and the device is connected to the USB cable, there is no SRP from the device. The device connection flow is as follows:

1. The device triggers the USBC_ISTAT.SESREQINT interrupt bit.
2. The device application programs the required bits in the USBC_CFG_D register.
3. When the host drives reset, the device triggers USBC_ISTAT.USBRST upon detecting the reset. The host then follows the USB 2.0 enumeration sequence.

VBUS is Off and the Device is Connected

When VBUS is off and the device is connected to the USB cable, the device initiates SRP in OTG revision 1.3 mode. The device connection flow is as follows:

1. The application initiates SRP by writing to the USBC_OTG_CTL.SESREQ bit. The USBC core perform data-line pulsing followed by VBUS pulsing.
2. The host starts a new session by turning on VBUS, indicating SRP success. The USBC core interrupts the application by setting (=1) the USBC_OTG_CTL.SESREQSCS bit.
3. The application reads the USBC_OTG_CTL.SESREQ bit and programs the required bits in USBC_CFG_D register.

4. When the host drives reset, the device triggers the USBC_ISTAT.USBRST upon detecting the reset. The host then follows the USB 2.0 enumeration sequence.

Disconnect the Device

The device session ends when the USB cable is disconnected or VBUS is switched off by the host. The device disconnect flow depends the OTG mode.

Case 1:

- When the USB cable is unplugged or the VBUS is switched off by the host, the device core triggers the USBC_ISTAT.OTGINT interrupt bit.
- When the device application detects the USBC_ISTAT.OTGINT interrupt, it checks that the USBC_OTG_IRQ.SESENDDET (session end detected) bit is set (=1).

Case 2:

- When the USB cable is unplugged or the VBUS is switched off by the host, the device core triggers the USBC_ISTAT.USBRST interrupt bit.
- When the device application detects the USBC_ISTAT.USBRST bit, the application sets a timeout check for SET ADDRESS Control Xfer from the host.
- When application does not receive SET ADDRESS Control Xfer from the host before the timeout period, it is treated as a device disconnection.

Initialize the Endpoint

The following sections describe how to initialize the endpoint upon USB reset and after enumeration.

Initialize upon USB Reset

This section describes what the application must do when it detects a USB reset.

1. Set the NAK bit for all OUT endpoints.
    a. USBC_CTL_OEP[n]_D.SNAK = 1 (for all OUT endpoints)
2. Unmask the following interrupt bits.
    a. USBC_IMSK_D.INEP0 = 1 (control 0 IN endpoint)
    b. USBC_IMSK_D.OUTEP0 = 1 (control 0 OUT endpoint)
    c. USBC_IMASK_OEP_D.SETUP = 1
    d. USBC_IMASK_OEP_D.XFERCOMPL = 1
    e. USBC_IMASK_IEP_D.XFERCOMPL = 1
    f. USBC_IMASK_IEP_D.TIMEOUT = 1

3. To transmit or receive data, the device must initialize registers as specified in the Initialize the Device section
4. Set up the data FIFO RAM for each of the FIFOs (only if dynamic FIFO sizing is enabled).
   - Program the USBC_RXFIFOSZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 DWORDs (for the status of the control OUT data packet) + 10 DWORDs (for setup packets). If thresholding is enabled, at a minimum, this must be equal to 2*(USBC_THR_CTL_D.RXTHRLEN/4 + 1) + 2 DWORDs (for the status of the control OUT data packet) + 10 DWORDs (for setup packets)
   - Program the USBC_TXFIFOSZ_NP register in shared FIFO operation or dedicated FIFO size register (depending on the FIFO number chosen) in Dedicated FIFO operation, to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0. If thresholding is enabled, this can be programmed to less than one max packet size.
5. Reset the device address field in device configuration register (USBC_CFG_D.ADDR)
6. (This step is not required if you are using scatter/gather DMA mode). Program the following fields in the endpoint-specific registers for the control OUT endpoint 0 to receive a SETUP packet.
   - USBC_TSIZ_OEP0_D.SUPCNT = 3 (to receive up to three back-to-back SETUP packets)
   - In DMA mode, USBC_DMA_ADDR_OEP0_D register with a memory address to store any SETUP packets received

   At this point, all initialization required to receive SETUP packets is done, except for enabling control OUT endpoint 0 in DMA mode.

Initialize upon Enumeration Completion

This section describes what the application must do when it detects an enumeration done interrupt.

1. On the enumeration done interrupt (USBC_ISTAT.ENUMDONE), read the USBC_STAT_D.ENUMSPD to determine the enumeration speed.
2. Program the USBC_CTL_IEP0_D.MPS field to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed
3. In DMA mode, program the USBC_CTL_OEP0_D register to enable control OUT endpoint 0, to receive a SETUP packet. In scatter/gather DMA mode, the descriptors must be set up in memory before enabling the endpoint.
   a. USBC_CTL_OEP0_D.EPENA = 1
4. Unmask the SOF interrupt.

   The device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

OTG Programming

OTG programming includes host negotiating protocols for the A device and B device.

A Device Host Negotiation

The host negotiation protocol (HNP) switches the USB host role from the A device to the B device. The application must set the USBC_CFG.HNPCAP bit to enable the USBC core to perform the protocol as an A device.

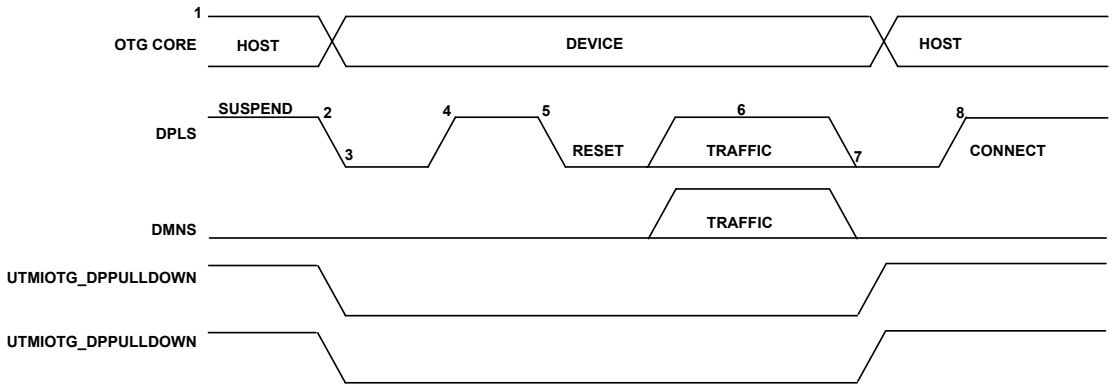


Figure 21-8: A Device HNP

The protocol completes the following sequence:

1. The USBC core sends the B device a SetFeature b_hnp_enable descriptor to enable HNP support. The ACK response of the B device indicates that the device supports HNP. (The application must set the USBC_OTG_CTL.HNPEN bit to indicate to the USBC core that the B device supports HNP).

2. When it has finished using the bus, the application suspends by writing to the USBC_PORT_CTL_H.SUSP bit.

3. When the B device detects a USB suspend operation and disconnects; the operation indicates the initial condition for the HNP. The B device initiates the HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

The USBC core sets the host negotiation detected interrupt bit (USBC_ISTAT.OTGINT), indicating the start of HNP.

The USBC core deasserts the UTMIOTG_DPPULLDOWN and UTMIOTG_DMPULLDOWN signals to indicate a device role. The PHY enable the D+ pull-up resistor indicates a connect for B device.

The application must read the USBC_OTG_CTL.CURMOD bit to determine the device mode operation.

4. The B device detects the connection, issues a USB reset, and enumerates the USBC core for data traffic.

5. The B device continues the host role, initiating traffic, and suspends the bus when the operation is complete.

The USBC core sets the USBC_ISTAT.ERLYSUSP bit after 3 ms of bus idleness.

6. In negotiated mode, the USBC core detects the suspend, disconnects, and switches back to the host role. The USBC core asserts the UTMIOTG_DPPULLDOWN and UTMIOTG_DMPULLDOWN signals to indicate its assumption of the host role.

7. The USBC core sets the USBC_ISTAT.CONIDSTSCHNG bit. The application must read the USBC_OTG_CTL.CONIDSTS bit to determine the USBC core operation as an A device. This indicates the completion of the HNP to the application. The application must read the USBC_OTG_CTL.CURMOD bit to determine host mode operation.

8. The B device connects, completing the HNP process.

B Device Host Negotiation

The host negotiation protocol (HNP) switches the USB host role from the B device to the A device. The application must set the USBC_CFG.HNPCAP bit to enable the USBC core to perform HNP as a B device.

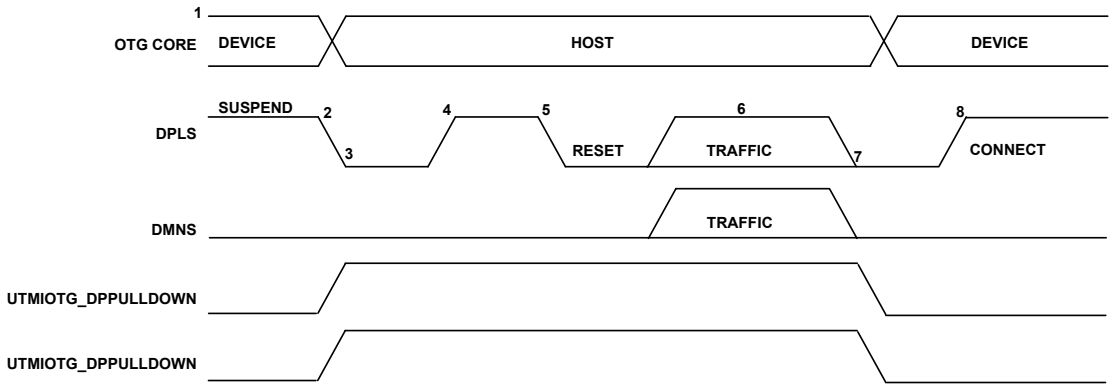


Figure 21-9: B Device HNP

1. The A device sends the SetFeature b_hnp_enable descriptor to enable HNP support. The USBC core ACK response indicates that it supports HNP. (The application must set the USBC_OTG_CTL.HNPEN bit to indicate HNP support).

The application sets the USBC_OTG_CTL.HNPREQ bit to indicate to the USBC core to initiate HNP.

2. When it has finished using the bus, the A device suspends by writing to the USBC_PORT_CTL_H.SUSP bit.

The USBC core sets the USBC_ISTAT.ERLYSUSP bit after 3 ms of bus idleness. Following this, the USBC core sets the USBC_ISTAT.ERLYSUSP bit.

The USBC core disconnects and the A device detects SE0 on the bus, indicating HNP. The USBC core asserts the UTMIOTG_DPPULLDOWN and UTMIOTG_DMPULLDOWN signals to indicate its assumption of the host role.

The A device responds by activating its D+ pull-up resistor within 3 ms of detecting SE0. The USBC core detects this as a connect.

The USBC core sets the USBC_OTG_IRQ.HNEGSSTSCHG bit, indicating the HNP status. The application must read the USBC_OTG_IRQ.HNEGSSTSCHG bit to determine host negotiation success. The application must read the USBC_ISTAT.CURMOD bit to determine host mode operation.

3. Program the USBC_PORT_CTL_H.PWR = 1 to drive VBUS on the USB.

4. Wait for the USBC_PORT_CTL_H.CONNDET interrupt. This indicates that a device is connected to the port.

5. The application sets the reset bit (USBC_PORT_CTL_H.RST) and the USBC core issues a USB reset and enumerates the A-device for data traffic.

6. Wait for the USBC_PORT_CTL_H.ENCHNG interrupt.

7. The USBC core continues the host role of initiating traffic, and when done, suspends the bus by writing the USBC_PORT_CTL_H.SUSP bit.

8. In negotiated mode, when the A device detects a suspend, it disconnects and switches back to the host role. The USBC core deasserts the UTMIOTG_DPPULLDOWN and UTMIOTG_DMPULLDOWN signals to indicate the assumption of the device role.

9. The application must read the USBC_ISTAT.CURMOD bit to determine the host mode operation.

10. The USBC core connects, completing the HNP process.

ADSP-SC59x USBC Register Descriptions

USB Controller (USBC) contains the following registers.

Table 21-4: ADSP-SC59x USBC Register List

Name	Description
USBC_ISTAT_D	Device All Endpoint Interrupt Status Register
USBC_IMSK_D	Device All Endpoint Interrupt Mask Register
USBC_CFG_D	Device Configuration Register
USBC_CTL_D	Device Control Register
USBC_CTL_IEP0_D	Device Control IN Endpoint 0 Control Register
USBC_CTL_IEP[n]_D	Device Control IN Endpoint n Control Register
USBC_DMA_ADDR_IEP0_D	Device Control IN Endpoint 0 DMA Address Register
USBC_DMA_BADDR_IEP0_D	Device IN Endpoint 0 Buffer Address Register
USBC_DMA_BADDR_IEP[n]_D	Device Control IN Endpoint n DMA Buffer Address Register
USBC_DMA_ADDR_IEP[n]_D	Device Control IN Endpoint n DMA Address Register
USBC_IMSK_IEP_FEMPT_D	Device IN Endpoint FIFO Empty Interrupt Mask Register
USBC_ISTAT_IEP0_D	Device Control IN Endpoint 0 Interrupt Control Register

Table 21-4: ADSP-SC59x USBC Register List (Continued)

Name	Description
USBC_ISTAT_IEP[n]_D	Device Control IN Endpoint n Interrupt Control Register
USBC_IMASK_IEP_D	Device IN Endpoint Common Interrupt Mask Register
USBC_TSIZ_IEP0_D	Device Control IN Endpoint 0 Transfer Size Register
USBC_TSIZ_IEP[n]_D	Device Control IN Endpoint n Transfer Size Register
USBC_TXFIFOSZ1_IEP_D	Device IN Endpoint 1 Transmit FIFO Size Register
USBC_TXFIFOSZ2_IEP_D	Device IN Endpoint 2 Transmit FIFO Size Register
USBC_TXFIFOSZ3_IEP_D	Device IN Endpoint 3 Transmit FIFO Size Register
USBC_CTL_OEP0_D	Device OUT Endpoint 0 Control Register
USBC_CTL_OEP[n]_D	Device OUT Endpoint n Control Register
USBC_DMA_ADDR_OEP0_D	Device OUT Endpoint 0 DMA Address Register
USBC_DMA_BADDR_OEP0_D	Device OUT Endpoint 0 Buffer Address Register
USBC_DMA_BADDR_OEP[n]_D	Device OUT Endpoint n Buffer Address Register
USBC_DMA_ADDR_OEP[n]_D	Device OUT Endpoint n DMA Address Register
USBC_ISTAT_OEP0_D	Device OUT Endpoint 0 Interrupt Register
USBC_ISTAT_OEP[n]_D	Device OUT Endpoint n Interrupt Register
USBC_IMASK_OEP_D	Device OUT Endpoint Common Interrupt Mask Register
USBC_TSIZ_OEP0_D	Device OUT Endpoint 0 Transfer Size Register
USBC_TSIZ_OEP[n]_D	Device OUT Endpoint n Transfer Size Register
USBC_STAT_D	Device Status Register
USBC_THR_CTL_D	Device Threshold Control Register
USBC_TXFSTAT_IEP0_D	Device Control IN Endpoint Transmit FIFO Status Register
USBC_TXFSTAT_IEP[n]_D	Device Control IN Endpoint Transmit FIFO Status Register
USBC_VBUSDIS_D	Device VBUS Discharge Time Register
USBC_VBUSPULSE_D	Device VBUS Pulsing Time Register
USBC_AHB_CFG	Bus Configuration Register
USBC_DFIFO_CFG	DFIFO Configuration Register
USBC_HWCFG1	User Hardware Configuration 1 Register
USBC_HWCFG2	User Hardware Configuration 2 Register
USBC_HWCFG3	User Hardware Configuration 3 Register
USBC_HWCFG4	User Hardware Configuration 4 Register
USBC_IMSK	Interrupt Mask Register

Table 21-4: ADSP-SC59x USBC Register List (Continued)

Name	Description
USBC_ISTAT	Interrupt Status Register
USBC_TXFIFOSZ_NP	Non-periodic Transmit FIFO Size Register
USBC_TXFIFO_STAT_NP	Non-periodic Transmit FIFO/Queue Status Register
USBC_OTG_CTL	OTG Control and Status Register
USBC_OTG_IRQ	OTG Interrupt Register
USBC_PHYIF_CTL	PHY Interface Control Register
USBC_RST_CTL	Reset Register
USBC_RXFIFOSZ	Receive FIFO Size Register
USBC_RXDATA_STAT	Receive Status Read/Pop Register
USBC_RXDBG_STAT	Receive Status Debug Read Register
USBC_MODID	Module ID Register
USBC_CFG	USB Configuration Register
USBC_ISTAT_H	Host All Channels Interrupt Register
USBC_IMSK_H	Host All Channels Interrupt Mask Register
USBC_CHAR[n]_H	Host Channel n Characteristics Register
USBC_DMA_BADDR[n]_H	Host Channel n DMA Buffer Address Register
USBC_DMA_ADDR[n]_H	Host Channel n DMA Address Register
USBC_CFG_H	Host Configuration Register
USBC_IMSK[n]_H	Host Channel n Interrupt Mask Register
USBC_ISTAT[n]_H	Host Channel n Interrupt Status Register
USBC_SPLT_CTL[n]_H	Host Channel n Split Control Register
USBC_TSIZ[n]_H	Host Channel n Transfer Size Register
USBC_FIR_H	Host Frame Interval Register
USBC_FL_BADDR_H	Host Frame List Base Address Register
USBC_FNUM_H	Host Frame Number/Frame Time Remaining Register
USBC_PORT_CTL_H	Host Port Control and Status Register
USBC_TXFIFOSZ_PER_H	Host Periodic Transmit FIFO Size Register
USBC_PWR_CTL	Power and Clock Gating Control Register

Device All Endpoint Interrupt Status Register

When a significant event occurs on an endpoint, the USBC_ISTAT_D register interrupts the application using the USBC_ISTAT.OEPINT or USBC_ISTAT.IEPINT register. There is one interrupt bit per endpoint, up to a maximum of 12 bits for OUT endpoints and 12 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used.

Bits in this register are set and cleared when the application sets and clears bits in the corresponding Device Endpoint n Interrupt register (USBC_ISTAT_IEP[n]_D/ USBC_ISTAT_OEP[n]_D).

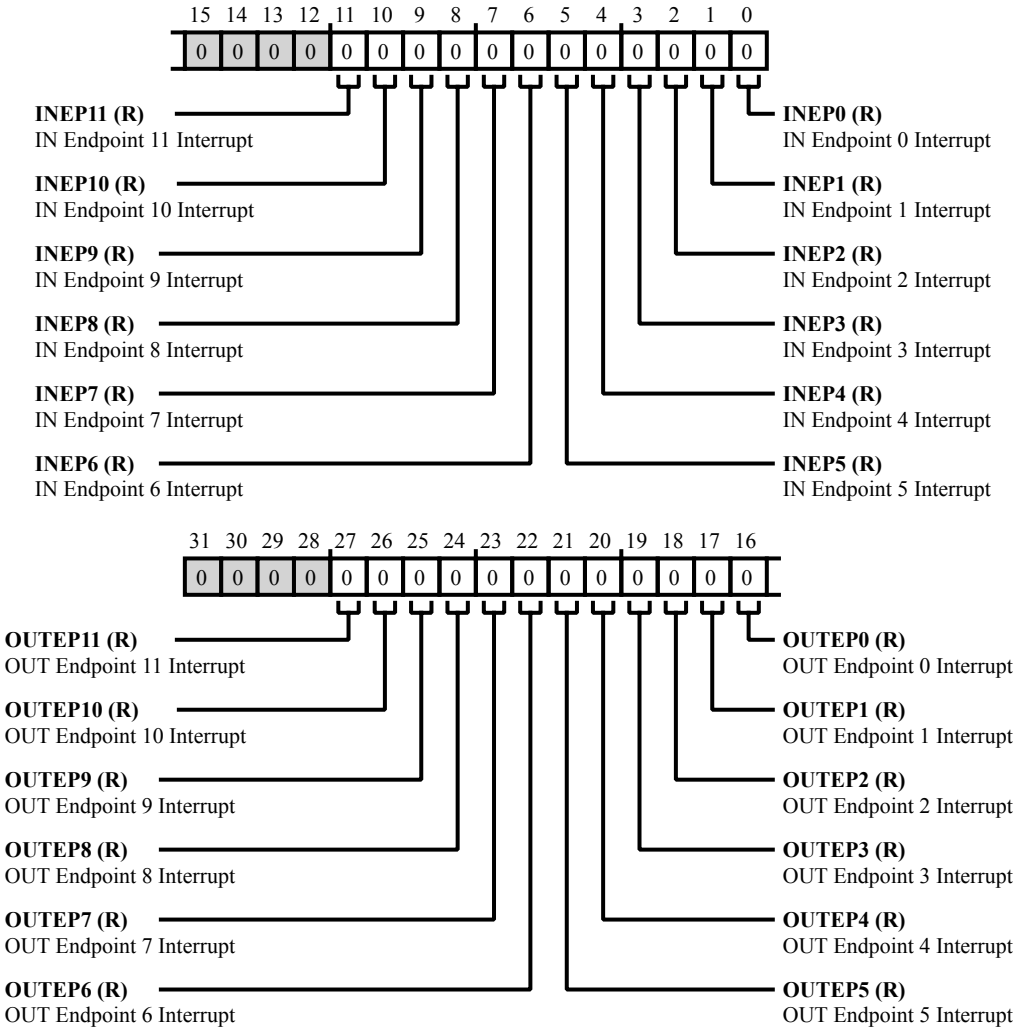


Figure 21-10: USBC_ISTAT_D Register Diagram

Table 21-5: USBC_ISTAT_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27 (R/NW)	OUTEP11	OUT Endpoint 11 Interrupt. The `USBC_ISTAT_D.OUTEP11` bit enables the interrupt request for the OUT endpoint.
26 (R/NW)	OUTEP10	OUT Endpoint 10 Interrupt. The `USBC_ISTAT_D.OUTEP10` bit enables the interrupt request for the OUT endpoint.
25 (R/NW)	OUTEP9	OUT Endpoint 9 Interrupt. The `USBC_ISTAT_D.OUTEP9` bit enables the interrupt request for the OUT endpoint.
24 (R/NW)	OUTEP8	OUT Endpoint 8 Interrupt. The `USBC_ISTAT_D.OUTEP8` bit enables the interrupt request for the OUT endpoint.
23 (R/NW)	OUTEP7	OUT Endpoint 7 Interrupt. The `USBC_ISTAT_D.OUTEP7` bit enables the interrupt request for the OUT endpoint.
22 (R/NW)	OUTEP6	OUT Endpoint 6 Interrupt. The `USBC_ISTAT_D.OUTEP6` bit enables the interrupt request for the OUT endpoint.
21 (R/NW)	OUTEP5	OUT Endpoint 5 Interrupt. The `USBC_ISTAT_D.OUTEP5` bit enables the interrupt request for the OUT endpoint.
20 (R/NW)	OUTEP4	OUT Endpoint 4 Interrupt. The `USBC_ISTAT_D.OUTEP4` bit enables the interrupt request for the OUT endpoint.
19 (R/NW)	OUTEP3	OUT Endpoint 3 Interrupt. The `USBC_ISTAT_D.OUTEP3` bit enables the interrupt request for the OUT endpoint.
18 (R/NW)	OUTEP2	OUT Endpoint 2 Interrupt. The `USBC_ISTAT_D.OUTEP2` bit enables the interrupt request for the OUT endpoint.
17 (R/NW)	OUTEP1	OUT Endpoint 1 Interrupt. The `USBC_ISTAT_D.OUTEP1` bit enables the interrupt request for the OUT endpoint.
16 (R/NW)	OUTEP0	OUT Endpoint 0 Interrupt. The `USBC_ISTAT_D.OUTEP0` bit enables the interrupt request for the OUT endpoint.

Table 21-5: USBC_ISTAT_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/NW)	INEP11	IN Endpoint 11 Interrupt. The `USBC_ISTAT_D.INEP11` bit enables the interrupt request for the IN endpoint.
10 (R/NW)	INEP10	IN Endpoint 10 Interrupt. The `USBC_ISTAT_D.INEP10` bit enables the interrupt request for the IN endpoint.
9 (R/NW)	INEP9	IN Endpoint 9 Interrupt. The `USBC_ISTAT_D.INEP9` bit enables the interrupt request for the IN endpoint.
8 (R/NW)	INEP8	IN Endpoint 8 Interrupt. The `USBC_ISTAT_D.INEP8` bit enables the interrupt request for the IN endpoint.
7 (R/NW)	INEP7	IN Endpoint 7 Interrupt. The `USBC_ISTAT_D.INEP7` bit enables the interrupt request for the IN endpoint.
6 (R/NW)	INEP6	IN Endpoint 6 Interrupt. The `USBC_ISTAT_D.INEP6` bit enables the interrupt request for the IN endpoint.
5 (R/NW)	INEP5	IN Endpoint 5 Interrupt. The `USBC_ISTAT_D.INEP5` bit enables the interrupt request for the IN endpoint.
4 (R/NW)	INEP4	IN Endpoint 4 Interrupt. The `USBC_ISTAT_D.INEP4` bit enables the interrupt request for the IN endpoint.
3 (R/NW)	INEP3	IN Endpoint 3 Interrupt. The `USBC_ISTAT_D.INEP3` bit enables the interrupt request for the IN endpoint.
2 (R/NW)	INEP2	IN Endpoint 2 Interrupt. The `USBC_ISTAT_D.INEP2` bit enables the interrupt request for the IN endpoint.
1 (R/NW)	INEP1	IN Endpoint 1 Interrupt. The `USBC_ISTAT_D.INEP1` bit enables the interrupt request for the IN endpoint.
0 (R/NW)	INEP0	IN Endpoint 0 Interrupt. The `USBC_ISTAT_D.INEP0` bit enables the interrupt request for the IN endpoint.

Device All Endpoint Interrupt Mask Register

The USBC_IMSK_D register works with the USBC_ISTAT_D register to interrupt the application when an event occurs on a device endpoint. However, the USBC_ISTAT_D register bit that corresponds to the interrupt is still set.

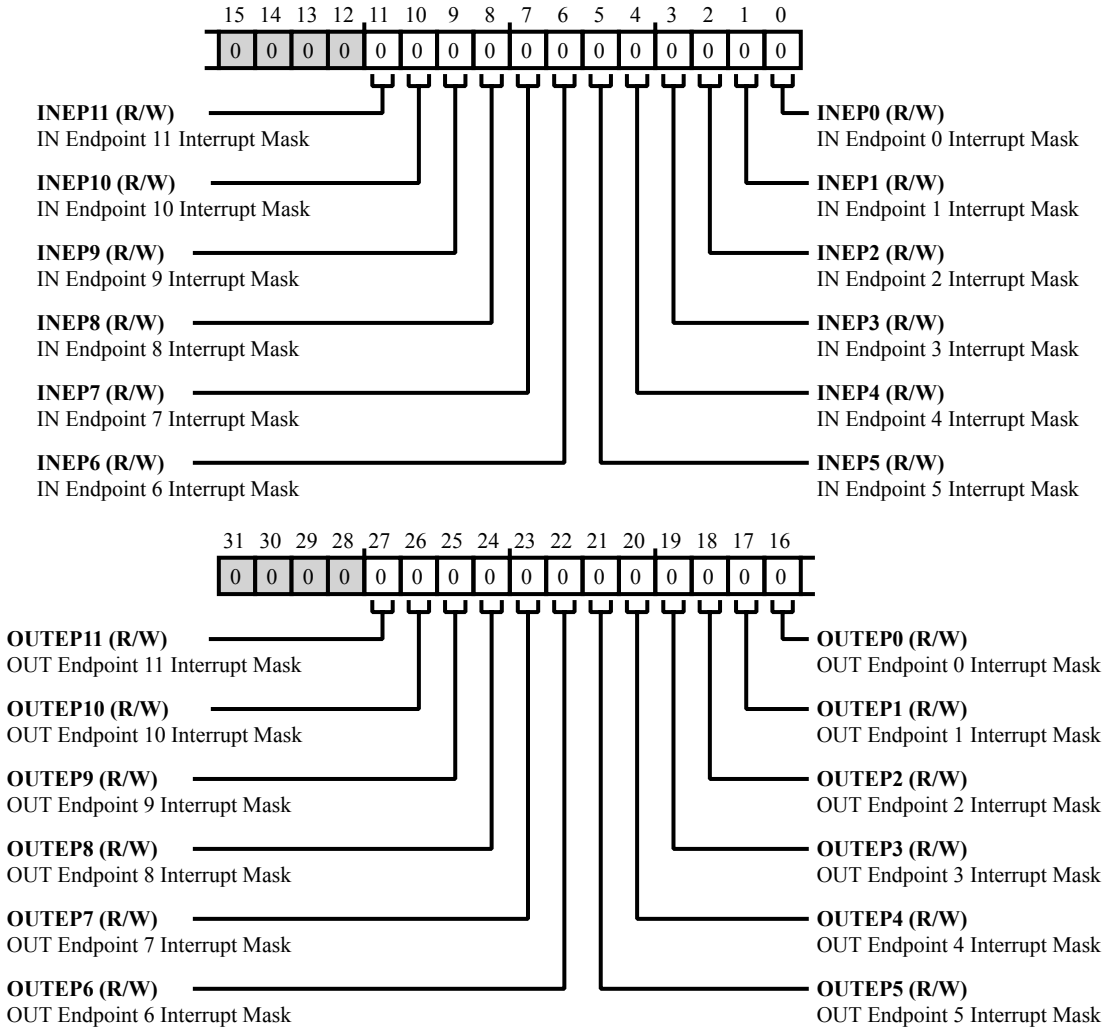


Figure 21-11: USBC_IMSK_D Register Diagram

Table 21-6: USBC_IMSK_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27 (R/W)	OUTEP11	OUT Endpoint 11 Interrupt Mask. The USBC_IMSK_D.OUTEP11 bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.

Table 21-6: USBC_IMSK_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
26 (R/W)	OUTEP10	OUT Endpoint 10 Interrupt Mask. The `USBC_IMSK_D.OUTEP10` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
25 (R/W)	OUTEP9	OUT Endpoint 9 Interrupt Mask. The `USBC_IMSK_D.OUTEP9` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
24 (R/W)	OUTEP8	OUT Endpoint 8 Interrupt Mask. The `USBC_IMSK_D.OUTEP8` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
23 (R/W)	OUTEP7	OUT Endpoint 7 Interrupt Mask. The `USBC_IMSK_D.OUTEP7` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
22 (R/W)	OUTEP6	OUT Endpoint 6 Interrupt Mask. The `USBC_IMSK_D.OUTEP6` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
21 (R/W)	OUTEP5	OUT Endpoint 5 Interrupt Mask. The `USBC_IMSK_D.OUTEP5` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
20 (R/W)	OUTEP4	OUT Endpoint 4 Interrupt Mask. The `USBC_IMSK_D.OUTEP4` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
19 (R/W)	OUTEP3	OUT Endpoint 3 Interrupt Mask. The `USBC_IMSK_D.OUTEP3` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
18 (R/W)	OUTEP2	OUT Endpoint 2 Interrupt Mask. The `USBC_IMSK_D.OUTEP2` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
17 (R/W)	OUTEP1	OUT Endpoint 1 Interrupt Mask. The `USBC_IMSK_D.OUTEP1` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.
16 (R/W)	OUTEP0	OUT Endpoint 0 Interrupt Mask. The `USBC_IMSK_D.OUTEP0` bit unmasks (enables) or masks (disables) the interrupt request for the OUT endpoint.

Table 21-6: USBC_IMSK_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/W)	INEP11	IN Endpoint 11 Interrupt Mask. The `USBC_IMSK_D.INEP11` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
10 (R/W)	INEP10	IN Endpoint 10 Interrupt Mask. The `USBC_IMSK_D.INEP10` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
9 (R/W)	INEP9	IN Endpoint 9 Interrupt Mask. The `USBC_IMSK_D.INEP9`bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
8 (R/W)	INEP8	IN Endpoint 8 Interrupt Mask. The `USBC_IMSK_D.INEP8`bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
7 (R/W)	INEP7	IN Endpoint 7 Interrupt Mask. The `USBC_IMSK_D.INEP7` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
6 (R/W)	INEP6	IN Endpoint 6 Interrupt Mask. The `USBC_IMSK_D.INEP6` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
5 (R/W)	INEP5	IN Endpoint 5 Interrupt Mask. The `USBC_IMSK_D.INEP5` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
4 (R/W)	INEP4	IN Endpoint 4 Interrupt Mask. The `USBC_IMSK_D.INEP4` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
3 (R/W)	INEP3	IN Endpoint 3 Interrupt Mask. The `USBC_IMSK_D.INEP3` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
2 (R/W)	INEP2	IN Endpoint 2 Interrupt Mask. The `USBC_IMSK_D.INEP2` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.
1 (R/W)	INEP1	IN Endpoint 1 Interrupt Mask. The `USBC_IMSK_D.INEP1` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.

Table 21-6: USBC_IMSK_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	INEP0	IN Endpoint 0 Interrupt Mask. The `USBC_IMSK_D.INEP0` bit unmasks (enables) or masks (disables) the interrupt request for the IN endpoint.

Device Configuration Register

The USBC_CFG_D register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

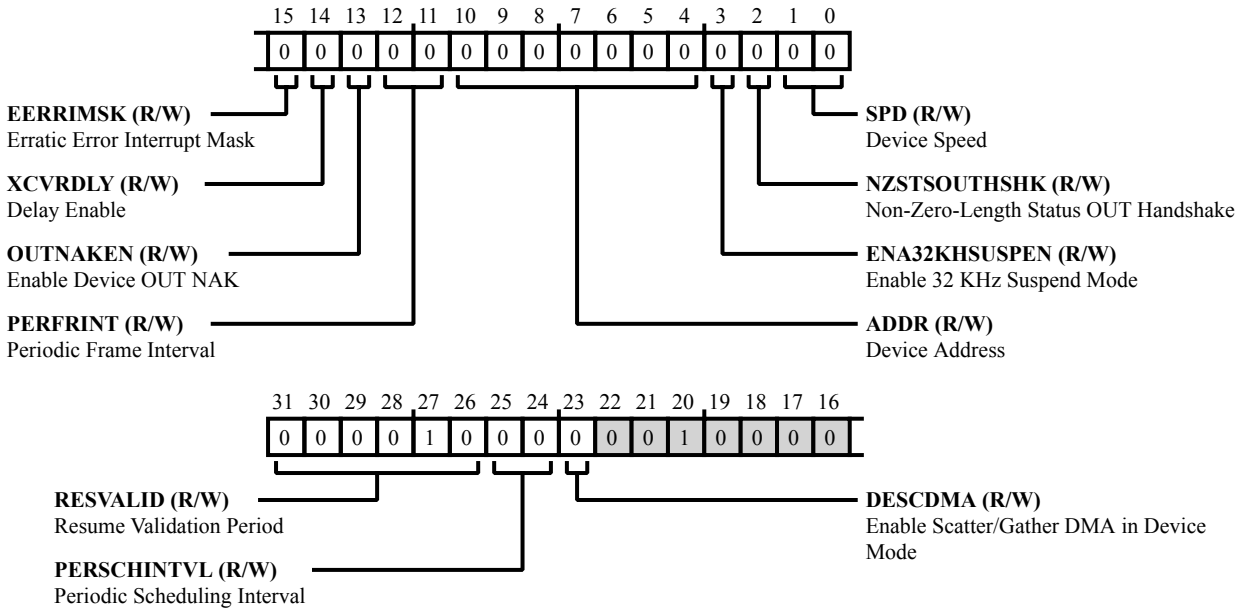


Figure 21-12: USBC_CFG_D Register Diagram

Table 21-7: USBC_CFG_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:26 (R/W)	RESVALID	Resume Validation Period. The USBC_CFG_D.RESVALID field controls the resume period when the core resumes after suspend. The core counts for USBC_CFG_D.RESVALID clock cycles to detect a valid resume. The USBC_CFG_D.RESVALID field is effective only when USBC_CFG_D.ENA32KHSUSPEN is set.

Table 21-7: USBC_CFG_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/W)	PERSCHINTVL	Periodic Scheduling Interval. The USBC_CFG_D.PERSCHINTVL field must be programmed for scatter/gather DMA mode. The USBC_CFG_D.PERSCHINTVL field specifies the amount of time the internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25, 50, or 75% of (micro)frame. When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data. When no periodic endpoints are active, the internal DMA engine services the non-periodic endpoints and ignores this field. After the specified time within a (micro)frame, the DMA switches to fetching for non-periodic endpoints.	
		0	25% of microframe
		1	50% of microframe
		2	75% of microframe
		3	Reserved
23 (R/W)	DESCDMA	Enable Scatter/Gather DMA in Device Mode. When the scatter/gather DMA option is selected during configuration, the application can set the USBC_CFG_D.DESCDMA bit during initialization to enable the scatter/gather DMA operation. Note: This bit must be modified only once after a reset. The following combinations are available for programming: Slave mode: USBC_AHB_CFG.DMAEN=0, USBC_CFG_D.DESCDMA=0 Invalid: USBC_AHB_CFG.DMAEN=0, USBC_CFG_D.DESCDMA=1 Buffered DMA mode: USBC_AHB_CFG.DMAEN=1, USBC_CFG_D.DESCDMA=0 Scatter/gather DMA mode: USBC_AHB_CFG.DMAEN=1, USBC_CFG_D.DESCDMA=1	
15 (R/W)	EERRIMSK	Erratic Error Interrupt Mask. The USBC_CFG_D.EERRIMSK bit unmasks (enables) or masks (disables) the interrupt request for early suspend on erratic error.	
14 (R/W)	XCVRDLY	Delay Enable. The USBC_CFG_D.XCVRDLY bit enables or disables the delay between xcvr_sel and txvalid during device chirp.	

Table 21-7: USBC_CFG_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	OUTNAKEN	Enable Device OUT NAK. The `USBC_CFG_D.OUTNAKEN` bit enables setting NAK for bulk OUT endpoints after the transfer is completed for device mode descriptor DMA. The `USBC_CFG_D.OUTNAKEN` bit is one time programmable after reset like any other `USBC_CFG_D` register bits.	
		0	Disable
		1	Enable
12:11 (R/W)	PERFRINT	Periodic Frame Interval. The `USBC_CFG_D.PERFRINT` field indicates the time within a (micro)frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine when all the isochronous traffic for that (micro)frame is complete.	
		0	80% of the microframe interval
		1	85% of the microframe interval
		2	90% of the microframe interval
		3	95% of the microframe interval
10:4 (R/W)	ADDR	Device Address. The `USBC_CFG_D.ADDR` field must be programmed after every SetAddress control command.	
3 (R/W)	ENA32KHSUSPEN	Enable 32 KHz Suspend Mode. The `USBC_CFG_D.ENA32KHSUSPEN` bit can be set only when the FS PHY interface is selected. Otherwise, this bit needs to be cleared. If the FS PHY interface is chosen and this bit is set, the PHY clock during suspend must be switched from 48 MHz to 32 KHz.	
2 (R/W)	NZSTSOUTHSHK	Non-Zero-Length Status OUT Handshake. The `USBC_CFG_D.NZSTSOUTHSHK` field selects the handshake that the core sends upon receiving a non-zero length data packet during the OUT transaction of a control transfer's status stage.	
		0	Send the received OUT packet to the application (zero length or non-zero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the Device Endpoint Control register.
		1	Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.

Table 21-7: USBC_CFG_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	SPD	Device Speed. The `USBC_CFG_D.SPD` bit indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the connect sequence is completed, and is based on the speed of the USB host to which the core is connected.	
		0	High Speed USB 2.0. PHY clock is 30 Mhz or 60 Mhz.
		1	Full Speed USB 2.0. PHY clock is 30 Mhz or 60 Mhz.
		2	Low Speed USB 1.1 Transceiver clock is 6 Mhz.
		3	Full Speed USB 1.1. Transceiver clock is 48 MHz.

Device Control Register

In device mode, the USBC_CTL_D register controls global functions such as wakeup and soft disconnect.

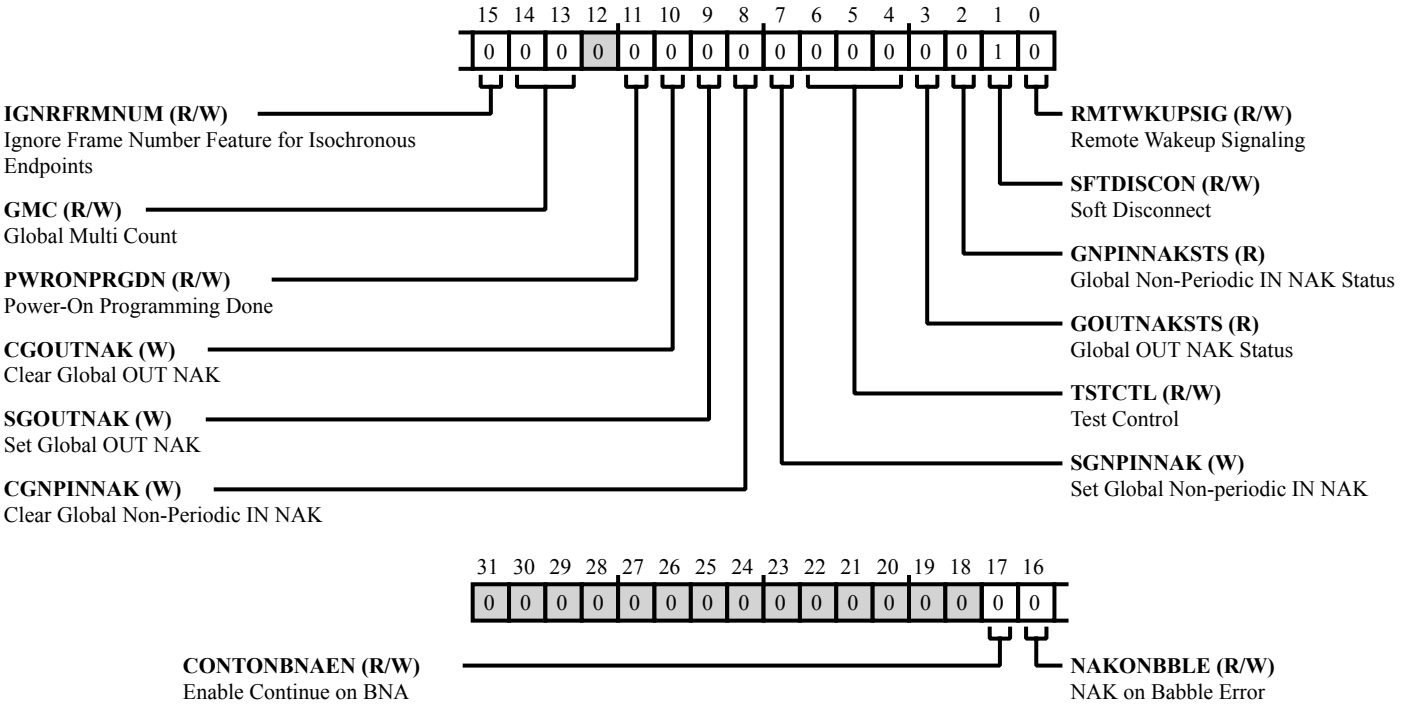


Figure 21-13: USBC_CTL_D Register Diagram

Table 21-8: USBC_CTL_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	CONTONBNAEN	Enable Continue on BNA. The USBC_CTL_D.CONTONBNAEN bit enables the core to continue on BNA for bulk OUT endpoints. With this feature enabled, when a bulk OUT or INTR OUT endpoint receives a BNA interrupt, the core starts processing the descriptor that caused the BNA interrupt after the endpoint re-enables the endpoint. After receiving a BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the DOEPDMA descriptor or from the descriptor that received the BNA interrupt. The USBC_CTL_D.CONTONBNAEN bit is a one-time programmable after reset bit (like the other USBC_CTL_D register bits).	
		0	Process from the DOEPDMA descriptor
		1	Process from the descriptor that received the BNA interrupt

Table 21-8: USBC_CTL_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	NAKONBBLE	NAK on Babble Error. When set (=1), the USBC_CTL_D.NAKONBBLE bit enables NAK automatically on babble (NakOnBble). The core sets NAK automatically for the endpoint on which babble is received.
15 (R/W)	IGNRFRMNUM	Ignore Frame Number Feature for Isochronous Endpoints. The USBC_CTL_D.IGNRFRMNUM bit controls the Periodic Transfer Interrupt (PTI) feature. When USBC_CTL_D.IGNRFRMNUM is set (=1), the core transmits the packets only in the frame number in which they are intended to be transmitted. When USBC_CTL_D.IGNRFRMNUM is cleared, the core ignores the frame number, sending packets immediately as the packets are ready. Note: Do not set (=1) the USBC_CTL_D.IGNRFRMNUM bit when the core is operating in threshold mode. This feature does not apply to high speed, high bandwidth transfers. When the USBC_CTL_D.IGNRFRMNUM bit is set (=1), there must be only one packet per descriptor. In scatter/gather DMA mode, if USBC_CTL_D.IGNRFRMNUM bit is enabled, the packets are not flushed when a ISOC IN token is received for an elapsed frame. In PTI mode, the application receive a transfer complete interrupt after transfers for multiple (micro)frames are completed.
14:13 (R/W)	GMC	Global Multi Count. The USBC_CTL_D.GMC field indicates the number of packets to be serviced for the endpoint before moving to the next endpoint. It applies only to non-periodic endpoints. The USBC_CTL_D.GMC bit must be programmed only once after initialization. The value of this field automatically changes to 2'h1 when USBC_CFG_D.DESCDMA = 1. 0 Invalid 1 1 packet 2 2 packets 3 3 packets
11 (R/W)	PWRONPRGDN	Power-On Programming Done. Th USBC_CTL_D.PWRONPRGDN bit indicates that register programming is completed after a wakeup from power down mode.
10 (RX/W)	CGOUTNAK	Clear Global OUT NAK. A write to the USBC_CTL_D.CGOUTNAK field clears the global OUT NAK.

Table 21-8: USBC_CTL_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (RX/W)	SGOUTNAK	Set Global OUT NAK. A write to the `USBC_CTL_D.SGOUTNAK` field sets the global OUT NAK. The application uses this bit to send a NAK handshake on all OUT endpoints. The application must set the this bit only after making sure that the `USBC_ISTAT.GOUTNAKEFF` bit is cleared (=0).	
8 (RX/W)	CGNPINNAK	Clear Global Non-Periodic IN NAK. A write to the `USBC_CTL_D.CGNPINNAK` field clears the global non-periodic IN NAK.	
7 (RX/W)	SGNPINNAK	Set Global Non-periodic IN NAK. A write to the `USBC_CTL_D.SGNPINNAK` field sets the global non-periodic IN NAK. The application uses this bit to send a NAK handshake on all non-periodic IN endpoints. The core can also set this bit when a timeout condition is detected on a non-periodic endpoint in shared FIFO operation. The application must set this bit only after ensuring that after making sure that the `USBC_ISTAT.GINNAKEFF` bit is cleared (=0).	
6:4 (R/W)	TSTCTL	Test Control. The `USBC_CTL_D.TSTCTL` field indicates the test mode state.	
		0	Test mode disabled
		1	Test_J mode. In this mode, the USB controller transmits a continuous J on the bus.
		2	Test_K mode. In this mode, the USB controller transmits a continuous K on the bus.
		3	Test_SE0 NAK mode
		4	Test_Packet mode
		5	Test_Force_Enable mode
3 (R/NW)	GOUTNAKSTS	Global OUT NAK Status. The `USBC_CTL_D.GOUTNAKSTS` bit indicates whether a handshake is sent based on the FIFO status and the NAK and STALL bit settings or sends a NAK handshake on all packets, except on SETUP transactions. No data is written to the RxFIFO (irrespective of space availability). All isochronous OUT packets are dropped	
		0	Send handshake
		1	Send NAK handshake

Table 21-8: USBC_CTL_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	GNPINNAKSTS	Global Non-Periodic IN NAK Status. The `USBC_CTL_D.GNPINNAKSTS` bit indicates whether a handshake is sent out based on the data availability in the transmit FIFO or an NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.	
		0	Handshake based on available data
		1	NAK Handshake
1 (R/W)	SFTDISCON	Soft Disconnect. The `USBC_CTL_D.SFTDISCON` bit signals the controller to do a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit. The following is the minimum duration under various conditions for which this bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application adds some extra delay to the specified minimum duration. For high speed, if the device state is suspended, the minimum duration is 1 ms + 2.5 us. If the device state is idle, the minimum duration is 3 ms + 2.5 us. If the device state is not idle or suspended (performing transactions), the minimum duration 125 us For full speed/low speed, if the device state is suspended, the minimum duration is 1 ms + 2.5 us. if the device state is idle, the minimum duration is 2.5 us. if the device state is not idle or suspended (performing transactions), the minimum duration 125 us. Note: This bit can be also used for ULPI/FS serial interfaces. This bit is not impacted by a soft reset.	
		0	Normal Operation When this bit is cleared after a soft disconnect, the controller generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.
		1	The controller generates a device disconnect event to the USB host.

Table 21-8: USBC_CTL_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	RMTWKUPSIG	Remote Wakeup Signaling. When the `USBC_CTL_D.RMTWKUPSIG` bit is set (=1), the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1-15 ms after setting it. If LPM is enabled and the core is in the L1 (sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the sleep state. As specified in the LPM specification, the hardware automatically clears the `USBC_CTL_D.RMTWKUPSIG` bit 50 ms after being set by the application.

Device Control IN Endpoint 0 Control Register

In device mode, the USBC_CTL_IEP0_D register controls IN endpoint 0.

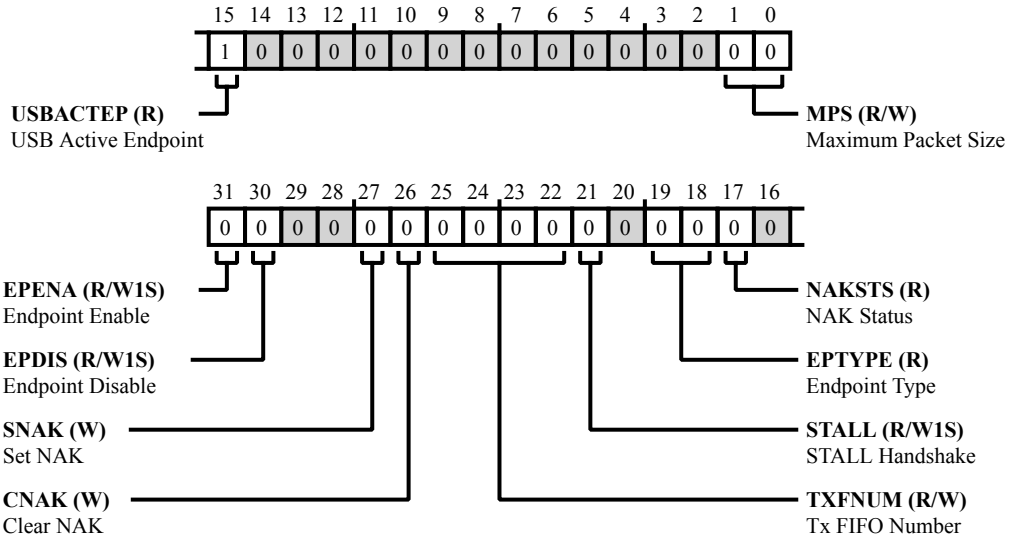


Figure 21-14: USBC_CTL_IEP0_D Register Diagram

Table 21-9: USBC_CTL_IEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	EPENA	Endpoint Enable. The USBC_CTL_IEP0_D.EPENA bit indicates that the descriptor structure and data buffer with data ready for transmission is configured.
30 (R/W1S)	EPDIS	Endpoint Disable. The application sets the USBC_CTL_IEP0_D.EPDIS bit to stop transmitting data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears the USBC_CTL_IEP0_D.EPDIS bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.
27 (RX/W)	SNAK	Set NAK. A write to the USBC_CTL_IEP0_D.SNAK bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.
26 (RX/W)	CNAK	Clear NAK. A write to the USBC_CTL_IEP0_D.CNAK bit clears the NAK bit for the endpoint.

Table 21-9: USBC_CTL_IEP0_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:22 (R/W)	TXFNUM	Tx FIFO Number. The `USBC_CTL_IEP0_D.TXFNUM` field indicates the FIFO number that is assigned to the IN endpoint.	
21 (R/W1S)	STALL	STALL Handshake. The application sets (=1) the `USBC_CTL_IEP0_D.STALL` bit, and the core clears (=0) it, when a SETUP token is received for the endpoint. If a NAK bit, the global non-periodic IN NAK, or global OUT NAK is set along with this bit, the `USBC_CTL_IEP0_D.STALL` bit takes priority.	
19:18 (R/NW)	EPTYPE	Endpoint Type. The `USBC_CTL_IEP0_D.EPTYPE` bit is hardcoded to 00 for control.	
17 (R/NW)	NAKSTS	NAK Status. When the `USBC_CTL_IEP0_D.NAKSTS` bit is set, either by the application or core, the core stops transmitting data, even when there is data available in the TxFIFO. Irrespective of the bit configuration, the core always responds to SETUP data packets with an ACK handshake.	
		0	Core is transmitting non-NAK handshakes based on FIFO status
		1	Core is transmitting NAK handshakes on the endpoint
15 (R/NW)	USBACTEP	USB Active Endpoint. The `USBC_CTL_IEP0_D.USBACTEP` bit is always set (= 1), indicating that control endpoint 0 is active in all configurations and interfaces.	
1:0 (R/W)	MPS	Maximum Packet Size. The `USBC_CTL_IEP0_D.MPS` field indicates the maximum packet size (in bytes) for the current logical endpoint. The field applies to IN and OUT endpoints.	
		0	64 bytes
		1	32 bytes
		2	16 bytes
		3	8 bytes

Device Control IN Endpoint n Control Register

In device mode, the `USBC_CTL_IEP[n]_D` register controls IN endpoint n.

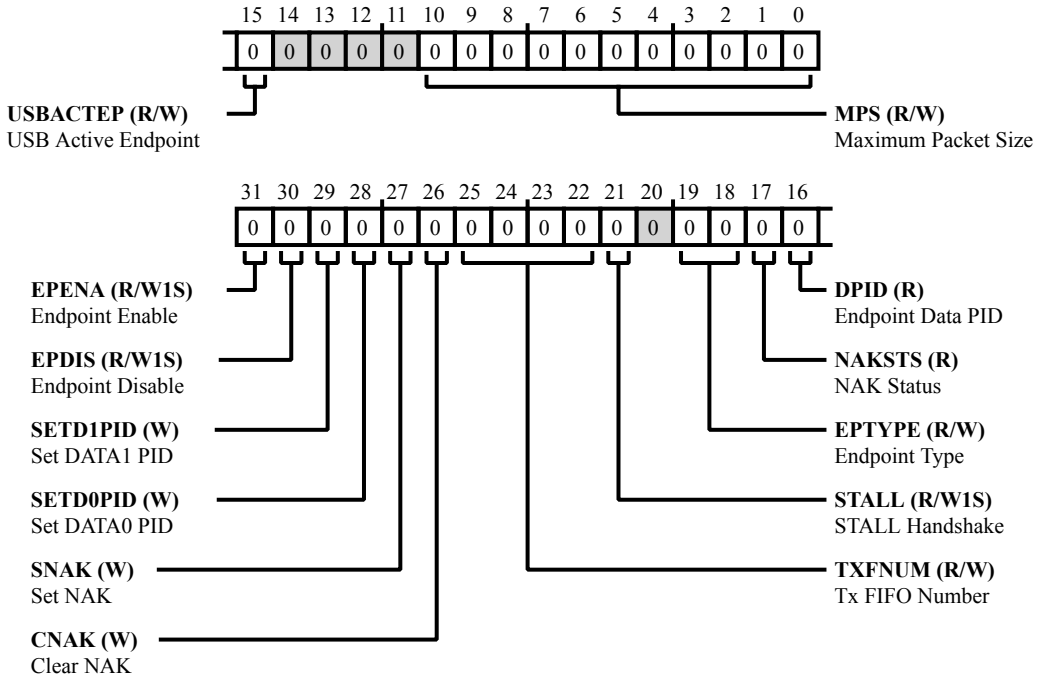


Figure 21-15: USBC_CTL_IEP[n]_D Register Diagram

Table 21-10: USBC_CTL_IEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	EPENA	Endpoint Enable. When scatter/gather DMA mode is enabled for IN endpoints, the `USBC_CTL_IEP[n]_D.EPENA` bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. For OUT endpoints, it indicates that the descriptor structure and data buffer to receive data is set up. When scatter/gather DMA mode is enabled, such as for buffer-pointer based DMA mode: For IN endpoints, the `USBC_CTL_IEP[n]_D.EPENA` bit indicates that data is ready to be transmitted on the endpoint. For OUT endpoints, the `USBC_CTL_IEP[n]_D.EPENA` bit indicates that the application has allocated the memory to start receiving data from the USB. The core clears the `USBC_CTL_IEP[n]_D.EPENA` bit before setting any of the following interrupts on this endpoint: SETUP Phase Done, Endpoint Disabled, Transfer Completed. Note: For control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.

Table 21-10: USBC_CTL_IEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W1S)	EPDIS	Endpoint Disable. The application sets the USBC_CTL_IEP[n]_D.EPDIS bit to stop transmitting data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears the USBC_CTL_IEP[n]_D.EPDIS bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint. The USBC_CTL_IEP[n]_D.EPDIS bit applies to IN and OUT endpoints.
29 (RX/W)	SETD1PID	Set DATA1 PID. The USBC_CTL_IEP[n]_D.SETD1PID bit applies to interrupt and bulk IN and OUT endpoints only.The USBC_CTL_IEP[n]_D.SETD1PID bit applies to both for scatter/gather DMA mode and non scatter/gather DMA mode. Writing to the USBC_CTL_IEP[n]_D.SETD1PID bit sets the Endpoint Data PID (DPID) field in this register to DATA1. In non scatter/gather DMA mode, set odd (micro)frame (SetOddFr). SetOddFr applies to isochronous IN and OUT endpoints only. Writing to the USBC_CTL_IEP[n]_D.SETD1PID bit sets the even and odd (micro)Frame (EO_FrNum) field to odd (micro)frame. The USBC_CTL_IEP[n]_D.SETD1PID bit is not applicable for scatter/gather DMA mode for isochronous endpoints.
28 (RX/W)	SETD0PID	Set DATA0 PID. The USBC_CTL_IEP[n]_D.SETD0PID bit applies to interrupt/bulk IN and OUT endpoints only. A write to the USBC_CTL_IEP[n]_D.SETD0PID bit sets the endpoint data PID (DPID) field in this register to DATA0. The USBC_CTL_IEP[n]_D.SETD0PID bit applies to both scatter/gather DMA mode and non-scatter/gather DMA mode. In non-scatter/gather DMA mode: Set even (micro)frame (SetEvenFr) SetEvenFr applies to isochronous IN and OUT endpoints only. A write to the USBC_CTL_IEP[n]_D.SETD0PID bit sets the even/odd (micro)frame (EO_FrNum) field to even (micro)frame. When scatter/gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.
27 (RX/W)	SNAK	Set NAK. A write to the USBC_CTL_IEP[n]_D.SNAK bit sets the NAK bit for the endpoint. Using the USBC_CTL_IEP[n]_D.SNAK bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.

Table 21-10: USBC_CTL_IEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (RX/W)	CNAK	Clear NAK. A write to the `USBC_CTL_IEP[n]_D.CNAK` bit clears the NAK bit for the endpoint.	
25:22 (R/W)	TXFNUM	Tx FIFO Number. The `USBC_CTL_IEP[n]_D.TXFNUM` field indicates the FIFO number that is assigned to the IN endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	
21 (R/W1S)	STALL	STALL Handshake. The `USBC_CTL_IEP[n]_D.STALL` bit indicates a handshake stall. The application sets (=1) the `USBC_CTL_IEP[n]_D.STALL` bit to stall all tokens from the USB host to this endpoint. If a NAK, global non-periodic IN NAK, or global OUT NAK bit is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. The `USBC_CTL_IEP[n]_D.STALL` bit applies to non-control, non-isochronous IN and OUT endpoints only. The application can only set the `USBC_CTL_IEP[n]_D.STALL` bit, and the core clears it, when a SETUP token is received for this endpoint. Irrespective of the bit configuration, the core always responds to SETUP data packets with an ACK handshake.	
19:18 (R/W)	EPTYPE	Endpoint Type. The `USBC_CTL_IEP[n]_D.EPTYPE` bit indicates the transfer type supported by this logical endpoint.	
		0	Control
		1	Isochronous
		2	Bulk
		3	Interrupt

Table 21-10: USBC_CTL_IEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/NW)	NAKSTS	NAK Status. When the `USBC_CTL_IEP[n]_D.NAKSTS` bit is set, either by the application or core, the core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of the bit configuration, the core always responds to SETUP data packets with an ACK handshake.	
		0	Core is transmitting non-NAK handshakes based on the FIFO status
		1	Core is transmitting NAK handshakes on the endpoint
16 (R/NW)	DPID	Endpoint Data PID. The `USBC_CTL_IEP[n]_D.DPID` field applies to interrupt/bulk IN and OUT endpoints only. The `USBC_CTL_IEP[n]_D.DPID` field indicates the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1 The `USBC_CTL_IEP[n]_D.DPID` field applies to both scatter/gather DMA mode and non-scatter/gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-scatter/gather DMA mode, the `USBC_CTL_IEP[n]_D.DPID` field applies to isochronous IN and OUT endpoints only. The `USBC_CTL_IEP[n]_D.DPID` field indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro)frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro)frame When scatter/gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.	

Table 21-10: USBC_CTL_IEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	USBACTEP	USB Active Endpoint. `USBC_CTL_IEP[n]_D.USBACTEP` bit indicates whether this endpoint is active in the current configuration and interface. The core clears the `USBC_CTL_IEP[n]_D.USBACTEP` bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.
10:0 (R/W)	MPS	Maximum Packet Size. The `USBC_CTL_IEP[n]_D.MPS` field indicates the maximum packet size (in bytes) for the current logical endpoint. This value is in bytes.

Device Control IN Endpoint 0 DMA Address Register

In device mode, the USBC_DMA_ADDR_IEP0_D register indicates the DWORD-aligned start address of the external memory for storing or fetching endpoint data. This register is incremented on every bus transaction.

When scatter/gather DMA mode is not enabled, the USBC_DMA_ADDR_IEP0_D register indicates the start address.

When scatter/gather DMA mode is enabled, the USBC_DMA_ADDR_IEP0_D register indicates the base pointer for the descriptor list.

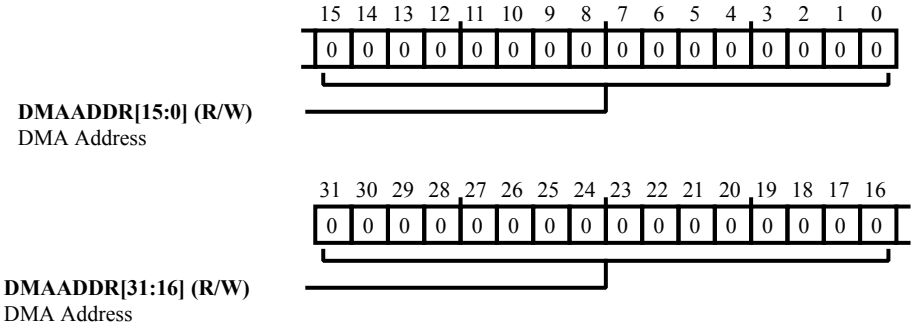


Figure 21-16: USBC_DMA_ADDR_IEP0_D Register Diagram

Table 21-11: USBC_DMA_ADDR_IEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DMAADDR	DMA Address. This USBC_DMA_ADDR_IEP0_D.DMAADDR field holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.

Device IN Endpoint 0 Buffer Address Register

In device mode, the USBC_DMA_BADDR_IEP0_D register indicates the buffer address for IN endpoint 0. It is updated when the data transfer for the endpoint 0 is in progress.

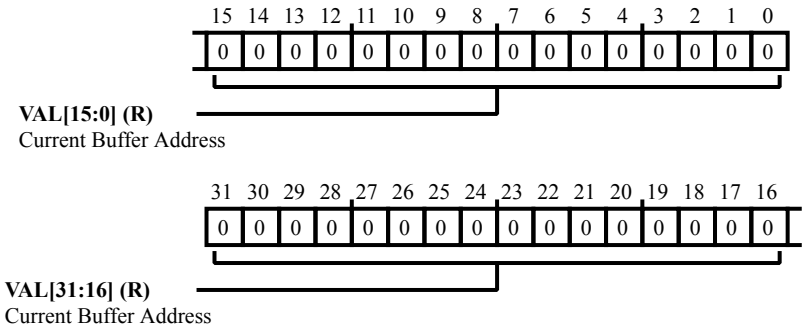


Figure 21-17: USBC_DMA_BADDR_IEP0_D Register Diagram

Table 21-12: USBC_DMA_BADDR_IEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VAL	Current Buffer Address. The USBC_DMA_BADDR_IEP0_D.VAL field holds the current buffer address.

Device Control IN Endpoint n DMA Buffer Address Register

In device mode, the USBC_DMA_BADDR_IEP[n]_D register indicates the buffer address for IN endpoint n. It is updated when the data transfer for the corresponding endpoint is in progress.

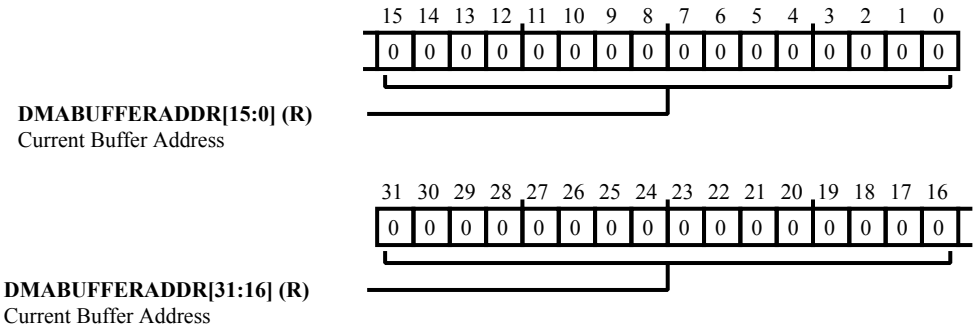


Figure 21-18: USBC_DMA_BADDR_IEP[n]_D Register Diagram

Table 21-13: USBC_DMA_BADDR_IEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DMABUFFERADDR	Current Buffer Address. The USBC_DMA_BADDR_IEP[n]_D.DMABUFFERADDR field holds the current buffer address. This field is present only in scatter/gather DMA mode. Otherwise, this field is reserved.

Device Control IN Endpoint n DMA Address Register

The USBC_DMA_ADDR_IEP[n]_D register indicates the DWORD-aligned start address of the external memory for storing or fetching endpoint data. This register is incremented on every bus transaction.

When scatter/gather DMA mode is not enabled, the USBC_DMA_ADDR_IEP[n]_D register indicates the start address.

When scatter/gather DMA mode is enabled, the USBC_DMA_ADDR_IEP[n]_D register indicates the base pointer for the descriptor list.

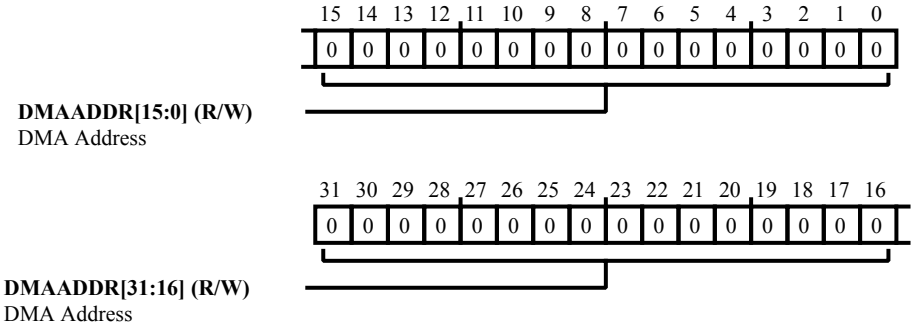


Figure 21-19: USBC_DMA_ADDR_IEP[n]_D Register Diagram

Table 21-14: USBC_DMA_ADDR_IEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DMAADDR	DMA Address. The USBC_DMA_ADDR_IEP[n]_D.DMAADDR field holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.

Device IN Endpoint FIFO Empty Interrupt Mask Register

The USBC_IMSK_IEP_FEMPT_D register is valid only in dedicated FIFO operation (OTG_EN_DED_TX_FIFO = 1). This register is used to control the IN endpoint FIFO empty interrupt generation.

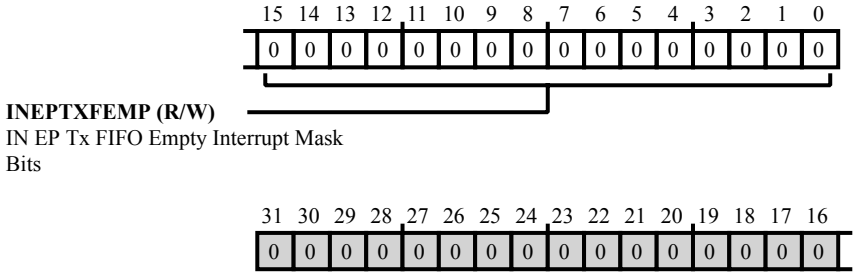


Figure 21-20: USBC_IMSK_IEP_FEMPT_D Register Diagram

Table 21-15: USBC_IMSK_IEP_FEMPT_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	INEPTXFEMP	IN EP Tx FIFO Empty Interrupt Mask Bits. The USBC_IMSK_IEP_FEMPT_D.INEPTXFEMP field acts as mask bits for the TXFEMP interrupt in the USBC_ISTAT_IEP[n]_D register; one bit per IN Endpoint. For example, bit 0 for IN endpoint 0, bit 15 for IN endpoint 15.

Device Control IN Endpoint 0 Interrupt Control Register

The USBC_ISTAT_IEP0_D register indicates the status of an endpoint with respect to USB- and bus-related events. The application must read this register when USBC_ISTAT.OEPINT or USBC_ISTAT.IEPINT is set. Before the application can read this register, it must first read the USBC_ISTAT_D register to get the exact endpoint number for the Device Endpoint n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the USBC_ISTAT_D and USBC_ISTAT registers.

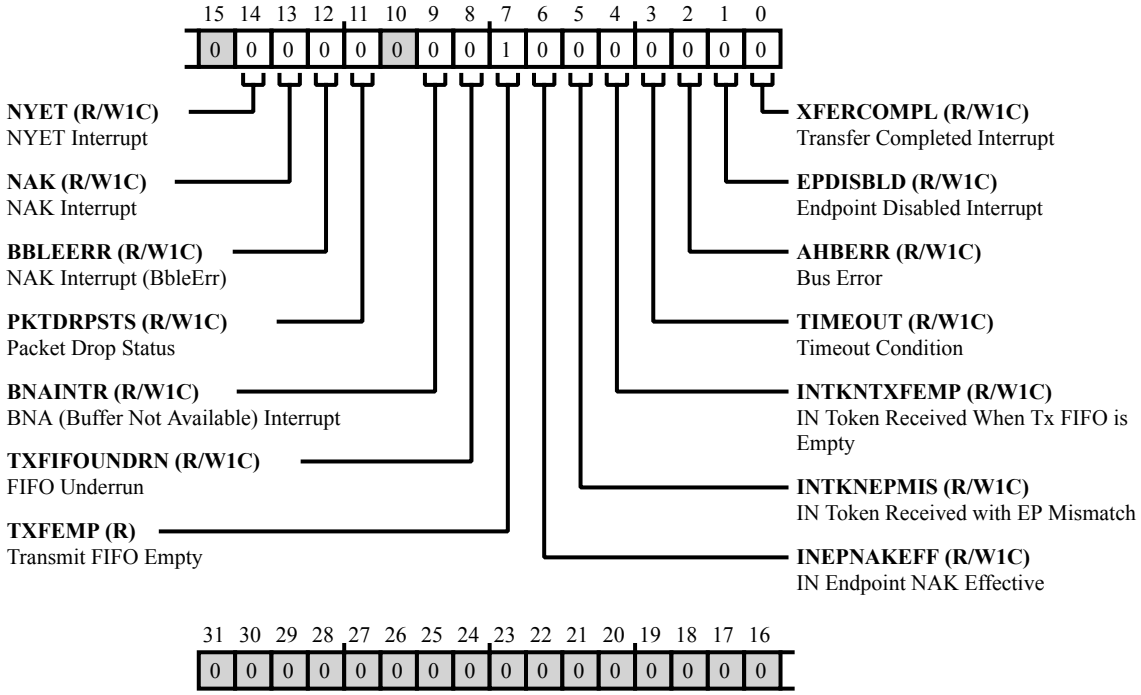


Figure 21-21: USBC_ISTAT_IEP0_D Register Diagram

Table 21-16: USBC_ISTAT_IEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W1C)	NYET	NYET Interrupt. The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
13 (R/W1C)	NAK	NAK Interrupt. The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints, the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.
12 (R/W1C)	BBLEERR	NAK Interrupt (BbleErr). The core generates the NAK interrupt when babble is received for the endpoint.

Table 21-16: USBC_ISTAT_IEP0_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/W1C)	PKTDRPSTS	Packet Drop Status. The USBC_ISTAT_IEP0_D.PKTDRPSTS bit indicates to the application that an ISOC OUT packet has been dropped. The USBC_ISTAT_IEP0_D.PKTDRPSTS bit does not have an associated mask bit and does not generate an interrupt. The USBC_ISTAT_IEP0_D.PKTDRPSTS bit is valid in non scatter/gather DMA mode when periodic transfer interrupt feature is selected.
9 (R/W1C)	BNAINTR	BNA (Buffer Not Available) Interrupt. The core generates this interrupt when the descriptor accessed is not ready for the core to process, such as host busy or DMA done.
8 (R/W1C)	TXFIFOUNDRN	FIFO Underrun. The USBC_ISTAT_IEP0_D.TXFIFOUNDRN bit applies to IN endpoints only. The core generates this interrupt when it detects a transmit FIFO underrun condition in threshold mode for this endpoint.
7 (R/NW)	TXFEMP	Transmit FIFO Empty. This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the USBC_AHB_CFG.NPTXFELVL bit. The USBC_ISTAT_IEP0_D.TXFEMP bit is valid only for IN endpoints.
6 (R/W1C)	INEPNAKEFF	IN Endpoint NAK Effective. The USBC_ISTAT_IEP0_D.INEPNAKEFF bit applies to periodic IN endpoints only. The USBC_ISTAT_IEP0_D.INEPNAKEFF bit can be cleared when the application clears the IN endpoint NAK by writing to the USBC_CTL_IEP[n]_D.CNAK bit. This interrupt indicates that the core has sampled the NAK bit. Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.
5 (R/W1C)	INTKNEPMIS	IN Token Received with EP Mismatch. The USBC_ISTAT_IEP0_D.INTKNEPMIS bit indicates that the data in the top of the non-periodic Tx FIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. The USBC_ISTAT_IEP0_D.INTKNEPMIS bit applies to non-periodic IN endpoints only.

Table 21-16: USBC_ISTAT_IEP0_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W1C)	INTKNTXFEMP	IN Token Received When Tx FIFO is Empty. The USBC_ISTAT_IEP0_D.INTKNTXFEMP bit indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. The USBC_ISTAT_IEP0_D.INTKNTXFEMP bit applies to non-periodic IN endpoints only.
3 (R/W1C)	TIMEOUT	Timeout Condition. The USBC_ISTAT_IEP0_D.TIMEOUT bit indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. In shared TX FIFO mode, the USBC_ISTAT_IEP0_D.TIMEOUT bit applies to non-isochronous IN endpoints only. In dedicated FIFO mode, the USBC_ISTAT_IEP0_D.TIMEOUT bit applies only to control IN endpoints. In scatter/gather DMA mode, the timeout interrupt is not asserted.
2 (R/W1C)	AHBERR	Bus Error. The USBC_ISTAT_IEP0_D.AHBERR bit bus error during a bus read/write. It applies only in internal DMA mode. The application can read the corresponding endpoint DMA address register to get the error address. The USBC_ISTAT_IEP0_D.AHBERR bit applies to IN and OUT endpoints.
1 (R/W1C)	EPDISBLD	Endpoint Disabled Interrupt. The USBC_ISTAT_IEP0_D.EPDISBLD bit indicates that the endpoint is disabled per the application's request. The USBC_ISTAT_IEP0_D.EPDISBLD bit applies to IN and OUT endpoints.
0 (R/W1C)	XFERCOMPL	Transfer Completed Interrupt. The USBC_ISTAT_IEP0_D.XFERCOMPL field applies to IN and OUT endpoints. When scatter/gather DMA mode is enabled: For IN endpoints, the USBC_ISTAT_IEP0_D.XFERCOMPL field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. For OUT endpoints, the USBC_ISTAT_IEP0_D.XFERCOMPL field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is set. When scatter/gather DMA mode is disabled, the USBC_ISTAT_IEP0_D.XFERCOMPL field indicates that the programmed transfer is complete on the bus as well as on the USB, for this endpoint.

Device Control IN Endpoint n Interrupt Control Register

The USBC_ISTAT_IEP[n]_D register indicates the status of an endpoint with respect to USB- and bus-related events. This register exists for endpoint [n] when the USBC_HWCFG.EPDIR bit is 0 or 1 for that endpoint.

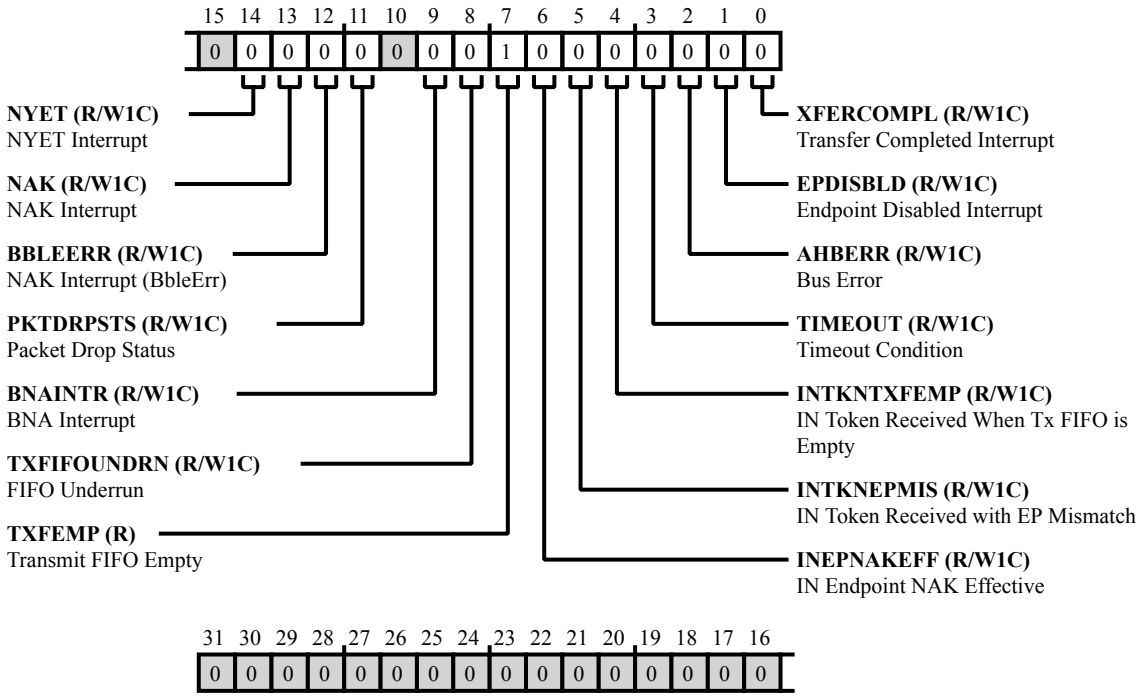


Figure 21-22: USBC_ISTAT_IEP[n]_D Register Diagram

Table 21-17: USBC_ISTAT_IEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W1C)	NYET	NYET Interrupt. The core generates USBC_ISTAT_IEP[n]_D.NYET (NYETInterrupt) when a NYET response is transmitted for a non isochronous OUT endpoint.
13 (R/W1C)	NAK	NAK Interrupt. The core generates the USBC_ISTAT_IEP[n]_D.NAK (NAKInterrupt) when a NAK is transmitted or received by the device. In case of isochronous IN endpoints, the interrupt is generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.
12 (R/W1C)	BBLEERR	NAK Interrupt (BbleErr). The core generates the USBC_ISTAT_IEP[n]_D.BBLEERR interrupt when babble is received for the endpoint.

Table 21-17: USBC_ISTAT_IEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/W1C)	PKTDRPSTS	Packet Drop Status. The `USBC_ISTAT_IEP[n]_D.PKTDRPSTS` bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. The `USBC_ISTAT_IEP[n]_D.PKTDRPSTS` bit is valid in non scatter/gather DMA mode when periodic transfer interrupt feature is selected.
9 (R/W1C)	BNAINTR	BNA Interrupt. The core generates the BNA (Buffer Not Available) interrupt when the descriptor accessed is not ready for the core to process, such as host busy or DMA done.
8 (R/W1C)	TXFIFOUNDRN	FIFO Underrun. The `USBC_ISTAT_IEP[n]_D.TXFIFOUNDRN` bit applies to IN endpoints only. The `USBC_ISTAT_IEP[n]_D.TXFIFOUNDRN` bit is valid only when thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint.
7 (R/NW)	TXFEMP	Transmit FIFO Empty. The `USBC_ISTAT_IEP[n]_D.TXFEMP` bit is valid only for IN endpoints. This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the `USBC_AHB_CFG.NPTXFELVL` bit.
6 (R/W1C)	INEPNAKEFF	IN Endpoint NAK Effective. The `USBC_ISTAT_IEP[n]_D.INEPNAKEFF` bit applies to periodic IN endpoints only. The `USBC_ISTAT_IEP[n]_D.INEPNAKEFF` bit can be cleared when the application clears the IN endpoint NAK by writing to `USBC_CTL_IEP[n]_D.CNAK`. This interrupt indicates that the core has sampled the NAK bit. When set (either by the application or by the core), the interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.
5 (R/W1C)	INTKNEPMIS	IN Token Received with EP Mismatch. The `USBC_ISTAT_IEP[n]_D.INTKNEPMIS` bit applies to non-periodic IN endpoints only. The `USBC_ISTAT_IEP[n]_D.INTKNEPMIS` bit indicates that the data in the top of the non-periodic Tx FIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Table 21-17: USBC_ISTAT_IEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W1C)	INTKNTXFEMP	IN Token Received When Tx FIFO is Empty. The `USBC_ISTAT_IEP[n]_D.INTKNTXFEMP` bit applies to non-periodic IN endpoints only. The `USBC_ISTAT_IEP[n]_D.INTKNTXFEMP` bit indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.
3 (R/W1C)	TIMEOUT	Timeout Condition. The `USBC_ISTAT_IEP[n]_D.TIMEOUT` bit indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. In dedicated FIFO mode, the `USBC_ISTAT_IEP[n]_D.TIMEOUT` bit applies only to control IN endpoints. In scatter/gather DMA mode, the timeout interrupt is not asserted.
2 (R/W1C)	AHBERR	Bus Error. The `USBC_ISTAT_IEP[n]_D.AHBERR` bit applies to IN and OUT endpoints. The `USBC_ISTAT_IEP[n]_D.AHBERR` bit is set (=1) when there is a bus error during a bus read/write. The application can read the corresponding endpoint DMA address register to get the error address.
1 (R/W1C)	EPDISBLD	Endpoint Disabled Interrupt. The `USBC_ISTAT_IEP[n]_D.EPDISBLD` bit applies to IN and OUT endpoints. The `USBC_ISTAT_IEP[n]_D.EPDISBLD` bit indicates that the endpoint is disabled per the application's request.
0 (R/W1C)	XFERCOMPL	Transfer Completed Interrupt. The `USBC_ISTAT_IEP[n]_D.XFERCOMPL` bit applies to IN and OUT endpoints. When scatter/gather DMA mode is enabled: For IN endpoints, the `USBC_ISTAT_IEP[n]_D.XFERCOMPL` bit indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. For OUT endpoints, the `USBC_ISTAT_IEP[n]_D.XFERCOMPL` bit indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is set. When scatter/gather DMA mode is disabled, the `USBC_ISTAT_IEP[n]_D.XFERCOMPL` bit indicates that the programmed transfer is complete on the bus as well as on the USB, for this endpoint.

Device IN Endpoint Common Interrupt Mask Register

The USBC_IMASK_IEP_D register works with each of the Device IN Endpoint Interrupt (USBC_ISTAT_IEP[n]_D) registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the USBC_ISTAT_IEP[n]_D register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

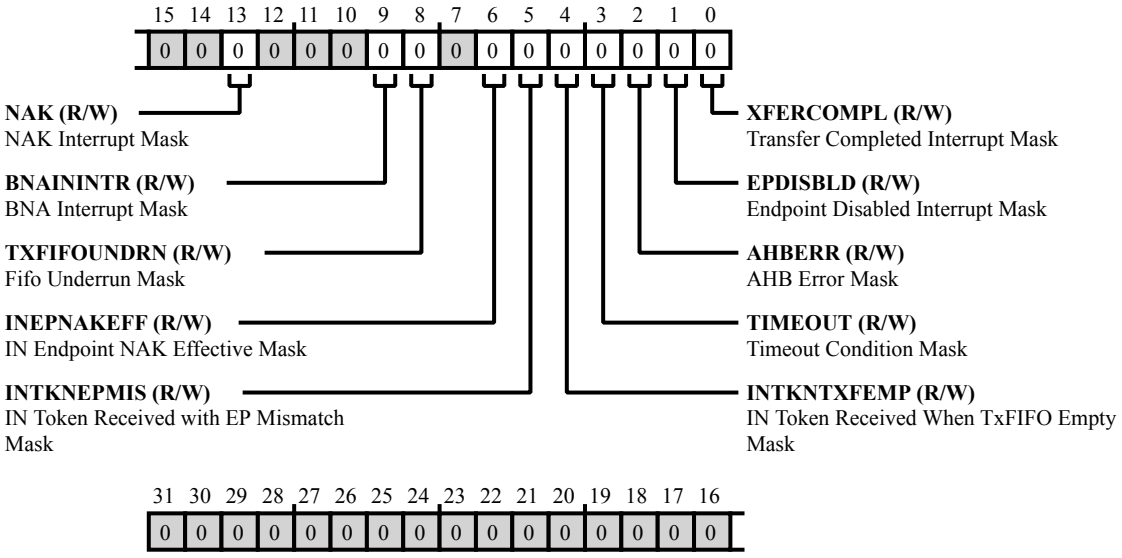


Figure 21-23: USBC_IMASK_IEP_D Register Diagram

Table 21-18: USBC_IMASK_IEP_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	NAK	NAK Interrupt Mask.
9 (R/W)	BNAININTR	BNA Interrupt Mask.
8 (R/W)	TXFIFOUNDRN	Fifo Underrun Mask.
6 (R/W)	INEPNAKEFF	IN Endpoint NAK Effective Mask.
5 (R/W)	INTKNEPMIS	IN Token Received with EP Mismatch Mask.
4 (R/W)	INTKNTXFEMP	IN Token Received When TxFIFO Empty Mask.

Table 21-18: USBC_IMASK_IEP_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W)	TIMEOUT	Timeout Condition Mask. Timeout Condition Mask (TimeOUTMsk) (Non-isochronous endpoints)
2 (R/W)	AHBERR	AHB Error Mask.
1 (R/W)	EPDISBLD	Endpoint Disabled Interrupt Mask.
0 (R/W)	XFERCOMPL	Transfer Completed Interrupt Mask.

Device Control IN Endpoint 0 Transfer Size Register

The application must modify the USBC_TSIZ_IEP0_D register before enabling endpoint 0. Once endpoint 0 is enabled using endpoint enable bit (USBC_CTL_IEP0_D.EPENA /USBC_CTL_OEP0_D.EPENA), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1-15.

When scatter/gather DMA mode is enabled, this register must not be programmed by the application. If the application reads this register when scatter/gather DMA mode is enabled, the core returns all zeros.

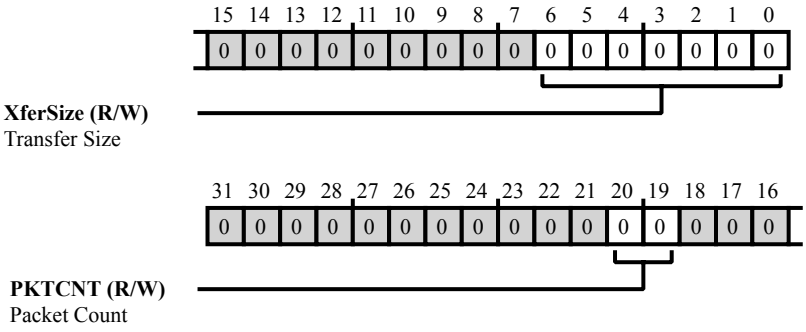


Figure 21-24: USBC_TSIZ_IEP0_D Register Diagram

Table 21-19: USBC_TSIZ_IEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:19 (R/W)	PKTCNT	Packet Count. The USBC_TSIZ_IEP0_D.PKTCNT field indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0. For IN endpoints, the USBC_TSIZ_IEP0_D.PKTCNT field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO. For OUT endpoints, the USBC_TSIZ_IEP0_D.PKTCNT field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.

Table 21-19: USBC_TSIZ_IEP0_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	XFERSIZE	Transfer Size. The `USBC_TSIZ_IEP0_D.XFERSIZE` field contains the transfer size in bytes for the current endpoint. The transfer size (XferSize) = Sum of buffer sizes across all descriptors in the list for the endpoint. In buffer DMA, the core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. For IN endpoints, the core decrements the `USBC_TSIZ_IEP0_D.XFERSIZE` field every time a packet from the external memory is written to the TxFIFO. For OUT endpoints, the core decrements the `USBC_TSIZ_IEP0_D.XFERSIZE` field every time a packet is read from the RxFIFO and written to the external memory.

Device Control IN Endpoint n Transfer Size Register

The USBC_TSIZ_IEP[n]_D register indicates the Device IN Endpoint n transfer size.

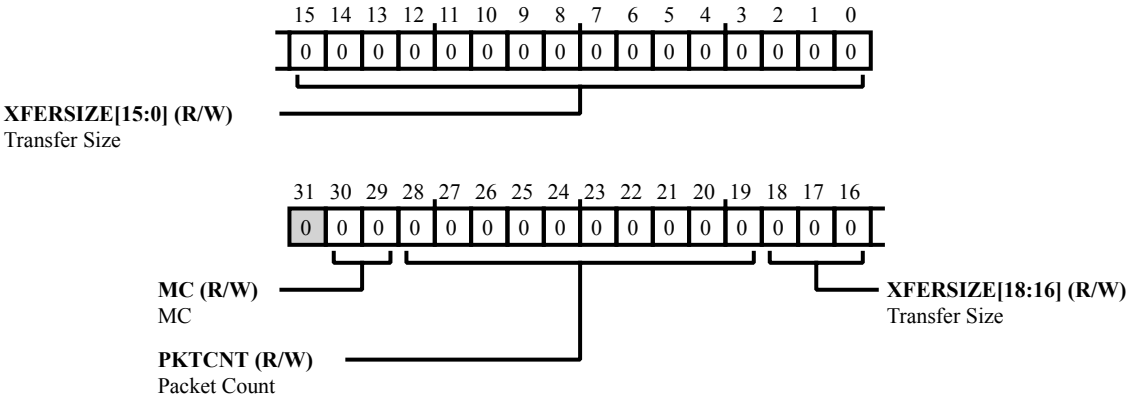


Figure 21-25: USBC_TSIZ_IEP[n]_D Register Diagram

Table 21-20: USBC_TSIZ_IEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:29 (R/W)	MC	MC. For periodic IN endpoints, the USBC_TSIZ_IEP[n]_D.MC field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. For non-periodic IN endpoints, the USBC_TSIZ_IEP[n]_D.MC field is valid only in internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the next endpoint. The USBC_TSIZ_IEP[n]_D.MC field applies to IN endpoints only.	
		0	Reserved
		1	1 packet
		2	2 packets
		3	3 packets
28:19 (R/W)	PKTCNT	Packet Count. The USBC_TSIZ_IEP[n]_D.PKTCNT field indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0. The USBC_TSIZ_IEP[n]_D.PKTCNT field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.	

Table 21-20: USBC_TSIZ_IEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
18:0 (R/W)	XFERSIZE	Transfer Size. The `USBC_TSIZ_IEP[n]_D.XFERSIZE` field indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be configured to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

Device IN Endpoint 1 Transmit FIFO Size Register

The USBC_TXFIFOSZ1_IEP_D register holds the size and memory start address of IN endpoint Tx FIFOs implemented in device mode. Each FIFO holds the data for one IN endpoint. The USBC_TXFIFOSZ1_IEP_D register is repeated for instantiated IN endpoint FIFOs 1 to 3.

For IN endpoint FIFO 0, use the USBC_TXFIFOSZ_PER_H register for programming the size and memory start address.

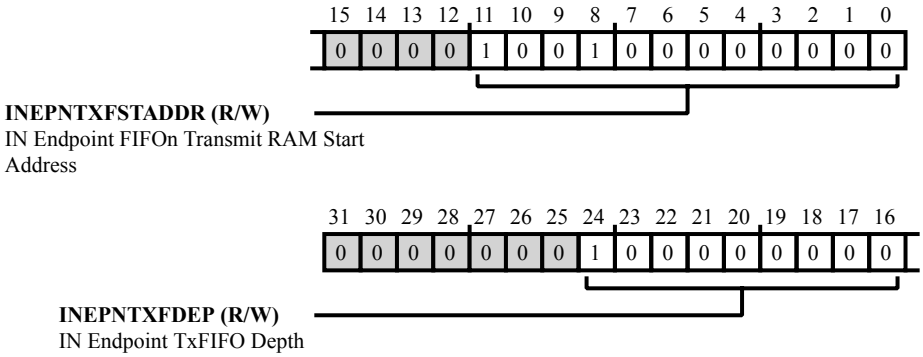


Figure 21-26: USBC_TXFIFOSZ1_IEP_D Register Diagram

Table 21-21: USBC_TXFIFOSZ1_IEP_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24:16 (R/W)	INEPNTXFDEP	IN Endpoint TxFIFO Depth. The USBC_TXFIFOSZ1_IEP_D.INEPNTXFDEP field indicates the IN endpoint TxFIFO depth. The value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 Programmed values must not exceed the power-on value.
11:0 (R/W)	INEPNTXFSTADDR	IN Endpoint FIFOn Transmit RAM Start Address. The USBC_TXFIFOSZ1_IEP_D.INEPNTXFSTADDR field contains the memory start address for IN endpoint transmit FIFOn (0<n< = 3). The power-on reset value of the USBC_TXFIFOSZ1_IEP_D register is specified as the largest Rx data FIFO depth.

Device IN Endpoint 2 Transmit FIFO Size Register

The USBC_TXFIFOSZ2_IEP_D register is valid only in dedicated FIFO mode (OTG_EN_DED_TX_FIFO=1). It holds the size and memory start address of IN endpoint TxFIFOs implemented in device mode. Each FIFO holds the data for one IN endpoint. The USBC_TXFIFOSZ2_IEP_D register is repeated for instantiated IN endpoint FIFOs 1 to 15. For IN endpoint FIFO 0, use the USBC_TXFIFOSZ_NP register for programming the size and memory start address.

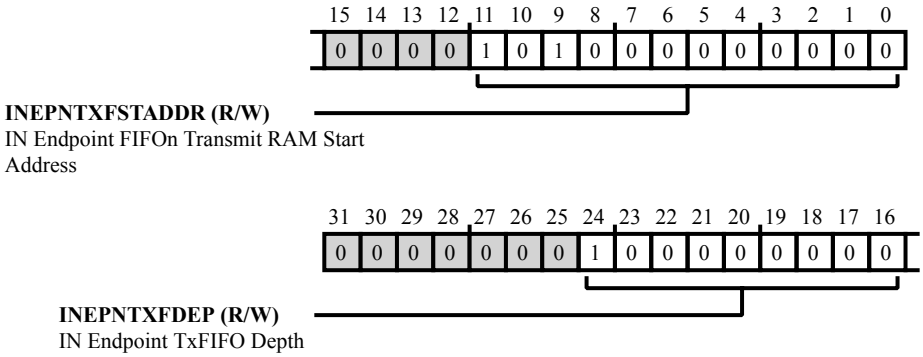


Figure 21-27: USBC_TXFIFOSZ2_IEP_D Register Diagram

Table 21-22: USBC_TXFIFOSZ2_IEP_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24:16 (R/W)	INEPNTXFDEP	IN Endpoint TxFIFO Depth. The USBC_TXFIFOSZ2_IEP_D.INEPNTXFDEP field indicates the IN endpoint TXFIFO depth. The value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of the USBC_TXFIFOSZ2_IEP_D register is specified as the largest IN endpoint FIFO number depth Programmed values must not exceed the power-on value.

Table 21-22: USBC_TXFIFOSZ2_IEP_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	INEPNTXFSTADDR	IN Endpoint FIFOn Transmit RAM Start Address. This `USBC_TXFIFOSZ2_IEP_D.INEPNTXFSTADDR` field contains the memory start address for IN endpoint transmit FIFOn (0<n< = 15). The power-on reset value of the `USBC_TXFIFOSZ2_IEP_D` register is specified as the largest Rx data FIFO depth. The power-on reset value of the register is calculated according to the following formula: OTG_RX_DFIFO_DEPTH + SUM of OTG_TX_DINEP_DFIFO_DEPTH_'i' (where x = 0 to n 1) If at POR the calculated value (C) exceeds 65535, then the reset value becomes reset value(A) = (C 65536). For example: If the start address of IN endpoint FIFO 1 is OTG_RX_DFIFO_DEPTH + OTG_TX_DINEP_DFIFO_DEPTH_ 0 and the start address of IN endpoint FIFO 2 is OTG_RX_DFIFO_DEPTH + OTG_TX_DINEP_DFIFO_DEPTH_ 0 + OTG_TX_DINEP_DFIFO_DEPTH_ 1. Programmed values must not exceed the power-on value.

Device IN Endpoint 3 Transmit FIFO Size Register

The USBC_TXFIFOSZ3_IEP_D register is valid only in dedicated FIFO mode (OTG_EN_DED_TX_FIFO=1). It holds the size and memory start address of IN endpoint TxFIFOs implemented in device mode. Each FIFO holds the data for one IN endpoint. The USBC_TXFIFOSZ3_IEP_D register is repeated for instantiated IN endpoint FIFOs 1 to 15. For IN endpoint FIFO 0, use the USBC_TXFIFOSZ_NP register for programming the size and memory start address.

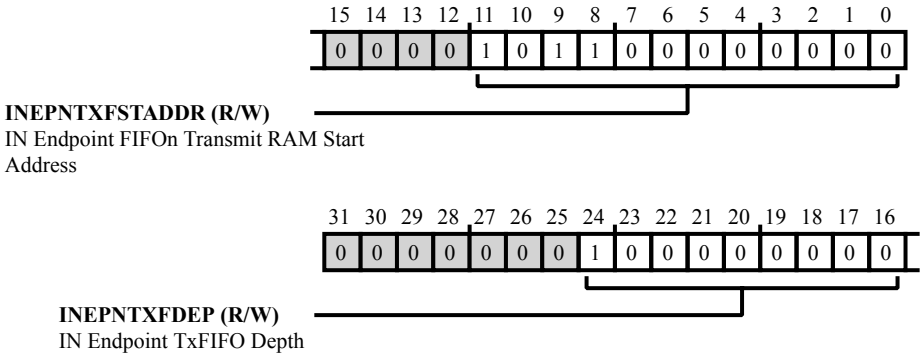


Figure 21-28: USBC_TXFIFOSZ3_IEP_D Register Diagram

Table 21-23: USBC_TXFIFOSZ3_IEP_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24:16 (R/W)	INEPNTXFDEP	IN Endpoint TxFIFO Depth. The USBC_TXFIFOSZ3_IEP_D.INEPNTXFDEP field indicates the IN endpoint TXFIFO depth. The value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of the USBC_TXFIFOSZ3_IEP_D register is specified as the largest IN endpoint FIFO number depth. Programmed values must not exceed the power-on value.

Table 21-23: USBC_TXFIFOSZ3_IEP_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	INEPNTXFSTADDR	IN Endpoint FIFOn Transmit RAM Start Address. The `USBC_TXFIFOSZ3_IEP_D.INEPNTXFSTADDR` field contains the memory start address for IN endpoint transmit FIFOn (0<n< = 15). The power-on reset value of the `USBC_TXFIFOSZ3_IEP_D` register is specified as the largest Rx data FIFO depth. The power-on reset value of the register is calculated according to the following formula: OTG_RX_DFIFO_DEPTH + SUM of OTG_TX_DINEP_DFIFO_DEPTH_'i' (where x = 0 to n 1) If at POR the calculated value (C) exceeds 65535, then the reset value becomes reset value(A) = (C 65536). For example: If the start address of IN endpoint FIFO 1 is OTG_RX_DFIFO_DEPTH + OTG_TX_DINEP_DFIFO_DEPTH_ 0 and the start address of IN endpoint FIFO 2 is OTG_RX_DFIFO_DEPTH + OTG_TX_DINEP_DFIFO_DEPTH_ 0 + OTG_TX_DINEP_DFIFO_DEPTH_ 1. If Enable Dynamic FIFO Sizing is deselected in coreConsultant (OTG_DFIFO_DYNAMIC = 0), this field is read-only and read value is the power-on reset value. Programmed values must not exceed the power-on value.

Device OUT Endpoint 0 Control Register

In device mode, the USBC_CTL_OEP0_D register controls OUT endpoint 0.

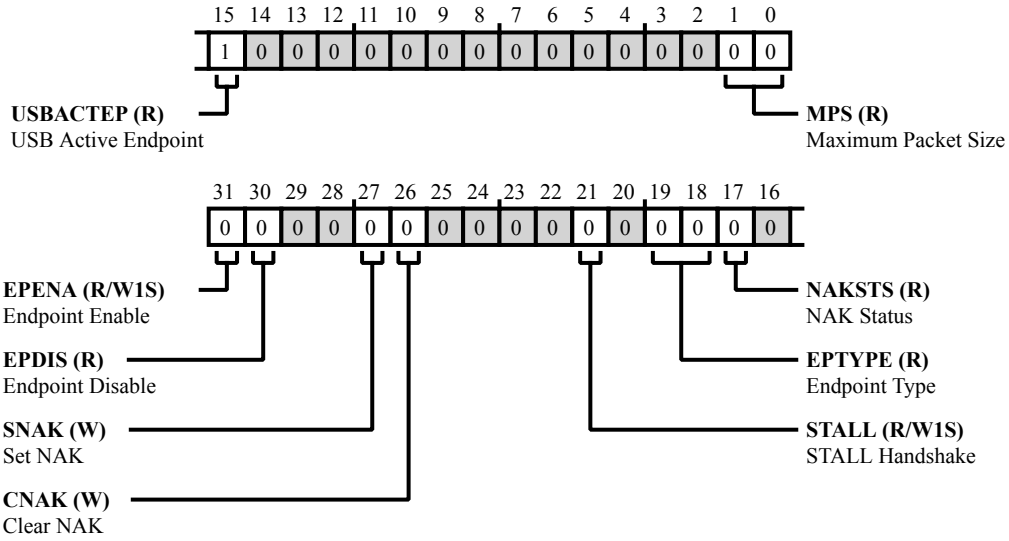


Figure 21-29: USBC_CTL_OEP0_D Register Diagram

Table 21-24: USBC_CTL_OEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	EPENA	Endpoint Enable. When Scatter/Gather DMA mode is enabled, for OUT endpoints, the USBC_CTL_OEP0_D.EPENA bit indicates that the descriptor structure and data buffer to receive data is set up. When Scatter/Gather DMA mode is disabled (such as for buffer-pointer based DMA mode), the USBC_CTL_OEP0_D.EPENA bit indicates that the application has allocated the memory to start receiving data from the USB. The core clears the USBC_CTL_OEP0_D.EPENA bit before setting any of the following interrupts on this endpoint: SETUP Phase Done Endpoint Disabled Transfer Completed Note: In DMA mode, the USBC_CTL_OEP0_D.EPENA bit must be set for the core to transfer SETUP data packets into memory.
30 (R/NW)	EPDIS	Endpoint Disable. The application cannot disable control OUT endpoint 0.

Table 21-24: USBC_CTL_OEP0_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (RX/W)	SNAK	Set NAK. A write to the USBC_CTL_OEP0_D.SNAK bit sets the NAK bit for the endpoint. Using the USBC_CTL_OEP0_D.SNAK bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set bit on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.	
26 (RX/W)	CNAK	Clear NAK. A write to the USBC_CTL_OEP0_D.CNAK bit clears the NAK bit for the endpoint.	
21 (R/W1S)	STALL	STALL Handshake. The application can only set the USBC_CTL_OEP0_D.STALL bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is Set along with the USBC_CTL_OEP0_D.STALL bit, the STALL bit takes priority. Irrespective of the configuration of the USBC_CTL_OEP0_D.STALL bit, the core always responds to SETUP data packets with an ACK handshake.	
19:18 (R/NW)	EPTYPE	Endpoint Type. The USBC_CTL_OEP0_D.EPTYPE field is hardcoded to 2'b00 for control.	
17 (R/NW)	NAKSTS	NAK Status. The USBC_CTL_OEP0_D.NAKSTS bit indicates whether the core is transmitting non-NAK handshakes based on the FIFO status or transmitting NAK handshakes on this endpoint. When either the application or the core sets the USBC_CTL_OEP0_D.NAKSTS bit, the core stops receiving data, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's configuration, the core always responds to SETUP data packets with an ACK handshake.	
		0	Transmit non-NAK handshakes
		1	Transmit NAK handshakes
15 (R/NW)	USBACTEP	USB Active Endpoint. The USBC_CTL_OEP0_D.USBACTEP bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.	
1:0 (R/NW)	MPS	Maximum Packet Size. The USBC_CTL_OEP0_D.MPS field indicates the maximum packet size for control OUT endpoint 0. It is the same size that is programmed in control IN Endpoint 0.	
		0	64 Bytes
		1	32 Bytes
		2	16 Bytes
		3	8 Bytes

Device OUT Endpoint n Control Register

In device mode, the `USBC_CTL_OEP[n]_D` register controls OUT endpoint n.

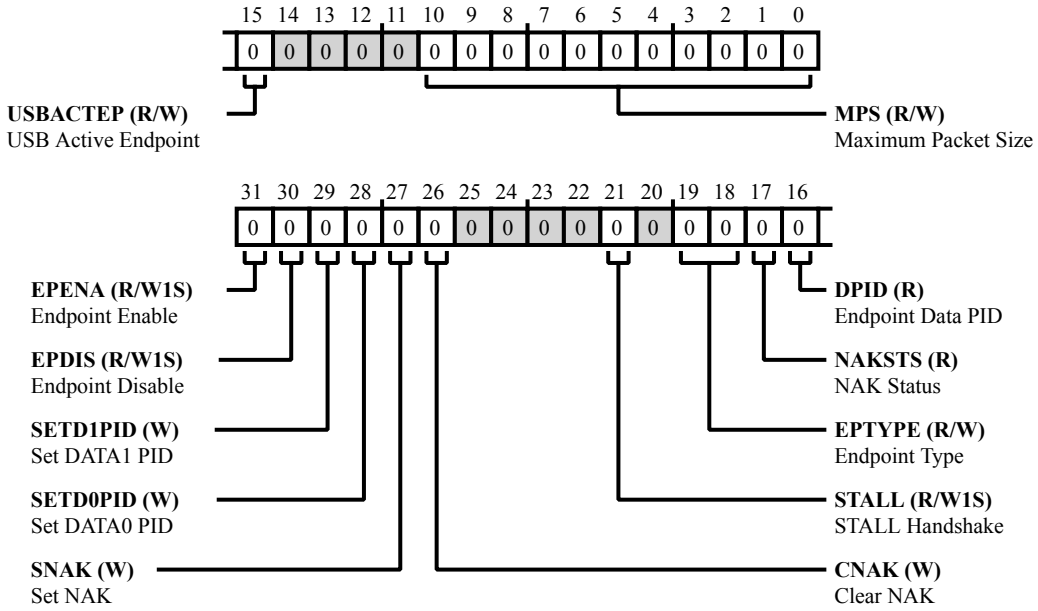


Figure 21-30: USBC_CTL_OEP[n]_D Register Diagram

Table 21-25: USBC_CTL_OEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	EPENA	Endpoint Enable. When Scatter/Gather DMA mode is enabled: For IN endpoints, the USBC_CTL_OEP[n]_D.EPENA bit indicates that the descriptor structure and data buffer with data ready to transmit is configured. For OUT endpoints, the USBC_CTL_OEP[n]_D.EPENA bit indicates that the descriptor structure and data buffer to receive data is configured. When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: For IN endpoints, the USBC_CTL_OEP[n]_D.EPENA bit indicates that data is ready to be transmitted on the endpoint. For OUT endpoints, the USBC_CTL_OEP[n]_D.EPENA bit indicates that the application has allocated the memory to start receiving data from the USB. The core clears the USBC_CTL_OEP[n]_D.EPENA bit before setting any of the following interrupts on this endpoint: SETUP Phase Done Endpoint Disabled Transfer Completed Applies to IN and OUT endpoints. Note: For control endpoints in DMA mode, the USBC_CTL_OEP[n]_D.EPENA bit must be configured to be able to transfer SETUP data packets in memory.
30 (R/W1S)	EPDIS	Endpoint Disable. The application sets the USBC_CTL_OEP[n]_D.EPDIS bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears the USBC_CTL_OEP[n]_D.EPDIS bit before setting the Endpoint Disabled interrupt. The application must set the USBC_CTL_OEP[n]_D.EPDIS bit only if Endpoint Enable is already set for this endpoint. Applies to IN and OUT endpoints.
29 (RX/W)	SETD1PID	Set DATA1 PID. In non scatter-gather DMA mode, the bit sets the odd (micro)frame (SetOddFr). Writing to the bit sets the even and odd (micro)frame (EO_FrNum) field to odd (micro)frame. Applies to isochronous IN and OUT endpoints only.

Table 21-25: USBC_CTL_OEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
28 (RX/W)	SETD0PID	Set DATA0 PID. A write to the USBC_CTL_OEP[n]_D.SETD0PID bit sets the endpoint data PID (DPID) field in this register to DATA0. The USBC_CTL_OEP[n]_D.SETD0PID bit applies to both scatter/gather DMA mode and non-scatter/gather DMA mode. In non-scatter/gather DMA mode: Set even (micro)frame (SetEvenFr). The SetEvenFr applies to isochronous IN and OUT endpoints only. A write to the USBC_CTL_OEP[n]_D.SETD0PID bit sets the even/odd (micro)frame (EO_FrNum) field to even (micro)frame. When scatter/gather DMA mode is enabled, the USBC_CTL_OEP[n]_D.SETD0PID bit is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. The USBC_CTL_OEP[n]_D.SETD0PID bit applies to interrupt/bulk IN and OUT endpoints only.
27 (RX/W)	SNAK	Set NAK. A write to the USBC_CTL_OEP[n]_D.SNAK bit sets the NAK bit for the endpoint. Using the USBC_CTL_OEP[n]_D.SNAK bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set the USBC_CTL_OEP[n]_D.SNAK bit for an endpoint after a SETUP packet is received on that endpoint.
26 (RX/W)	CNAK	Clear NAK. A write to the USBC_CTL_OEP[n]_D.CNAK bit clears the NAK bit for the endpoint.
21 (R/W1S)	STALL	STALL Handshake. For non-control, non-isochronous IN and OUT endpoints: The application sets the USBC_CTL_OEP[n]_D.STALL bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear the USBC_CTL_OEP[n]_D.STALL bit, never the core. For control endpoints: The application can only set the USBC_CTL_OEP[n]_D.STALL bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's configuration, the core always responds to SETUP data packets with an ACK handshake.

Table 21-25: USBC_CTL_OEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19:18 (R/W)	EPTYPE	Endpoint Type. The `USBC_CTL_OEP[n]_D.EPTYPE` field indicates the transfer type supported by this logical endpoint.	
		0	Control
		1	Isochronous
		2	Bulk
		3	Interrupt
17 (R/NW)	NAKSTS	NAK Status. When the `USBC_CTL_OEP[n]_D.NAKSTS` bit is set, either by the application or core, the core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the Tx FIFO. For isochronous IN endpoints: The core sends out a zero length data packet, even if there data is available in the TxFIFO. Irrespective of the bit configuration, the core always responds to SETUP data packets with an ACK handshake.	
		0	Transmit non NAK handshakes
		1	Transmit NAK handshakes

Table 21-25: USBC_CTL_OEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/NW)	DPID	Endpoint Data PID. The USBC_CTL_OEP[n]_D.DPID bit applies to interrupt/bulk IN and OUT endpoints only. The USBC_CTL_OEP[n]_D.DPID bit indicates the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID bits of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1 The USBC_CTL_OEP[n]_D.DPID bit applies to both scatter/gather DMA mode and non-scatter/gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-scatter/gather DMA mode, the USBC_CTL_OEP[n]_D.DPID bit applies to isochronous IN and OUT endpoints only. The USBC_CTL_OEP[n]_D.DPID bit indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro)frame number in which it intends to transmit/ receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro)frame When scatter/gather DMA mode is enabled, the USBC_CTL_OEP[n]_D.DPID bit is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.
15 (R/W)	USBACTEP	USB Active Endpoint. The USBC_CTL_OEP[n]_D.USBACTEP bit indicates whether this endpoint is active in the current configuration and interface. The core clears the USBC_CTL_OEP[n]_D.USBACTEP bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.
10:0 (R/W)	MPS	Maximum Packet Size. The USBC_CTL_OEP[n]_D.MPS field indicates the maximum packet size for the current logical endpoint. This value is in bytes.

Device OUT Endpoint 0 DMA Address Register

In device mode, the USBC_DMA_ADDR_OEP0_D register indicates the DWORD-aligned start address of the external memory for storing or fetching endpoint data. This register is incremented on every bus transaction.

When scatter/gather DMA mode is not enabled, the USBC_DMA_ADDR_OEP0_D register indicates the start address.

When scatter/gather DMA mode is enabled, the USBC_DMA_ADDR_OEP0_D register indicates the base pointer for the descriptor list.

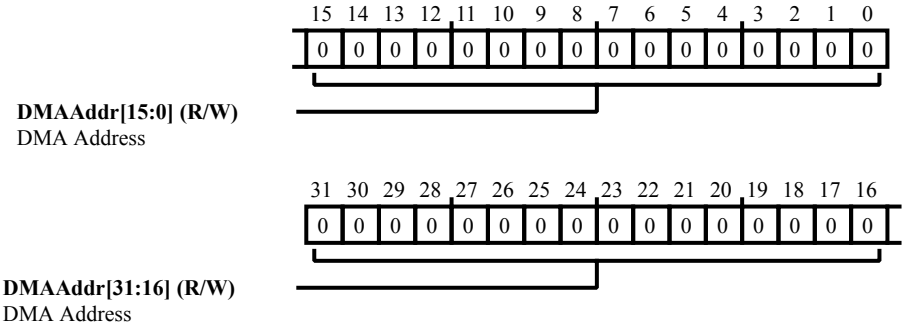


Figure 21-31: USBC_DMA_ADDR_OEP0_D Register Diagram

Table 21-26: USBC_DMA_ADDR_OEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DMAADDR	DMA Address. The USBC_DMA_ADDR_OEP0_D.DMAADDR field holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to- back, the SETUP data packet in the memory is overwritten.

Device OUT Endpoint 0 Buffer Address Register

In device mode, the USBC_DMA_BADDR_OEP0_D register indicates the buffer address for OUT endpoint 0. It is updated when the data transfer for the endpoint 0 is in progress.

The USBC_DMA_BADDR_OEP0_D register is available only in scatter/gather DMA mode and is otherwise reserved.

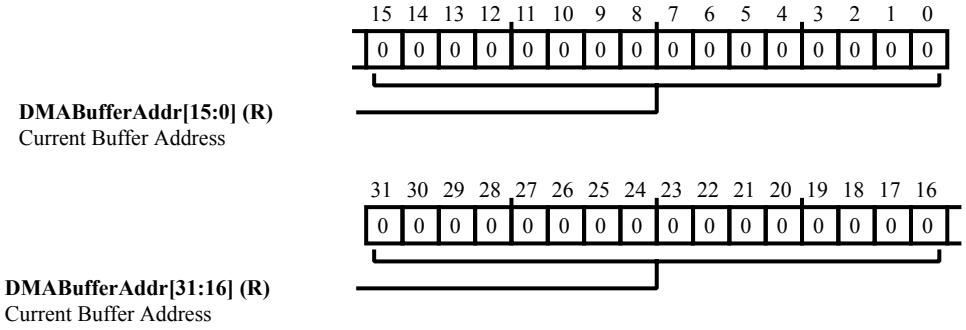


Figure 21-32: USBC_DMA_BADDR_OEP0_D Register Diagram

Table 21-27: USBC_DMA_BADDR_OEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DMABUFFERADDR	Current Buffer Address. The USBC_DMA_BADDR_OEP0_D.DMABUFFERADDR bit filed contains the current buffer address.

Device OUT Endpoint n Buffer Address Register

In device mode, the USBC_DMA_BADDR_OEP[n]_D register indicates the buffer address for OUT endpoint n. It is updated when the data transfer for the corresponding endpoint is in progress.

The USBC_DMA_BADDR_OEP[n]_D register is available only in scatter/gather DMA mode and is otherwise reserved.

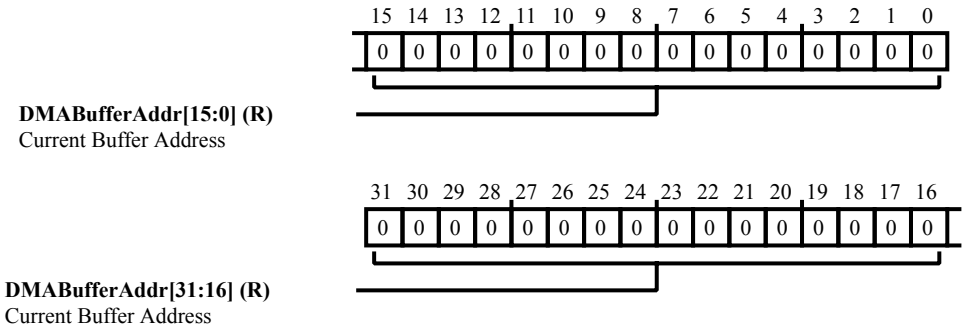


Figure 21-33: USBC_DMA_BADDR_OEP[n]_D Register Diagram

Table 21-28: USBC_DMA_BADDR_OEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DMABUFFERADDR	Current Buffer Address. The USBC_DMA_BADDR_OEP[n]_D.DMABUFFERADDR bit filed contains the current buffer address.

Device OUT Endpoint n DMA Address Register

The USBC_DMA_ADDR_OEP[n]_D register indicates the DWORD-aligned address. This register is incremented on every bus transaction.

When scatter/gather DMA mode is not enabled, the USBC_DMA_ADDR_OEP[n]_D register indicates the start address.

When scatter/gather DMA mode is enabled, the USBC_DMA_ADDR_OEP[n]_D register indicates the base pointer for the descriptor list.

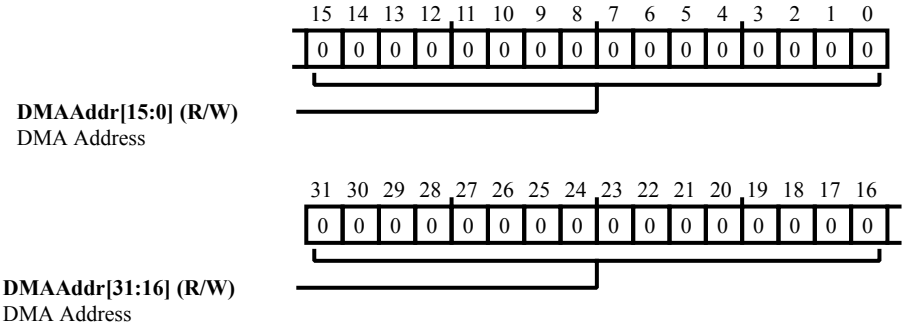


Figure 21-34: USBC_DMA_ADDR_OEP[n]_D Register Diagram

Table 21-29: USBC_DMA_ADDR_OEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DMAADDR	DMA Address. The USBC_DMA_ADDR_OEP[n]_D.DMAADDR field holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to- back, the SETUP data packet in the memory is overwritten.

Device OUT Endpoint 0 Interrupt Register

The USBC_ISTAT_OEP0_D register is the device OUT endpoint 0 interrupt register.

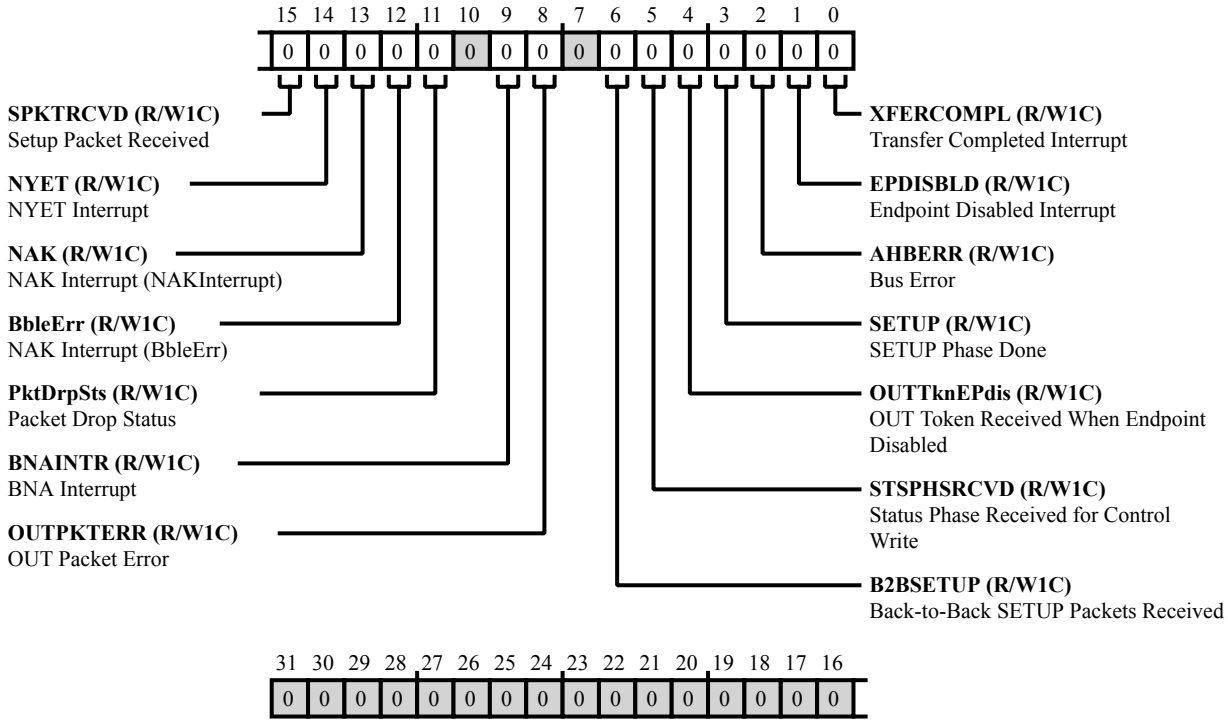


Figure 21-35: USBC_ISTAT_OEP0_D Register Diagram

Table 21-30: USBC_ISTAT_OEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	SPKTRCVD	Setup Packet Received. The USBC_ISTAT_OEP0_D.SPKTRCVD bit applies to control OUT endpoints in only in the buffer DMA mode. Set by the controller, the USBC_ISTAT_OEP0_D.SPKTRCVD bit indicates that this buffer holds 8 bytes of setup data. There is only one SETUP packet per buffer. On receiving a SETUP packet, the controller closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the control transfer and reprogram the buffer start address. Note: Because of the above behavior, the controller can receive any number of back to back setup packets and one buffer for every SETUP packet is used.	
		0	No SETUP packet received
		1	SETUP packet received

Table 21-30: USBC_ISTAT_OEP0_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W1C)	NYET	NYET Interrupt. The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
13 (R/W1C)	NAK	NAK Interrupt (NAKInterrupt). The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.
12 (R/W1C)	BBLEERR	NAK Interrupt (BbleErr). The core generates this interrupt when babble is received for the endpoint.
11 (R/W1C)	PKTDRPSTS	Packet Drop Status. The `USBC_ISTAT_OEP0_D.PKTDRPSTS` bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. The `USBC_ISTAT_OEP0_D.PKTDRPSTS` bit is valid in non scatter/gather DMA mode when periodic transfer interrupt feature is selected.
9 (R/W1C)	BNAINTR	BNA Interrupt. The `USBC_ISTAT_OEP0_D.BNAINTR` bit is valid only when scatter/gather DMA mode is enabled. The core generates the BNA (Buffer Not Available) interrupt when the descriptor accessed is not ready for the core to process, such as host busy or DMA done.
8 (R/W1C)	OUTPKTERR	OUT Packet Error. The `USBC_ISTAT_OEP0_D.OUTPKTERR` bit applies to OUT endpoints only. This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet.
6 (R/W1C)	B2BSETUP	Back-to-Back SETUP Packets Received. The `USBC_ISTAT_OEP0_D.B2BSETUP` bit applies to control OUT endpoints only. The `USBC_ISTAT_OEP0_D.B2BSETUP` bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.

Table 21-30: USBC_ISTAT_OEP0_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W1C)	STSPHSRCVD	Status Phase Received for Control Write. This interrupt is valid only for control OUT endpoints and only in scatter/gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase. This is applicable only in case of scatter/gather DMA mode.
4 (R/W1C)	OUTTKNEPDIS	OUT Token Received When Endpoint Disabled. The `USBC_ISTAT_OEP0_D.OUTTKNEPDIS` bit applies only to control OUT endpoints. The `USBC_ISTAT_OEP0_D.OUTTKNEPDIS` bit indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.
3 (R/W1C)	SETUP	SETUP Phase Done. The `USBC_ISTAT_OEP0_D.SETUP` bit applies to control OUT endpoints only. The `USBC_ISTAT_OEP0_D.SETUP` bit indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.
2 (R/W1C)	AHBERR	Bus Error. The `USBC_ISTAT_OEP0_D.AHBERR` bit applies to IN and OUT endpoints. The `USBC_ISTAT_OEP0_D.AHBERR` bit is generated only in internal DMA mode when there is a bus error during an bus read/write. The application can read the corresponding endpoint DMA address register to get the error address.
1 (R/W1C)	EPDISBLD	Endpoint Disabled Interrupt. The `USBC_ISTAT_OEP0_D.EPDISBLD` bit applies to IN and OUT endpoints. The `USBC_ISTAT_OEP0_D.EPDISBLD` bit indicates that the endpoint is disabled per the application's request.

Table 21-30: USBC_ISTAT_OEP0_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W1C)	XFERCOMPL	Transfer Completed Interrupt. The `USBC_ISTAT_OEP0_D.XFERCOMPL` bit applies to IN and OUT endpoints. When scatter/gather DMA mode is enabled: For IN endpoints, the `USBC_ISTAT_OEP0_D.XFERCOMPL` bit indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. For OUT endpoints, the `USBC_ISTAT_OEP0_D.XFERCOMPL` bit indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is set. Note: In DMA mode, this bit must be set for the core to transfer SETUP data packets into memory. When scatter/gather DMA mode is disabled, the bit indicates that the programmed transfer is complete on the bus as well as on the USB, for this endpoint.

Device OUT Endpoint n Interrupt Register

The `USBC_ISTAT_OEP[n]_D` register is the device OUT endpoint n interrupt register.

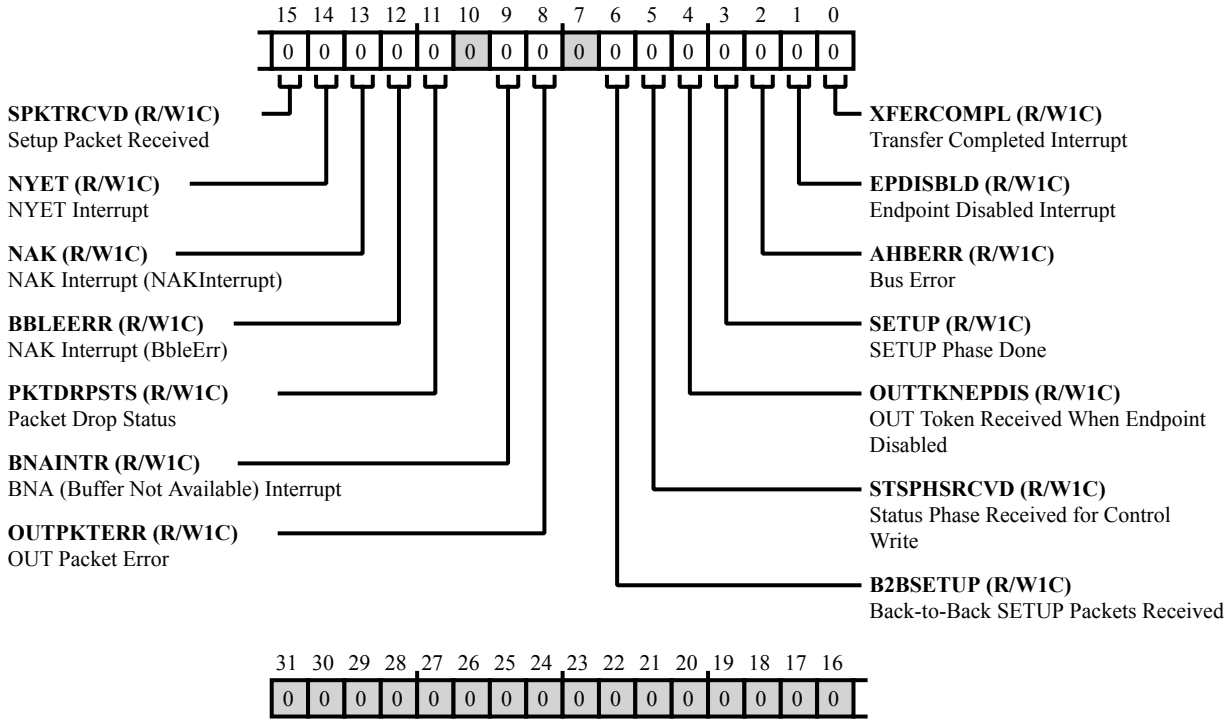


Figure 21-36: USBC_ISTAT_OEP[n]_D Register Diagram

Table 21-31: USBC_ISTAT_OEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	SPKTRCVD	Setup Packet Received. The `USBC_ISTAT_OEP[n]_D.SPKTRCVD` bit applies to control OUT endpoints in only in the buffer DMA mode. Set by the controller, the `USBC_ISTAT_OEP[n]_D.SPKTRCVD` bit indicates whether the SETUP packet has been received. The buffer holds 8 bytes of SETUP data. There is only one SETUP packet per buffer. On receiving a SETUP packet, the controller closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the control transfer and reprogram the buffer start address. Note: Because of the above behavior, the controller can receive any number of back-to-back SETUP packets and one buffer for every SETUP packet is used.	
		0	No SETUP packet received
		1	SETUP Packet Received

Table 21-31: USBC_ISTAT_OEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W1C)	NYET	NYET Interrupt. The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
13 (R/W1C)	NAK	NAK Interrupt (NAKInterrupt). The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.
12 (R/W1C)	BBLEERR	NAK Interrupt (BbleErr). The core generates this interrupt when babble is received for the endpoint.
11 (R/W1C)	PKTDRPSTS	Packet Drop Status. The `USBC_ISTAT_OEP[n]_D.PKTDRPSTS` bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. The `USBC_ISTAT_OEP[n]_D.PKTDRPSTS` bit is valid in non scatter/gather DMA mode when periodic transfer interrupt feature is selected.
9 (R/W1C)	BNAINTR	BNA (Buffer Not Available) Interrupt. The `USBC_ISTAT_OEP[n]_D.BNAINTR` bit is valid only when scatter/gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the core to process, such as host busy or DMA done.
8 (R/W1C)	OUTPKTERR	OUT Packet Error. The `USBC_ISTAT_OEP[n]_D.OUTPKTERR` bit applies to OUT endpoints only. This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet.
6 (R/W1C)	B2BSETUP	Back-to-Back SETUP Packets Received. The `USBC_ISTAT_OEP[n]_D.B2BSETUP` bit applies to control OUT endpoints only. The `USBC_ISTAT_OEP[n]_D.B2BSETUP` bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.

Table 21-31: USBC_ISTAT_OEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W1C)	STSPHSRCVD	Status Phase Received for Control Write. The `USBC_ISTAT_OEP[n]_D.STSPHSRCVD` bit is valid only for control OUT endpoints and only in scatter/gather DMA mode. The `USBC_ISTAT_OEP[n]_D.STSPHSRCVD` bit indicates an interrupt that is generated after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase. The `USBC_ISTAT_OEP[n]_D.STSPHSRCVD` bit is applicable only in case of scatter/gather DMA mode.
4 (R/W1C)	OUTTKNEPDIS	OUT Token Received When Endpoint Disabled. The `USBC_ISTAT_OEP[n]_D.OUTTKNEPDIS` bit applies only to control OUT endpoints. The `USBC_ISTAT_OEP[n]_D.OUTTKNEPDIS` bit indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.
3 (R/W1C)	SETUP	SETUP Phase Done. The `USBC_ISTAT_OEP[n]_D.SETUP` bit applies to control OUT endpoints only. The `USBC_ISTAT_OEP[n]_D.SETUP` bit indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.
2 (R/W1C)	AHBERR	Bus Error. The `USBC_ISTAT_OEP[n]_D.AHBERR` bit applies to IN and OUT endpoints. The `USBC_ISTAT_OEP[n]_D.AHBERR` bit is generated when there is a bus error during an bus read/write. The application can read the corresponding endpoint DMA address register to get the error address.
1 (R/W1C)	EPDISBLD	Endpoint Disabled Interrupt. The `USBC_ISTAT_OEP[n]_D.EPDISBLD` bit applies to IN and OUT endpoints. The `USBC_ISTAT_OEP[n]_D.EPDISBLD` bit indicates that the endpoint is disabled per the application's request.

Table 21-31: USBC_ISTAT_OEP[n]_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W1C)	XFERCOMPL	Transfer Completed Interrupt. The `USBC_ISTAT_OEP[n]_D.XFERCOMPL` bit applies to IN and OUT endpoints. When scatter/gather DMA mode is enabled: For IN endpoints, the `USBC_ISTAT_OEP[n]_D.XFERCOMPL` bit indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. For OUT endpoints, the `USBC_ISTAT_OEP[n]_D.XFERCOMPL` bit indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is set. Note: In DMA mode, this bit must be set for the core to transfer SETUP data packets into memory. When scatter/gather DMA mode is disabled, the `USBC_ISTAT_OEP[n]_D.XFERCOMPL` bit indicates that the programmed transfer is complete on the bus as well as on the USB, for this endpoint.

Device OUT Endpoint Common Interrupt Mask Register

The USBC_IMASK_OEP_D register works with each of the device OUT endpoint interrupt (USBC_ISTAT_OEP[n]_D) registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the USBC_ISTAT_OEP[n]_D register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

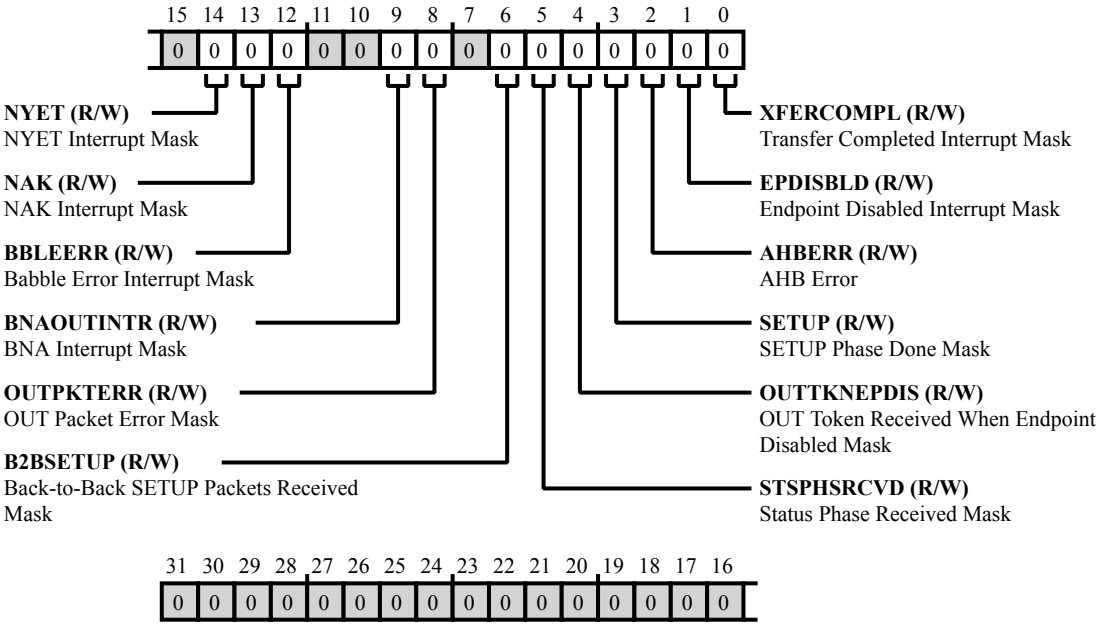


Figure 21-37: USBC_IMASK_OEP_D Register Diagram

Table 21-32: USBC_IMASK_OEP_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W)	NYET	NYET Interrupt Mask.
13 (R/W)	NAK	NAK Interrupt Mask.
12 (R/W)	BBLEERR	Babble Error Interrupt Mask.
9 (R/W)	BNAOUTINTR	BNA Interrupt Mask. BNA interrupt Mask (BnaOutIntrMsk)
8 (R/W)	OUTPKTERR	OUT Packet Error Mask.

Table 21-32: USBC_IMASK_OEP_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/W)	B2BSETUP	Back-to-Back SETUP Packets Received Mask. The `USBC_IMASK_OEP_D.B2BSETUP` bit only applies to control OUT endpoints.
5 (R/W)	STSPHSRCVD	Status Phase Received Mask. The `USBC_IMASK_OEP_D.STSPHSRCVD` bit only applies to control OUT endpoints.
4 (R/W)	OUTTKNEPDIS	OUT Token Received When Endpoint Disabled Mask. The `USBC_IMASK_OEP_D.OUTTKNEPDIS` bit only applies to control OUT endpoints.
3 (R/W)	SETUP	SETUP Phase Done Mask. The `USBC_IMASK_OEP_D.SETUP` bit only applies to control endpoints.
2 (R/W)	AHBERR	AHB Error.
1 (R/W)	EPDISBLD	Endpoint Disabled Interrupt Mask.
0 (R/W)	XFERCOMPL	Transfer Completed Interrupt Mask.

Device OUT Endpoint 0 Transfer Size Register

The USBC_TSIZ_OEP0_D register holds the transfer size of OUT endpoint 0 implemented in device mode.

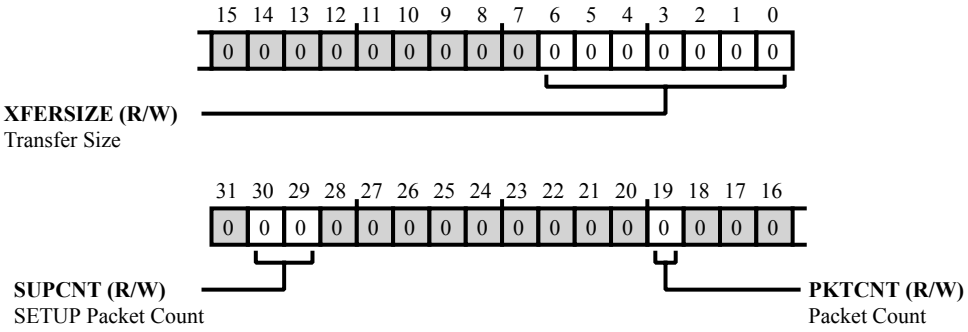


Figure 21-38: USBC_TSIZ_OEP0_D Register Diagram

Table 21-33: USBC_TSIZ_OEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:29 (R/W)	SUPCNT	SETUP Packet Count. The USBC_TSIZ_OEP0_D.SUPCNT field specifies the number of back-to-back SETUP data packets the endpoint can receive.	
		0	Reserved
		1	1 packet
		2	2 packets
		3	3 packets
19 (R/W)	PKTCNT	Packet Count. The USBC_TSIZ_OEP0_D.PKTCNT field is decremented to zero after a packet is written into the Rx FIFO.	
6:0 (R/W)	XFERSIZE	Transfer Size. The USBC_TSIZ_OEP0_D.XFERSIZE field indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements the USBC_TSIZ_OEP0_D.XFERSIZE field every time a packet is read from the Rx FIFO and written to the external memory.	

Device OUT Endpoint n Transfer Size Register

The USBC_TSIZ_OEP[n]_D register holds the transfer size of OUT endpoint n implemented in device mode.

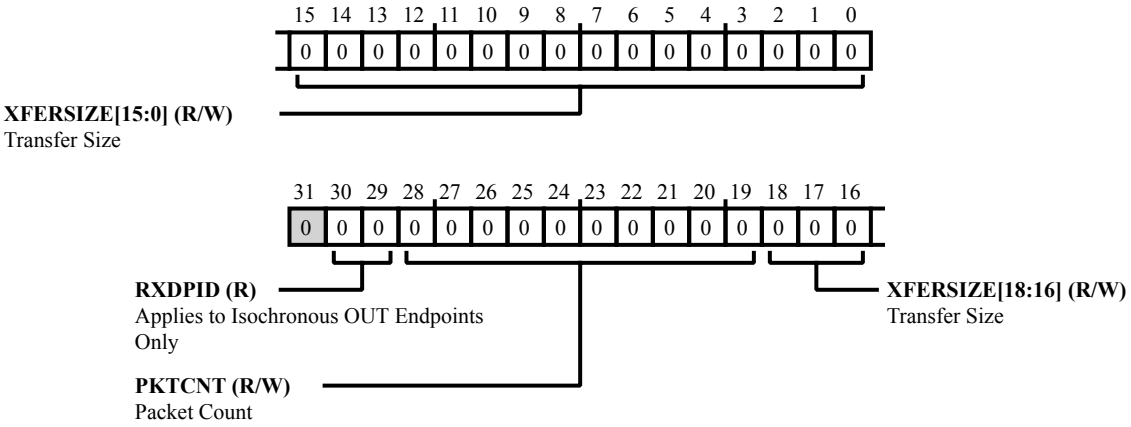


Figure 21-39: USBC_TSIZ_OEP[n]_D Register Diagram

Table 21-34: USBC_TSIZ_OEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:29 (R/NW)	RXDPID	Applies to Isochronous OUT Endpoints Only. The USBC_TSIZ_OEP[n]_D.RXDPID field indicates the data PID received in the last packet for this endpoint.	
		0	DATA0
		1	DATA2
		2	DATA1
		3	MDATA
28:19 (R/W)	PKTCNT	Packet Count. The USBC_TSIZ_OEP[n]_D.PKTCNT field is decremented to zero after a packet is written into the Rx FIFO.	
18:0 (R/W)	XFERSIZE	Transfer Size. The USBC_TSIZ_OEP[n]_D.XFERSIZE field indicates the transfer size in bytes for endpoint n. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	

Device Status Register

The USBC_STAT_D register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the USBC_ISTAT_D register.

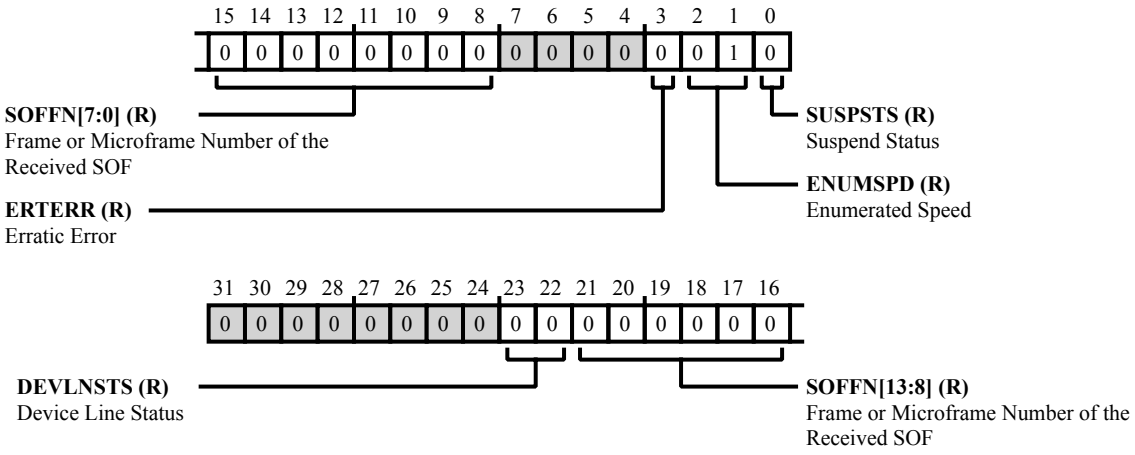


Figure 21-40: USBC_STAT_D Register Diagram

Table 21-35: USBC_STAT_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
23:22 (R/NW)	DEVLNSTS	Device Line Status. The USBC_STAT_D.DEVLNSTS field indicates the current logic level USB data lines.	
		0	Logic Level of D+
		1	Logic Level of D-
21:8 (R/NW)	SOFFN	Frame or Microframe Number of the Received SOF. When the core is operating at high speed, the USBC_STAT_D.SOFFN field contains a microframe number. When the core is operating at full or low speed, this field contains a frame number. Note: This register may return a non-zero value if read immediately after power-on reset. In case the register bit reads non-zero immediately after power-on reset, it does not indicate that SOF has been received from the host. The read value of this interrupt is valid only after a valid connection between host and device is established.	

Table 21-35: USBC_STAT_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	ERTERR	Erratic Error. The core sets the USBC_STAT_D.ERTERR bit to report any erratic errors (due to PHY error) seen on the ULPI. Due to erratic errors, the USBC_otg core goes into a SUSPEND state and an interrupt is generated to the application with the USBC_ISTAT.ERLYSUSP bit. If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recovery.	
2:1 (R/NW)	ENUMSPD	Enumerated Speed. The USBC_STAT_D.ENUMSPD field indicates the speed at which the controller has come up after speed detection through a connect or reset sequence. Low speed is not supported for devices using a ULPI PHY.	
		0	High speed. PHY clock is running at 30 or 60 Mhz.
		1	Full speed. PHY clock is running at 30 or 60 Mhz.
		2	Low speed. PHY clock is running at 6 Mhz.
		3	Full speed. PHY clock is running at 48 Mhz.
0 (R/NW)	SUSPSTS	Suspend Status. In device mode, the USBC_STAT_D.SUSPSTS bit is set (=1) as long as a SUSPEND condition is detected on the USB. The core enters the SUSPEND state when there is no activity on the linestate signal for an extended period of time. The core comes out of the suspend under the following conditions : - If there is any activity on the linestate signal, or - If the application writes to the Remote Wakeup Signaling bit in the USBC_CTL_D register. When the core comes out of the suspend, the USBC_STAT_D.SUSPSTS bit is cleared (=0).	

Device Threshold Control Register

The USBC_THR_CTL_D register indicates the device threshold control information.

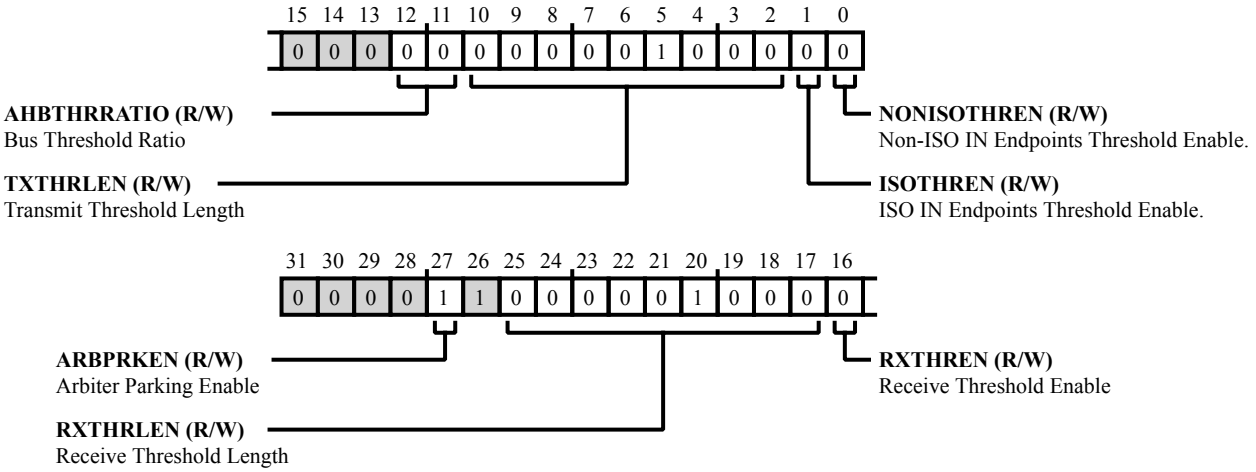


Figure 21-41: USBC_THR_CTL_D Register Diagram

Table 21-36: USBC_THR_CTL_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27 (R/W)	ARBPRKEN	Arbiter Parking Enable. The USBC_THR_CTL_D.ARBPRKEN bit controls internal DMA arbiter parking for IN endpoints. If thresholding is enabled and the USBC_THR_CTL_D.ARBPRKEN bit is set (=1), the arbiter parks on the IN endpoint for which there is a token received on the USB. This avoids getting into underrun conditions. By default, arbiter parking is enabled.
25:17 (R/W)	RXTHRLEN	Receive Threshold Length. The USBC_THR_CTL_D.RXTHRLEN field specifies receive thresholding size in DWORDS. The USBC_THR_CTL_D.RXTHRLEN field also specifies the amount of data received on the USB before the core can start transmitting on the bus. The threshold length has to be at least eight DWORDS. The recommended value for ThrLen must be the same as USBC_AHB_CFG.HBSTLEN.
16 (R/W)	RXTHREN	Receive Threshold Enable. When the USBC_THR_CTL_D.RXTHREN bit is set, the core enables thresholding in the receive direction. Note: Do not enable the USBC_THR_CTL_D.RXTHREN bit, because it may cause issues in the RxFIFO, especially during error conditions such as RxError and Babble.

Table 21-36: USBC_THR_CTL_D Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12:11 (R/W)	AHBTHRRATIO	Bus Threshold Ratio. The USBC_THR_CTL_D.AHBTHRRATIO field defines the ratio between the bus threshold and the MAC threshold for the transmit path only. The bus threshold always remains less than or equal to the USB threshold, because this does not increase overhead. Both the bus and the MAC threshold must be DWORD-aligned. The application must program USBC_THR_CTL_D.TXTHRLEN and the USBC_THR_CTL_D.AHBTHRRATIO bits to make the bus threshold value DWORD aligned. If the bus threshold value is not DWORD aligned, the core might not behave correctly. When programming the USBC_THR_CTL_D.TXTHRLEN and USBC_THR_CTL_D.AHBTHRRATIO, the application must ensure that the minimum bus threshold value does not go below 8 DWORDS to meet the USB turnaround time requirements.	
		0	MAC threshold
		1	MAC threshold/2
		2	MAC threshold/4
		3	MAC threshold/8
10:2 (R/W)	TXTHRLEN	Transmit Threshold Length. The USBC_THR_CTL_D.TXTHRLEN field specifies the transmit thresholding size in DWORDS. It also forms the MAC threshold and specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmit on the USB. The threshold length must be at least eight DWORDS when the value of USBC_THR_CTL_D.AHBTHRRATIO is zero. When the USBC_THR_CTL_D.AHBTHRRATIO bit field is non zero, the application must ensure that the bus threshold value does not go below the recommended eight DWORD. The USBC_THR_CTL_D.TXTHRLEN field controls both isochronous and non-isochronous IN endpoint thresholds. The recommended value for the USBC_THR_CTL_D.TXTHRLEN field is the same as the programmed bus burst length in the USBC_AHB_CFG.HBSTLEN field.	
1 (R/W)	ISOTHREN	ISO IN Endpoints Threshold Enable.. When the USBC_THR_CTL_D.ISOTHREN bit is set, the core enables thresholding for isochronous IN endpoints.	
0 (R/W)	NONISOTHREN	Non-ISO IN Endpoints Threshold Enable.. When the USBC_THR_CTL_D.NONISOTHREN bit is set, the core enables thresholding for non Isochronous IN endpoints.	

Device Control IN Endpoint Transmit FIFO Status Register

The USBC_TXFSTAT_IEP0_D register indicates the amount of free space available in the endpoint Tx FIFO.

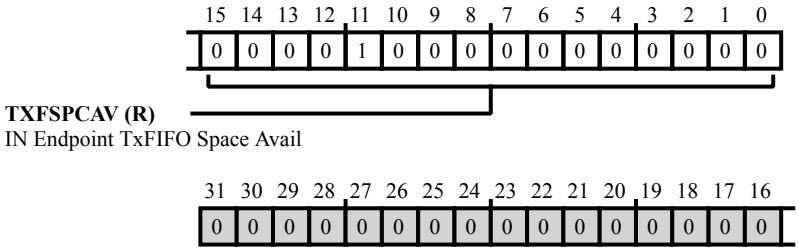


Figure 21-42: USBC_TXFSTAT_IEP0_D Register Diagram

Table 21-37: USBC_TXFSTAT_IEP0_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXFSPCAV	IN Endpoint TxFIFO Space Avail. The USBC_TXFSTAT_IEP0_D.TXFSPCAV field indicates the amount of free space available in the endpoint TxFIFO. Values are in terms of 32-bit words. - 16'h0: Endpoint TxFIFO is full - 16'h1: 1 word available - 16'h2: 2 words available - 16'hn: n words available (where 0 n 32,768) - 16'h8000: 32,768 words available - Others: Reserved

Device Control IN Endpoint Transmit FIFO Status Register

In device mode, the `USBC_TXFSTAT_IEP[n]_D` register indicates the status of the Tx FIFO in the endpoint.

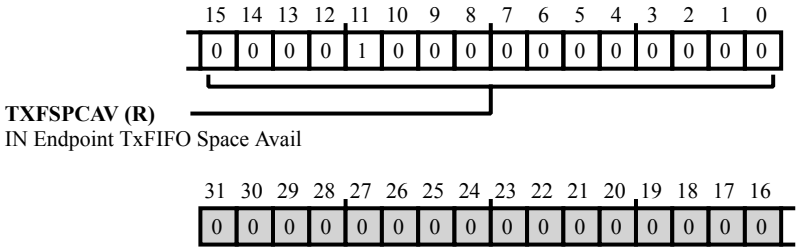


Figure 21-43: USBC_TXFSTAT_IEP[n]_D Register Diagram

Table 21-38: USBC_TXFSTAT_IEP[n]_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXFSPCAV	IN Endpoint TxFIFO Space Avail. The `USBC_TXFSTAT_IEP[n]_D.TXFSPCAV` field indicates the amount of free space available in the endpoint TxFIFO. Values are in terms of 32-bit words. - 16'h0: Endpoint TxFIFO is full - 16'h1: 1 word available - 16'h2: 2 words available - 16'hn: n words available (where 0 n 32,768) - 16'h8000: 32,768 words available - Others: Reserved

Device VBUS Discharge Time Register

The USBC_VBUSDIS_D register specifies the VBUS discharge time after VBUS pulsing during SRP.

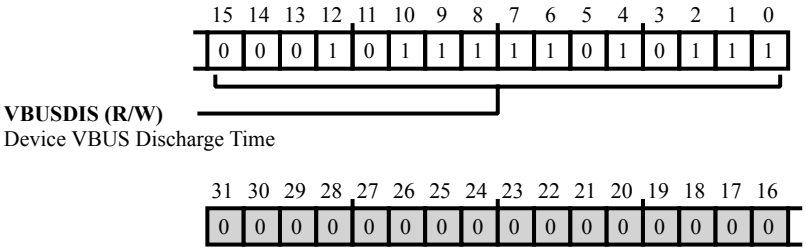


Figure 21-44: USBC_VBUSDIS_D Register Diagram

Table 21-39: USBC_VBUSDIS_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VBUSDIS	Device VBUS Discharge Time. The USBC_VBUSDIS_D.VBUSDIS bit indicates the VBUS discharge time after VBUS pulsing during SRP. This value equals (VBUS discharge time in PHY clocks) / 1,024. The value used depends whether the PHY is operating at 30 MHz (16-bit data width) or 60 MHz (8-bit data width). Depending on the VBUS load, this value can need adjustment.

Device VBUS Pulsing Time Register

The USBC_VBUSPULSE_D register indicates the pulsing time during SRP.

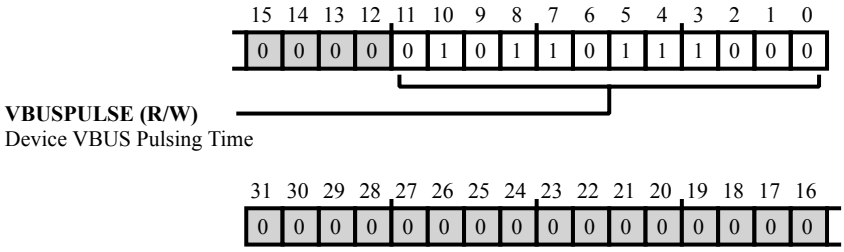


Figure 21-45: USBC_VBUSPULSE_D Register Diagram

Table 21-40: USBC_VBUSPULSE_D Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	VBUSPULSE	Device VBUS Pulsing Time. The USBC_VBUSPULSE_D.VBUSPULSE field specifies the VBUS pulsing time during SRP. This value equals (VBUS pulsing time in PHY clocks) / 1,024. The value used depends whether the PHY is operating at 30 MHz (16-bit data width) or 60 MHz (8-bit data width).

Bus Configuration Register

The USBC_AHB_CFG register can be used to configure the core after power-on or a change in mode. This register mainly contains bus system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the bus or the USB.

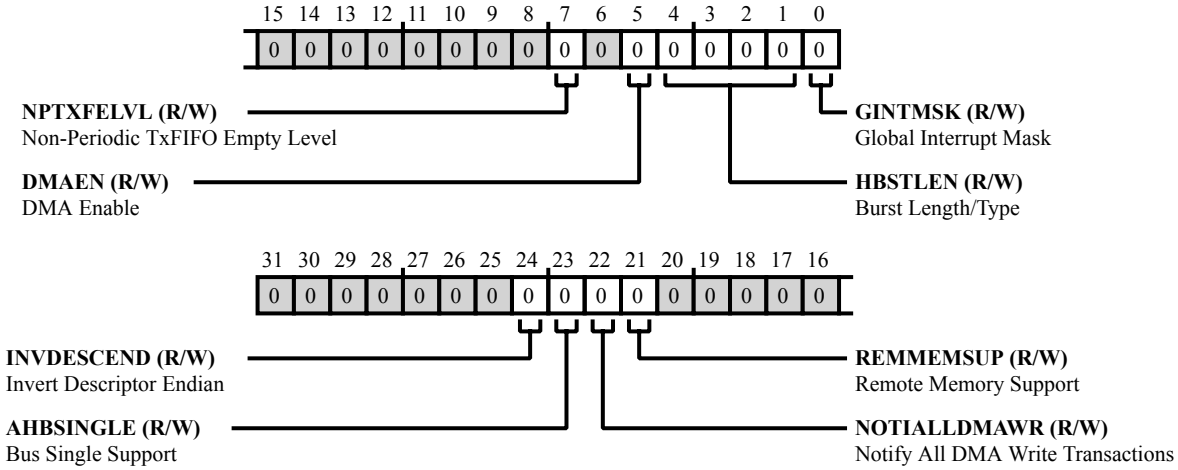


Figure 21-46: USBC_AHB_CFG Register Diagram

Table 21-41: USBC_AHB_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	INVDESCEND	Invert Descriptor Endian. The USBC_AHB_CFG.INVDESCEND bit is applicable in host and device modes. The USBC_AHB_CFG.INVDESCEND bit indicates whether the byte order function (endian) of the bus requester is the same as the descriptor or is inverted.	
		0	Endian configuration is the same as the bus requester
		1	Endian configuration is the opposite of the bus requester. If the byte order function of the bus requester is big endian, the descriptor is little endian. If the byte order function of bus requester is little endian, the descriptor is big endian.

Table 21-41: USBC_AHB_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
23 (R/W)	AHBSINGLE	Bus Single Support. When the core is operating in DMA mode, the USBC_AHB_CFG.AHBSINGLE bit supports single transfers for the remaining data in a transfer. It determines whether the remaining data is sent in an INCR burst size or a single burst size. The USBC_AHB_CFG.AHBSINGLE bit is active in both host and device modes. Note: If this feature is enabled, the bus RETRY and SPLIT transfers still have INCR burst type. Enable this feature when the bus responder connected to the core does not support INCR burst (and when Split, and Retry transactions are not being used in the bus). 0 – INCR Burst Size 1 – Single Burst Size
22 (R/W)	NOTIALLDMAWR	Notify All DMA Write Transactions. The USBC_AHB_CFG.NOTIALLDMAWR bit is applicable in host and device modes. The USBC_AHB_CFG.NOTIALLDMAWR bit is programmed to enable the system DMA Done functionality for all the DMA write Transactions corresponding to the channel/endpoint. The USBC_AHB_CFG.NOTIALLDMAWR bit is valid only when USBC_AHB_CFG.REMMEMSUP = 1 and USBC_AHB_CFG.NOTIALLDMAWR = 1. The core asserts dma request for all the DMA write transactions on the bus interface. The core waits for system dma done signal for all the DMA write transactions in order to complete the transfer of a particular channel/endpoint. When USBC_AHB_CFG.NOTIALLDMAWR = 0, the core asserts dma request signal only for the last transaction of DMA write transfer corresponding to a particular channel/endpoint. Similarly, the core waits for system dma done signal only for that transaction of DMA write to complete the transfer of a particular channel/endpoint.
21 (R/W)	REMMEMSUP	Remote Memory Support. The USBC_AHB_CFG.REMMEMSUP bit is applicable in host and device modes. The USBC_AHB_CFG.REMMEMSUP bit is programmed to enable the functionality to wait for the system DMA Done signal for the DMA write transfers. When USBC_AHB_CFG.REMMEMSUP = 1, the dma request output signal is asserted when the DMA starts write transfer to the external memory. When the core is done with the transfers it asserts dma done signal to flag the completion of DMA writes from the controller. The core then waits for system dma done signal from the system to proceed further and complete the data transfer corresponding to a particular channel/endpoint. When USBC_AHB_CFG.REMMEMSUP = 0, the dma request and dma done signals are not asserted and the core proceeds with the assertion of the XferComp interrupt as soon as the DMA write transfer is done at the core boundary and it does not wait for the system dma done signal to complete the DATA transfers.

Table 21-41: USBC_AHB_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	NPTXFELVL	Non-Periodic TxFIFO Empty Level. The USBC_AHB_CFG.NPTXFELVL bit is applicable in host and device modes. In host mode and with shared FIFO and device mode: - 1'b0: the USBC_ISTAT.NPTXFEMP interrupt indicates that the non-periodic TxFIFO is half empty. - 1'b1: the USBC_ISTAT.NPTXFEMP interrupt indicates that the non-periodic TxFIFO is completely empty. In dedicated FIFO in device mode: - 1'b0: the USBC_ISTAT_IEP[n]_D.TXFEMP interrupt indicates that the IN endpoint TxFIFO is half empty. - 1'b1: the USBC_ISTAT_IEP[n]_D.TXFEMP interrupt indicates that the IN endpoint TxFIFO is completely empty.	
		0	TxFIFO half empty: device mode and host mode with shared FIFO USBC_ISTAT.NPTXFEMPinterrupt / device mode with dedicated FIFO USBC_ISTAT_IEP[n]_D.TXFEMP interrupt
		1	TxFIFO completely empty: device mode and host mode with shared FIFO USBC_ISTAT.NPTXFEMP interrupt / device mode with dedicated FIFO USBC_ISTAT_IEP[n]_D.TXFEMP interrupt
5 (R/W)	DMAEN	DMA Enable. The USBC_AHB_CFG.DMAEN bit is applicable in host and device modes. The USBC_AHB_CFG.DMAEN bit is always 0 when bus responder only mode has been selected. Reset: 1'b0	
4:1 (R/W)	HBSTLEN	Burst Length/Type. The USBC_AHB_CFG.HBSTLEN bit field indicates the burst length. It is applicable in host and device modes.	
		0	Single
		1	INCR
		2	Reserved
		3	INCR4
		4	Reserved
		5	INCR8
		6	Reserved
		7	INCR16

Table 21-41: USBC_AHB_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	GINTMSK	Global Interrupt Mask. The application uses the `USBC_AHB_CFG.GINTMSK` bit to mask or unmask the interrupt line assertion to itself. Irrespective of the configuration of the `USBC_AHB_CFG.GINTMSK` bit, the interrupt status registers are updated by the controller. The `USBC_AHB_CFG.GINTMSK` bit is applicable in host and device modes.	
		0	Mask
		1	Unmask

DFIFO Configuration Register

The USBC_DFIFO_CFG register represents the global DFIFO configuration.

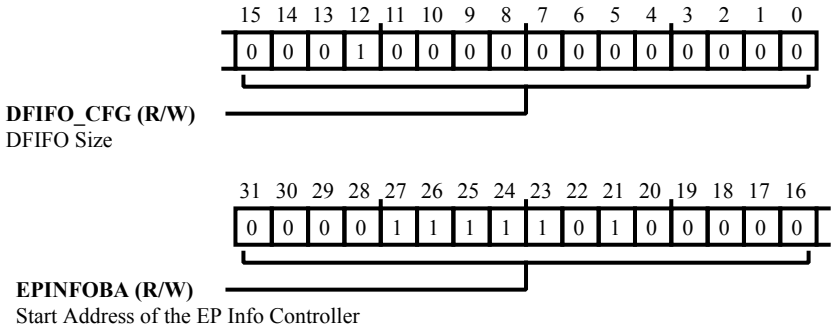


Figure 21-47: USBC_DFIFO_CFG Register Diagram

Table 21-42: USBC_DFIFO_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	EPINFOBA	Start Address of the EP Info Controller. The USBC_DFIFO_CFG.EPINFOBA field provides the start address of the EP info controller.
15:0 (R/W)	DFIFO_CFG	DFIFO Size. The USBC_DFIFO_CFG.DFIFO_CFG field is for dynamic programming of the DFIFO Size. This value takes effect only when the application programs a non zero value to this register. The value programmed must conform to the guidelines described in 'FIFO RAM Allocation'. The core does not have any corrective logic if the FIFO sizes are programmed incorrectly.

User Hardware Configuration 1 Register

The USBC_HWCFG1 register contains the logical endpoint direction(s).

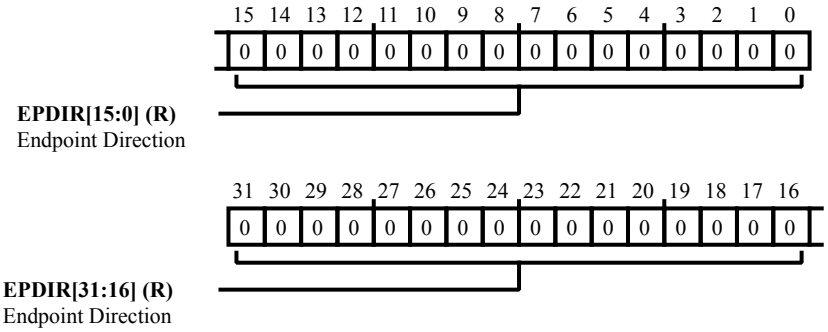


Figure 21-48: USBC_HWCFG1 Register Diagram

Table 21-43: USBC_HWCFG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/NW)	EPDIR	Endpoint Direction. The USBC_HWCFG1.EPDIR field uses two bits per endpoint to determine the endpoint direction. Endpoint - Bits [31:30]: Endpoint 15 direction - Bits [29:28]: Endpoint 14 direction ... - Bits [3:2]: Endpoint 1 direction - Bits[1:0]: Endpoint 0 direction (always BIDIR)	
		0	BIDIR (IN and OUT) endpoint
		1	IN endpoint
		2	OUT endpoint
		3	Reserved

User Hardware Configuration 2 Register

The USBC_HWCFG2 register contains configuration options.

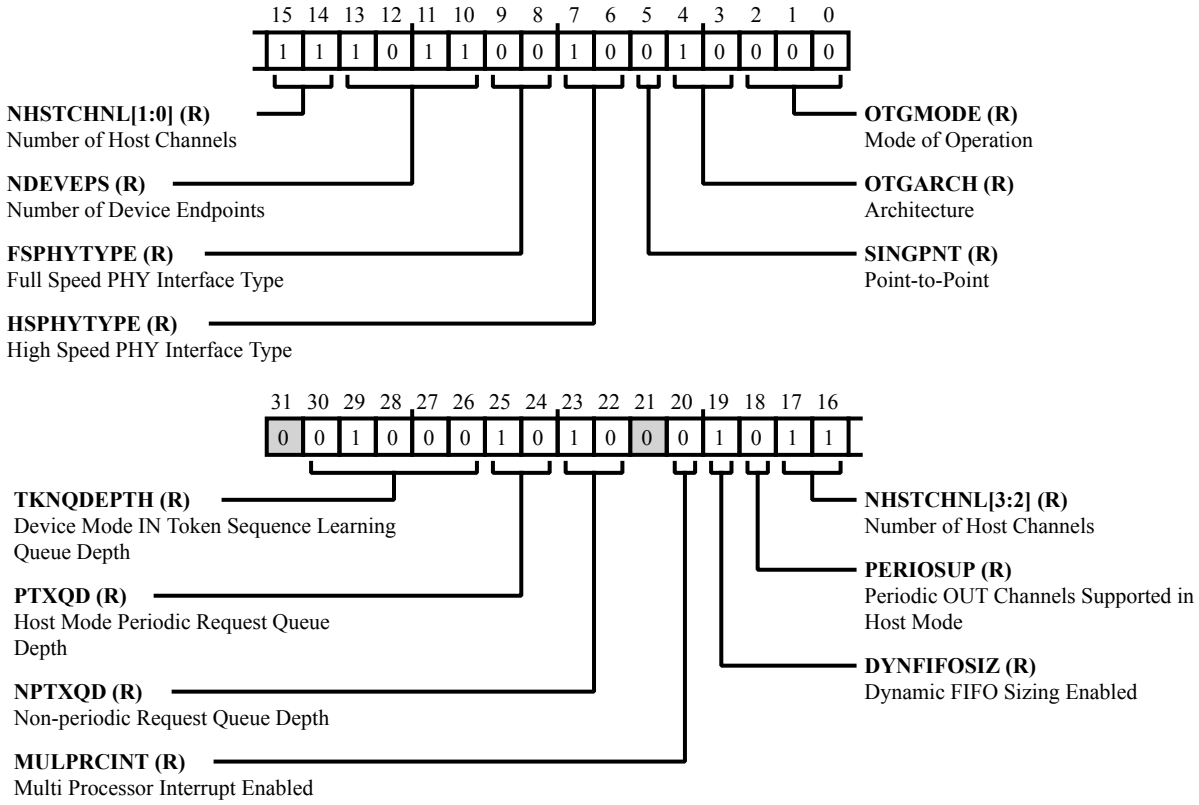


Figure 21-49: USBC_HWCFG2 Register Diagram

Table 21-44: USBC_HWCFG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:26 (R/NW)	TKNQDEPTH	Device Mode IN Token Sequence Learning Queue Depth. The USBC_HWCFG2.TKNQDEPTH field indicates the device mode IN token sequence learning queue depth. The depth can range from 0 to 30.	
25:24 (R/NW)	PTXQD	Host Mode Periodic Request Queue Depth. The USBC_HWCFG2.PTXQD field indicates the host mode periodic request queue depth.	
		0	2
		1	4
		2	8
		3	16

Table 21-44: USBC_HWCFG2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:22 (R/NW)	NPTXQD	Non-periodic Request Queue Depth. The `USBC_HWCFG2.NPTXQD` field indicates the non-periodic request queue depth.	
		0	2
		1	4
		2	8
		3	Reserved
20 (R/NW)	MULPRCINT	Multi Processor Interrupt Enabled. The `USBC_HWCFG2.MULPRCINT` bit indicates if the multiprocessor interrupt is enabled.	
		0	No
		1	Yes
19 (R/NW)	DYNFIFOSIZ	Dynamic FIFO Sizing Enabled. The `USBC_HWCFG2.DYNFIFOSIZ` bit indicates if dynamic FIFO sizing is enabled.	
18 (R/NW)	PERIOSUP	Periodic OUT Channels Supported in Host Mode. The ${.::} field indicates if the periodic OUT channel is supported in host mode.	
		0	No
		1	Yes
17:14 (R/NW)	NHSTCHNL	Number of Host Channels. The `USBC_HWCFG2.NHSTCHNL` field indicates the number of host channels supported by the core in host mode. The range of this field is 0-15: 0 specifies 1 channel, 15 specifies 16 channels.	
13:10 (R/NW)	NDEVEPS	Number of Device Endpoints. The `USBC_HWCFG2.NDEVEPS` field indicates the number of device endpoints supported by the core in device mode. The range of this field is 0-15.	
9:8 (R/NW)	FSPHYTYPE	Full Speed PHY Interface Type. The `USBC_HWCFG2.FSPHYTYPE` field indicates the type of full speed PHY interface.	
		0	Full speed interface not supported
		1	Dedicated full speed interface
		2	FS pins shared with UTMI+ pins
		3	FS pins shared with ULPI pins

Table 21-44: USBC_HWCFG2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7:6 (R/NW)	HSPHYTYPE	High Speed PHY Interface Type. The `USBC_HWCFG2.HSPHYTYPE` field indicates the type of high speed PHY interface.	
		0	High speed interface is not supported
		1	UTMI+
		2	ULPI
		3	UTMI+ and ULPI
5 (R/NW)	SINGPNT	Point-to-Point. The `USBC_HWCFG2.SINGPNT` bit indicates whether the application is a single point application with hub and split support or a multipoint application with no hub and split support.	
		0	Multipoint Application
		1	Single Point Application
4:3 (R/NW)	OTGARCH	Architecture. The `USBC_HWCFG2.OTGARCH` field indicates the architecture.	
		0	Responder Only
		1	External DMA
		2	Internal DMA
		3	Reserved
2:0 (R/NW)	OTGMODE	Mode of Operation. The `USBC_HWCFG2.OTGMODE` field indicates the mode of operation.	
		0	HNP and SRP Capable OTG (Host and Device)
		1	SRP Capable OTG (Host and Device)
		2	Non-HNP and Non-SRP Capable OTG (Host and Device)
		3	SRP Capable Device
		4	Non-OTG Device
		5	SRP Capable Host
		6	Non-OTG Host
		7	Reserved

User Hardware Configuration 3 Register

The USBC_HWCFG3 register contains configuration options.

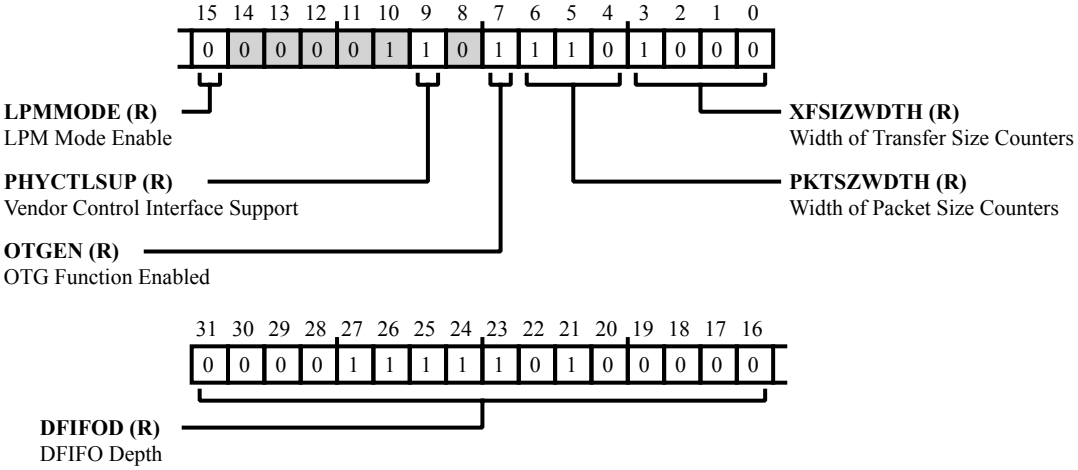


Figure 21-50: USBC_HWCFG3 Register Diagram

Table 21-45: USBC_HWCFG3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16 (R/NW)	DFIFOD	DFIFO Depth. The USBC_HWCFG3.DFIFOD field indicates the DFIFO depth. This value is in terms of 32-bit words, with a minimum value is 32 and a maximum value is 32,768.	
15 (R/NW)	LPMMODE	LPM Mode Enable. The USBC_HWCFG3.LPMMODE bit enables the LPM mode of operation.	
9 (R/NW)	PHYCTLSUP	Vendor Control Interface Support. The USBC_HWCFG3.PHYCTLSUP bit indicates whether the vendor control interface is available on the core.	
		0	Not available
		1	Available
7 (R/NW)	OTGEN	OTG Function Enabled. The application uses the USBC_HWCFG3.OTGEN bit to indicate the OTG capabilities of the controller.	
		0	Not OTG capable
		1	OTG capable
6:4 (R/NW)	PKTSZWDTH	Width of Packet Size Counters. The USBC_HWCFG3.PKTSZWDTH field indicates the width of the transfer size counters.	

Table 21-45: USBC_HWCFG3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	4 Bits
		1	5 Bits
		2	6 Bits
		3	7 Bits
		4	8 Bits
		5	9 Bits
		6	10 Bits
3:0 (R/NW)	XFSIZWDTH	Width of Transfer Size Counters. The `USBC_HWCFG3.XFSIZWDTH` field indicates the width of the transfer size counters.	
		0	11 Bits
		1	12 Bits
		2	13 Bits
		3	14 Bits
		4	15 Bits
		5	16 Bits
		6	17 Bits
		7	18 Bits
		8	19 Bits

User Hardware Configuration 4 Register

The USBC_HWCFG4 register contains configuration options.

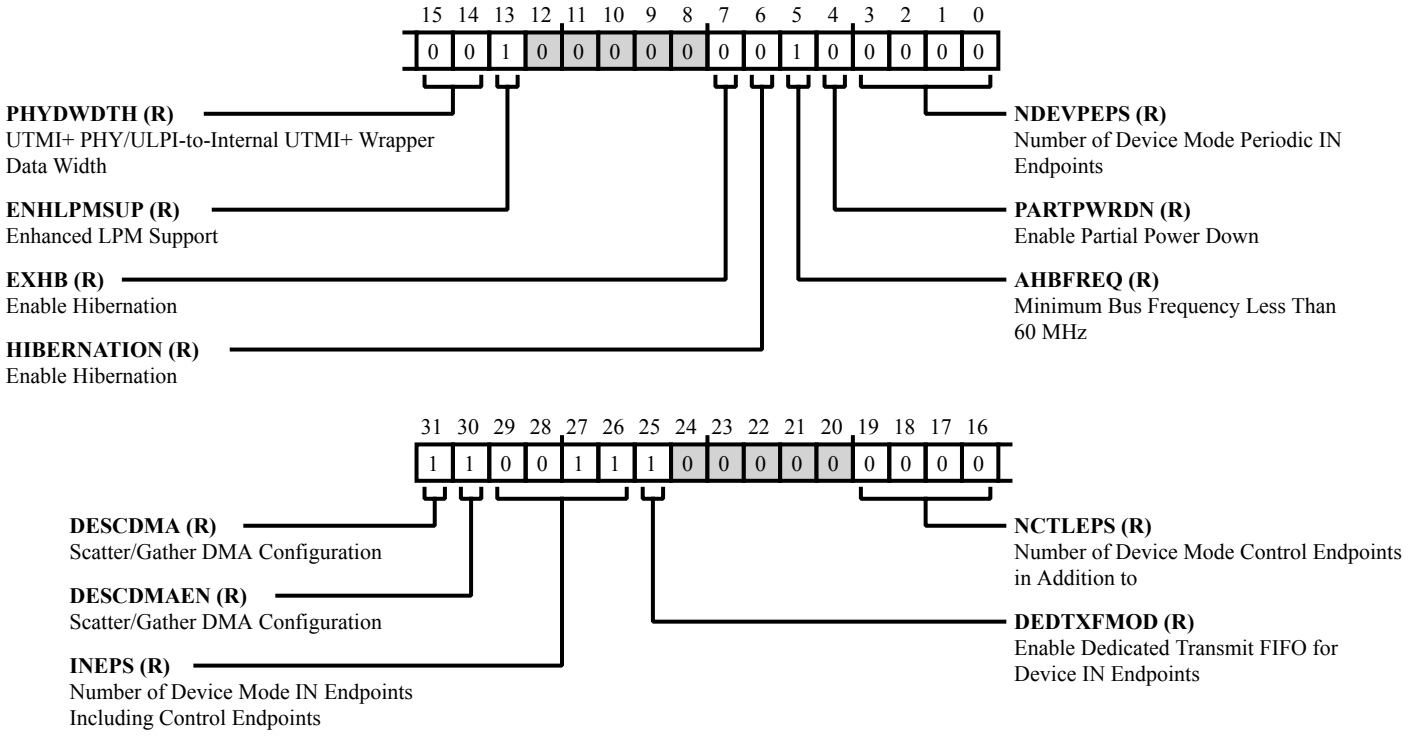


Figure 21-51: USBC_HWCFG4 Register Diagram

Table 21-46: USBC_HWCFG4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	DESCDMA	Scatter/Gather DMA Configuration. The USBC_HWCFG4.DESCDMA bit enables dynamic scatter/gather DMA configuration.	
		0	Disable
		1	Enable
30 (R/NW)	DESCDMAEN	Scatter/Gather DMA Configuration. The USBC_HWCFG4.DESCDMAEN bit enables scatter/gather DMA configuration.	
		0	Non scatter/gather DMA
		1	Scatter/gather DMA
29:26 (R/NW)	INEPS	Number of Device Mode IN Endpoints Including Control Endpoints. The USBC_HWCFG4.INEPS field indicates the number of device mode IN endpoints (including control endpoints).	

Table 21-46: USBC_HWCFG4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	1 IN Endpoint
		1	2 IN Endpoints
		2	3 IN Endpoints
		3	4 IN Endpoints
		4	5 IN Endpoints
		5	6 IN Endpoints
		6	7 IN Endpoints
		7	8 IN Endpoints
		8	9 IN Endpoints
		9	10 IN Endpoints
		10	11 IN Endpoints
		11	12 IN Endpoints
		12	13 IN Endpoints
		13	14 IN Endpoints
		14	15 IN Endpoints
		15	16 IN Endpoints
25 (R/NW)	DEDTXFMOD	Enable Dedicated Transmit FIFO for Device IN Endpoints. The `USBC_HWCFG4.DEDTXFMOD` bit enables a dedicated transmit FIFO for device IN endpoints (DedFifoMode).	
		0	Disable
		1	Enable
19:16 (R/NW)	NCTLEPS	Number of Device Mode Control Endpoints in Addition to. Endpoint 0 (NCTLEPS) Range: 0-15	
15:14 (R/NW)	PHYDWDTH	UTMI+ PHY/ULPI-to-Internal UTMI+ Wrapper Data Width. When a ULPI PHY is used, an internal wrapper converts ULPI to UTMI+.	
		0	8 bits
		1	16 bits
		2	8/16 bits, software selectable
		3	Reserved

Table 21-46: USBC_HWCFG4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/NW)	ENHLPMSUP	Enhanced LPM Support. The `USBC_HWCFG4.ENHLPMSUP` bit indicates that the controller supports the following behavior: L1 entry behavior is based on FIFO status: - TX FIFO - Accept L1 request even if ISOC OUT TX FIFO is not empty. - Reject L1 request if non-periodic TX FIFO is not empty. - Ensure application can flush the TX FIFO while the controller is in L1. - RX FIFO - Accept L1 request even if RX FIFO (common to periodic and non-periodic) is not empty. - Accept L1 request but delay SLEEPM assertion until RX SINK buffer is empty. Prevent L1 entry if a control transfer is in progress on any control Endpoint. Ability to Flush TxFIFO even if PHY Clock is gated.	
7 (R/NW)	EXHB	Enable Hibernation. The `USBC_HWCFG4.EXHB` bit enables extended hibernation.	
		0	Disable
		1	Enable
6 (R/NW)	HIBERNATION	Enable Hibernation. The `USBC_HWCFG4.HIBERNATION` bit enables hibernation.	
		0	Disable
		1	Enable
5 (R/NW)	AHBFREQ	Minimum Bus Frequency Less Than 60 MHz. The `USBC_HWCFG4.AHBFREQ` bits indicates whether the minimum bus frequency is less than 60 MHz.	
		0	No
		1	Yes
4 (R/NW)	PARTPWRDN	Enable Partial Power Down. The `USBC_HWCFG4.PARTPWRDN` bit enables or disables partial power down.	
		0	Disable
		1	Enable
3:0 (R/NW)	NDEVPEPS	Number of Device Mode Periodic IN Endpoints. The `USBC_HWCFG4.NDEVPEPS` field indicates the number of periodic IN endpoints (0-15).	

Interrupt Mask Register

The USBC_IMSK register works with the Interrupt Register (USBC_ISTAT) to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the USBC_ISTAT register bit corresponding to that interrupt is still set.

Note: The fields of this register change depending on host or device mode.

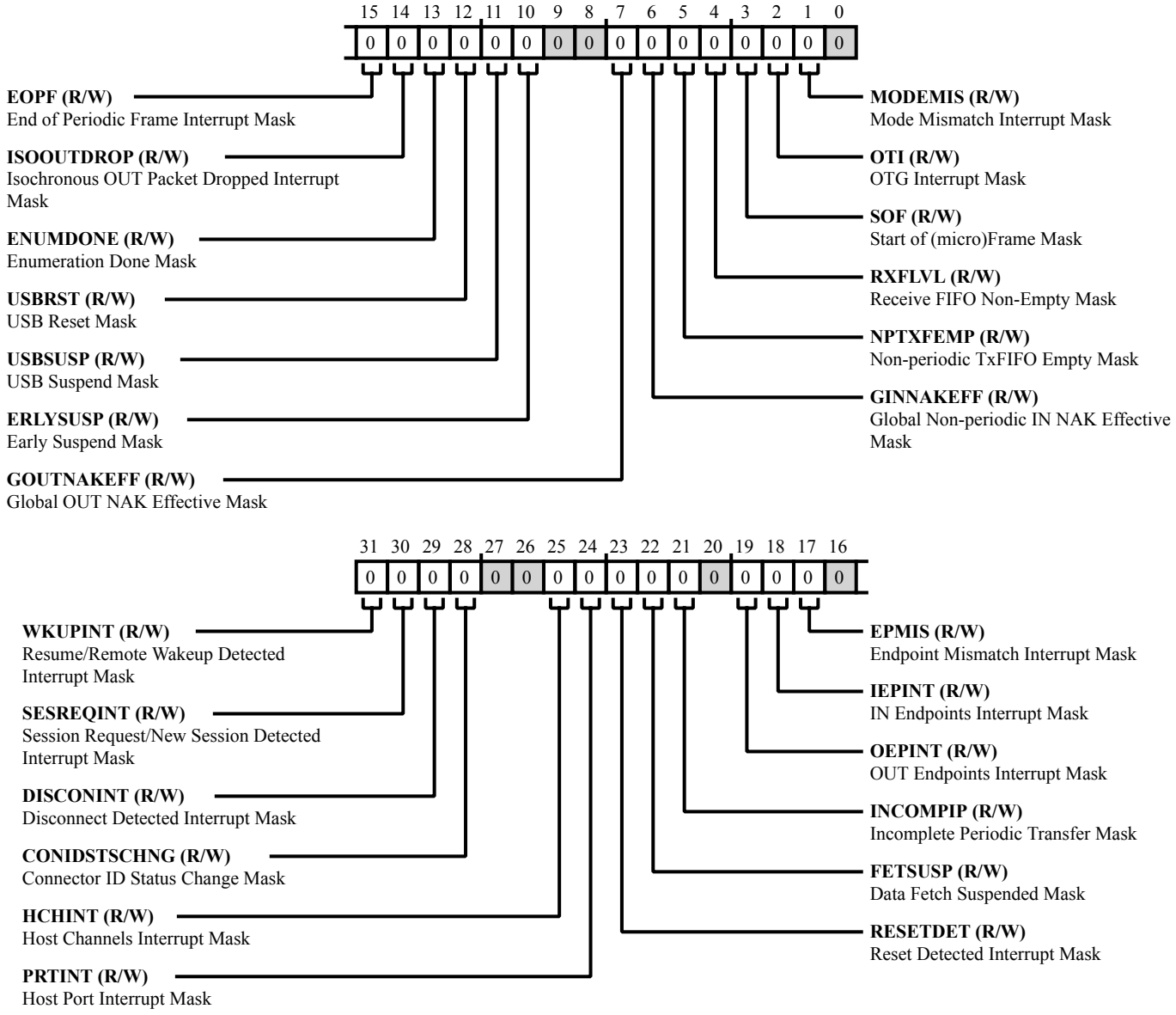


Figure 21-52: USBC_IMSK Register Diagram

Table 21-47: USBC_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	WKUPINT	Resume/Remote Wakeup Detected Interrupt Mask. Mode: Host and Device The WakeUp bit is used for LPM state wake up in a way similar to that of wake up in suspend state.
30 (R/W)	SESREQINT	Session Request/New Session Detected Interrupt Mask. Mode: Host and Device
29 (R/W)	DISCONINT	Disconnect Detected Interrupt Mask. Mode: Host and Device Disconnect Detected Interrupt Mask (DisconnIntMsk)
28 (R/W)	CONIDSTSCHNG	Connector ID Status Change Mask. Mode: Host and Device Connector ID Status Change Mask (ConIDStsChngMsk)
25 (R/W)	HCHINT	Host Channels Interrupt Mask. Mode: Host only
24 (R/W)	PRTINT	Host Port Interrupt Mask. Mode: Host only
23 (R/W)	RESETDET	Reset Detected Interrupt Mask. Mode: Device only
22 (R/W)	FETSUSP	Data Fetch Suspended Mask. Mode: Device only
21 (R/W)	INCOMPIP	Incomplete Periodic Transfer Mask. Mode: Host only Incomplete Isochronous OUT Transfer Interrupt Mask (incompISOOUTMsk) Mode: Device only
19 (R/W)	OEPINT	OUT Endpoints Interrupt Mask. Mode: Device only
18 (R/W)	IEPINT	IN Endpoints Interrupt Mask. Mode: Device only
17 (R/W)	EPMIS	Endpoint Mismatch Interrupt Mask. Mode: Device only
15 (R/W)	EOPF	End of Periodic Frame Interrupt Mask. Mode: Device only
14 (R/W)	ISOOUTDROP	Isochronous OUT Packet Dropped Interrupt Mask. Mode: Device only
13 (R/W)	ENUMDONE	Enumeration Done Mask. Mode: Device only

Table 21-47: USBC_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
12 (R/W)	USBRST	USB Reset Mask. Mode: Device only
11 (R/W)	USBSUSP	USB Suspend Mask. Mode: Device only
10 (R/W)	ERLYSUSP	Early Suspend Mask. Mode: Device only
7 (R/W)	GOUTNAKEFF	Global OUT NAK Effective Mask. Mode: Device only
6 (R/W)	GINNAKEFF	Global Non-periodic IN NAK Effective Mask. Mode: Device only,
5 (R/W)	NPTXFEMP	Non-periodic TxFIFO Empty Mask. Mode: Host and Device
4 (R/W)	RXFLVL	Receive FIFO Non-Empty Mask. Mode: Host and Device
3 (R/W)	SOF	Start of (micro)Frame Mask. Mode: Host and Device
2 (R/W)	OTI	OTG Interrupt Mask. Mode: Host and Device
1 (R/W)	MODEMIS	Mode Mismatch Interrupt Mask. Mode: Host and Device

Interrupt Status Register

The USBC_ISTAT register interrupts the application for system-level events in the current mode (device mode or host mode). Some of the bits in this register are valid only in host mode, while others are valid in device mode only. This register also indicates the current mode. To clear the interrupt status bits of type R_SS_WC, the application must write 1'b1 into the bit.

The FIFO status interrupts are read only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the USBC_ISTAT register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Note: Read the reset value of USBC_ISTAT.CURMOD only after the following conditions:

If IDDIG_FILTER is disabled, read only after PHY clock is stable.

If IDDIG_FILTER is enabled, read only after the filter timer expires.

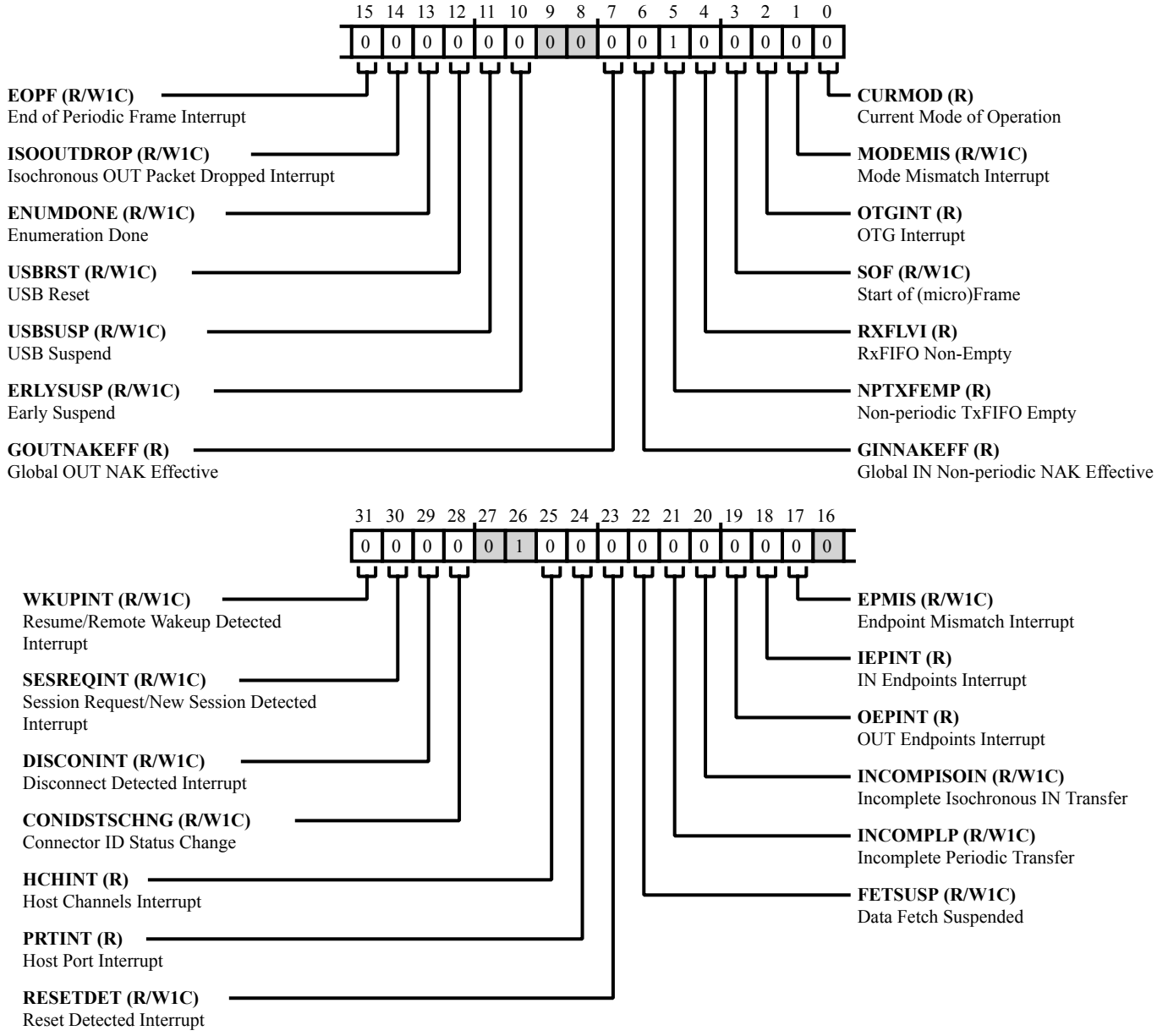


Figure 21-53: USBC_ISTAT Register Diagram

Table 21-48: USBC_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	WKUPINT	Resume/Remote Wakeup Detected Interrupt. The USBC_ISTAT.WKUPINT bit is applicable in host and device mode. Wakeup interrupt during Suspend(L2) or LPM(L1) state. During Suspend(L2): In device mode, the USBC_ISTAT.WKUPINT bit is set only when a host-initiated resume is detected on the USB. In host mode, the USBC_ISTAT.WKUPINT bit is set only when a device-initiated remote wakeup is detected on USB. During LPM(L1): In device mode, the USBC_ISTAT.WKUPINT bit is set for either host-initiated resume or device-initiated remote wakeup on USB. In host mode, the USBC_ISTAT.WKUPINT bit is set for either host-initiated resume or device-initiated remote wakeup on USB.
30 (R/W1C)	SESREQINT	Session Request/New Session Detected Interrupt. The USBC_ISTAT.SESREQINT bit is applicable in host and device mode.
29 (R/W1C)	DISCONINT	Disconnect Detected Interrupt. The USBC_ISTAT.DISCONINT bit is applicable in host mode. Disconnect Detected Interrupt (DisconnInt) The USBC_ISTAT.DISCONINT bit is set (=1) when a device disconnect is detected.
28 (R/W1C)	CONIDSTSCHNG	Connector ID Status Change. The USBC_ISTAT.CONIDSTSCHNG bit is applicable in host and device mode. The core sets the USBC_ISTAT.CONIDSTSCHNG bit when there is a change in connector ID status.
25 (R/NW)	HCHINT	Host Channels Interrupt. The USBC_ISTAT.HCHINT bit is applicable in host mode. The core sets the USBC_ISTAT.HCHINT bit to indicate that an interrupt is pending on one of the channels of the core (in Host mode). The application must read the USBC_ISTAT_H register to determine the exact number of the channel on which the interrupt occurred, and Then read the corresponding USBC_ISTAT[n]_H register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the USBC_ISTAT[n]_H register to clear this bit.

Table 21-48: USBC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/NW)	PRTINT	Host Port Interrupt. The USBC_ISTAT.PRTINT bit is applicable in host mode. The core sets the USBC_ISTAT.PRTINT bit to indicate a change in port status of one of the controller ports in host mode. The application must read the USBC_PORT_CTL_H register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the Host Port Control and Status register to clear the USBC_ISTAT.PRTINT bit.
23 (R/W1C)	RESETDET	Reset Detected Interrupt. The USBC_ISTAT.RESETDET bit is applicable in device mode. In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in Suspend. In host mode, this interrupt is not asserted.
22 (R/W1C)	FETSUSP	Data Fetch Suspended. The USBC_ISTAT.FETSUSP bit is applicable in device mode. This interrupt indicates that the core has stopped fetching data. For IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application: - Sets a global non-periodic IN NAK handshake - Disables IN endpoints - Flushes the FIFO - Determines the token sequence from the IN token sequence learning queue - Re-enables the endpoints - Clears the global non-periodic IN NAK handshake If the global non-periodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received. The core generates an 'IN token received when FIFO empty' interrupt. The USBC_otg then sends the host a NAK response. To avoid this scenario, the application can check the USBC_ISTAT.FETSUSP interrupt, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the IN token received when FIFO empty interrupt when clearing a global IN NAK handshake.

Table 21-48: USBC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
21 (R/W1C)	INCOMPLP	Incomplete Periodic Transfer. The USBC_ISTAT.INCOMPLP bit is applicable in host and device mode. In host mode, the core sets the USBC_ISTAT.INCOMPLP bit when there are incomplete periodic transactions still pending which are scheduled for the current microframe. Incomplete Isochronous OUT Transfer (incompISOOUT) In device mode, the core sets (=1) the USBC_ISTAT.INCOMPLP bit to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current microframe. The USBC_ISTAT.INCOMPLP bit is set along with the USBC_ISTAT.EOPF bit.
20 (R/W1C)	INCOMPISOIN	Incomplete Isochronous IN Transfer. The USBC_ISTAT.INCOMPISOIN bit is applicable in device mode. The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current microframe. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register. Note: This interrupt is not asserted in scatter/gather DMA mode.
19 (R/NW)	OEPINT	OUT Endpoints Interrupt. The USBC_ISTAT.OEPINT bit is applicable in device mode. The controller sets the USBC_ISTAT.OEPINT bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the USBC_ISTAT_D register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding USBC_ISTAT_OEP[n]_D register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding USBC_ISTAT_OEP[n]_D register to clear the USBC_ISTAT.OEPINT bit.
18 (R/NW)	IEPINT	IN Endpoints Interrupt. The USBC_ISTAT.IEPINT bit is applicable in device mode. The core sets the USBC_ISTAT.IEPINT bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the USBC_ISTAT_D register to determine the exact number of the IN endpoint on the USBC_ISTAT_IEP[n]_D register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding USBC_ISTAT_IEP[n]_D register to clear the USBC_ISTAT.IEPINT bit.

Table 21-48: USBC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W1C)	EPMIS	Endpoint Mismatch Interrupt. The USBC_ISTAT.EPMIS bit is applicable in device mode. Note: This interrupt is valid only in shared FIFO operation. The USBC_ISTAT.EPMIS bit indicates that an IN token has been received for a non-periodic endpoint, but the data for another endpoint is present in the top of the non-periodic transmit FIFO and the IN endpoint mismatch count programmed by the application has expired.
15 (R/W1C)	EOPF	End of Periodic Frame Interrupt. The USBC_ISTAT.EOPF bit is applicable in device mode. The USBC_ISTAT.EOPF bit indicates that the period specified in the USBC_CFG_D.PERFRINT field has been reached in the current microframe.
14 (R/W1C)	ISOOUTDROP	Isochronous OUT Packet Dropped Interrupt. The USBC_ISTAT.ISOOUTDROP bit is applicable in device mode. The controller sets the USBC_ISTAT.ISOOUTDROP bit when it fails to write an isochronous OUT packet into the RxFIFO because the RxFIFO does not have enough space to accommodate a maximum packet size packet for the isochronous OUT endpoint.
13 (R/W1C)	ENUMDONE	Enumeration Done. The USBC_ISTAT.ENUMDONE bit is applicable in device mode. The core sets the USBC_ISTAT.ENUMDONE bit to indicate that speed enumeration is complete. The application must read the USBC_STAT_D register to obtain the enumerated speed.
12 (R/W1C)	USBRST	USB Reset. The USBC_ISTAT.USBRST bit is applicable in device mode. The controller sets the USBC_ISTAT.USBRST bit to indicate that a reset is detected on the USB.
11 (R/W1C)	USBSUSP	USB Suspend. The USBC_ISTAT.USBSUSP bit is applicable in device mode. The controller sets the USBC_ISTAT.USBSUSP bit to indicate that a suspend was detected on the USB. The controller enters the suspended state when there is no activity on the linestate signal for an extended period of time.
10 (R/W1C)	ERLYSUSP	Early Suspend. The USBC_ISTAT.ERLYSUSP bit is applicable in device mode. The controller sets the USBC_ISTAT.ERLYSUSP bit to indicate that an idle state has been detected on the USB for 3 ms.

Table 21-48: USBC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	GOUTNAKEFF	Global OUT NAK Effective. The USBC_ISTAT.GOUTNAKEFF bit is applicable in device mode. The USBC_ISTAT.GOUTNAKEFF bit indicates that the USBC_CTL_D.SGOUTNAK bit, set by the application, has taken effect in the core. The USBC_ISTAT.GOUTNAKEFF bit can be cleared by writing to the USBC_CTL_D.CGOUTNAK bit.
6 (R/NW)	GINNAKEFF	Global IN Non-periodic NAK Effective. The USBC_ISTAT.GINNAKEFF bit is applicable in device mode. The USBC_ISTAT.GINNAKEFF bit indicates that the USBC_CTL_D.SGNPINNAK bit, set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. The USBC_ISTAT.GINNAKEFF bit can be cleared by clearing the USBC_CTL_D.CGNPINNAK bit. This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.
5 (R/NW)	NPTXFEMP	Non-periodic TxFIFO Empty. The USBC_ISTAT.NPTXFEMP bit is applicable in host and device mode. The USBC_ISTAT.NPTXFEMP bit indicates that an interrupt is asserted when the non-periodic TxFIFO is either half or completely empty, and there is space for at least one Entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the USBC_AHB_CFG.NPTXFELVL bit. In host mode, the application can use the USBC_ISTAT.NPTXFEMP bit In device mode, the application uses uses USBC_ISTAT_IEP[n]_D.TXFEMP.
4 (R/NW)	RXFLVI	RxFIFO Non-Empty. The USBC_ISTAT.RXFLVI bit is applicable in host and device mode. The USBC_ISTAT.RXFLVI bit indicates that there is at least one packet pending to be read from the RxFIFO.

Table 21-48: USBC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W1C)	SOF	Start of (micro)Frame. The USBC_ISTAT.SOF bit is applicable in host and device mode. In host mode, the core sets the USBC_ISTAT.SOF bit to indicate that an SOF (FS), micro-SOF (HS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to the USBC_ISTAT.SOF bit to clear the interrupt. In device mode, the controller sets the USBC_ISTAT.SOF bit to indicate that an SOF token has been received on the USB. The application can read the Device Status register to get the current (micro)Frame number. This interrupt is seen only when the core is operating at either HS or FS. The USBC_ISTAT.SOF bit can be set only by the core; the application must write 1 to clear it. Note: This register may return 1'b1 if read immediately after power-on reset. If the register bit reads 1'b1 immediately after power-on reset, it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.
2 (R/NW)	OTGINT	OTG Interrupt. The USBC_ISTAT.OTGINT bit is applicable in host and device mode. The controller sets the USBC_ISTAT.OTGINT bit to indicate an OTG protocol event. The application must read the USBC_OTG_IRQ register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the USBC_OTG_IRQ register to clear thE USBC_ISTAT.OTGINT bit.
1 (R/W1C)	MODEMIS	Mode Mismatch Interrupt. The USBC_ISTAT.MODEMIS bit is applicable in host and device mode. The core sets the USBC_ISTAT.MODEMIS bit when the application is trying to access: - A host mode register, when the controller is operating in device mode - A device mode register, when the controller is operating in host mode The register access is completed on the bus with an OKAY response, but is ignored by the controller internally and does not affect the operation of the controller. The USBC_ISTAT.MODEMIS bit can be set only by the core; the application should write 1 to clear it.

Table 21-48: USBC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	CURMOD	Current Mode of Operation. The `USBC_ISTAT.CURMOD` bit is applicable in host and device mode. Current Mode of Operation (CurMod) The `USBC_ISTAT.CURMOD` bit indicates the current mode. Note: The reset value of this register field can be read only after the PHY clock is stable.	
		0	Device mode
		1	Host mode

Non-periodic Transmit FIFO Size Register

The application can program the RAM size and the memory start address for the non-periodic TxFIFO.

Note: The fields of the USBC_TXFIFOSZ_NP register change depending on host or device mode.

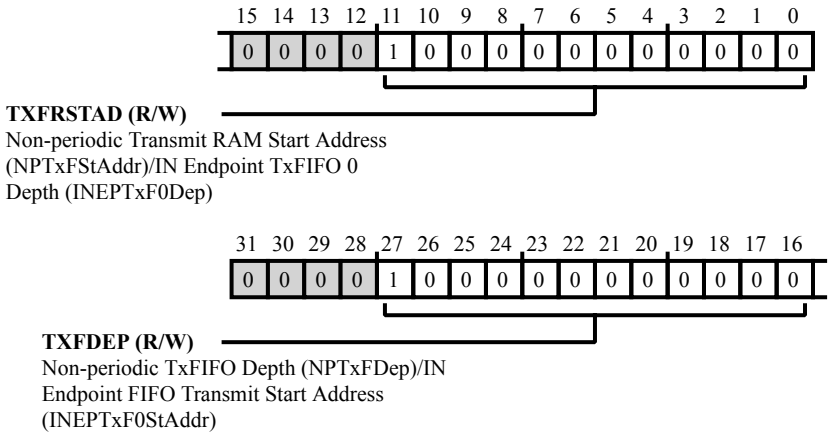


Figure 21-54: USBC_TXFIFOSZ_NP Register Diagram

Table 21-49: USBC_TXFIFOSZ_NP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:16 (R/W)	TXFDEP	Non-periodic TxFIFO Depth (NPTxFDep)/IN Endpoint FIFO Transmit Start Address (INEPTxF0StAddr). In host mode, this field contains the Non-periodic TxFIFO Depth (NPTxFDep). In device mode, this field contains the IN Endpoint FIFO Transmit Start Address (INEPTxF0StAddr). Programmed values must not exceed the power-on value. The power-on reset value of this field is 2048.
11:0 (R/W)	TXFRSTAD	Non-periodic Transmit RAM Start Address (NPTxFStAddr)/IN Endpoint TxFIFO 0 Depth (INEPTxF0Dep). In host mode, this field is Non-periodic Transmit RAM Start Address (NPTxFStAddr). In device mode, this field is IN Endpoint TxFIFO 0 Depth (INEPTxF0Dep). This value is in terms of 32-bit words. The minimum value is 16, the maximum value is 32,768, and the reset value is 256.

Non-periodic Transmit FIFO/Queue Status Register

In device mode, the USBC_TXFIFO_STAT_NP register is valid only in shared FIFO operation.

The read-only USBC_TXFIFO_STAT_NP register contains the free space information for the non-periodic TxFIFO and the non-periodic transmit request queue.

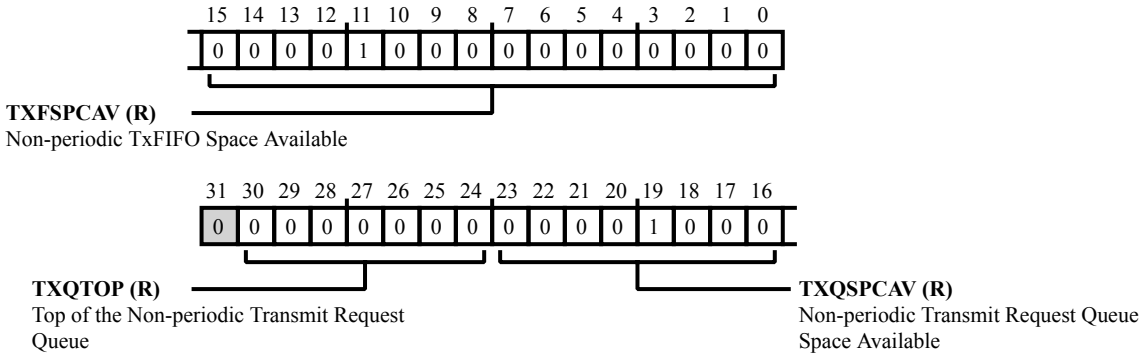


Figure 21-55: USBC_TXFIFO_STAT_NP Register Diagram

Table 21-50: USBC_TXFIFO_STAT_NP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:24 (R/NW)	TXQTOP	Top of the Non-periodic Transmit Request Queue. Entry in the Non-periodic Tx Request Queue that is currently being processed by the MAC. - Bits [30:27]: Channel/endpoint number - Bits [26:25]: - 2'b00: IN/OUT token -- 2'b01: Zero-length transmit packet (device IN/host OUT) -- 2'b10: PING/CSPLIT token -- 2'b11: Channel halt command - Bit [24]: Terminate (last Entry for selected channel/endpoint)	
23:16 (R/NW)	TXQSPCAV	Non-periodic Transmit Request Queue Space Available. The USBC_TXFIFO_STAT_NP.TXQSPCAV bit field indicates the amount of free space available in the non-periodic transmit request queue. This queue holds both IN and OUT requests in host mode. Device mode has only IN requests. Values are in terms of 32-bit words.	
		0	Queue Full
		1	1 Location Available
		2	2 Locations Available

Table 21-50: USBC_TXFIFO_STAT_NP Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	3 Locations Available
		4	4 Locations Available
		5	5 Locations Available
		6	6 Locations Available
		7	7 Locations Available
		8	8 Locations Available
15:0 (R/NW)	TXFSPCAV	Non-periodic TxFIFO Space Available. The `USBC_TXFIFO_STAT_NP.TXFSPCAV` bit field indicates the amount of free space available in the Non-periodic TxFIFO. Values are in terms of 32-bit words. 16'hn: n words available, where 0 <= n <= 32,768) reserved where n > 32,768, and full where n = 0.	
		0	TxFIFO Full

OTG Control and Status Register

The USBC_OTG_CTL register controls the behavior and reflects the status of the OTG function of the controller.

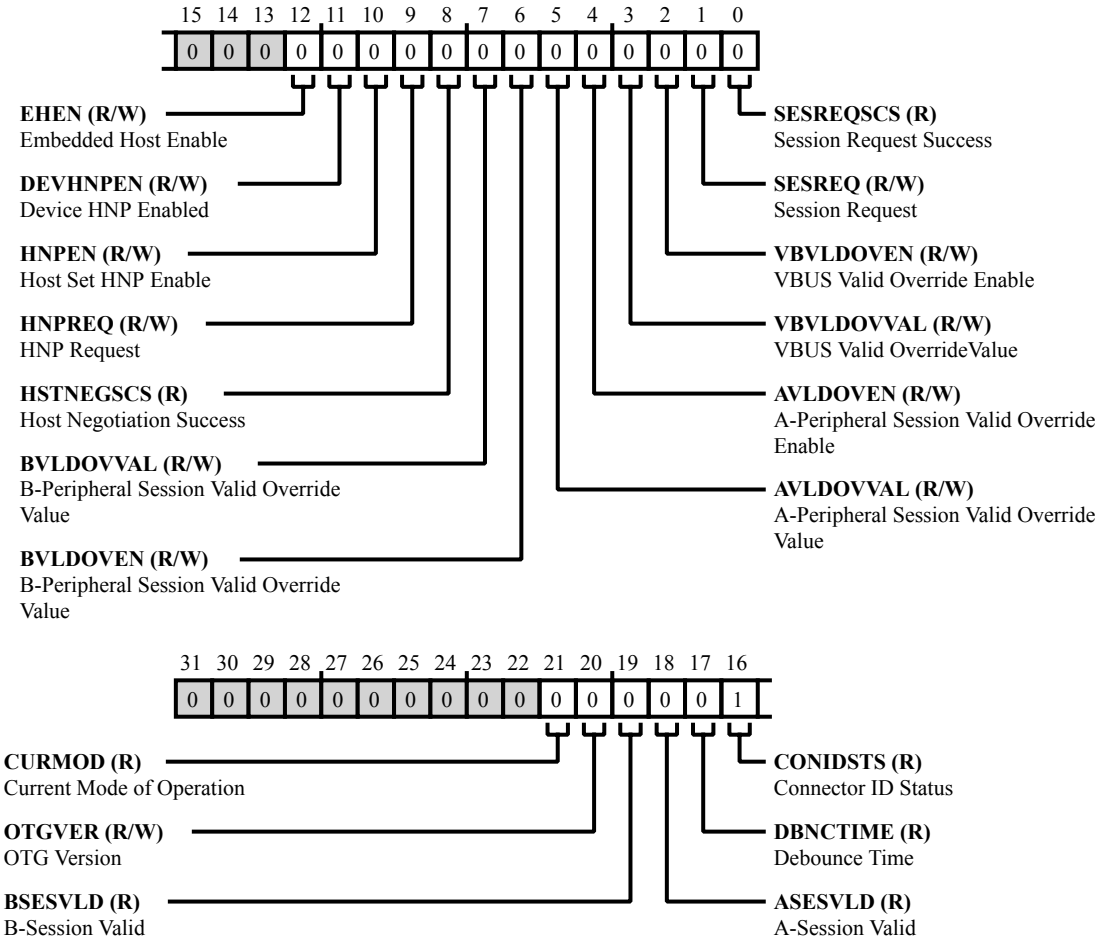


Figure 21-56: USBC_OTG_CTL Register Diagram

Table 21-51: USBC_OTG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/NW)	CURMOD	Current Mode of Operation. Current Mode of Operation (CurMod) Mode: Host and Device The USBC_OTG_CTL.CURMOD bit indicates the current mode. Reset:1'b0 Note: The reset value of this register field can be read only after the PHY clock is stable.	
		0	Host Mode
		1	Device Mode

Table 21-51: USBC_OTG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	OTGVER	OTG Version. The USBC_OTG_CTL.OTGVER bit indicates the OTG revision.	
		0	OTG Version 1.3: Core supports data line pulsing and VBus pulsing for SRP
		1	OTG Version 2.0: Core supports only data line pulsing for SRP
19 (R/NW)	BSESVLD	B-Session Valid. Mode: Device only The USBC_OTG_CTL.BSESVLD bit indicates the device mode transceiver status. In OTG mode, the USBC_OTG_CTL.BSESVLD bit is used to determine if the device is connected or disconnected. Note:The reset value of the USBC_OTG_CTL.BSESVLD bit is 1'b0. - The reset value of this register field can be read only after the PHY clock is stable.	
		0	B-session is not valid
		1	B-session is valid
18 (R/NW)	ASESVLD	A-Session Valid. Mode: Host only The USBC_OTG_CTL.ASESVLD bit indicates the host mode transceiver status. Note:The reset value of the USBC_OTG_CTL.ASESVLD bit is 1'b0.	
		0	A-session is not valid
		1	A-session is valid
17 (R/NW)	DBNCTIME	Debounce Time. Mode: Host only The USBC_OTG_CTL.DBNCTIME bit indicates the debounce time of a detected connection.	
		0	Long debounce time, used for physical connections (100 ms + 2.5 micro-sec)
		1	Short debounce time, used for soft connections (2.5 micro-sec)

Table 21-51: USBC_OTG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	CONIDSTS	Connector ID Status. Mode: Host and Device The `USBC_OTG_CTL.CONIDSTS` bit indicates the connector ID status on a connect event. Note: The reset value of this register field can be read only after the PHY clock is stable	
		0	The core is in A-Device mode
		1	The core is in B-Device mode
12 (R/W)	EHEN	Embedded Host Enable. Mode: SRP Capable Host The `USBC_OTG_CTL.EHEN` bit is used to select between OTG A device state machine and embedded host state machine.	
		0	OTG A Device state machine is selected
		1	Embedded Host State Machine is selected
11 (R/W)	DEVHNPEN	Device HNP Enabled. Mode: HNP Capable OTG Device The application sets the `USBC_OTG_CTL.DEVHNPEN` bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.	
		0	HNP Not Enabled
		1	HNP Enabled
10 (R/W)	HNPEN	Host Set HNP Enable. Mode: HNP Capable OTG Host The application sets the `USBC_OTG_CTL.HNPEN` bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.	
		0	Host Set HNP is not enabled
		1	Host Set HNP is enabled

Table 21-51: USBC_OTG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	HNPREQ	HNP Request. Mode: HNP Capable OTG Device The application sets the USBC_OTG_CTL.HNPREQ bit to initiate an HNP request to the connected USB host. The application can clear the USBC_OTG_CTL.HNPREQ bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (USBC_OTG_IRQ.HNEGSSTSCHG) is set. The controller clears the USBC_OTG_CTL.HNPREQ bit when the USBC_OTG_IRQ.HNEGSSTSCHG bit is cleared.	
		0	No HNP request
		1	HNP request
8 (R/NW)	HSTNEGSCS	Host Negotiation Success. Mode: HNP-capable Device The controller sets the USBC_OTG_CTL.HSTNEGSCS bit when host negotiation is successful. The controller clears the USBC_OTG_CTL.HNPREQ bit is set.	
		0	Host negotiation failure
		1	Host negotiation success
7 (R/W)	BVLDOVVAL	B-Peripheral Session Valid Override Value. Mode: Device only The USBC_OTG_CTL.BVLDOVVAL bit is used to set Override value for Bvalid signal when the USBC_OTG_CTL.BVLDOVEN bit is set.	
		0	Bvalid value is 0
		1	Bvalid value is 1
6 (R/W)	BVLDOVEN	B-Peripheral Session Valid Override Value. Mode: Device only The USBC_OTG_CTL.BVLDOVEN bit is used to enable/disable the software to override the Bvalid signal using the USBC_OTG_CTL.BVLDOVVAL bit.	
		0	Override is disabled and bvalid signal from the respective PHY selected is used internally by the force
		1	Internally Bvalid received from the PHY is overridden

Table 21-51: USBC_OTG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	AVLDOVVAL	A-Peripheral Session Valid Override Value. Mode: Host only The USBC_OTG_CTL.AVLDOVVAL bit is used to set override value for the Avalid signal when the USBC_OTG_CTL.AVLDOVEN bit is set.	
		0	Avalid value is 0
		1	Avalid value is 1
4 (R/W)	AVLDOVEN	A-Peripheral Session Valid Override Enable. Mode: Host only The USBC_OTG_CTL.AVLDOVEN bit is used to enable/disable the software to override the Avalid signal using the USBC_OTG_CTL.AVLDOVVAL bit.	
		0	Override is disabled and Avalid signal from the respective PHY selection is used internally by the core
		1	Internally Avalid received from the PHY is overridden
3 (R/W)	VBVLDOVVAL	VBUS Valid OverrideValue. Mode: Host only The USBC_OTG_CTL.VBVLDOVVAL bit is used to set override value for vbusvalid signal when the USBC_OTG_CTL.VBVLDOVEN is set.	
		0	vbusvalid value is 0
		1	vbusvalid value is 1
2 (R/W)	VBVLDOVEN	VBUS Valid Override Enable. Mode: Host only The USBC_OTG_CTL.VBVLDOVEN bit is used to enable/disable the software to override the Bvalid signal using the USBC_OTG_CTL.VBVLDOVVAL bit.	
		0	Override is disabled and bvalid signal from the respective PHY selected is used internally by the controller
		1	Internally Bvalid received from the PHY is overridden

Table 21-51: USBC_OTG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	SESREQ	Session Request. Mode: Device only The application sets the USBC_OTG_CTL.SESREQ bit to initiate a session request on the USB. The application can clear the USBC_OTG_CTL.SESREQ bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (USBC_OTG_IRQ.HNEGSSTSCHG) is set. The core clears the USBC_OTG_CTL.SESREQ bit when the USBC_OTG_IRQ.HNEGSSTSCHG bit is cleared. If you use the USB 1.1 Full-Speed Serial Transceiver interface to initiate the session request, the application must wait until the VBUS discharges to 0.2 V, after the B-Session Valid bit in this register (USBC_OTG_CTL.BSESVLD) is cleared. This discharge time varies between different PHYs and can be obtained from the PHY vendor.	
		0	No session request
		1	Session request
0 (R/NW)	SESREQSCS	Session Request Success. Mode: Device only The core sets the USBC_OTG_CTL.SESREQSCS bit when a session request initiation is successful.	
		0	Failure
		1	Success

OTG Interrupt Register

The application reads the USBC_OTG_IRQ register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

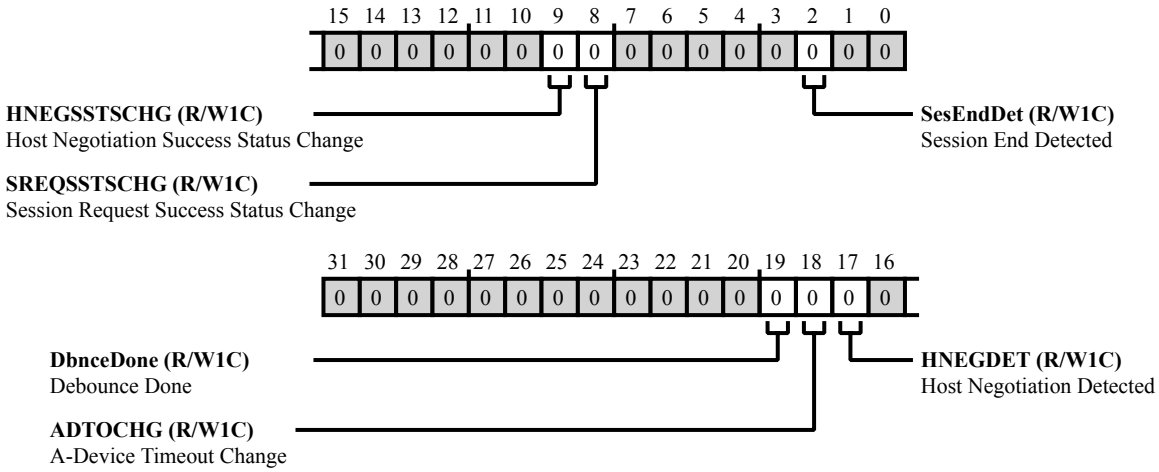


Figure 21-57: USBC_OTG_IRQ Register Diagram

Table 21-52: USBC_OTG_IRQ Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W1C)	DBNCEDONE	Debounce Done. Mode: Host only The core sets the USBC_OTG_IRQ.DBNCEDONE bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable (USBC_CFG.HNPCAP) or SRP Capable (${.:GUSBCFG:SRPCap) is set. This bit can be set only by the core and the application should write 1 to clear it.
18 (R/W1C)	ADTOCHG	A-Device Timeout Change. Mode: Host and Device The core sets the USBC_OTG_IRQ.ADTOCHG bit to indicate that the A-device has timed out while waiting for the B-device to connect. This bit can be set only by the core and the application should write 1 to clear it.
17 (R/W1C)	HNEGDET	Host Negotiation Detected. Mode:Host and Device The core sets the USBC_OTG_IRQ.HNEGDET bit when it detects a host negotiation request on the USB. This bit can be set only by the core and the application should write 1 to clear it.

Table 21-52: USBC_OTG_IRQ Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W1C)	HNEGSSTSCHG	Host Negotiation Success Status Change. Mode: Host and Device The core sets the `USBC_OTG_IRQ.HNEGSSTSCHG` bit on the success or failure of a USB host negotiation request. The application must read the Host Negotiation Success bit (`USBC_OTG_CTL.HSTNEGSCS`) to check for success or failure. This bit can be set only by the core and the application should write 1 to clear it.
8 (R/W1C)	SREQSSTSCHG	Session Request Success Status Change. Mode: Host and Device The core sets the `USBC_OTG_IRQ.SREQSSTSCHG` bit on the success or failure of a session request. The application must read the Session Request Success bit in the OTG Control and Status register (`USBC_OTG_CTL.SESREQSCS`) to check for success or failure. This bit can be set only by the core and the application should write 1 to clear it.
2 (R/W1C)	SESENDDET	Session End Detected. Mode: Host and Device This bit can be set only by the core and the application should write 1 to clear it.

PHY Interface Control Register

The application can use the USBC_PHYIF_CTL register to access PHY registers.

The application sets the USBC_PHYIF_CTL register for PHY register access. The application polls the VStatus Done bit in USBC_PHYIF_CTL register for the completion of the PHY register access.

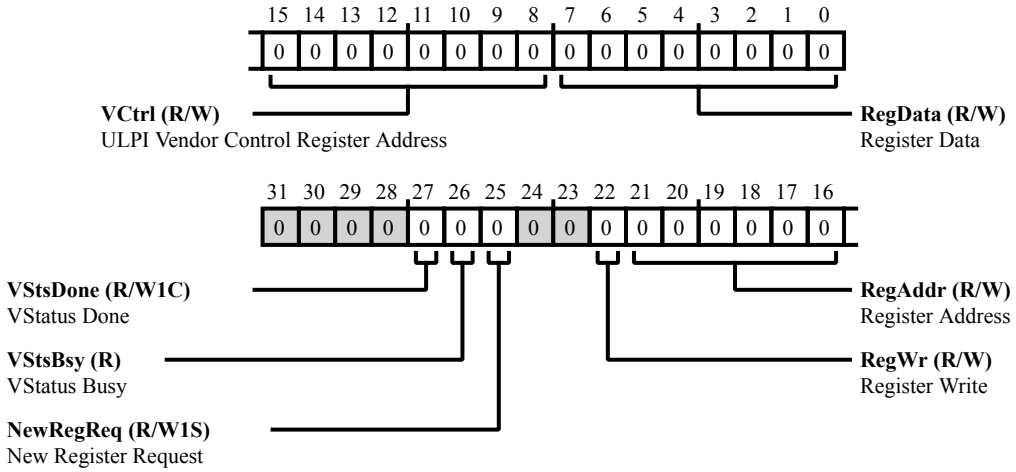


Figure 21-58: USBC_PHYIF_CTL Register Diagram

Table 21-53: USBC_PHYIF_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27 (R/W1C)	VSTSDONE	VStatus Done. The core sets the USBC_PHYIF_CTL.VSTSDONE bit when the vendor control access is done. This bit is cleared by the core when the application sets the New Register Request bit (USBC_PHYIF_CTL.NEWREGREQ).
26 (R/NW)	VSTSBSY	VStatus Busy. The core sets the USBC_PHYIF_CTL.VSTSBSY bit when the vendor control access is in progress and clears this bit when done.
25 (R/W1S)	NEWREGREQ	New Register Request. The application sets the USBC_PHYIF_CTL.NEWREGREQ bit for a new vendor control access.
22 (R/W)	REGWR	Register Write. Set the USBC_PHYIF_CTL.REGWR bit for register writes and clear it for register reads.
21:16 (R/W)	REGADDR	Register Address. The 6-bit PHY register address for immediate PHY Register set access. Set the USBC_PHYIF_CTL.REGADDRfield to 0x2F for Extended PHY Register set access.

Table 21-53: USBC_PHYIF_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	VCTRL	ULPI Vendor Control Register Address. The 4-bit register address a vendor defined 4-bit parallel output bus. Bits 11:8 of this field are placed on utmi_vcontrol[3:0]. ULPI Extended Register Address (ExtRegAddr) The 6-bit PHY extended register address.
7:0 (R/W)	REGDATA	Register Data. The `USBC_PHYIF_CTL.REGDATA` bit field contains the write data for register write. Read data for register read is valid when the `USBC_PHYIF_CTL.VSTSDONE` bit is set.

Reset Register

The USBC_RST_CTL register resets various hardware features inside the controller.

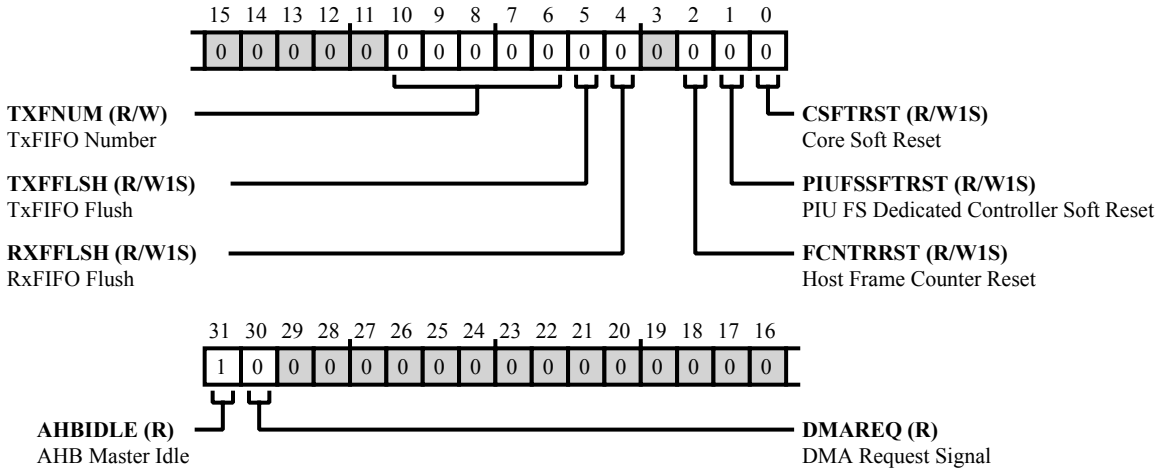


Figure 21-59: USBC_RST_CTL Register Diagram

Table 21-54: USBC_RST_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	AHBIDLE	AHB Master Idle. Mode: Host and Device The USBC_RST_CTL.AHBIDLE bit indicates that the bus master state machine is in the IDLE condition.	
30 (R/NW)	DMAREQ	DMA Request Signal. Mode: Host and Device The USBC_RST_CTL.DMAREQ bit indicates that the DMA request is in progress. Used for debug.	
10:6 (R/W)	TXFNUM	TxFIFO Number. Mode: Host and Device This is the FIFO number that must be flushed using the TxFIFO Flush bit. This field must not be changed until the core clears the TxFIFO Flush bit.	
		0	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO flush in device mode when in shared FIFO operation Tx FIFO 0 flush in device mode when in dedicated FIFO mode

Table 21-54: USBC_RST_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		1	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 1 flush in device mode when in shared FIFO operation Tx FIFO 1 flush in device mode when in dedicated FIFO mode
		2	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 2 flush in device mode when in shared FIFO operation Tx FIFO 2 flush in device mode when in dedicated FIFO mode
		3	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 3 flush in device mode when in shared FIFO operation Tx FIFO 3 flush in device mode when in dedicated FIFO mode
		4	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 4 flush in device mode when in shared FIFO operation Tx FIFO 4 flush in device mode when in dedicated FIFO mode
		5	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 5 flush in device mode when in shared FIFO operation Tx FIFO 5 flush in device mode when in dedicated FIFO mode
		6	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 6 flush in device mode when in shared FIFO operation Tx FIFO 6 flush in device mode when in dedicated FIFO mode
		7	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 7 flush in device mode when in shared FIFO operation Tx FIFO 7 flush in device mode when in dedicated FIFO mode

Table 21-54: USBC_RST_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		8	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 8 flush in device mode when in shared FIFO operation Tx FIFO 8 flush in device mode when in dedicated FIFO mode
		9	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 9 flush in device mode when in shared FIFO operation Tx FIFO 9 flush in device mode when in dedicated FIFO mode
		10	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 10 flush in device mode when in shared FIFO operation Tx FIFO 10 flush in device mode when in dedicated FIFO mode
		11	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 11 flush in device mode when in shared FIFO operation Tx FIFO 11 flush in device mode when in dedicated FIFO mode
		12	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 12 flush in device mode when in shared FIFO operation Tx FIFO 12 flush in device mode when in dedicated FIFO mode
		13	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 13 flush in device mode when in shared FIFO operation Tx FIFO 13 flush in device mode when in dedicated FIFO mode
		14	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 14 flush in device mode when in shared FIFO operation Tx FIFO 14 flush in device mode when in dedicated FIFO mode

Table 21-54: USBC_RST_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		15	Non-periodic TxFIFO flush in host mode Non-periodic TxFIFO 15 flush in device mode when in shared FIFO operation Tx FIFO 15 flush in device mode when in dedicated FIFO mode
		16	Flush all transmit FIFOs in device or host mode
5 (R/W1S)	TXFFLSH	TxFIFO Flush. Mode: Host and Device The USBC_RST_CTL.TXFFLSH bit selectively flushes a single or all transmit FIFOs, but cannot do so If the core is in the midst of a transaction. The application must write this bit only after checking that the core is neither writing to the TxFIFO nor reading from the TxFIFO. Verify using these registers: - Read the NAK Effective Interrupt to ensure the core is not reading from the FIFO. - Write the USBC_RST_CTL.AHBIDLE bit to ensure the core is not writing anything to the FIFO. Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower of PHY or bus clock.	
4 (R/W1S)	RXFFLSH	RxFIFO Flush. Mode: Host and Device The application can flush the entire RxFIFO using the USBC_RST_CTL.RXFFLSH bit, but must first ensure that the core is not in the middle of a transaction. The application must only write to the USBC_RST_CTL.RXFFLSH bit after checking that the controller is neither reading from the RxFIFO nor writing to the RxFIFO. The application must wait until the USBC_RST_CTL.RXFFLSH bit is cleared before performing any other operations. This bit requires eight clocks (slowest of PHY or bus clock) to clear.	
2 (R/W1S)	FCNTRRST	Host Frame Counter Reset. Mode: Host only The application writes to the USBC_RST_CTL.FCNTRRST bit to reset the (micro)frame number counter inside the core. When the (micro)frame counter is reset, the subsequent SOF sent out by the core has a (micro)frame number of 0. When application writes 1 to the bit, it might not be able to read back the value as it will get cleared by the core in a few clock cycles.	

Table 21-54: USBC_RST_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W1S)	PIUFSSFTRST	PIU FS Dedicated Controller Soft Reset. Mode: Host and Device The `USBC_RST_CTL.PIUFSSFTRST` bit resets the PIU FS dedicated controller. All module state machines in FS dedicated controller of the PIU are reset to the IDLE state. The bit is used to reset the FS dedicated controller in PIU in case of any PHY errors like loss of activity or babble error resulting in the PHY remaining in RX state for more than one frame boundary. The `USBC_RST_CTL.PIUFSSFTRST` bit is a self clearing bit and core clears this bit after all the necessary logic is reset in the core.
0 (R/W1S)	CSFTRST	Core Soft Reset. Mode: Host and Device Resets the hclk and phy_clock domains as follows: - Clears the interrupts and all the CSR registers except the following register bits: `USBC_PWR_CTL.RSTPDWNMODULE`, `USBC_CFG.FRCDEVMOD`, `USBC_CFG.FRCHSTMOD`, `USBC_CFG.DDRSEL`, `USBC_CFG.PHYSEL`, `USBC_CFG.FSINTF`, `USBC_CFG.ULPI_UTMI_SEL`, `USBC_CFG.PHYIF`, `USBC_CFG.TXENDDLY`, `USBC_CFG.TSDLPULSE`, `USBC_CFG.CLKSUSM`, `USBC_CFG.AUTORES`, `USBC_CFG_H.PCLKSEL`, `USBC_CFG_D.SPD`, and `USBC_CTL_D.SFTDISCON`: - All module state machines: - All module state machines (except the bus completer unit) are reset to the IDLE state, and all the transmit FIFOs and the receive FIFO are flushed.: - Any transactions on the bus requester are terminated as soon as possible, after gracefully completing the last data phase of a bus transfer. Any transactions on the USB are terminated immediately.: The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit is cleared software must wait at least 3 PHY clocks before doing any access to the PHY domain (synchronization delay). Software must also must check that bit 31 of this register is 1 (bus requester is IDLE) before starting any operation. Typically software reset is used during software development and also when dynamically changing the PHY selection bits in the USB configuration registers listed above. When the PHY is changed, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.

Receive FIFO Size Register

The USBC_RXFIFOSZ register indicates the RAM size that must be allocated to the RxFIFO.

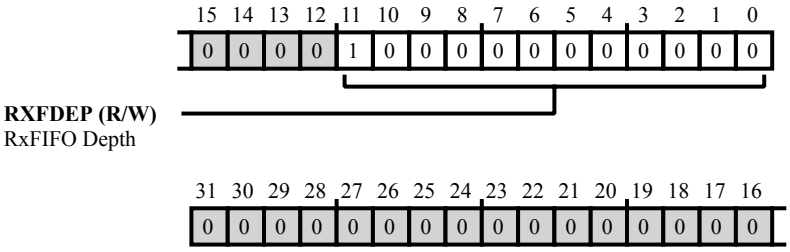


Figure 21-60: USBC_RXFIFOSZ Register Diagram

Table 21-55: USBC_RXFIFOSZ Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	RXFDEP	RxFIFO Depth. Mode: Host and Device The USBC_RXFIFOSZ.RXFDEP field indicates the RxFIFO depth. This value is in terms of 32-bit words. - Minimum value is 16 - Maximum value is 32,768 The power-on reset value of this register is specified as the largest Rx data FIFO depth The reset value is 2048.

Receive Status Read/Pop Register

A read to the USBC_RXDATA_STAT register returns the contents of the top of the receive FIFO and additionally pops the top data entry out of the RxFIFO.

The receive status contents must be interpreted differently in host and device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 32'h0000_0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (USBC_ISTAT.RXFLVI) is asserted.

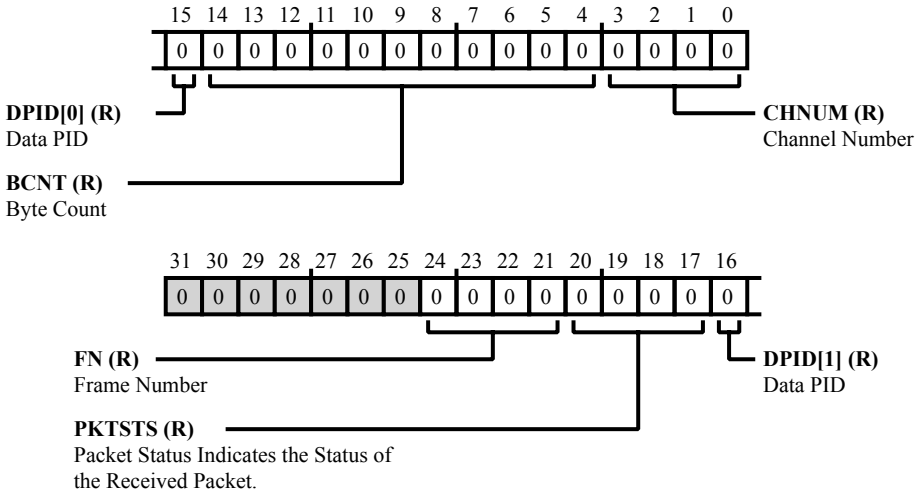


Figure 21-61: USBC_RXDATA_STAT Register Diagram

Table 21-56: USBC_RXDATA_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24:21 (R/NW)	FN	Frame Number. Mode: Device only The USBC_RXDATA_STAT.FN bit field is the least significant 4 bits of the (micro)Frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Table 21-56: USBC_RXDATA_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20:17 (R/NW)	PKTSTS	Packet Status Indicates the Status of the Received Packet.. In host mode, - 4'b0010: IN data packet received - 4'b0011: IN transfer completed (triggers an interrupt) - 4'b0101: Data toggle error (triggers an interrupt) - 4'b0111: Channel halted (triggers an interrupt) - Others: Reserved Reset:4'b0 In device mode, - 4'b0001: Global OUT NAK (triggers an interrupt) - 4'b0010: OUT data packet received - 4'b0011: OUT transfer completed (triggers an interrupt) - 4'b0100: SETUP transaction completed (triggers an interrupt) - 4'b0110: SETUP data packet received - Others: Reserved Reset:4'h0
16:15 (R/NW)	DPID	Data PID. In host mode, the `USBC_RXDATA_STAT.DPID` bit field indicates the data PID of the received packet. In device mode, the `USBC_RXDATA_STAT.DPID` bit field indicates the data PID of the received OUT data packet. Reset: 2'h0
		0 DATA0
		1 DATA1
		2 DATA2
		3 MDATA
14:4 (R/NW)	BCNT	Byte Count. In host mode, the `USBC_RXDATA_STAT.BCNT` bit indicates the byte count of the received IN data packet. In device mode, the `USBC_RXDATA_STAT.BCNT` bit indicates the byte count of the received data packet.

Table 21-56: USBC_RXDATA_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/NW)	CHNUM	Channel Number. Mode: Host only The `USBC_RXDATA_STAT.CHNUM` bit indicates the channel number to which the current received packet belongs. Endpoint Number (EPNum) Mode: Device only The `USBC_RXDATA_STAT.CHNUM` bit indicates the endpoint number to which the current received packet belongs.

Receive Status Debug Read Register

A read to the USBC_RXDBG_STAT register returns the contents of the top of the receive FIFO.

The receive status contents must be interpreted differently in Host and device modes. The core ignores the receive status read when the receive FIFO is empty and returns a value of 32'h0000_0000.

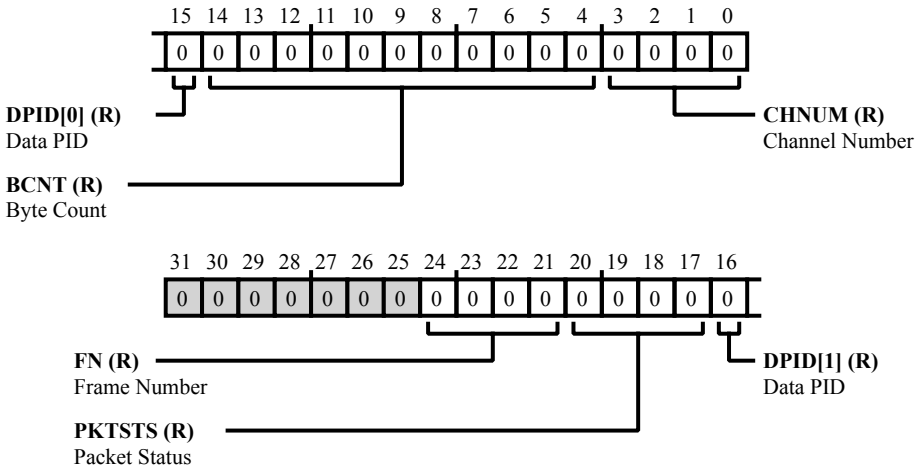


Figure 21-62: USBC_RXDBG_STAT Register Diagram

Table 21-57: USBC_RXDBG_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24:21 (R/NW)	FN	Frame Number. Mode: Device only The USBC_RXDBG_STAT.FN field is the least significant 4 bits of the (micro)frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Table 21-57: USBC_RXDBG_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20:17 (R/NW)	PKTSTS	Packet Status. In host mode, the `USBC_RXDBG_STAT.PKTSTS`field indicates the status of the received packet. - 4'b0010: IN data packet received - 4'b0011: IN transfer completed (triggers an interrupt) - 4'b0101: Data toggle error (triggers an interrupt) - 4'b0111: Channel halted (triggers an interrupt) - Others: Reserved Reset:4'b0 In device mode, - 4'b0001: Global OUT NAK (triggers an interrupt) - 4'b0010: OUT data packet received - 4'b0011: OUT transfer completed (triggers an interrupt) - 4'b0100: SETUP transaction completed (triggers an interrupt) - 4'b0110: SETUP data packet received - Others: Reserved Reset:4'h0	
16:15 (R/NW)	DPID	Data PID. In host mode, the `USBC_RXDBG_STAT.DPID` field indicates the data PID of the received packet. In device mode, the `USBC_RXDBG_STAT.DPID` field indicates the data PID of the received OUT data packet. Reset: 2'h0	
		0	DATA0
		1	DATA2
		2	DATA1
		3	MDATA
14:4 (R/NW)	BCNT	Byte Count. In host mode, the `USBC_RXDBG_STAT.BCNT` field indicates the byte count of the received IN data packet. In device mode, the `USBC_RXDBG_STAT.BCNT` field indicates the byte count of the received data packet.	

Table 21-57: USBC_RXDBG_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/NW)	CHNUM	Channel Number. Mode: Host only The `USBC_RXDBG_STAT.CHNUM` field indicates the channel number to which the current received packet belongs. Endpoint Number (EPNum) Mode: Device only The `USBC_RXDBG_STAT.CHNUM` field indicates the endpoint number to which the current received packet belongs.

Module ID Register

The USBC_MODID register (read-only) contains the module ID.

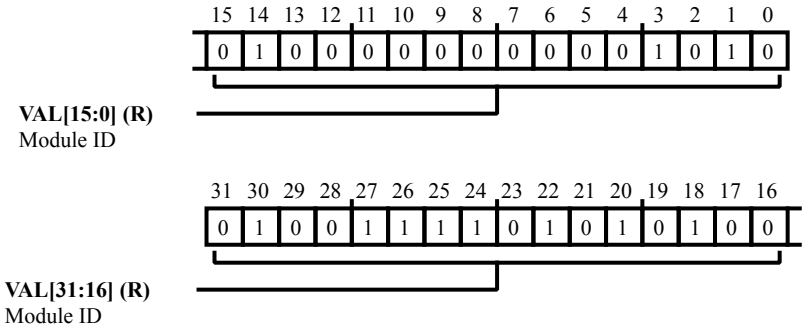


Figure 21-63: USBC_MODID Register Diagram

Table 21-58: USBC_MODID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VAL	Module ID. Module ID of the controller being used currently.

USB Configuration Register

The USBC_CFG register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the bus or the USB. Do not make changes to this register after the initial programming.

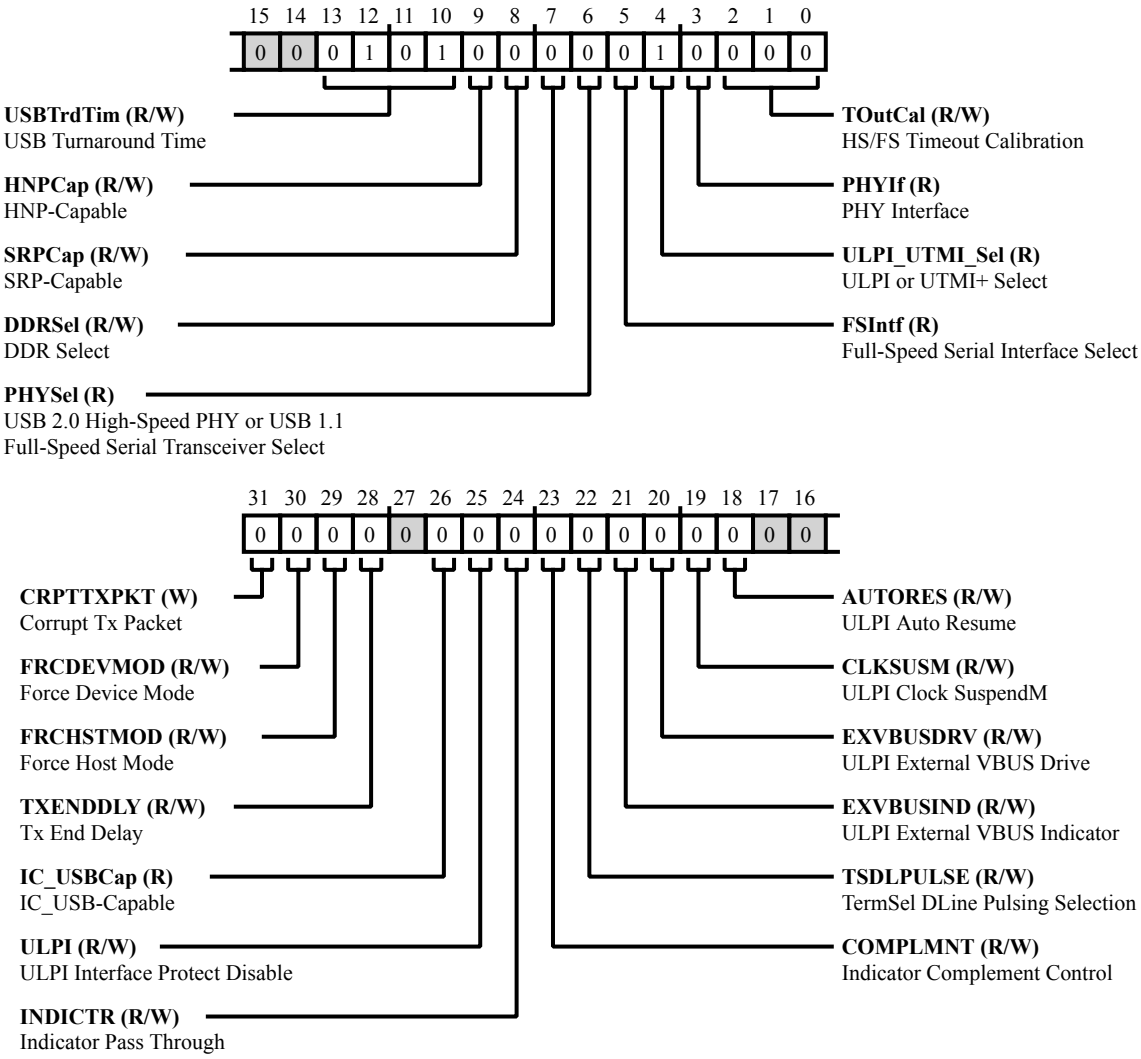


Figure 21-64: USBC_CFG Register Diagram

Table 21-59: USBC_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (RX/W)	CRPTTXPKT	Corrupt Tx Packet. Mode: Host and device The `USBC_CFG.CRPTTXPKT` bit is for debug purposes only. Never set this bit to 1. The application should always write zero to this bit.	
30 (R/W)	FRCDEVMOD	Force Device Mode. Mode:Host and device Writing a 1 to the `USBC_CFG.FRCDEVMOD` bit forces the controller to device mode. After setting the force bit, the application must wait at least 25 ms before the change to take effect.	
		0	Normal Mode
		1	Force Device Mode
29 (R/W)	FRCHSTMOD	Force Host Mode. Mode: Host and device Writing a 1 to the `USBC_CFG.FRCHSTMOD` bit forces the core to host mode. After setting the force bit, the application must wait at least 25 ms before the change to take effect.	
		0	Normal Mode
		1	Force Host Mode
28 (R/W)	TXENDDLY	Tx End Delay. Mode: Device only Writing a one to the `USBC_CFG.TXENDDLY` bit enables the controller to follow the TxEndDelay timings as per UTMI+ specification 1.05 section 4.1.5 for opmode signal during remote wakeup.	
		0	Normal Mode
		1	Tx End Delay
26 (R/NW)	IC_USBCAP	IC_USB-Capable. Mode: Host and Device The application uses the `USBC_CFG.IC_USBCAP` bit to control the core's IC_USB capabilities.	
		0	IC_USB PHY Interface Not Selected
		1	IC_USB PHY Interface Selected

Table 21-59: USBC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/W)	ULPI	ULPI Interface Protect Disable. Mode: Host only Controls circuitry built into the PHY for protecting the ULPI interface when the link tri-states STP and data. Any pull-ups or pull-downs employed by this feature can be disabled.	
		0	Enable Interface Protect Circuit
		1	Disable Interface Protect Circuit
24 (R/W)	INDICTR	Indicator Pass Through. Mode: Host only Controls whether the complement output is qualified with the internal Vbus valid comparator before being used in the Vbus state in the RX CMD.	
		0	Complement Output Signal Qualified with Internal VbusValid Comparator
		1	Complement Output Signal Not Qualified with Internal VbusValid Comparator
23 (R/W)	COMPLMNT	Indicator Complement Control. Mode: Host only The `USBC_CFG.COMPLMNT` bit controls the PHY to invert the ExternalVbusIndicator input signal, generating the complement output.	
		0	Do Not Invert ExternalVbusIndicator Signal
		1	Invert ExternalVbusIndicator Signal
22 (R/W)	TSDLPULSE	TermSel DLine Pulsing Selection. Mode: Device only The `USBC_CFG.TSDLPULSE` bit selects utmi_termselect to drive data line pulse during SRP.	
		0	Data Line Pulsing Using utmi_txvalid (Default)
		1	Data Line Pulsing Using utmi_termsel
21 (R/W)	EXVBUSIND	ULPI External VBUS Indicator. Mode: Host only The `USBC_CFG.EXVBUSIND` bit indicates to the ULPI PHY to use an external VBUS overcurrent indicator.	
		0	Internal VBUS Valid Comparator
		1	External VBUS Valid Comparator

Table 21-59: USBC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	EXVBUSDRV	ULPI External VBUS Drive. Mode: Host only The USBC_CFG.EXVBUSDRV bit selects between internal or external supply to drive 5V on VBUS, in ULPI PHY.	
		0	Internal Charge Pump (Default)
		1	External Supply
19 (R/W)	CLKSUSM	ULPI Clock SuspendM. Mode: Host and Device The USBC_CFG.CLKSUSM bit sets the ClockSuspendM bit in the interface control register on the PHY. This bit applies only in serial or carkit modes.	
		0	PHY powers down internal clock during suspend
		1	PHY does not power down internal clock
18 (R/W)	AUTORES	ULPI Auto Resume. Mode: Host and Device The USBC_CFG.AUTORES bit sets the AutoResume bit in the interface control register on the PHY.	
		0	PHY does not use AutoResume feature
		1	HY uses AutoResume feature
13:10 (R/W)	USBTRDTIM	USB Turnaround Time. Mode: Device only The USBC_CFG.USBTRDTIM bit field the turnaround time in PHY clocks. Specifies the response time for a MAC request to the Packet FIFO Controller (PFC) to fetch data from the DFIFO (SPRAM). This must be programmed as follows: - 4'h5: When the MAC interface is 16-bit UTMI - 4'h9: When the MAC interface is 8-bit UTMI Note: The previous values are calculated for the minimum bus frequency of 30 MHz. USB turnaround time is critical for certification where long cables and 5-Hubs are used. If the bus needs to run at less than 30 MHz and the USB turnaround time is not critical, these bits can be programmed to a larger value.	

Table 21-59: USBC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	HNPCAP	HNP-Capable. Mode: Host and Device The `USBC_CFG.HNPCAP` bit is for configuring the controller HNP capabilities. If HNP functionality is disabled by the software, the OTG signals on the PHY domain must be tied to the appropriate values.	
		0	HNP Capability Disabled
		1	HNP Capability Enabled
8 (R/W)	SRPCAP	SRP-Capable. Mode: Host and Device The `USBC_CFG.SRPCAP` bit is for configuring the controller SRP capabilities. If the core operates as a non-SRP-capable B-device, it cannot request the connected A-device (host) to activate VBUS and start a session. If SRP functionality is disabled by the software, the OTG signals on the PHY domain must be tied to the appropriate values.	
		0	SRP Capability Disabled
		1	SRP Capability Enabled
7 (R/W)	DDRSEL	DDR Select. Mode: Host and Device The application uses the `USBC_CFG.DDRSEL` bit to select a single data rate (SDR) or double data rate (DDR) for the USBC interface.	
		0	Single Data Rate USBC Interface: 8-bit-wide data bus
		1	Double Data Rate USBC Interface: 4-bit-wide data bus
6 (R/NW)	PHYSEL	USB 2.0 High-Speed PHY or USB 1.1 Full-Speed Serial Transceiver Select. Mode: Host and Device The application uses the `USBC_CFG.PHYSEL` bit to select either a high-speed UTMI+ or ULPI PHY, or a full-speed transceiver. This bit is always 0, with read only access.	
		0	USB 2.0 high-speed UTMI+ or ULPI PHY
		1	USB 1.1 full-speed serial transceiver

Table 21-59: USBC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	FSINTF	Full-Speed Serial Interface Select. Mode: Host and Device The application uses the `USBC_CFG.FSINTF` bit to select either a unidirectional or bidirectional USB 1.1 full-speed serial transceiver interface.	
		0	6-pin unidirectional full-speed serial interface
		1	3-pin bidirectional full-speed serial interface
4 (R/NW)	ULPI_UTMI_SEL	ULPI or UTMI+ Select. Mode: Host and Device The application uses the `USBC_CFG.ULPI_UTMI_SEL` bit to select either a UTMI + interface or ULPI Interface.	
		0	UTMI+ Interface
		1	ULPI Interface
3 (R/NW)	PHYIF	PHY Interface. Mode: Host and Device The application uses this bit to configure the core to support a ULPI PHY with an 8- or 16-bit interface.	
		0	8 bits
		1	16 bits

Table 21-59: USBC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	TOUTCAL	HS/FS Timeout Calibration. Mode: Host and Device The number of PHY clocks that the application programs in this field is added to the high-speed/full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the linestate condition can vary from one PHY to another. The USB standard timeout value for high-speed operation is 736 to 816 (inclusive) bit times. The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock are as follows: High-speed operation: - One 30-MHz PHY clock = 16 bit times - One 60-MHz PHY clock = 8 bit times Full-speed operation: - One 30-MHz PHY clock = 0.4 bit times - One 60-MHz PHY clock = 0.2 bit times - One 48-MHz PHY clock = 0.25 bit times

Host All Channels Interrupt Register

When a significant event occurs on a channel, the `USBC_ISTAT_H` register interrupts the application using the `USBC_ISTAT.HCHINT` bit. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel n interrupt register.

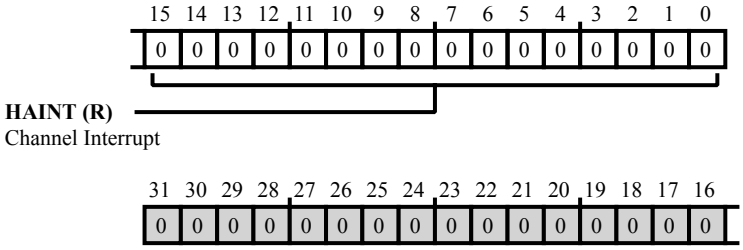


Figure 21-65: USBC_ISTAT_H Register Diagram

Table 21-60: USBC_ISTAT_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	HAINT	Channel Interrupt. Channel Interrupt for channel no.

Host All Channels Interrupt Mask Register

The USBC_IMSK_H register works with the Host All Channel Interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

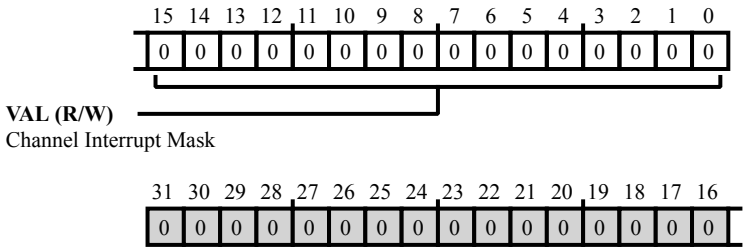


Figure 21-66: USBC_IMSK_H Register Diagram

Table 21-61: USBC_IMSK_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VAL	Channel Interrupt Mask. One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

Host Channel n Characteristics Register

The USBC_CHAR[n]_H register indicates the characteristics of the host channel n.

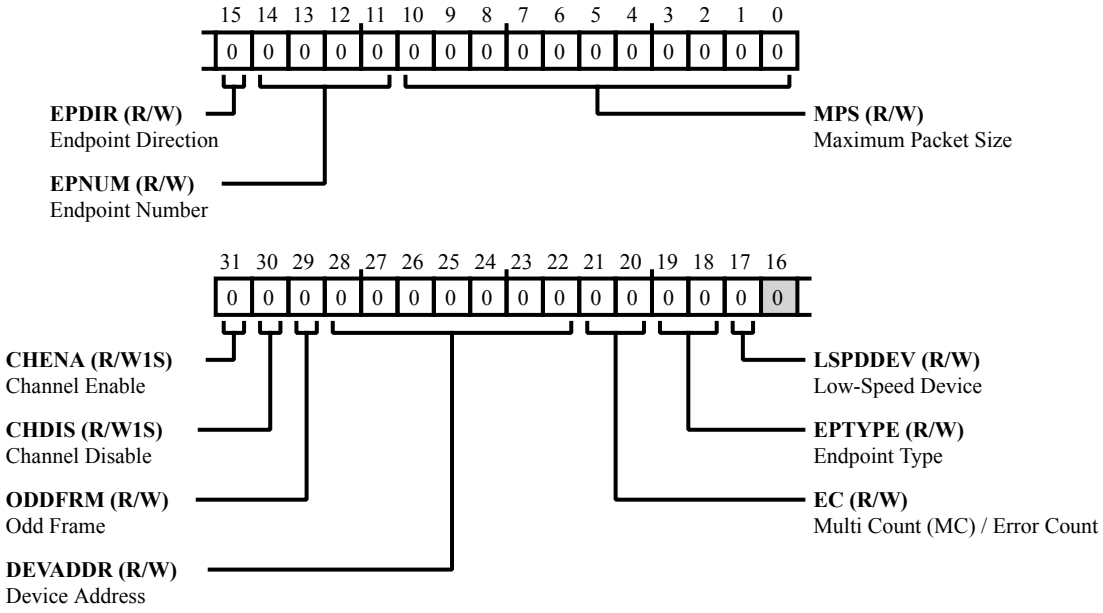


Figure 21-67: USBC_CHAR[n]_H Register Diagram

Table 21-62: USBC_CHAR[n]_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1S)	CHENA	Channel Enable. When scatter/gather mode is enabled, the USBC_CHAR[n]_H.CHENA bit indicates that: - 1'b0: the descriptor structure is not yet ready. - 1'b1: the descriptor structure and data buffer with data is setup and this channel can access the descriptor. When scatter/gather mode is disabled, the USBC_CHAR[n]_H.CHENA bit set by the application and cleared by the OTG host. - 1'b0: Channel disabled - 1'b1: Channel enabled
30 (R/W1S)	CHDIS	Channel Disable. The application sets the USBC_CHAR[n]_H.CHDIS bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the channel disabled interrupt before treating the channel as disabled.

Table 21-62: USBC_CHAR[n]_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	ODDFRM	Odd Frame. The USBC_CHAR[n]_H.ODDFRM bit is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd (micro)frame. The USBC_CHAR[n]_H.ODDFRM bit is applicable for only periodic (isochronous and interrupt) transactions.	
		0	Even (micro)frame
		1	Odd (micro)frame
28:22 (R/W)	DEVADDR	Device Address. The USBC_CHAR[n]_H.DEVADDR field selects the specific device serving as the data source or sink.	
21:20 (R/W)	EC	Multi Count (MC) / Error Count. When the Split Enable bit of the Host Channel-n Split Control register (USBC_SPLT_CTL[n]_H.SPLTENA) is reset (=0), the USBC_CHAR[n]_H.EC field indicates to the host the number of transactions that must be executed per micro-frame for this periodic endpoint. For non periodic transfers, the USBC_CHAR[n]_H.EC field is used only in DMA mode and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When the USBC_SPLT_CTL[n]_H.SPLTENA is set (=1), the USBC_CHAR[n]_H.EC field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. The USBC_CHAR[n]_H.EC bit field must be set to at least 2'b01.	
		0	Reserved
		1	1
		2	2
		3	3
19:18 (R/W)	EPTYPE	Endpoint Type. The USBC_CHAR[n]_H.EPTYPE field indicates the transfer type selected.	
		0	Control
		1	Isochronous
		2	Bulk
		3	Interrupt
17 (R/W)	LSPDDEV	Low-Speed Device. The USBC_CHAR[n]_H.LSPDDEV bit is set by the application to indicate that this channel is communicating to a low-speed device.	

Table 21-62: USBC_CHAR[n]_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	EPDIR	Endpoint Direction. The `USBC_CHAR[n]_H.EPDIR` bit indicates whether the transaction is IN or OUT.	
		0	Out
		1	In
14:11 (R/W)	EPNUM	Endpoint Number. The `USBC_CHAR[n]_H.EPNUM` field indicates the endpoint number on the device serving as the data source or sink.	
10:0 (R/W)	MPS	Maximum Packet Size. The `USBC_CHAR[n]_H.MPS` field indicates the maximum packet size of the associated endpoint.	

Host Channel n DMA Buffer Address Register

The USBC_DMA_BADDR[n]_H register holds the current buffer address. It is present only for scatter/gather DMA. It is implemented in RAM instead of a flop-based implementation. The USBC_DMA_BADDR[n]_H register is updated during the data transfer for the corresponding end point.

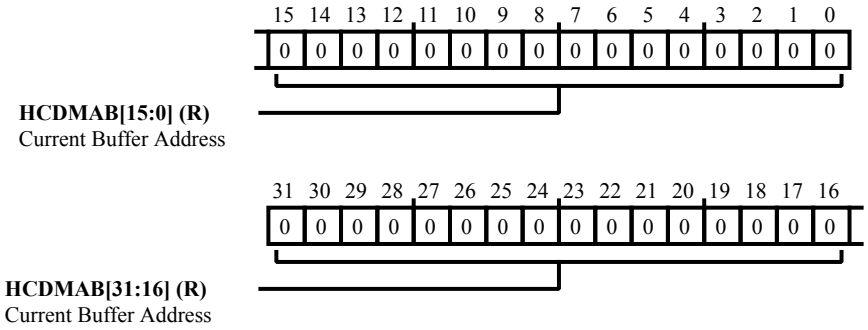


Figure 21-68: USBC_DMA_BADDR[n]_H Register Diagram

Table 21-63: USBC_DMA_BADDR[n]_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	HCDMAB	Current Buffer Address. The USBC_DMA_BADDR[n]_H.HCDMAB bit field indicates the current buffer address. The USBC_DMA_BADDR[n]_H register is present only in scatter/gather DMA mode. This field is reserved in other modes.

Host Channel n DMA Address Register

The USBC_DMA_ADDR[n]_H register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.This register is incremented on every bus transaction.

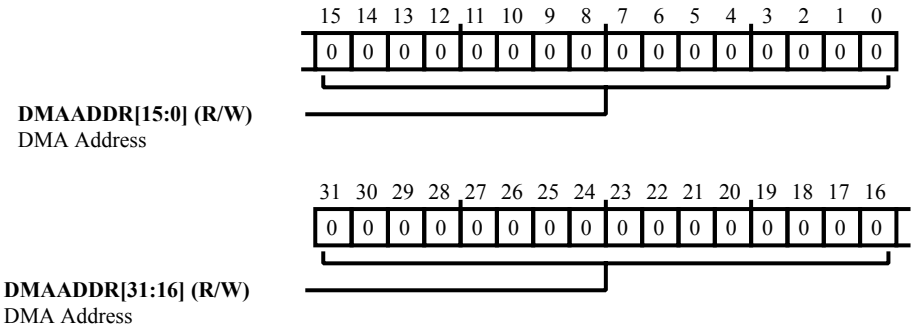


Figure 21-69: USBC_DMA_ADDR[n]_H Register Diagram

Table 21-64: USBC_DMA_ADDR[n]_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DMAADDR	DMA Address. In buffer DMA mode: [31:0]: DMA Address (DMAAddr) The USBC_DMA_ADDR[n]_H.DMAADDR field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. For control endpoints, the USBC_DMA_ADDR[n]_H.DMAADDR field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every bus transaction. The application can give only a DWORD-aligned address. When scatter/gather DMA mode is not enabled, the application programs the start address value in the USBC_DMA_ADDR[n]_H.DMAADDR field. When scatter/gather DMA mode is enabled, the USBC_DMA_ADDR[n]_H.DMAADDR field indicates the base pointer for the descriptor list.

Host Configuration Register

The USBC_CFG_H register is the host configuration register.

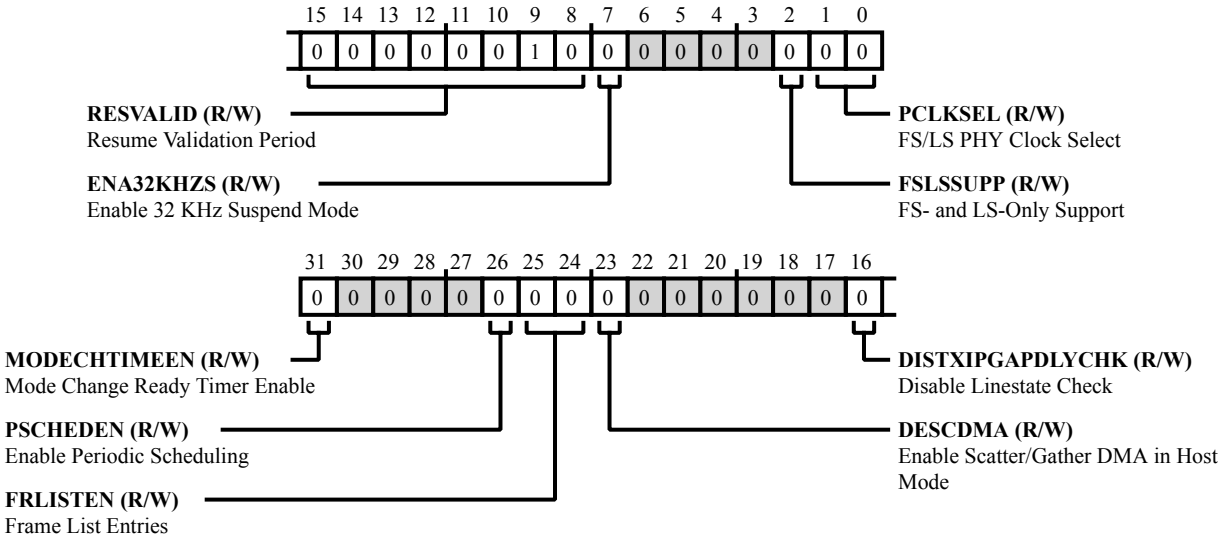


Figure 21-70: USBC_CFG_H Register Diagram

Table 21-65: USBC_CFG_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	MODECHTIMEEN	Mode Change Ready Timer Enable. The USBC_CFG_H.MODECHTIMEEN bit is used to enable/disable the host core to wait 200 PHY clock cycles at the end of resume to change the opmode signal to the PHY to 00 after suspend or LPM.	
		0	Host core waits for either 200 PHY clock cycles or a line state of SE0 at the end of resume, to the change the op-mode from 2'b10 to 2'b00
		1	Host core only waits for a line state of SE0 at the end of resume, to change the opmode from 2'b10 to 2'b0
26 (R/W)	PSCHEDEN	Enable Periodic Scheduling. The USBC_CFG_H.PSCHEDEN bit applies in host DMA mode only. The USBC_CFG_H.PSCHEDEN bit enables periodic scheduling within the core. Initially, the bit is reset. The core will not process any periodic channels. When the USBC_CFG_H.PSCHEDEN bit is set, the core will get ready to start scheduling periodic channels. When the USBC_CFG_H.PSCHEDEN bit is reset, the core will get ready to stop scheduling periodic channels.	

Table 21-65: USBC_CFG_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/W)	FRLISTEN	Frame List Entries. The USBC_CFG_H.FRLISTEN field specifies the number of entries in the frame list. The USBC_CFG_H.FRLISTEN field is valid only in scatter/gather DMA mode.	
		0	8 Entries
		1	16 Entries
		2	32 Entries
		3	64 Entries
23 (R/W)	DESCDMA	Enable Scatter/Gather DMA in Host Mode. The application can set the USBC_CFG_H.DESCDMA bit during initialization to enable the scatter/gather DMA operation. Note: The USBC_CFG_H.DESCDMA bit must be modified only once after a reset. The following combinations are available for programming: - USBC_AHB_CFG.DMAEN = 0, USBC_CFG_H.DESCDMA = 0 => Slave mode - USBC_AHB_CFG.DMAEN = 0, USBC_CFG_H.DESCDMA = 1 => Invalid - USBC_AHB_CFG.DMAEN = 1, USBC_CFG_H.DESCDMA = 0 => Buffered DMA mode - USBC_AHB_CFG.DMAEN = 1, USBC_CFG_H.DESCDMA = 1 => Scatter/Gather DMA mode	
16 (R/W)	DISTXIPGAPDLYCHK	Disable Linestate Check. The USBC_CFG_H.DISTXIPGAPDLYCHK field applies only in HS mode of operation. When enabled, the controller implements counter based logic to detect the end of token transmission, otherwise controller monitors the linestate for end of token transmission, to start the token-to-token and token-to-data interpacket delay counter.	
15:8 (R/W)	RESVALID	Resume Validation Period. The USBC_CFG_H.RESVALID field is effective only when USBC_CFG_H.ENA32KHZS is set (=1). The USBC_CFG_H.RESVALID field controls the resume period when the core resumes from suspend. The core counts for USBC_CFG_H.RESVALID number of clock cycles to detect a valid resume when this is set.	
7 (R/W)	ENA32KHZS	Enable 32 KHz Suspend Mode. The USBC_CFG_H.ENA32KHZS bit can be set only in FS PHY interface is selected. Otherwise, this bit needs to be set to zero.	

Table 21-65: USBC_CFG_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	FSLSSUPP	FS- and LS-Only Support. The `USBC_CFG_H.FSLSSUPP` bit controls the core's enumeration speed. Using this bit, the application can make the core enumerate as a full speed host, even if the connected device supports high speed traffic. Do not make changes to this field after initial programming.	
		0	HS/FS/LS based on maximum speed suported by the connected device
		1	FS/LS only, even if the connected device can support high speed.
1:0 (R/W)	PCLKSEL	FS/LS PHY Clock Select. The `USBC_CFG_H.PCLKSEL` field selects the full speed and low speed PHY clock rates. When the core is in full-speed host mode: 0: PHY clock is running at 30/60 MHz 1: PHY clock is running at 48 MHz Others: Reserved When the core is in low speed host mode 00: PHY clock is running at 30/60 MHz. When the ULPI PHY low power mode is not selected, use 30/60 MHz. 01: PHY clock is running at 48 MHz. When the ULPI PHY low power mode is selected, use 48 MHz if the PHY supplies a 48 MHz clock during low speed mode. 10: PHY clock is running at 6 MHz. In USB 1.1 FS mode, use 6 MHz when the ULPI PHY low power mode is selected and the PHY supplies a 6 MHz clock during LS mode. If a 6 MHz clock is selected during LS mode, a soft reset is required. 11: Reserved Notes: When core in FS mode, the internal and external clocks have the same frequency. When core in LS mode, If `USBC_CFG_H.PCLKSEL` = 00: Internal and external clocks have the same frequency If `USBC_CFG_H.PCLKSEL` = 10: Internal clock is the divided by eight version of external 48 MHz clock	

Host Channel n Interrupt Mask Register

The USBC_IMSK[n]_H register reflects the mask for each channel status.

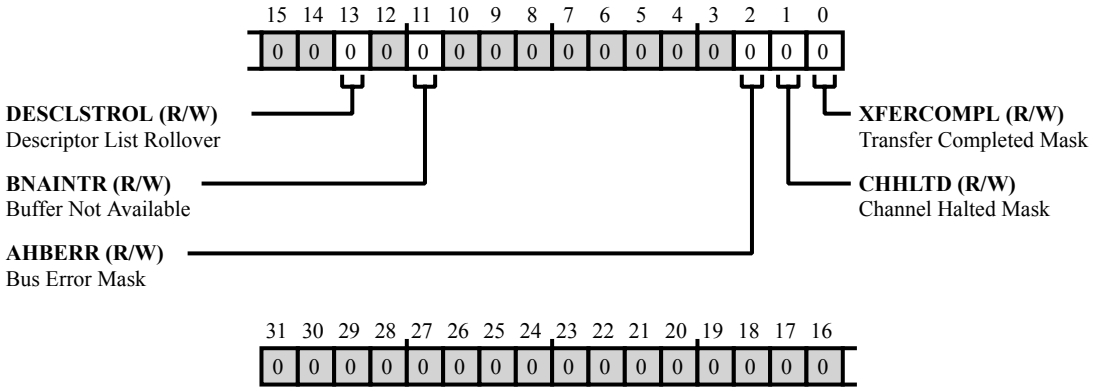


Figure 21-71: USBC_IMSK[n]_H Register Diagram

Table 21-66: USBC_IMSK[n]_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	DESCLSTROL	Descriptor List Rollover. This bit is valid only when Scatter/Gather DMA mode is enabled.
11 (R/W)	BNAINTR	Buffer Not Available. This bit indicates that the buffer is not available. It is valid only when scatter/gather DMA mode is enabled.
2 (R/W)	AHBERR	Bus Error Mask. In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.
1 (R/W)	CHHLTD	Channel Halted Mask.
0 (R/W)	XFERCOMPL	Transfer Completed Mask.

Host Channel n Interrupt Status Register

The USBC_ISTAT[n]_H register indicates the status of an endpoint with respect to USB and bus related events. The application must read this register when the USBC_ISTAT.HCHINT bit is set.

Before the application can read this register, it must first read the USBC_ISTAT_H register to get the exact channel number for the Host Channel n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the USBC_ISTAT_H and USBC_ISTAT registers.

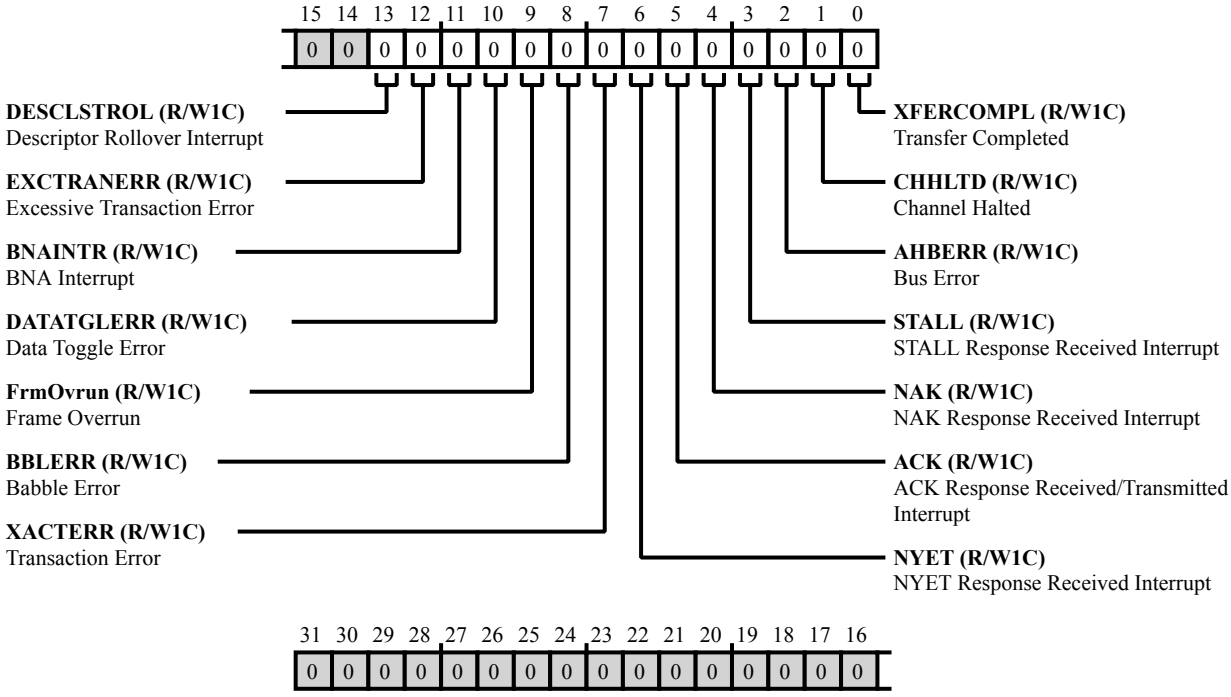


Figure 21-72: USBC_ISTAT[n]_H Register Diagram

Table 21-67: USBC_ISTAT[n]_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W1C)	DESCLSTROL	Descriptor Rollover Interrupt. The USBC_ISTAT[n]_H.DESCLSTROL bit is valid only when scatter/gather DMA mode is enabled. The core sets the USBC_ISTAT[n]_H.DESCLSTROL bit when the corresponding channel's descriptor list rolls over. For non scatter/gather DMA mode, the USBC_ISTAT[n]_H.DESCLSTROL bit is reserved.

Table 21-67: USBC_ISTAT[n]_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
12 (R/W1C)	EXCTRANERR	Excessive Transaction Error. The `USBC_ISTAT[n]_H.EXCTRANERR` bit is valid only when scatter/gather DMA mode is enabled. The core sets this bit when three consecutive transaction errors occurred on the USB bus. The `USBC_ISTAT[n]_H.EXCTRANERR` bit is not generated for isochronous channels. For non scatter/gather DMA mode, the `USBC_ISTAT[n]_H.EXCTRANERR` bit is reserved.
11 (R/W1C)	BNAINTR	BNA Interrupt. The `USBC_ISTAT[n]_H.BNAINTR` bit is valid only when scatter/gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA (Buffer Not Available) is not generated for isochronous channels. For non scatter/gather DMA mode, the `USBC_ISTAT[n]_H.BNAINTR` bit is reserved.
10 (R/W1C)	DATATGLERR	Data Toggle Error. The `USBC_ISTAT[n]_H.DATATGLERR` bit can be set only by the core and the application should write 1 to clear it. In scatter/gather DMA mode, the interrupt due to the `USBC_ISTAT[n]_H.DATATGLERR` bit is masked in the core.
9 (R/W1C)	FRMOVRUN	Frame Overrun. In scatter/gather DMA mode, the interrupt due to the `USBC_ISTAT[n]_H.FRMOVRUN` bit is masked in the core. The `USBC_ISTAT[n]_H.FRMOVRUN` bit bit can be set only by the core and the application should write 1 to clear it.
8 (R/W1C)	BBLERR	Babble Error. In scatter/gather DMA mode, the interrupt due to the `USBC_ISTAT[n]_H.BBLERR` bit is masked in the core. The `USBC_ISTAT[n]_H.BBLERR` bit can be set only by the core and the application should write 1 to clear it.
7 (R/W1C)	XACTERR	Transaction Error. The `USBC_ISTAT[n]_H.XACTERR` bit indicates one of the following errors occurred on the USB: CRC check failure, Timeout, Bit stuff error, or False EOP. In scatter/gather DMA mode, the interrupt due to the `USBC_ISTAT[n]_H.XACTERR` bit is masked in the core. The `USBC_ISTAT[n]_H.XACTERR` bit can be set only by the core and the application should write 1 to clear it.

Table 21-67: USBC_ISTAT[n]_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/W1C)	NYET	NYET Response Received Interrupt. In scatter/gather DMA mode, the interrupt due to the `USBC_ISTAT[n]_H.NYET` bit is masked in the core. The `USBC_ISTAT[n]_H.NYET` bit can be set only by the core and the application should write 1 to clear it.
5 (R/W1C)	ACK	ACK Response Received/Transmitted Interrupt. In scatter/gather DMA mode, the interrupt due to the `USBC_ISTAT[n]_H.ACK` bit is masked in the core. The `USBC_ISTAT[n]_H.ACK` bit can be set only by the core and the application should write 1 to clear it.
4 (R/W1C)	NAK	NAK Response Received Interrupt. In scatter/gather DMA mode, the interrupt due to the `USBC_ISTAT[n]_H.NAK` bit is masked in the core. The `USBC_ISTAT[n]_H.NAK` bit can be set only by the core and the application should write 1 to clear it.
3 (R/W1C)	STALL	STALL Response Received Interrupt. In scatter/gather DMA mode, the interrupt due to the `USBC_ISTAT[n]_H.STALL` bit is masked in the core. The `USBC_ISTAT[n]_H.STALL` bit can be set only by the core and the application should write 1 to clear it.
2 (R/W1C)	AHBERR	Bus Error. The `USBC_ISTAT[n]_H.AHBERR` bit is generated only in internal DMA mode when there is a bus error during a bus read/write. The application can read the corresponding channel's DMA address register to get the error address.
1 (R/W1C)	CHHLTD	Channel Halted. In non scatter/gather DMA mode, the `USBC_ISTAT[n]_H.CHHLTD` bit indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In scatter/gather DMA mode, the `USBC_ISTAT[n]_H.CHHLTD` bit indicates that transfer completed due to any of the following: EOL being set in descriptor Bus error Excessive transaction errors Babble Stall

Table 21-67: USBC_ISTAT[n]_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W1C)	XFERCOMPL	Transfer Completed. The `USBC_ISTAT[n]_H.XFERCOMPL` bit indicates that the transfer completed normally without any errors. The `USBC_ISTAT[n]_H.XFERCOMPL` bit can be set only by the core and the application should write 1 to clear it. For scatter/gather DMA mode, the `USBC_ISTAT[n]_H.XFERCOMPL` bit indicates that current descriptor processing completed with an IOC bit set in its descriptor. In non scatter/gather DMA mode, the `USBC_ISTAT[n]_H.XFERCOMPL` bit indicates that transfer completed normally without any errors.

Host Channel n Split Control Register

The USBC_SPLT_CTL[n]_H register controls split transactions for a channel.

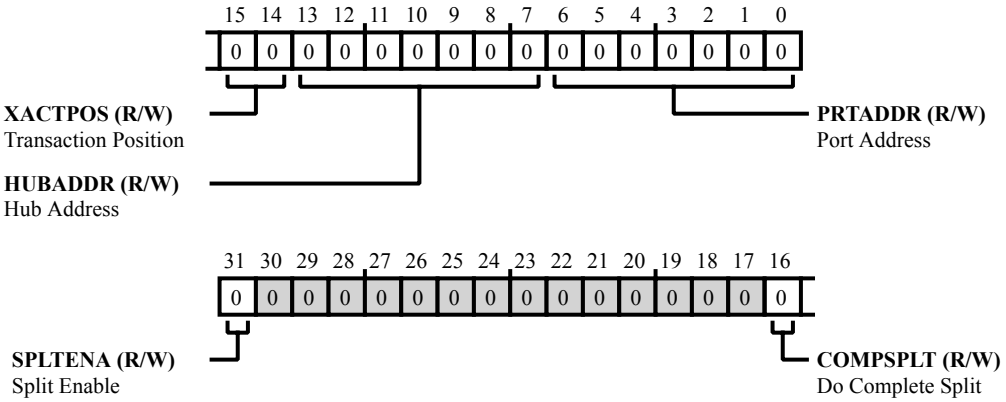


Figure 21-73: USBC_SPLT_CTL[n]_H Register Diagram

Table 21-68: USBC_SPLT_CTL[n]_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	SPLTENA	Split Enable. When set (=1), the USBC_SPLT_CTL[n]_H.SPLTENA bit enables the channel to perform split transactions.	
16 (R/W)	COMPSPLT	Do Complete Split. The USBC_SPLT_CTL[n]_H.COMPSPLT bit enables the request to the OTG host to perform a complete split transaction.	
15:14 (R/W)	XACTPOS	Transaction Position. The USBC_SPLT_CTL[n]_H.XACTPOS field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.	
		0	Mid. The middle payload of the transaction (which is larger than 188 bytes).
		1	End. The last payload of the transaction (which is larger than 188 bytes).
		2	Begin. The first data payload of the transaction (which is larger than 188 bytes).
		3	All. The entire data payload is the transaction (which is less than or equal to 188 bytes).
13:7 (R/W)	HUBADDR	Hub Address. The USBC_SPLT_CTL[n]_H.HUBADDR field holds the device address of the transaction translator's hub.	

Table 21-68: USBC_SPLT_CTL[n]_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	PRTADDR	Port Address. The `USBC_SPLT_CTL[n]_H.PRTADDR` field indicates the port number of the recipient transaction translator.

Host Channel n Transfer Size Register

The USBC_TSIZ[n]_H register represents the host channel n transfer size register.

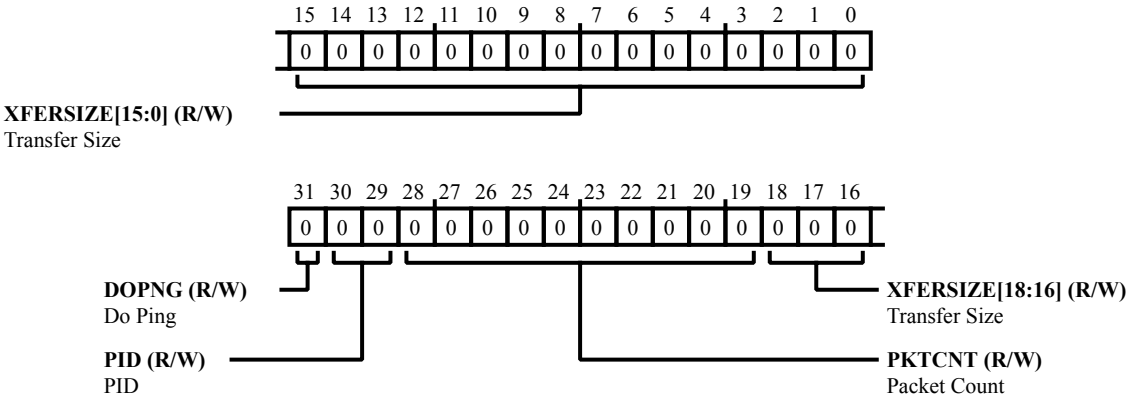


Figure 21-74: USBC_TSIZ[n]_H Register Diagram

Table 21-69: USBC_TSIZ[n]_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	DOPNG	Do Ping. The USBC_TSIZ[n]_H.DOPNG bit is used only for OUT transfers. Setting the USBC_TSIZ[n]_H.DOPNG bit (=1) directs the host to do PING protocol. Note: Do not set USBC_TSIZ[n]_H.DOPNG bit for IN transfers. If the USBC_TSIZ[n]_H.DOPNG bit is set for for IN transfers, it disables the channel.	
30:29 (R/W)	PID	PID. The USBC_TSIZ[n]_H.PID field indicates the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.	
		0	DATA0
		1	DATA2
		2	DATA1
		3	MDATA (non-control) / SETUP (control)
28:19 (R/W)	PKTCNT	Packet Count. The USBC_TSIZ[n]_H.PKTCNT field indicates the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.	

Table 21-69: USBC_TSIZ[n]_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
18:0 (R/W)	XFERSIZE	Transfer Size. For an OUT, the `USBC_TSIZ[n]_H.XFERSIZE` field is the number of data bytes the host sends during the transfer. For an IN, the `USBC_TSIZ[n]_H.XFERSIZE` field is the buffer size that the application has reserved for the transfer. The `USBC_TSIZ[n]_H.XFERSIZE` field must be programmed as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic). Scatter/Gather DMA Mode: [18:16]: Reserved [15:8]: NTD (Number of Transfer Descriptors) (Non Isochronous) This value is in terms of number of descriptors. Maximum number of descriptor that can be present in the list is 64. The values can be from 0 to 63. 0 1 descriptor 63 64 descriptors This field indicates the total number of descriptors present in that list. The core will wrap around after servicing NTD number of descriptors for that list. (Isochronous) This field indicates the number of descriptors present in that list microframe. The possible values for FS are 1-2, 3-4, 7-8, 15-16, 31-32, and 63-64 descriptors The possible values for HS are 7-8, 15-16, 31-32, 63-64, 127-128, and 255-256 descriptors [7:0]: SCHED_INFO (Schedule information) Every bit in this 8 bit register indicates scheduling for that microframe. Bit 0 indicates scheduling for 1st microframe and bit 7 indicates scheduling for 8th microframe in that frame. A value of 8b11111111 indicates that the corresponding interrupt channel is scheduled to issue a token every microframe in that frame. A value of 8b10101010 indicates that the corresponding interrupt channel is scheduled to issue a token every alternate microframe starting with second microframe. Note that this field is applicable only for periodic (Isochronous and Interrupt) channels.

Host Frame Interval Register

The USBC_FIR_H register indicates the frame interval. It also control the reloading of the USBC_FIR_H register during runtime.

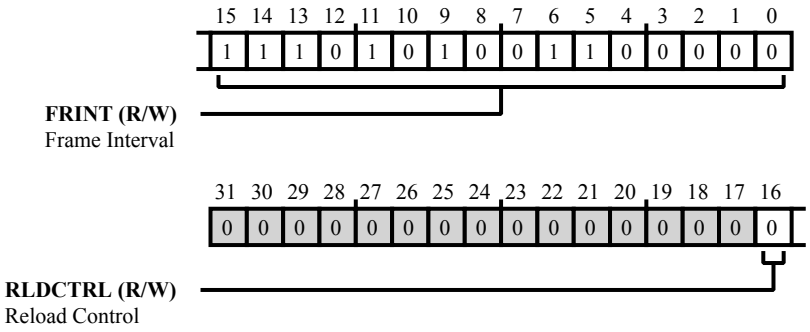


Figure 21-75: USBC_FIR_H Register Diagram

Table 21-70: USBC_FIR_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	RLDCTRL	Reload Control. The USBC_FIR_H.RLDCTRL bit enables dynamic reloading of the USBC_FIR_H register during runtime. The USBC_FIR_H.RLDCTRL bit must be programmed during initial configuration and its value should not be changed during runtime.	
		0	Disable
		1	Enable
15:0 (R/W)	FRINT	Frame Interval. The USBC_FIR_H.FRINT field specifies the interval between two consecutive SPFs (FS) or micro-SOFs (HS) or keep-alive tokens (HS). The field contains the number of PHY clocks that constitute the required frame interval. The default value set in the USBC_FIR_H.FRINT field is for full-speed operation when the PHY clock frequency is 60 MHz. The application can write a value to this register only after the USBC_PORT_CTL_H.ENA bit is set (=1). If no value is programmed, the core calculates the value based on the PHY clock specified in the USBC_CFG_H.PCLKSEL field. Do not change the value of USBC_FIR_H.FRINT field after the initial configuration. 125 s * (PHY clock frequency for HS) 1 ms * (PHY clock frequency for FS/LS)	

Host Frame List Base Address Register

The USBC_FL_BADDR_H register holds the starting address of the frame list information. This register is used only for isochronous and interrupt channels.

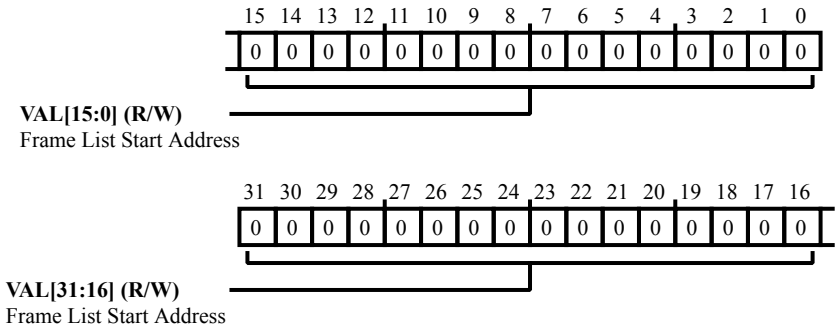


Figure 21-76: USBC_FL_BADDR_H Register Diagram

Table 21-71: USBC_FL_BADDR_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VAL	Frame List Start Address.

Host Frame Number/Frame Time Remaining Register

The USBC_FNUM_H register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current (micro)frame.

Note: Read the reset value of this register only after the following conditions:

If IDDIG_FILTER is disabled, read only when the PHY clock is stable.

If IDDIG_FILTER is enabled, read only after the filter timer expires.

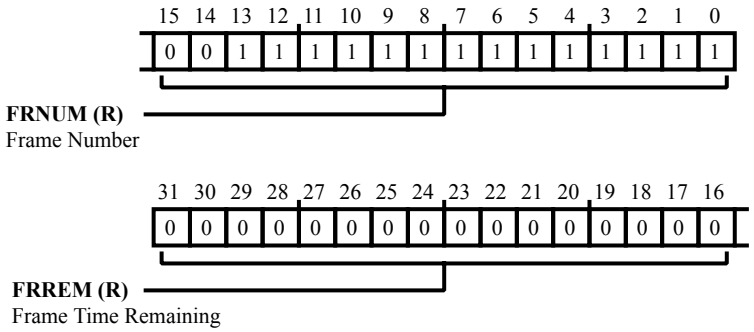


Figure 21-77: USBC_FNUM_H Register Diagram

Table 21-72: USBC_FNUM_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/NW)	FRREM	Frame Time Remaining. The USBC_FNUM_H.FRREM field indicates the amount of time remaining in the current microframe (HS) or frame (FS/LS), in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, the USBC_FNUM_H.FRREM field is reloaded with the value in the frame interval register and a new SOF is transmitted on the USB.
15:0 (R/NW)	FRNUM	Frame Number. The USBC_FNUM_H.FRNUM field increments when a new SOF is transmitted on the USB. It is reset to 0 when it reaches 16'h3FFF.

Host Port Control and Status Register

The USBC_PORT_CTL_H register is available only in host mode. Currently, the OTG host supports only one port. A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port.

The R_SS_WC bits in this register can trigger an interrupt to the application through the USBC_ISTAT.PRTINT bit. On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the R_SS_WC bits, the application must write a 1 to the bit to clear the interrupt.

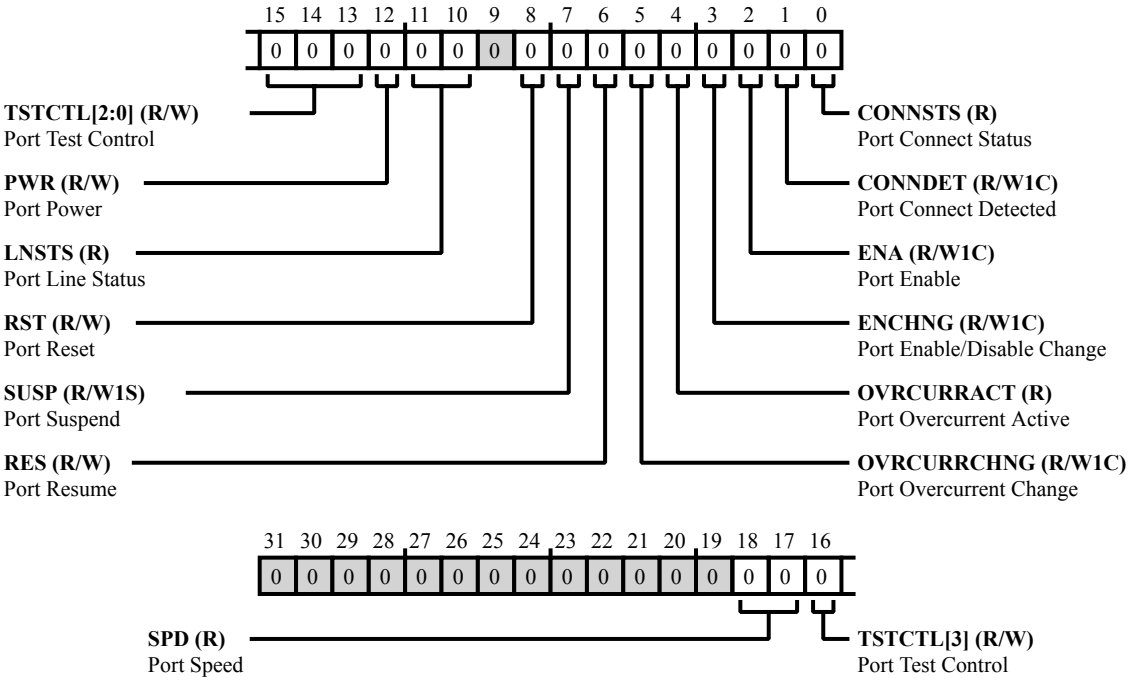


Figure 21-78: USBC_PORT_CTL_H Register Diagram

Table 21-73: USBC_PORT_CTL_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18:17 (R/NW)	SPD	Port Speed. The USBC_PORT_CTL_H.SPD field indicates the speed of the device attached to this port.	
		0	High speed
		1	Full speed
		2	Low speed
		3	Reserved

Table 21-73: USBC_PORT_CTL_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16:13 (R/W)	TSTCTL	Port Test Control. The application writes a nonzero value to the `USBC_PORT_CTL_H.TSTCTL` field to put the port into a test mode, and the corresponding pattern is signaled on the port. To move the USBC_otg controller to test mode, this field must be set. Complete the following steps to move the USBC_otg core to test mode: 1. Power on the core. 2. Load the USBC_otg driver. 3. Connect an HS device and enumerate to HS mode. 4. Access the `USBC_PORT_CTL_H` register to send test packets. 5. Remove the device and connect to fixture (OPT) port. The USBC_otg host core continues sending out test packets. 6. Test the eye diagram.	
		0	Test mode disabled
		1	Test_J mode
		2	Test_K mode
		3	Test_SEO_NAK mode
		4	Test_Packet mode
		5	Test_Force_Enable
		6	Reserved
		7	Reserved
		8	Reserved
		9	Reserved
		10	Reserved
		11	Reserved
		12	Reserved
		13	Reserved
		14	Reserved
		15	Reserved

Table 21-73: USBC_PORT_CTL_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	PWR	Port Power. The USBC_PORT_CTL_H.PWR bit controls power to the port (write 1 to set to 1 and write 0 to set to 0). The core can clear the USBC_PORT_CTL_H.PWR field bit on an overcurrent condition. Note: The USBC_PORT_CTL_H.PWR bit is interface independent. The application needs to program the USBC_PORT_CTL_H.PWR bit for all interfaces.	
		0	Power off
		1	Power on
11:10 (R/NW)	LNSTS	Port Line Status. The USBC_PORT_CTL_H.LNSTS field indicates the current logic level USB data lines Bit [10]: Logic level of D+ Bit [11]: Logic level of D-	
8 (R/W)	RST	Port Reset. When the application sets the USBC_PORT_CTL_H.RST bit, a reset sequence is started on this port. The application must time the reset period and clear the USBC_PORT_CTL_H.RST bit after the reset sequence is complete. The application must leave the USBC_PORT_CTL_H.RST bit set for at least a minimum duration mentioned below to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard. The USBC_PORT_CTL_H.RST bit is cleared by the core even if there is no device connected to the host. High speed: 50 ms Full speed/low speed: 10 ms	
		0	Port not in reset
		1	Port in reset

Table 21-73: USBC_PORT_CTL_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1S)	SUSP	Port Suspend. The application sets the USBC_PORT_CTL_H.SUSP bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY. The USBC_PORT_CTL_H.SUSP bit indicates the current suspend status of the port. The USBC_PORT_CTL_H.SUSP bit is cleared by the core after a remote wakeup signal is detected or the application sets the port reset bit or port resume bit in this register or the (USBC_ISTAT.WKUPINT or USBC_ISTAT.DISCONINT bit. The USBC_PORT_CTL_H.SUSP bit is cleared by the core even if there is no device connected to the host.	
		0	Port not in suspend mode
		1	Port in suspend mode
6 (R/W)	RES	Port Resume. The application sets the USBC_PORT_CTL_H.RES bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit. If the core detects a USB remote wakeup sequence, as indicated by the USBC_ISTAT.WKUPINT bit, the core starts driving resume signaling without application intervention and clears the USBC_PORT_CTL_H.RES bit when it detects a disconnect condition. When LPM is enabled, In L1 state the behavior of the USBC_PORT_CTL_H.RES bit is as follows: The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal. If the core detects a USB remote wakeup sequence, as indicated by the Port L1Resume/Remote L1Wakeup Detected Interrupt bit of the core Interrupt register the (USBC_ISTAT.L1WkUpInt), the core starts driving resume signaling without application intervention and clears the USBC_PORT_CTL_H.RES bit at the end of resume. This bit can be set by both core or application and also cleared by core or application. The USBC_PORT_CTL_H.RES bit is cleared by the core even if there is no device connected to the host.	
		0	No resume driven
		1	Resume driven
5 (R/W1C)	OVRCURRCHNG	Port Overcurrent Change. The core sets the USBC_PORT_CTL_H.OVRCURRCHNG bit when the status of the USBC_PORT_CTL_H.OVRCURRACT bit changes. The USBC_PORT_CTL_H.OVRCURRCHNG bit can be set only by the core and the application should write 1 to clear it.	

Table 21-73: USBC_PORT_CTL_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	OVRCURRACT	Port Overcurrent Active. The `USBC_PORT_CTL_H.OVRCURRACT` bit indicates the overcurrent condition of the port.	
		0	No overcurrent condition
		1	Overcurrent condition
3 (R/W1C)	ENCHNG	Port Enable/Disable Change. The core sets the `USBC_PORT_CTL_H.ENCHNG` bit when the status of the `USBC_PORT_CTL_H.ENA` bit changes. The `USBC_PORT_CTL_H.ENCHNG` bit can be set only by the core and the application should write 1 to clear it.	
2 (R/W1C)	ENA	Port Enable. The `USBC_PORT_CTL_H.ENA` bit enables the port. A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set the `USBC_PORT_CTL_H.ENA` bit by a register write. It can only clear the `USBC_PORT_CTL_H.ENA` bit to disable the port by writing 1. This bit does not trigger any interrupt to the application.	
		0	Disable
		1	Enable
1 (R/W1C)	CONNDET	Port Connect Detected. The core sets the `USBC_PORT_CTL_H.CONNDET` bit when a device connection is detected to trigger an interrupt to the application using the `USBC_ISTAT.PRTINT` bit. The `USBC_PORT_CTL_H.CONNDET` bit can be set only by the core and the application should write 1 to clear it. The application must write a 1 to this bit to clear the interrupt.	
0 (R/NW)	CONNSTS	Port Connect Status. The `USBC_PORT_CTL_H.CONNSTS` bit indicates whether a device is connected to the port.	
		0	No device attached to port
		1	Device attached to port

Host Periodic Transmit FIFO Size Register

The USBC_TXFIFOSZ_PER_H register holds the size and the memory start address of the periodic TxFIFO.

Note: Read the reset value of this register only after the following conditions:

If IDDIG_FILTER is disabled, read only after PHY clock is stable.

If IDDIG_FILTER is enabled, read only after the filter timer expires.

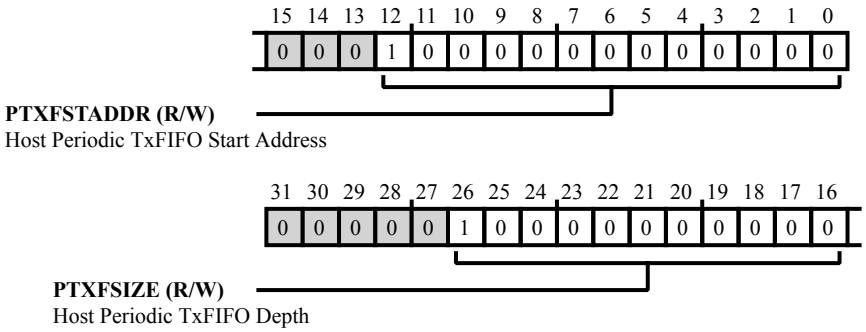


Figure 21-79: USBC_TXFIFOSZ_PER_H Register Diagram

Table 21-74: USBC_TXFIFOSZ_PER_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
26:16 (R/W)	PTXFSIZE	Host Periodic TxFIFO Depth. The USBC_TXFIFOSZ_PER_H.PTXFSIZE field indicates the host periodic TXFIFO depth. The value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value is specified as the largest host mode periodic Tx Data FIFO depth. Programmed values must not exceed the power-on value.
12:0 (R/W)	PTXFSTADDR	Host Periodic TxFIFO Start Address.

Power and Clock Gating Control Register

The USBC_PWR_CTL register controls power down, sleep, and clock stopping features.

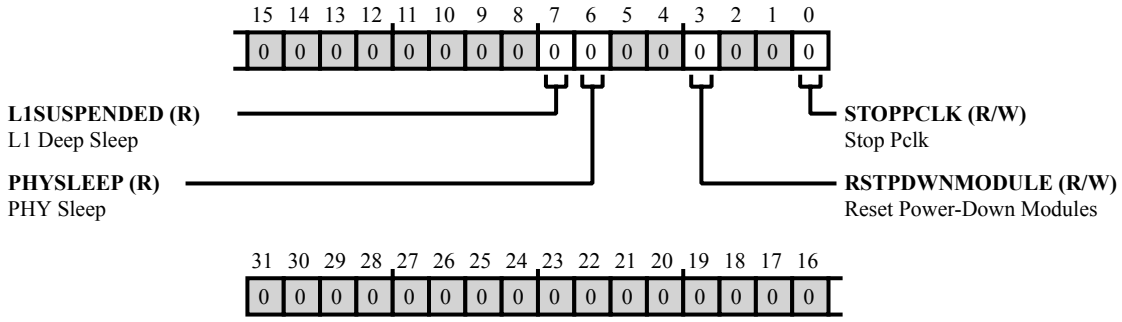


Figure 21-80: USBC_PWR_CTL Register Diagram

Table 21-75: USBC_PWR_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	L1SUSPENDED	L1 Deep Sleep. The USBC_PWR_CTL.L1SUSPENDED bit indicates that the PHY is in deep sleep when in L1 state.
6 (R/NW)	PHYSLEEP	PHY Sleep. The USBC_PWR_CTL.PHYSLEEP indicates that the PHY is in a sleep state.
3 (R/W)	RSTPDWNMODULE	Reset Power-Down Modules. The USBC_PWR_CTL.RSTPDWNMODULE bit is valid only in Partial Power-Down mode. - The application sets this bit when the power is turned off. - The application clears this bit after the power is turned on and the PHY clock is up. Note: The R/W of all core registers are possible only when this bit is set to 1b0.
0 (R/W)	STOPPCLK	Stop Pclk. The USBC_PWR_CTL.STOPPCLK bit stops the PHY clock (phy_clk), when set (=1) and the USB is suspended, the session is not valid, or the device is disconnected. The USBC_PWR_CTL.STOPPCLK bit, when clear (=0), the USB resumes or a new session starts.

22 Serial Peripheral Interface (SPI)

The serial peripheral interface is an industry-standard synchronous serial link that supports communication with multiple SPI-compatible devices. The baseline SPI peripheral is a synchronous, four-wire interface consisting of two data pins, one device select pin, and a gated clock pin. The two data pins allow full-duplex operation to other SPI-compatible devices. Two extra (optional) data pins are provided on specific SPIs to support quad SPI operation. Enhanced modes of operation such as flow control, fast mode, and dual-I/O mode (DIOM) are also supported. In addition, a direct memory access (DMA) mode allows for transferring several words with minimal CPU interaction.

With a range of configurable options, the SPI ports provide a glueless hardware interface with other SPI-compatible devices in controller mode, target mode, and multi-controller environments. The SPI peripheral includes programmable baud rates, clock phase, and clock polarity. The peripheral can operate in a multi-controller environment by interfacing with several other devices, acting as either a controller device or a target device. In a multi-controller environment, the SPI peripheral uses open-drain outputs to avoid data bus contention. The flow control features enable slow target devices to interface with fast controller devices by providing an SPI ready pin which flexibly controls the transfers.

NOTE: SPI peripherals on the processor operate in the SCLK0 domain. For more details on SCLK programming, refer the *Clock Generation Unit (CGU)* chapter.

SPI Features

The SPI module supports the following features:

- Full-duplex, synchronous serial interface
- 8, 16-bit and 32-bit word sizes
- Programmable baud rate, clock phase, and polarity
- Programmable interframe latency
- Flow control
- Support for Fast and DIOM modes
- SPI1 and SPI2 support quad mode. Memory-mapped mode is supported by SPI2 only.
- Independent receive and transmit DMA channels

- Burst transfer mode for non-DMA write accesses

SPI Functional Description

This section provides information on the function of the SPI module.

Shift register functionality

The SPI is a shift register that serially transmits and receives data bits to or from other SPI devices. During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling of the information on the two serial data lines.

Controller/target functionality

During a data transfer, one SPI system acts as the link controller which controls the data flow. The other system acts as the target, which has data shifted into and out of it by the controller. Different devices can take turn being controllers, and one controller can simultaneously shift data into multiple targets (broadcast mode). However, only one target can drive its output to write data back to the controller at any given time. This rule must be enforced in the broadcast mode. Several targets can be selected to receive data from the controller in this mode. But, only one target can be enabled to send data back to the controller.

Enhanced operating modes

SPI supports enhanced modes of operation like fast mode, DIOM, and Quad-SPI, and optional flow control. In fast mode, received data is sampled on the transmit edge instead of the standard receive edge, thus enabling a full-cycle path for the received data. In DIOM, both MOSI and MISO are configured as input or output pins, and 2 bits are shifted in or out on each receive or transmit edge. In Quad-SPI mode, SPI_D3:0 are configured as input or output pins and 4 bits are shifted in or out on each receive or transmit edge. A slower target can use flow control to stall a faster controller device.

Single and multi-controller use

The SPI can be used in a single controller as well as multi-controller environment. The `SPI_MOSI`, `SPI_MISO`, and the `SPI_CLK` signals are all tied together in both configurations. SPI transmission and reception can be enabled simultaneously or individually, depending on `SPI_RXCTL` and `SPI_TXCTL` settings. In broadcast mode, several target can be enabled to receive, but only one target must be in transmit mode and driving the `SPI_MISO` line.

ADSP-SC59x SPI Register List

The Serial Peripheral Interface (SPI) provides a full-duplex, synchronous serial interface, which supports both master/slave modes and multi-master environments. The SPI's baud rate and clock phase/polarities are programmable, and it has integrated DMA channels for both transmit and receive data streams. A set of registers governs SPI operations. For more information on SPI functionality, see the SPI register descriptions.

Table 22-1: ADSP-SC59x SPI Register List

Name	Description
SPI_CLK	Clock Rate Register
SPI_CTL	Control Register
SPI_DLY	Delay Register
SPI_ILAT	Masked Interrupt Condition Register
SPI_ILAT_CLR	Masked Interrupt Clear Register
SPI_IMSK	Interrupt Mask Register
SPI_IMSK_CLR	Interrupt Mask Clear Register
SPI_IMSK_SET	Interrupt Mask Set Register
SPI_MMRDH	Memory Mapped Read Header
SPI_MMTOP	SPI Memory Top Address
SPI_RFIFO	Receive FIFO Data Register
SPI_RWC	Received Word Count Register
SPI_RWCR	Received Word Count Reload Register
SPI_RXCTL	Receive Control Register
SPI_SLVSEL	Slave Select Register
SPI_STAT	Status Register
SPI_TFIFO	Transmit FIFO Data Register
SPI_TWC	Transmitted Word Count Register
SPI_TWCR	Transmitted Word Count Reload Register
SPI_TXCTL	Transmit Control Register

ADSP-SC59x SPI Interrupt List

Table 22-2: ADSP-SC59x SPI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
121	SPI0_TXDMA	SPI0 TX DMA Channel	Level	22
122	SPI0_RXDMA	SPI0 RX DMA Channel	Level	23
123	SPI0_STAT	SPI0 Status	Level	
124	SPI0_ERR	SPI0 Error	Level	
125	SPI1_TXDMA	SPI1 TX DMA Channel	Level	24
126	SPI1_RXDMA	SPI1 RX DMA Channel	Level	25

Table 22-2: ADSP-SC59x SPI Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
127	SPI1_STAT	SPI1 Status	Level	
128	SPI1_ERR	SPI1 Error	Level	
129	SPI2_TXDMA	SPI2 TX DMA Channel	Level	26
130	SPI2_RXDMA	SPI2 RX DMA Channel	Level	27
131	SPI2_STAT	SPI2 Status	Level	
132	SPI2_ERR	SPI2 Error	Level	
133	SPI3_TXDMA	SPI3 TX DMA Channel	Level	55
134	SPI3_RXDMA	SPI3 RX DMA Channel	Level	56
135	SPI3_STAT	SPI3 Status	Level	
136	SPI3_ERR	SPI3 Error	Level	
261	SPI0_TXDMA_ERR	SPI0 TX DMA Channel Error	Level	
262	SPI0_RXDMA_ERR	SPI0 RX DMA Channel Error	Level	
263	SPI1_TXDMA_ERR	SPI1 TX DMA Channel Error	Level	
264	SPI1_RXDMA_ERR	SPI1 RX DMA Channel Error	Level	
265	SPI2_TXDMA_ERR	SPI2 TX DMA Channel Error	Level	
266	SPI2_RXDMA_ERR	SPI2 RX DMA Channel Error	Level	
267	SPI3_TXDMA_ERR	SPI3 TX DMA Channel Error	Level	
268	SPI3_RXDMA_ERR	SPI3 RX DMA Channel Error	Level	

ADSP-SC59x SPI Trigger List

Table 22-3: ADSP-SC59x SPI Trigger List Generators

Trigger ID	Name	Description	Sensitivity
77	SPI0_TXDMA	SPI0 TX DMA Channel	Edge
78	SPI0_RXDMA	SPI0 RX DMA Channel	Edge
79	SPI1_TXDMA	SPI1 TX DMA Channel	Edge
80	SPI1_RXDMA	SPI1 RX DMA Channel	Edge
81	SPI3_TXDMA	SPI3 TX DMA Channel	Edge
82	SPI3_RXDMA	SPI3 RX DMA Channel	Edge
83	SPI2_TXDMA	SPI2 TX DMA Channel	Edge
84	SPI2_RXDMA	SPI2 RX DMA Channel	Edge

Table 22-4: ADSP-SC59x SPI Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
52	SPI0_TXDMA	SPI0 TX DMA Channel	Pulse
53	SPI0_RXDMA	SPI0 RX DMA Channel	Pulse
54	SPI1_TXDMA	SPI1 TX DMA Channel	Pulse
55	SPI1_RXDMA	SPI1 RX DMA Channel	Pulse
56	SPI3_TXDMA	SPI3 TX DMA Channel	Pulse
57	SPI3_RXDMA	SPI3 RX DMA Channel	Pulse
58	SPI2_TXDMA	SPI2 TX DMA Channel	Pulse
59	SPI2_RXDMA	SPI2 RX DMA Channel	Pulse

ADSP-SC59x SPI DMA Channel List

Table 22-5: ADSP-SC59x SPI DMA Channel List

DMA ID	DMA Channel Name	Description
DMA22	SPI0_TXDMA	SPI0 TX DMA Channel
DMA23	SPI0_RXDMA	SPI0 RX DMA Channel
DMA24	SPI1_TXDMA	SPI1 TX DMA Channel
DMA25	SPI1_RXDMA	SPI1 RX DMA Channel
DMA26	SPI2_TXDMA	SPI2 TX DMA Channel
DMA27	SPI2_RXDMA	SPI2 RX DMA Channel
DMA55	SPI3_TXDMA	SPI3 TX DMA Channel
DMA56	SPI3_RXDMA	SPI3 RX DMA Channel

SPI Block Diagram

The ***SPI Controller Block Diagram*** illustrates the blocks of the SPI module. The module is comprised of three primary parts:

- SPI core contains the receive and transmit FIFOs and their associated shift registers
- Control blocks contain the synchronizer and logic to control the data flow through the data pipelines
- Register block

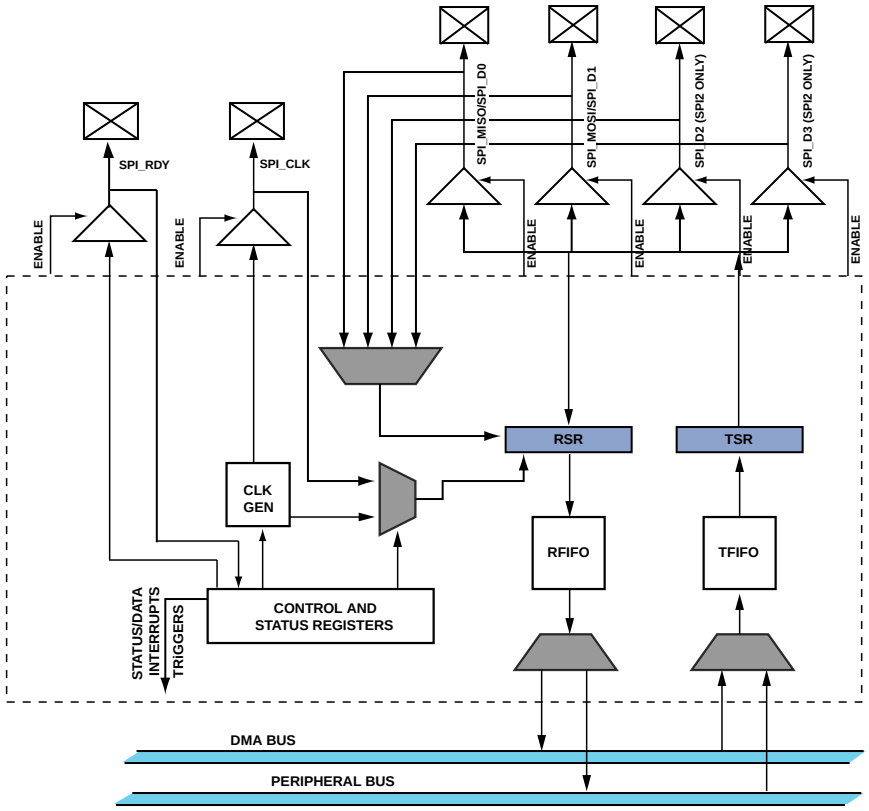


Figure 22-1: SPI Controller Block Diagram, Quad Mode

Transfer Protocol

The SPI module implements two channels that are independent of each other. The SPI module uses the SPI_RXCTL and SPI_TXCTL dedicated control registers to control these channels. Except in dual and quad modes, SPI can enable and use both channels simultaneously.

The SPI protocol supports four different combinations of serial clock phase and polarity. These combinations are selected through the SPI_CTL.CPOL and SPI_CTL.CPHA bits.

The ***SPI Transfer Protocol*** figures demonstrate the two basic transfer formats as defined by the CPHA bit. Two waveforms are shown for SPI_CLK; one for SPI_CTL.CPOL=0 and the other for SPI_CTL.CPOL=1. The diagrams can be interpreted as controller or target timing diagrams since the SPI_CLK, SPI_MISO, and SPI_MOSI pins are directly connected between the controller and the target. The SPI_MISO signal is the output from the target (target transmission), and the SPI_MOSI signal is the output from the controller (controller transmission). The controller generates the SPI_CLK signal. The SPI_SS signal is the target device select input to the target from the controller. The diagrams represent an 8-bit transfer (SPI_CTL.SIZE=0) with the MSB first (SPI_CTL.LSBF=0). Any combination of the SPI_CTL.SIZE and SPI_CTL.LSBF bits is permissible. For example, a 16-bit transfer with the LSB first is another possible configuration.

The clock polarity and the clock phase could be identical for the controller device and the target device involved in the communication link. The transfer format from the controller can be changed between transfers to adjust for various requirements of a target device.

The SPI module uses the SPI_CTL.ASSEL bit to determine when the SPI hardware or software control the SPI_SEL[n] line. When SPI_CTL.ASSEL=1, the target select line must be set to the polarity set in the SPI_CTL.SELST field between each serial transfer. The actual behavior of SPI_SEL[n] also depends on the parameters programmed into the SPI_DLY register. The SPI hardware logic automatically controls this functionality. When SPI_CTL.ASSEL=0, SPI_SEL[n] can either remain active between successive transfers or be inactive. The software must control this activity through manipulation of the SPI_SLVSEL register.

The ***SPI Transfer Protocol*** pair of figures illustrates the case when SPI_CTL.ASSEL = 1 and the SPI_SEL[n] line is inactive between frames. If ASSEL = 0, the SPI_SEL[n] line can remain active between frames; however, the first bit is only driven when an active transition of SPI_CLK occurs.

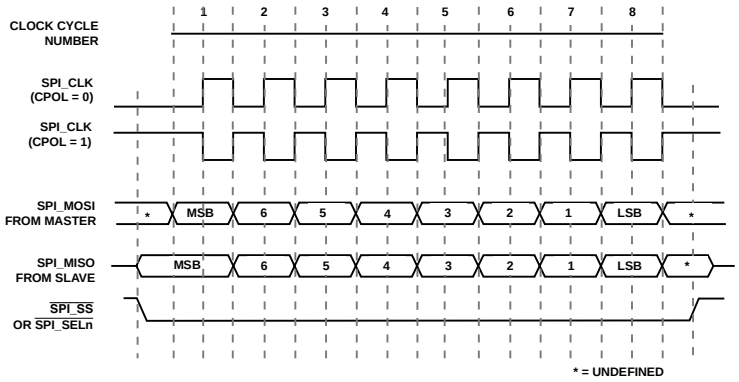


Figure 22-2: SPI Transfer Protocol for CPHA=0

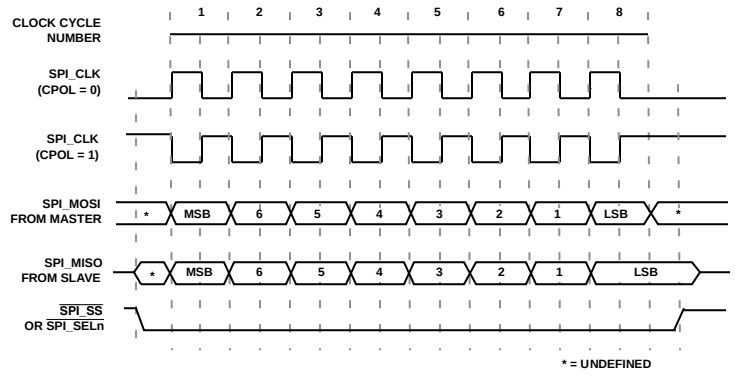


Figure 22-3: SPI Transfer Protocol for CPHA=1

Clock Considerations

The SPI_CLK signal is a gated clock that is only active during data transfers, for the time of the transferred word. In normal mode, the number of active edges is equal to the number of bits to be transmitted or received. In dual-I/O mode, it is half of the number of bits to be transmitted or received, and in quad-SPI mode it is one-fourth of the number. The clock rate can be derived using both even and odd dividers from CLKO6.

For controller devices, the SPI uses the SPI_CLK register value to determine the clock rate, whereas this value is ignored for target devices.

When the SPI controller is a controller, SPI_CLK is an output signal. Conversely, when the SPI controller is a target, SPI_CLK is an input signal. Peripheral devices ignore the SPI clock when the target select input is driven inactive. The SPI uses the SPI_CLK signal to shift out and shift in the data driven onto the SPI_MISO and SPI_MOSI lines. The data is always shifted out on one edge of the clock (the active edge) and sampled on the opposite edge of the clock (the sampling edge). Clock polarity and clock phase relative to data are programmable through the SPI_CTL register and define the transfer format.

Controlling Delay Between Frames

The ***SPI Timing with Lead and Lag Programming (Independent of SPI_CTL.CPHA Setting)*** figure illustrates SPI timing using the SPI_DLY.LEADX and SPI_DLY.LAGX programming. The SPI uses the SPI_DLY.LAGX bits to control the timing between the chip select (SPI_SS) signal assertion and the first SPI_CLK edge. The SPI uses the SPI_DLY.LEADX bits to control the timing between the last SPI_CLK edge and deassertion of the SPI_SS signal. The lead and lag timing can be extended by a 1 SPI_CLK duration to ease timing restrictions on the target device.

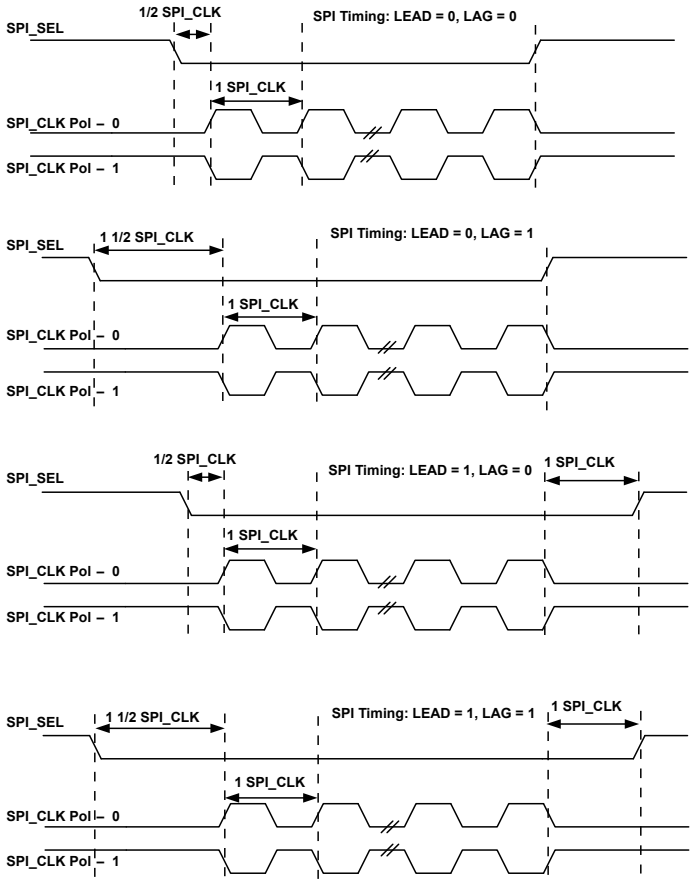


Figure 22-4: SPI Timing with Lead and Lag Programming (Independent of SPI_CTL.CPHA Setting)

The ***SPI Timing with SPI_DLY.STOP Programming (Independent of SPI_CTL.CPHA Setting)*** figure illustrates SPI timing with STOP programming. The SPI module uses this timing to insert multiples of `SPI_CLK` period delays between transfers. The `SPI_SS` line is deasserted for the duration specified in the `SPI_DLY.STOP` bit field, assuming the `SPI_CTL.SELST` bit is configured for deassertion between transfers.

If the `SPI_DLY.STOP` bit =0, the controller operates in a *continuous mode*. This mode causes an immediate start of the second word after the last bit is transferred from the first word. During this mode of operation, the chip select line is continuously asserted.

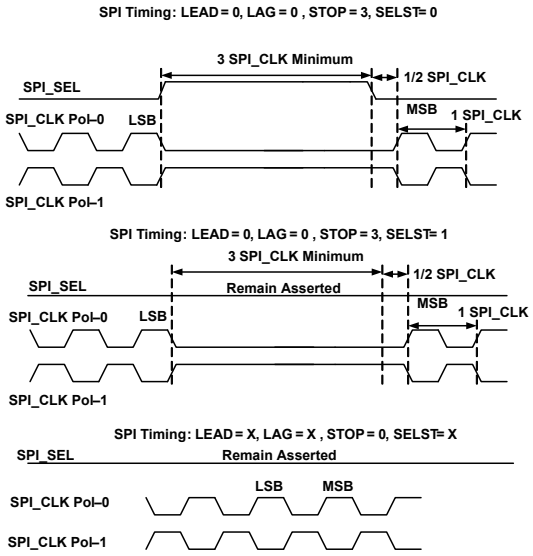


Figure 22-5: SPI Timing with SPI_DLY.STOP Programming (Independent of SPI_CTL.CPHA Setting)

When the SPI_DLY.STOP bit =0 and initial conditions for a transfer are not met, the interface pauses before the next transfer. During this pause, the SPI uses the SPI_CTL.SELST bit to determine the state of the chip select pin. The SPI uses the SPI_DLY.LEADX and SPI_DLY.LAGX bits to determine the timing between SPI_CLK edges and the chip select line.

Flow Control

In controller mode, the target device must drive the SPI_RDY pin. The pin acts as an input signal. The target can deassert the SPI_RDY pin to stop the controller from initiating any new transfer. If SPI_RDY is deasserted in the middle of a transfer, the current transfer continues, and the next transfer will not start unless the target asserts the SPI_RDY signal. Whenever the target deasserts SPI_RDY and stalls the controller, the SPI controller goes into a waiting state, and the SPI_STAT.FCS bit is set. When the target asserts SPI_RDY, the SPI controller resumes operation, and the SPI_STAT.FCS bit is cleared.

In target mode, the SPI_RDY pin acts as an output signal. Flow control can be configured on either the TX channel or the RX channel. The SPI uses the SPI_CTL.FCCH bit to control this configuration. If flow control is configured on the TX channel, as the SPI_TFIFO status nears the empty condition, the SPI_RDY pin is deasserted. If flow control is configured on the RX channel, as the SPI_RFIFO status nears the full condition, the SPI_RDY pin is deasserted. The SPI uses the SPI_CTL.FCWM bits to control the FIFO status at which SPI_RDY deassertion takes place. Flow control in target mode is purely based on the FIFO status and does not depend on the word counters.

The ***SPI Flow Control Timing in Controller Mode*** figure illustrates this timing.

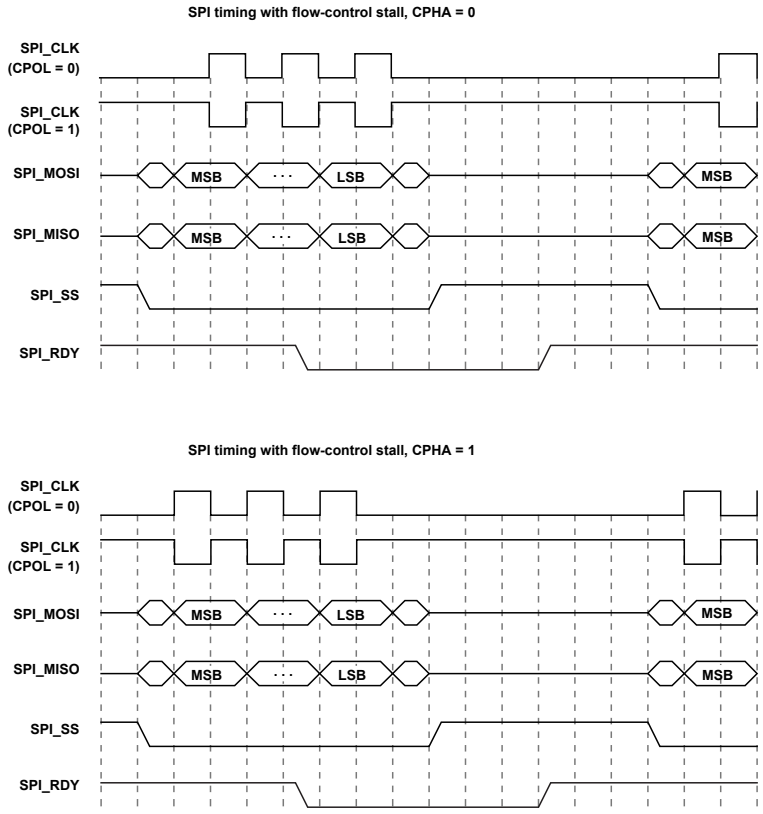


Figure 22-6: SPI Flow Control Timing in Controller Mode.

Target Select Operation

If the SPI is in target mode, $\overline{\texttt{SPI_SS}}$ acts as the target select input. When SPI is enabled as a controller, $\overline{\texttt{SPI_SS}}$ can serve as an error detection input for the SPI in a multi-controller environment. The `SPI_CTL.PSSE` bit enables this feature. When `SPI_CTL.PSSE`=1, the $\overline{\texttt{SPI_SS}}$ input is the controller mode error input. Otherwise, $\overline{\texttt{SPI_SS}}$ is ignored.

The $\overline{\texttt{SPI_SS}}$ signal is an active-low signal. The controller asserts the signal during the transfer. The signal can be deasserted or remain asserted between transfers. When $\overline{\texttt{SPI_SS}}$ is deasserted, `SPI_CLK` and inputs are ignored, and outputs are three-stated.

The target select bits (`SPI_SLVSEL.SSEL1` – `SPI_SLVSEL.SSEL7`) are used in a multiple-target SPI environment. For example, if there are eight SPI devices in the system including a processor controller, the controller processor can support the SPI mode transactions across the other seven devices. This configuration requires only one controller processor in this multi-target environment.

For example, assume that the SPI of the processor is the controller. The `SPI_SLVSEL.SSEL1` – `SPI_SLVSEL.SSEL7` bits on the processor can be connected to the target select pin of each target device. In this configuration, the target select bits can be used in three ways. In cases 1 and 2, the processor is the controller and the seven microcontrollers or targets with SPI interfaces are targets. The processor can do one of the following:

1. Transmit to all seven SPI devices at the same time in a broadcast mode. Here, all target select bits are set.

2. Receive and transmit from one SPI device by enabling only one target SPI device at a time.
3. If all the targets are also processors, then the requester can receive data from only one processor at a time. (The functionality is enabled by clearing the SPI_CTL.EMISO bit in the six other slave processors.) The requestor can transmit broadcast data to all seven at the same time. This MISO enabling feature is available in some other microcontrollers. Therefore, it is possible to use the MISO enabling feature with any other SPI device that includes this functionality.

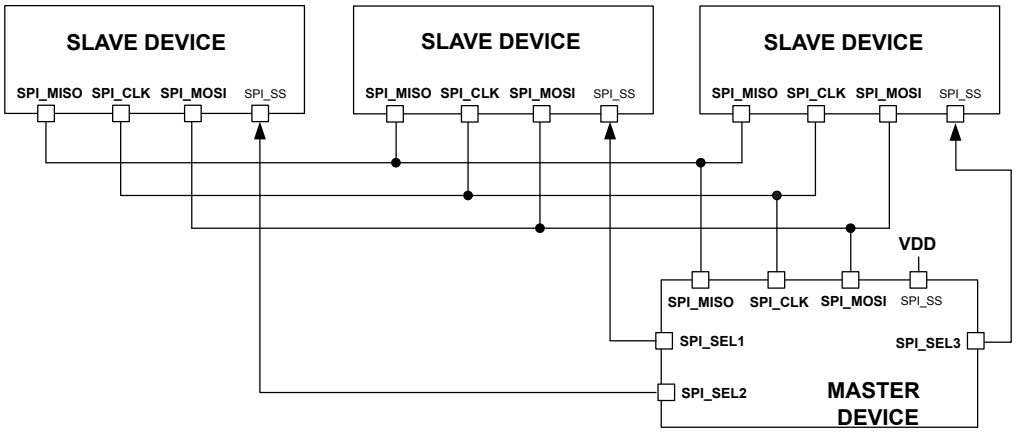


Figure 22-7: Single-Controller, Multiple-Target Configuration

Beginning and Ending a Non-DMA SPI Transfer

The start and finish of a non-DMA SPI transfer depend on the following settings.

1. Whether the device is configured as a controller or a target.
2. The state of the SPI_CTL.ASSEL bit, which selects between hardware and software control over SPI_SLVSEL.

When SPI_CTL.CPHA=0, the enabled target select outputs are driven active. However, the SPI_CLK signal remains inactive for the first half of the first cycle of SPI_CLK. For a target with SPI_CTL.CPHA=0, the transfer starts as soon as the SPI_SS input goes low.

When SPI_CTL.CPHA=1, a transfer starts with the first active edge of SPI_CLK for both target and controller devices. For a controller device, a transfer is complete after it sends the last data and simultaneously receives the last data bit. A transfer for a target device ends after the last sampling edge of SPI_CLK. If SPI_CTL.ASSEL=1, the hardware maintains responsibility for toggling SPI_SS between frames. If SPI_CTL.ASSEL=0, software controls the SPI_SS line and can keep it active between frames.

The SPI_STAT.RFE bit defines when the receive buffer can be read, indicating that SPI_RFIFO is not empty. The SPI_STAT.TFF bit defines when the transmit buffer can be written, indicating that the SPI_TFIFO is not full. The end of a single word transfer occurs when the SPI_STAT.RFE bit is cleared. The status indicates that a new word has been received and written into the receive FIFO. The SPI_STAT.RFE bit remains cleared when the receive FIFO has valid data.

To maintain software compatibility with other SPI devices, the SPI_STAT.SPIF bit is also available for polling. This bit can have a slightly different behavior from other commercially available devices.

In controller mode with the SPI_CTL.ASSEL bit cleared, software manually asserts the required target select signal before starting the transaction. After all data transfers, software typically releases the target select line.

When the receive or transmit word counters are enabled in the SPI_TXCTL or SPI_RXCTL registers, the SPI generates a finish interrupt at the end of the transfer. It signals the end of all transfers related to that transaction.

Transmit Operation in Non-DMA Mode

The transmit operation in non-DMA mode is enabled through the SPI_TXCTL.TEN bit. It can be enabled independently from the receive operation, and the transmit channel can become the initiating channel based on the SPI_TXCTL.TTI bit setting.

Transmit underrun is not possible in this mode, as no new transfer initiates unless the transmit FIFO is empty (in the case that SPI_TXCTL.TTI =1). A receive overflow is detected when data from a new frame transfer replaces older data in a full receive FIFO. This event can occur if SPI_TXCTL.TTI =1 and the receive channel is enabled in a non-initiating capacity.

An SPI transmit interrupt request is signaled once the transmit channel has been enabled and the transmit FIFO is not full. The SPI uses the SPI_TXCTL.TDR bit setting to control the frequency of the interrupt request.

Receive Operation in Non-DMA Mode

The receive operation in non-DMA mode is enabled through the SPI_RXCTL.REN bit. It can be enabled independently from the transmit operation, and the receive channel can become the initiating channel based on the SPI_RXCTL.RTI bit setting.

Receive overflow is not possible in this mode, as no new transfer initiates when the receive FIFO is full (in the case of SPI_RXCTL.RTI =1). A transmit underrun can occur (SPI_TXCTL.TDU bit) when no valid data is in the SPI_TFIFO register when a transfer is initiated. This event can occur if SPI_RXCTL.RTI =1 and the transmit channel is enabled in a non-initiating capacity.

An SPI receive interrupt request is signaled once the receive channel has been enabled and there is data waiting to be read. The SPI uses the SPI_RXCTL.RDR bit setting to control the frequency of the interrupt request.

Dual I/O Mode

In Dual I/O mode, the SPI_MISO and SPI_MOSI pins are configured to operate in the same direction which doubles bandwidth. The SPI uses the SPI_CTL.SOSI bit to determine the order of bits on the pins. When set, the processor sends the first bit on the SPI_MOSI pin and the second bit on the SPI_MISO pin. If the SPI_CTL.SOSI bit is cleared, the order is reversed. Since dual I/O mode uses both pins to transmit or receive data, only one channel can be enabled, either transmit or receive. Flow control through the SPI_RDY pin is supported. Interrupt request generation is unaffected by dual I/O mode. However, the interrupt service interval is reduced because the individual transfer latency is halved.

Changing to quad SPI mode must be done when the SPI is in a quiescent state.

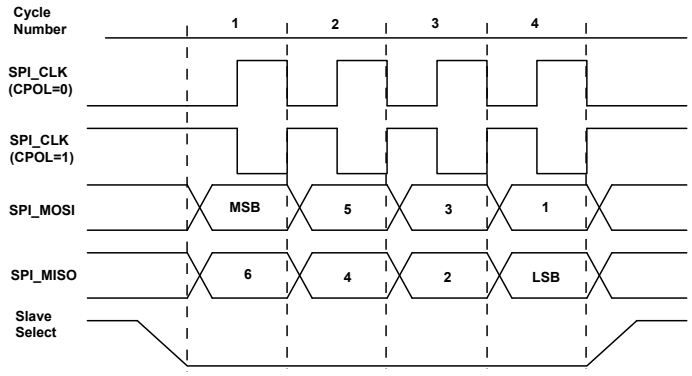


Figure 22-8: Dual I/O Mode Transfer Protocol for CPHA=0, SOSI=1, 8-Bit Transfer, LSBF=0.

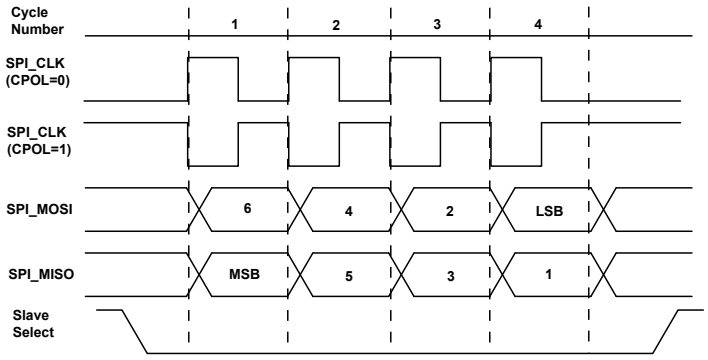


Figure 22-9: Dual I/O Mode Transfer Protocol for CPHA=1, SOSI=0, 8-Bit Transfer, LSBF=0.

Quad I/O Mode (SPI1/SPI2)

In quad SPI mode, the SPI_MISO and SPI_MOSI pins, in tandem with the SPI_D2 and SPI_D3 pins, are configured to operate in the same direction. The SPI uses the SPI_CTL.SOSI bit to determine the order of bits on the pins. When set, the processor sends:

- The first bit on the SPI_MOSI pin
- The second bit on the SPI_MISO pin
- The third bit on the SPI_D2 pin
- The fourth bit on the SPI_D3 pin

If the SPI_CTL.SOSI bit is cleared, the order is reversed. Since quad SPI mode uses all four pins to transmit or receive data, only one channel can be enabled as either transmit or receive. Flow control through the SPI_RDY pin is supported. Interrupt generation is unaffected by quad SPI mode.

Changing to quad SPI mode must be done when the SPI is in a quiescent state.

While using dual or quad I/O mode for communicating with flash devices, program the SPI_CTL.CPHA and the SPI_CTL.CPOL bits =1. This programming avoids bus contention during read operations, because the flash device starts driving out the bits immediately after dummy cycles in read header.

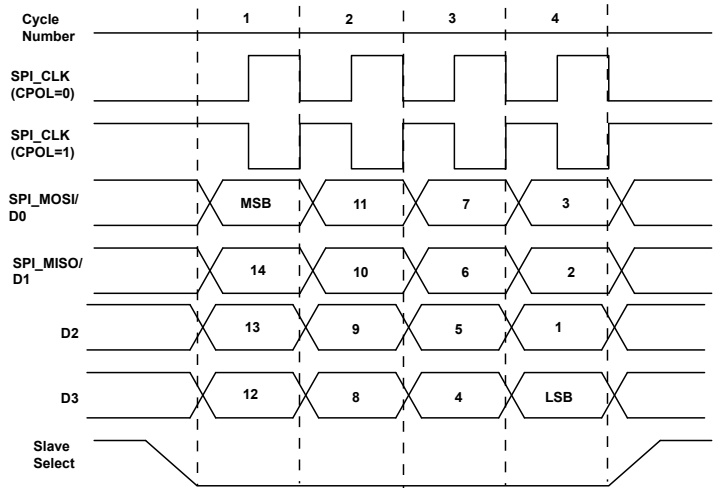


Figure 22-10: Quad Mode Timing for CPHA=0, SOSI=1, 16-Bit Transfer, LSBF=0.

NOTE: The SPI does support quad SPI 8-bit transfer in target continuous mode of operation with an CLKO6:SPI_CLK ratio of less than 1:2. A minimum of 2 CLKO6 cycles is required between transfers in 8-bit quad SPI target mode with an CLKO6:SPI_CLK ratio of less than 1:2.

Fast Mode

Fast mode is like the normal mode of operation when transmitting. When receiving, data is sampled at the next transmit edge allowing a full cycle of timing in the receive direction. This mode is valid in controller mode operation only. When the SPI operates in fast mode, the target drives the data for one full cycle.

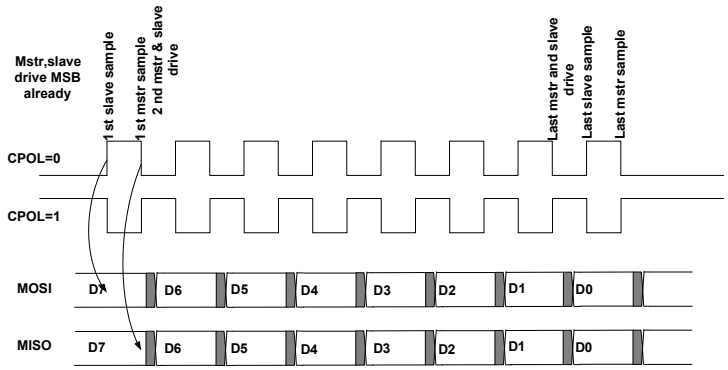


Figure 22-11: SPI Transfer Protocol in Fast Mode for SPI_CTL.CPHA = 0

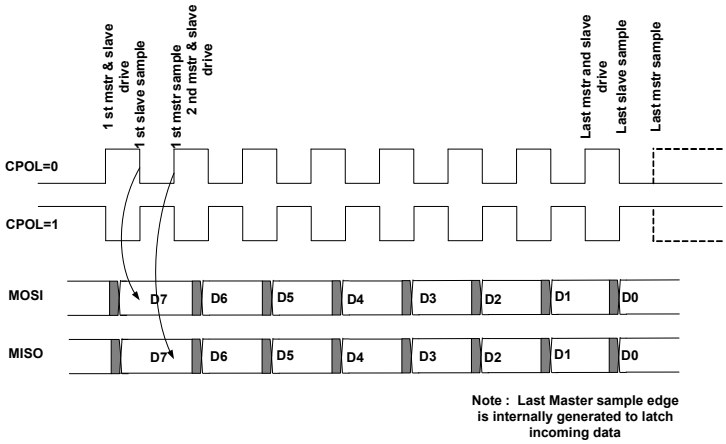


Figure 22-12: SPI Transfer Protocol in Fast Mode for SPI_CTL.CPHA = 1

Memory-Mapped Mode (SPI2)

The SPI supports direct memory-mapped read accesses from a SPI memory device, enabled by setting the `SPI_CTL.MMSE` bit. This mode allows for direct execution of instructions from a SPI memory device without the need for a low-level software driver, as hardware handles all overhead tasks (for example, transmission of the read header, pin turnaround timing, and receive data sizing). The SPI features configurable options in the memory-mapped read header register (`SPI_MMRDH`) to provide compatibility with a wide range of SPI memory devices.

In non-memory-mapped mode, the software is responsible for providing the command and required dummy words for the read response, whereas this is all handled by hardware when the SPI is in memory-mapped mode. The memory of the SPI device is accessible directly through reads of the processor address space. The read accesses can be code or data accesses in core mode or when using memory DMA (MDMA). These accesses allow code to execute directly from SPI memory devices (true eXecute-In-Place operations), and the contents can be cached to improve performance. It is not necessary to access the SPI data buffer registers nor poll status bits; however, the hardware does not support peripheral DMA accesses nor write operations to the SPI memory space.

The ***Types of Operations*** table is a comparison of the permitted operations in the non-memory-mapped and memory-mapped modes supported by the SPI controller.

Table 22-6: Types of Operations

SPI Operation	Non-Memory-Mapped Mode	Memory-Mapped Mode
Core data write	Yes	No
Core data read	Yes	Yes
Code fetch: Execute-In-Place (XIP)	No	Yes
Read/Write accesses using SPI peripheral DMA	Yes	No
Read/Write accesses by other peripheral DMA channels	No	No
MDMA read*	No	Yes

Table 22-6: Types of Operations (Continued)

SPI Operation	Non-Memory-Mapped Mode	Memory-Mapped Mode
MDMA write	No	No

Memory-Mapped Description of Operation

Memory-mapped mode is enabled by setting the SPI_CTL.MMSE bit. When enabled, the SPI (if ready) accepts the read requests through a dedicated on-chip target interface. The memory subsystem controller drives this dedicated interface through the SCB fabric.

In a typical scenario, the memory subsystem controller issues read requests to the fabric, and the fabric routes these requests to the target port of the SPI peripheral. The controller describes the read access using a number of parameters such as starting address, transfer size, and burst type. The SPI responds to this read access request when it is ready for a new transfer. It loads the opcode, a specified number of address bytes, and an optional mode byte into the transmit FIFO. The SPI memory state machine begins when both the transmit and receive channels of the SPI are enabled with:

- the transmit transfer initiation bit is set (SPI_TXCTL.TTI=1), *and*
- the receive initiation bit is cleared (SPI_RXCTL.RTI=0)

The SPI memory read sequence starts with the assertion of $\overline{\text{SPI_SEL1}}$. If the SPI memory state machine is in the reset state, it looks for a command. The SPI hardware then sends the specific 8-bit read command (which can be optionally skipped), followed by the SPI memory read address. After this, a dummy period is inserted, in which a mode byte is optionally sent, and the pins are held or three-stated during the dummy clocking period.

NOTE: This read header is transmitted over the SPI standard protocol pins (SPI_CLK, SPI_MOSI, SPI_MISO, $\overline{\text{SPI_SEL1}}$) or over the extended SPI protocol pins (SPI_CLK, SPI_MOSI, SPI_MISO, SPI_D2, SPI_D3, $\overline{\text{SPI_SEL1}}$), based on the SPI_MMRDH.CMDPINS, SPI_MMRDH.ADRPINS, and SPI_CTL.MIOM bit settings. SPI memory devices usually support communication in MSB-first mode. In dual mode, the SPI typically uses SPI_MISO as IO1 and SPI_MOSI as IO0. In quad mode, the SPI typically uses SPI_D3 pin as IO3, SPI_D2 as IO2, SPI_MISO as IO1, and SPI_MOSI as IO0.

When all I/O data pins are three-stated, the SPI continues clocking the SPI memory device, which drives out the data bits at the addressed location, until all bytes are received. The SPI hardware reads the data as configured by the SPI_CTL.MIOM bit setting. Upon reception of the last byte, the SPI typically deasserts $\overline{\text{SPI_SEL1}}$ to prepare for the next requested read header.

Application code must ensure that the opcode sent is consistent with multiple I/O programming and that the parameters specified in the memory-mapped read header register are consistent with flash read access timing.

The ***SPI Memory-Mapped Register Operations Flow*** diagram shows how the fields of the SPI_MMRDH register determine the read header while initiating transfers in memory-mapped mode.

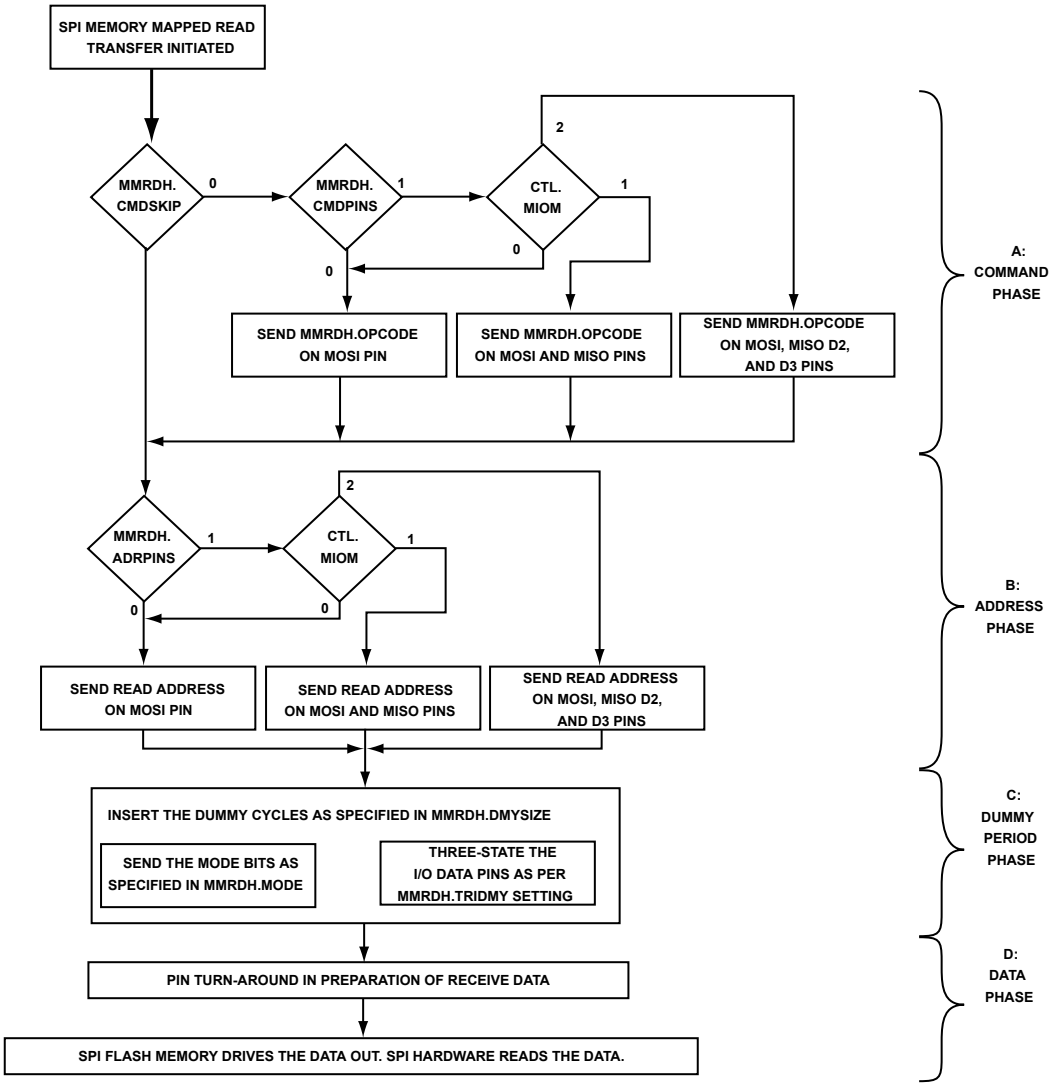


Figure 22-13: SPI Memory-Mapped Register Operations Flow

Memory-Mapped Architectural Concepts

In memory-mapped mode, the SPI accepts read requests through a dedicated on-chip slave interface. The SPI (when ready) accepts these requests and begins the process of assembling the read header based on access attributes described in both the SPI_MMRDH register and the internal bus request. After the read header transmission completes, a pin turnaround period is timed, and the receiver is enabled. The SPI continues clocking the SPI memory device until all bytes are received.

The SPI memory-mapped hardware accommodates various memory devices with different read timing. The capabilities include extra mode bits, flexible dummy period timing, and three-state control, as configured in the SPI_MMRDH register.

The ***Memory-Mapped Protocol*** figure shows the protocol for the SPI controller in memory-mapped mode.

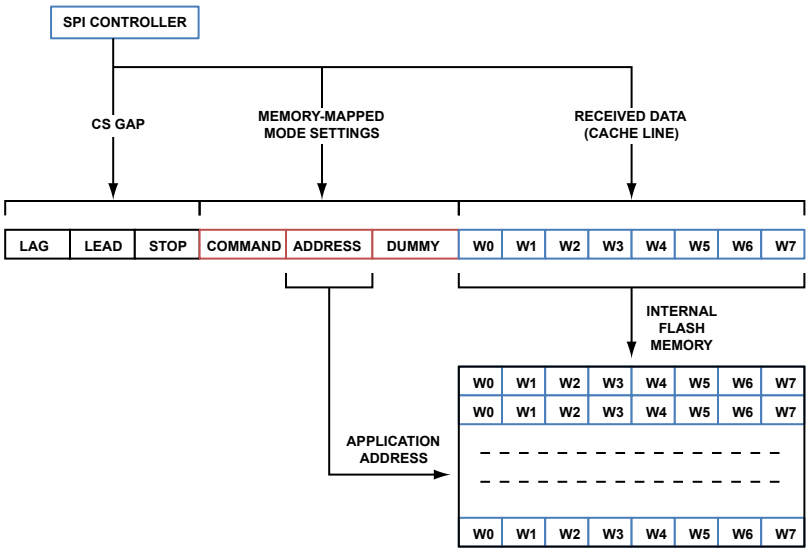


Figure 22-14: Memory-Mapped Protocol

As shown in the figure, the COMMAND field (`SPI_MMRDH.OPCODE`) is transmitted upon assertion of the `SPI_SEL[n]` signal. The SPI memory interprets this 8-bit value as a read command. Any 8-bit read opcode whose timing is compliant with the processor SPI and features provided by memory-mapped hardware is allowed, the most common being:

- Standard Read (0x03)
- Fast Read (0x0B)
- Fast Read Dual Output (0x3B)
- Fast Read Dual I/O (0x6B)
- Fast Read Quad Output (0xBB)
- Fast Read Quad I/O (0xEB)

NOTE: The SPI hardware does not validate the content of the `SPI_MMRDH.OPCODE` field prior to transmitting.

DMYSIZE (Number of Dummy Bytes)

When operating at a high clock frequency in multi-IO modes, most flash devices require some dummy clocks after the address bits. These dummy clock cycles allow the internal circuits of the device extra time for setting up the initial address. These bits specify the number of bytes separating address transmission and read data return.

The number of dummy cycles required varies per manufacturer, the read command used, and the SPI access time. The SPI hardware allows dummy cycles to be programmed in bytes in the `SPI_MMRDH.DMYSIZE` field, the value of which is a function of the number of pins used to transmit the address (`SPI_MMRDH.ADRPINS`), as shown in the ***Pins Used to Transmit the Address (ADRPINS)*** table.

Table 22-7: Pins Used to Transmit the Address (ADRPINS)

	Dummy clock cycles		
SPI_MMRDH.DMYSIZE	(**SPI_MMRDH.ADRPINS**=0, **SPI_CTL.MIOM**=x) Dummy bytes elapse over 1-pin	(**SPI_MMRDH.ADRPINS**=1, **SPI_CTL.MIOM**=1) Dummy bytes elapse over 2-pins	(**SPI_MMRDH.ADRPINS**=1, MIOM=2) Dummy bytes elapse over 4-pins
000	0	0	0
001	8	4	2
010	16	8	4
011	24	12	6
100	32	16	8
101	40	20	10
110	48	24	12
111	56	28	14

This dummy clocking period allows the mode bits to be sent, the pins to be three-stated, and the pins to be turned around in preparation for the receive data.

Memory-Mapped Read Accesses

The SPI hardware supports the most used read operations.

- Two standard SPI reads (read and read fast), which use the unidirectional `SPI_MOSI` and `SPI_MISO` pins in addition to `SPI_SEL[n]` and `SPI_CLK`
- Four extended SPI multiple I/O reads: dual output, quad output, dual I/O, and quad I/O reads

The ***SPI Read Operations*** table and ***SPI Flash Fast Read Sequence*** figures summarize the types of read operations. Program each read operation in a way that is compatible with the description given in the SPI flash data sheet.

Table 22-8: SPI Read Operations

Operation	Read Command (Opcode)	CMDPIN	ADRPIN	DMYSIZE	Three-state	Multiple I/O Mode	Data Pins
Read	0x03	1	1	Zero	No	No	1
Fast Read	0x0B	1	1	Non-Zero	Yes	No	1
Dual Output Read	0x3B	1	1	Non-Zero	Yes	Yes(IO0-1)	2
Quad Output Read	0x6B	1	1	Non-Zero	Yes	Yes(IO0-3)	4
Dual I/O Read	0xBB	1, 2	2	Non-Zero	Yes	Yes (IO0-1)	2
Quad I/O Read	0xEB	1, 4	4	Non-Zero	Yes	Yes (IO0-3)	4

Some memory devices also support word quad I/O read (0xE7) and octal quad I/O read (0xE3) operations. These operations require fewer dummy cycles than normal quad I/O read operations.

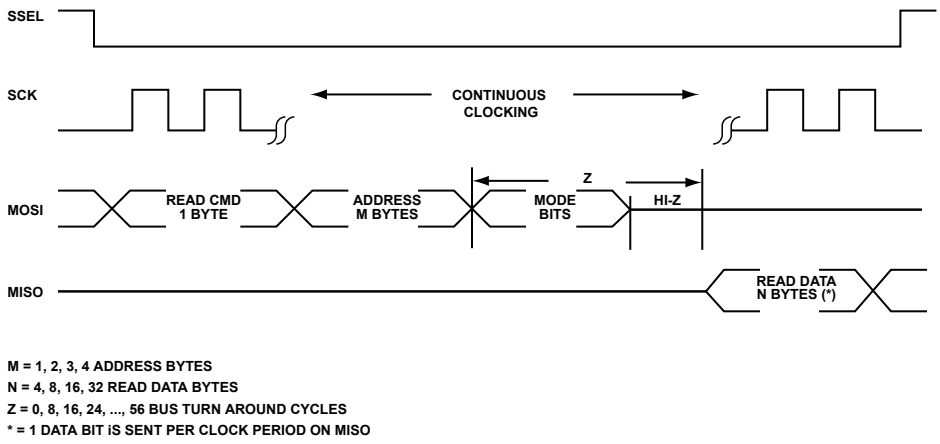


Figure 22-15: SPI Flash Fast Read Sequence

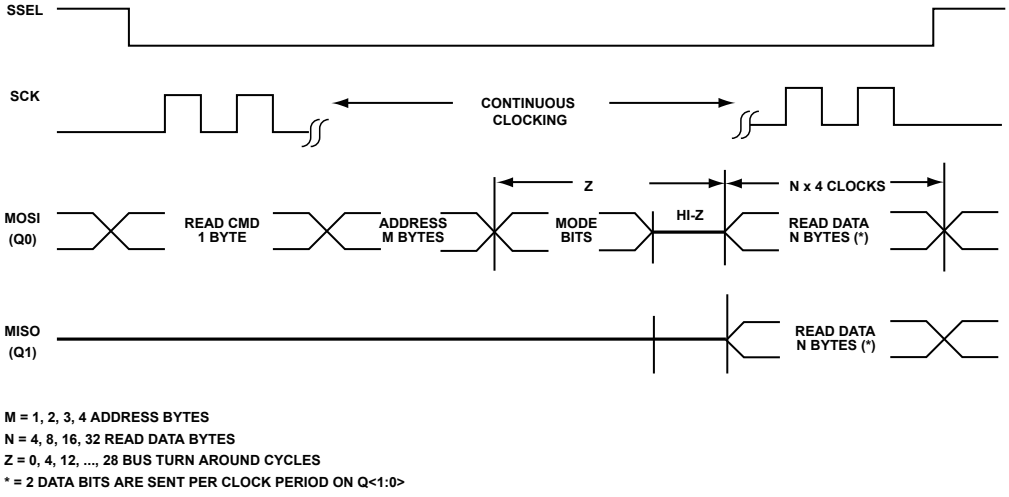


Figure 22-16: SPI Flash Fast Read (Dual Output) Sequence

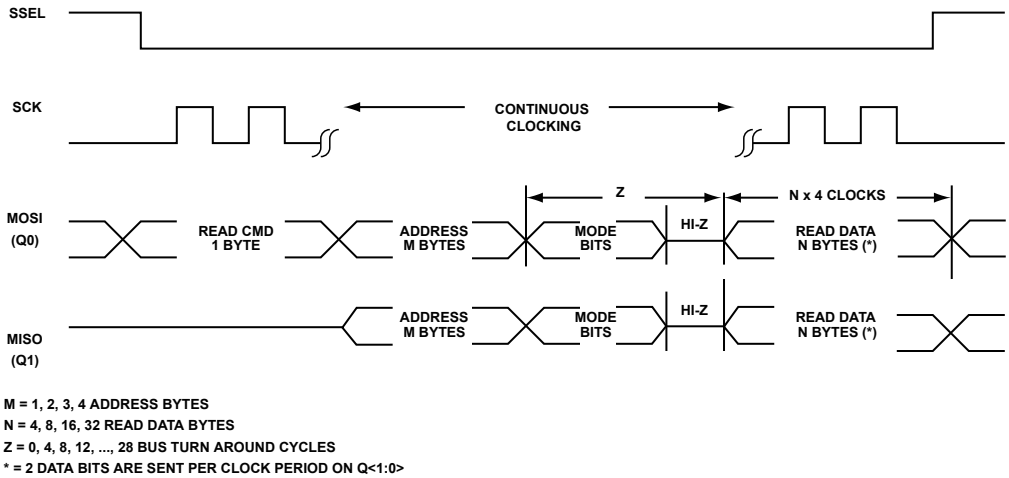


Figure 22-17: SPI Flash Fast Read (Dual I/O) Sequence

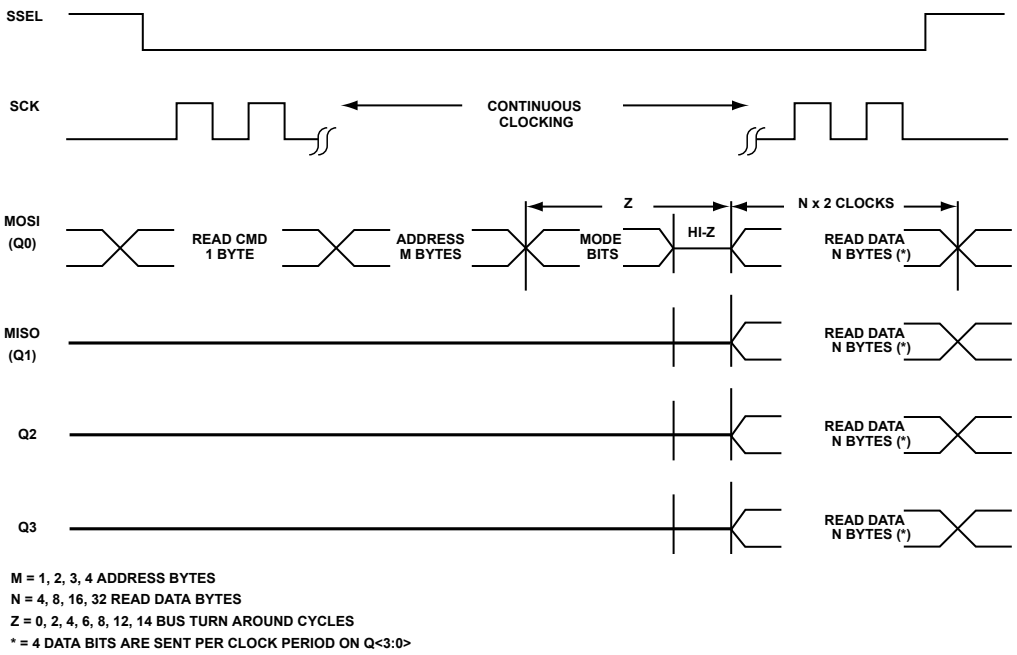


Figure 22-18: SPI Flash Quad Output Read Sequence

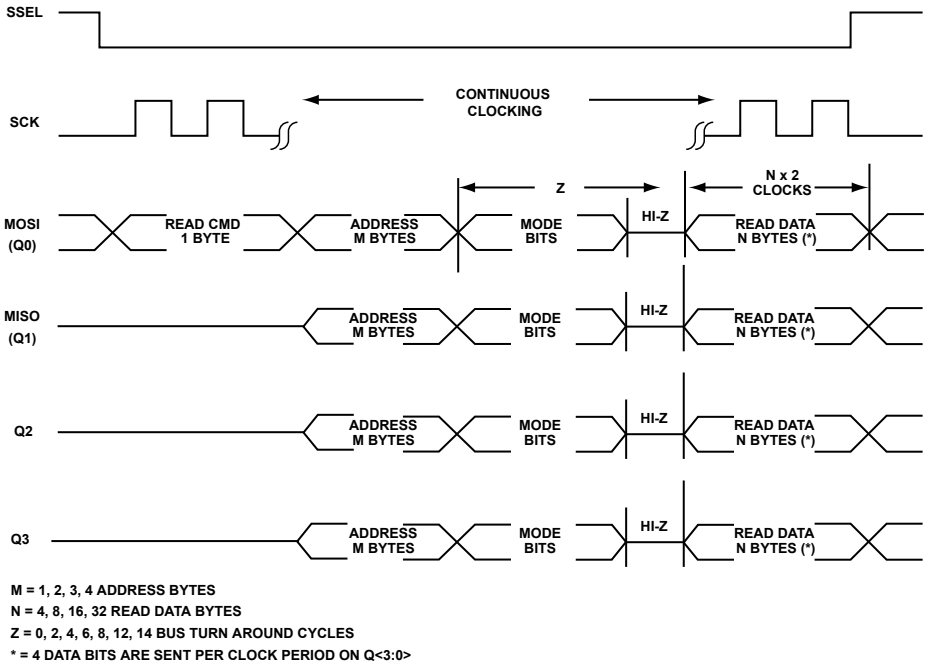


Figure 22-19: SPI Flash Quad I/O Read Sequence

SPI memory-mapped reads can be made cacheable in the core's internal memory by properly configuring the region as cacheable memory without bypass (see the related core's cache configuration documentation for details). In the figures, the number of read data bytes (N) is based on the following:

- For an instruction fetch by core (when in XIP mode); the number of instruction bytes to be fetched depends on the cache line size of the cache.

- For a data fetch by the core (data read), the number of data bytes to be fetched depends on the cache line size of the cache.

Although the minimum size of a memory-mapped data read transfer is 4 bytes, applications can fetch a single byte or a 2-byte data. (For example, it can fetch an unsigned char or short access in C code). In this case, only the required bytes are provided to the core and the other bytes are cached.

The on-chip memory subsystem controller provides a starting address for the burst and the SPI hardware issues this address as part of the read header. The address provided is N-byte aligned. For example, to read the 30th byte from SPI memory, then the typical address to provide is:

- 28 (0x0000_001C) for a 32-bit cache line
- 24 (0x0000_0018) for a 64-bit cache line
- 16 (0x0000_0010) for a 128-bit cache line
- 0 (0x0000_0000) for a 256-bit cache line

The read data is returned to the memory subsystem in the order provided by the SPI memory. There can be considerable delay for the expected data provided to the controller.

To minimize this delay, the wrap feature can be used where the memory subsystem provides the address of the critical word.

- For MDMA reads, the number of read data bytes (N) is always equal to 4 bytes. The MDMA read does not depend on the cache setting. For MDMA reads, limit the `DMA_CFG.MSIZE` field to one, two or four bytes. The address provided by the memory subsystem controller to the SPI hardware is always 4-bytealigned.

Memory-Mapped High-Performance Features

In addition to automating the SPI memory read accesses, the memory-mapped hardware also provides some features to improve SPI memory fetches and increase the system performance. The following sections describe these features.

Merged Read Accesses

It is common for the memory subsystem to fetch two or more cache lines from consecutive addresses (the address sequencing is linear without any jumps). To take advantage of this situation, the SPI memory-mapped hardware provides a feature called merging. Enable merging by setting the `SPI_MMRDH.MERGE` bit.

When enabled, the hardware compares the address of an incoming read request to the address of a request the SPI memory is actively servicing. It can decide to merge two accesses when the address for the second access is incremental. For example, if the first address of a 32-byte cache line fetch is 0x0000_0000 and the second fetch is to address 0x0000_0020, then these two accesses can be merged. Merging increases efficiency and overall fetch bandwidth by eliminating the read header for those accesses which only require continuation of the SPI clock.

Wrap Around Accesses

Many SPI flash memory devices support wrapping which is used to enhance critical word fetching of cache lines. In this mode, the SPI device automatically wraps the read address to the base of a cache line once the end of the cache line is reached.

Wrap around accesses are enabled by setting the SPI_MMRDH.WRAP bit.

Some flash devices require programs to send a ***Set Wrap*** command to place the device in wrap mode. Other flash devices provide a configuration register which must be programmed to set the flash in wrap mode. Since the SPI memory-mapped hardware does not support any write operations to flash, perform this step in non-memory-mapped mode (SPI_CTL.MMSE=0) by accessing the SPI registers.

Data access is limited to 8-byte, 16-byte, or 32-byte sections of flash page in wrap mode. The Arm core uses the Wrap 4 access (64-bit data) for L1 cache. The Arm core uses Wrap 4 and Wrap 8 accesses (64-bit data) for L2 cache. The cores use the Wrap 8 accesses for unaligned accesses. During the read request to the SPI memory-mapped hardware, the memory subsystem controller of the processor provides the address of a critical word instead of the line base. The read-data starts at the address specified in the instruction. Once it reaches the end boundary of the 8, 16, or 32-byte section, the output automatically wraps around to the beginning boundary to the line base address. The data fetch continues. It is not necessary to deassert the SPI SPI_SEL[n] signal or resend the read header to wrap to the cache line base when servicing misaligned cache fill requests.

The ***Byte Sequence in Wrap Modes*** table shows byte sequences in various wrap modes.

Table 22-9: Byte Sequence in Wrap Modes

Starting Address	8-Byte Wrap (cache_line = 8 byte)
0	0-1-2- . . . -6-7
1	1-2- 3-. . . -7-0
7	7-0-1- . . . -5-6
15	15-8-9- . . . -13-14
31	31-24-25-. . . .-29-30

The burst with wrap feature allows applications to fetch a critical address quickly. Applications then fill the cache afterwards within a fixed length (8/16/32-byte) of data without issuing multiple read commands. Certain applications can benefit from this feature to improve cache fill efficiency and overall performance of system code execution.

NOTE: Do not use the merge and wrap feature together. Using wrap bursts can unintentionally disable merging (merging cannot occur for unaligned wrapping bursts). A wrap burst can start fetching data words in the middle of the cache line and cannot be merged with the next access.

Execute-In-Place (XIP, SPI2 only)

Execute-In-Place, known as XIP, permits software code to execute directly from an SPI flash device rather than downloading the code and executing it out of RAM. XIP, also known as Command Skip mode, is a general term and can be applied to fetching data as well.

There is a difference between XIP mode and standard mode. In XIP mode, after the SPI memory device is selected (CS# =LOW), the memory device does not decode the first input byte as command code. Instead, it expects the read header to directly start with address bytes. In standard mode, the memory decodes the first input byte it receives as a command code.

The XIP mode dramatically reduces random access time for applications that require fast code execution without shadowing the memory content on a RAM. The SPI memory-mapped hardware provides a control bit, SPI_MMRDH.CMDSKIP to skip the command from read header.

Some SPI memory devices require configuration of their control register to enable the XIP mode of operation, using the non-memory-mapped mode of the processor SPI. Typically, during the dummy cycle period, the mode bits are used to confirm the XIP operation and the SPI_MMRDH.MODE field must be set appropriately. A dummy memory-mapped access may be needed before setting the SPI_MMRDH.CMDSKIP bit to set the SPI memory device in Command Skip mode.

For more details about how to configure SPI memories into XIP mode, refer to the device data sheet.

NOTE: When configuring the flash to XIP mode from the SHARC+ core, ensure that the routine that configures flash to XIP is not routed through the L2CC. This is accomplished by first configuring the flash to XIP mode, then enabling the L2CC from the core.

Memory-Mapped Mode Error Status Bits

The SPI memory-mapped hardware provides bits in the SPI_STAT register to report errors. It provides these bits for notification only and their state has no effect on SPI operations. The status register bits are sticky. A W1C (write-1-to-clear) operation clears the bits.

- Memory-Mapped Write Error (SPI_STAT.MMWE). This bit is set (=1) if an attempt is made to write to address space that is reserved for memory-mapped SPI memory. The SPI memory-mapped hardware does not support automated write access to SPI memory space.
- Memory-Mapped Read Error (SPI_STAT.MMRE). This bit is set (=1) if an attempt is made to read address space reserved for memory-mapped SPI memory while memory mapping is disabled (SPI_CTL.MMSE =0).
- Memory-Mapped Access Error (SPI_STAT.MMAE). This bit is set (=1) if an attempt is made to access either the TX or RX FIFO while memory-mapped access of SPI memory is enabled. In this case, attempts to communicate with the SPI device using legacy methods are blocked and receive fabric reports an error. Legacy methods include any direct access made to the TX and RX FIFOs, whether by DMA or processor MMR.
- Memory-Mapped Write Error Mask (SPI_CTL.MMWEM) bit specifies whether an error response is returned to the fabric on write attempts to address space that is reserved for memory-mapped SPI memory reads. Regardless of whether a write error response is masked using this bit, the memory-mapped write error (SPI_STAT.MMWE) sticky notification bit is still set.

  NOTE: Unlike other bits in the SPI_STAT register, these memory-mapped mode error bits do not have associated bits in the SPI interrupt mask (SPI_IMSK) and SPI interrupt condition (SPI_ILAT) registers.

The memory-mapped top register (SPI_MMTOP) is used to specify the upper limit of the SPI memory address. The memory-mapped accesses to SPI memory addresses equal to or above this range are considered illegal. The accesses are blocked, and a bus error response is generated.

This register is useful to block the invalid SPI memory address accesses. Some SPI memory vendors do not clearly specify (guarantee) that overrange address bits are ignored (address spaces can be wrapped).

Memory-Mapped Programming Guidelines

Setting the SPI_CTL.MMSE bit enables SPI memory-mapped mode. When enabled, the SPI interface is forced to be consistent with SPI memory requirements regardless the settings of certain control bits. The following tables specify typical settings for configuring the SPI in memory-mapped mode:

Table 22-10: SPI Control (SPI_CTL) Register

Bits	Typical values to set	Description	Comments
SPI_CTL.MSTR	1	Controller mode enable	
SPI_CTL.PSSE	0	Protected target select enable	
SPI_CTL.ODM	0	Open-drain mode enable	
SPI_CTL.CPHASPI_CTL.CPOL	0–0 or 1–1	SPI mode of communication	Flash dependent, usually SPI flash supports mode-0 (CPHA=CPOL=0) and mode-3 (CPHA=CPOL=1)
SPI_CTL.ASSEL	1	Hardware target select pin control	
SPI_CTL.SELST	1	Target select asserted between transfers	
SPI_CTL.EMISO	1	MISO pin enable	
SPI_CTL.SIZE	2	32-bit transfer size	
SPI_CTL.LSBF	0	MSB bit first mode	Flash dependent, usually SPI flash communicates in MSB bit first mode
SPI_CTL.FCEN SPI_CTL.FCCH SPI_CTL.FCPL SPI_CTL.FCWM	0	Hardware flow control related bits	
SPI_CTL.FMODE	1	Fast mode enable	Typically set to 1 for full cycle timing, 0 only works at low speed
SPI_CTL.SOSI	0	Treat SPI_MOSI pin as IO0 pin.	

Table 22-11: SPI Receive Control Register

Bits	Typical values to set	Description
SPI_RXCTL.REN	1	Receive channel enable
SPI_RXCTL.RTI	0	Receive transfer initiation disable
SPI_RXCTL.RWCEN	0	Receive word counter disable

Table 22-11: SPI Receive Control Register (Continued)

Bits	Typical values to set	Description
SPI_RXCTL.RDR	0	Receive data request disable
SPI_RXCTL.RDO	0	Discard incoming data if RFIFO is full
SPI_RXCTL.RRWM	0	Receive FIFO regular watermark
SPI_RXCTL.RUWM	0	Receive FIFO urgent watermark disable

Table 22-12: SPI Transmit Control Register

Bits	Typical values to set	Description
SPI_TXCTL.TEN	1	Transmit channel enable
SPI_TXCTL.TTI	1	Transmit transfer initiation disable
SPI_TXCTL.TWCEN	0	Transmit word counter disable
SPI_TXCTL.TDR	0	Transmit data request disable
SPI_TXCTL.TDU	0	Send last word when TFIFO is empty
SPI_TXCTL.TRWM	0	Transmit FIFO regular watermark
SPI_TXCTL.TUWM	0	Transmit FIFO urgent watermark disable

Table 22-13: SPI DLY Control Register

Bits	Typical Values to Set	Description	Comments See Flash data sheet for CS (for example, SSEL) Timing Specs
SPI_DLY.LAGX	1	Extended lag timing	
SPI_DLY.LEADX	1	Extended lead timing	
SPI_DLY.STOP	3	Stop bit between the transfers	Can be set to 1 at lower SPI clock frequencies.

The multiple I/O mode (SPI_CTL.MIOM) bits are partially ignored:

- The command (opcode) is transmitted using either just one or the number of pins specified by the SPI_CTL.MIOM bits, depending on SPI_MMRDH.CMDPINS bit setting.
- The address is then transmitted using either just one or the number of pins specified by the SPI_CTL.MIOM bits, depending on SPI_MMRDH.ADRPINS bit setting.
- The data is always read with the number of pins specified by the SPI_CTL.MIOM bits.

NOTE: Set the SPI module enable bits SPI_CTL.EN last after configuring all registers.

Use the following programming guidelines for memory-mapped mode:

- The SPI memory-mapped hardware does not check the flash status before initiating the access. It assumes that SPI memory is always able to respond to a read access. Before enabling memory-mapped mode (for example, setting the SPI_CTL.MMSE bit) ensure that SPI flash is ready for a read access. When using non-memory-mapped mode, a write-complete status can be examined prior to enabling the SPI in memory-mapped mode. (See the write in progress bit in the SPI flash memory status register.) Also, immediately after initial power-up, SPI memory devices can be inaccessible for a vendor-specified period.
- When SPI is enabled in memory-mapped mode, attempts to communicate with the SPI device using legacy methods are blocked. Legacy methods include any direct access made to the transmit or receive FIFOs, whether initiated by DMA or by a processor MMR access.
- To use some of the features offered by SPI memory devices, programs can first configure the SPI memory device by setting its control word or sending some commands. Because SPI memory-mapped hardware does not allow any type of SPI write operations, configure the SPI in non-memory-mapped mode prior to enabling memory-mapped mode.
- The memory-mapped hardware does not interpret the opcode. It does not check the validity of the timing that is specified in the SPI_MMRDH register for a particular opcode. Programs must set the fields of the SPI_MMRDH register to be consistent with the read-type selected.
- When the core requests the data or code fetch, the memory-mapped transfer depends on cache settings. The cache configuration register in the SPI memory device must be appropriately configured before enabling memory-mapped mode. Some of the high performance modes like merge, wrap, and transfer size depend on cache parameters.
- SPI memory-mapped MDMA reads do not support wrapping. For MDMA reads, limit the DMA_CFG.MSIZE field to 1 byte, 2 bytes or 4 bytes.
- There is not always tool support to change the SPI memory-mapped hardware setting or cache settings on-the-fly. Changing these settings can optimize the performance of code that accesses SPI memory in memory-mapped mode. It is expected that the SPI memory, SPI peripheral, and cache are programmed to one specific set of control settings for the whole application. Profiling or benchmarking of the actual application can be done to find the setting that works best.

SPI Interrupt Signals

The SPI controller supports three types of interrupt request signals that correspond to data, status, and error conditions.

Data Interrupts

The SPI target supports two data interrupt channels – receive and transmit. These interrupt signals are multiplexed into the DMA request lines. Since the target interfaces with separate read and write interfaces with DMA, the read and write data interrupts are independent. When the DMA channels are not used, the interrupts are routed directly to the system event controller. The interrupts occupy the same vector locations as the corresponding DMA channels.

Each of the data interrupt requests can be individually controlled. Program the SPI_RXCTL.RDR and SPI_TXCTL.TDR bit fields for receive and transmit, respectively. When receive is enabled, the RX interrupt request is issued whenever there is data available in the receive datapath for reading. (The event occurs according to the SPI_RXCTL.RDR bit setting.) When transmit is enabled, the TX interrupt request is issued whenever the transmit datapath can be written. (The event occurs according to the SPI_TXCTL.TDR setting.) DMA data interrupts are compatible with second-generation DMA to incorporate urgent data requests and transfer finish interrupt requests apart from the usual data request interrupts. Transmit interrupt requests operate independently from the word counter-value in the SPI_TWC register.

Status Interrupts

The SPI controller supports several status interrupt requests to indicate different conditions of the receiver and transmitter. All status interrupt requests can be masked. Status interrupt requests are signaled directly through a single SPI status IRQ line. The line cannot be combined with the SPI error IRQ line for some processors. The ***SPI Status Interrupts*** table describes the status interrupt requests that are available for the SPI controller.

Table 22-14: SPI Status Interrupts

SPI_STAT Bit	Description
SPI_STAT.RUWM	Receive FIFO urgent watermark interrupt request. Issued when the level of the RFIFO breaches the watermark set in the SPI_RXCTL.RUWM field. It is cleared when the level of the RFIFO reaches the watermark set in the SPI_RXCTL.RRWM field. If the RX channel is configured in DMA mode, SPI_RXCTL.RUWM is multiplexed with the data request.
SPI_STAT.TUWM	Transmit FIFO urgent watermark interrupt request. Issued when the level of the TFIFO breaches the watermark set using the SPI_TXCTL.TUWM bit. It is cleared when the level of the TFIFO reaches the watermark set in the SPI_TXCTL.TRWM field. If the TX channel is configured in DMA mode, SPI_STAT.TUWM is multiplexed with the data request.
SPI_STAT.TS	Transmit start interrupt request. Issued when the start of a transmit burst is detected by loading of the SPI_TWC register with the contents of the SPI_TWCR register.
SPI_STAT.RS	Receive start interrupt request. Issued when the start of a receive burst is detected by the loading of SPI_RWC with the contents of SPI_RWCR.
SPI_STAT.TF	Transmit finish interrupt request. Issued when a transmit burst completes (SPI_TWC decrements to zero).
SPI_STAT.RF	Receive finish interrupt request. Issued when a receive burst completes (SPI_RWC decrements to zero).

Error Conditions

The SPI controller supports interrupt requests upon several different error conditions. All interrupt requests are maskable. The individual error indications combine into a single SPI error IRQ signal, which can be multiplexed on some processors with the aggregated SPI status IRQ signal. The ***SPI Error Interrupts*** table details the possible error indications.

Error conditions arise depending on which of the channels (transmit or receive) are enabled. If a channel is disabled, all errors related to it are ignored. When both channels are enabled, errors from both channels are enabled.

Table 22-15: SPI Error Interrupts

Bit	Description
SPI_STAT.MF	Mode fault. Signaled when another device also tries to be a controller in a multi-controller system and drives the SPI_SS input low. This error is signaled in controller mode operation.
SPI_STAT.TUR	Transmission error. Signaled when an underflow condition occurs on the transmit channel. This event occurs when a new transfer starts but SPI_TFIFO is empty. This error does not occur in controller transmit initiating mode since SPI_TFIFO *Not Empty* is one of the conditions for transfer initiation.
SPI_STAT.ROR	Reception error. Signaled when an overflow condition occurs on the receive channel. This event occurs when a new data word is received, but the SPI_RFIFO is full. This error condition does not occur in controller receive initiating mode since SPI_RFIFO *Not Full* is one of the conditions for transfer initiation.
SPI_STAT.TC	Transmit collision error. Signaled when loading data to the transmit shift register happens near the first transmitting edge of SPI_CLK. In target mode of operation, the SPI controller is unaware of when the next transfer starts. Loading of data to the transmit shift register can happen just after the transmitting edge. This event results in the setup time not being met for the first bit transmitted. The transmitted data is corrupt. In SPI_CTL.CPHA 1 mode, the first SPI_CLK edge is taken as the first transmitting edge. If SPI_CTL.CPHA =0, then the last SPI_CLK edge of the last transmission (SPI_CTL.SELST =1) or target select deassertion (SPI_CTL.SELST =0) is taken as the first transmitting edge. This error is signaled only in the target mode of operation. In controller mode of operation, loading of data happens before the first transmitting edge of SPI_CLK.

SPI Programming Concepts

The following sections provide general programming guidelines and procedures.

Programming Guidelines

It is acceptable to program SPI_RXCTL and SPI_TXCTL registers after programming the SPI_CTL register. However, program the initiating mode register and its counter-register, if enabled, after the non-initiating mode register. For example, if transmit is the initiating mode and receive is the non-initiating mode, then program the SPI_RXCTL and SPI_RWC registers before the SPI_TXCTL and SPI_TWC registers. If enabling both transmit and receive in initiating mode, enable the SPI_CTL register after programming both the SPI_RXCTL and SPI_TXCTL registers.

These programming guidelines prevent SPI from starting a transfer when SPI registers are not fully programmed. Other ways of programming are also allowed when the initiating conditions prevent the start of communication until after programming of SPI registers is complete.

Avoid data corruption when changing the SPI module configuration. Do not change the configuration during a data transfer. Additionally, change the clock polarity only when no target is selected. However, an exception to this rule exists. When an SPI communication link consists of a single controller and target, SPI_CTL.ASSEL = 0. The target select input of the slave is permanently tied low. In this case, the target is always selected. Avoid data corruption by enabling the target only after both the controller and target devices are configured.

The module supports 8, 16-bit and 32-bit word sizes. To ensure correct operation, configure both the controller and target with the same word size.

Controller Operation in Non-DMA Modes

This section describes the operation of the SPI as a controller in non-DMA mode.

1. Write to the SPI_SLVSEL register, setting one or more of the SPI select enable bits. This operation ensures that the desired targets are properly deselected while the controller is configured.
2. The SPI_RXCTL.RTI and SPI_TXCTL.TTI bits determine the SPI initiating mode. The initiating mode defines the primary transfer channel, and also the initiating condition for the transfer.
3. Write to the SPI_CLK, SPI_CTL, SPI_RXCTL, and SPI_TXCTL registers. This operation enables the device as a controller and configures the SPI system. It specifies the transfer modes and channels, appropriate word length, transfer format, baud rate, and other control information.

   ADDITIONAL INFORMATION: If SPI_RXCTL.RTI is enabled and SPI_TXCTL.TTI is not, write to the SPI_RXCTL register after writing into SPI_CTL, SPI_TXCTL, and SPI_TFIFO registers to prevent a transmit underrun for the first transfer.
4. If SPI_CTL.ASSEL=0, activate the desired targets by clearing one or more of the SPI_SLVSEL flag bits. Otherwise, the SPI hardware performs target activation.
5. The SPI controller then generates the programmed clock pulses on SPI_CLK and simultaneously shifts data out of SPI_MOSI while shifting data in from SPI_MISO. Before a shift, the shift register is loaded with the contents of the SPI_TFIFO register. At the end of the transfer, the contents of the shift register are loaded into SPI_RFIFO.
6. Whenever the initiating conditions are satisfied, the SPI continues to send and receive words. If the transmit buffer remains empty or the receive buffer remains full, the device operates according to the states of the SPI_TXCTL.TDU and SPI_RXCTL.RDO bits.
7. It is possible to program a secondary channel in addition to the initiating channel. This feature allows usage of available channel resources for receives or transmits simultaneously with the initiating channel.

Target Operation in Non-DMA Modes

When a device is enabled as a target in a non-DMA mode, a transition of the $\overline{\text{SPI_SS}}$ select signal to the active state (low) triggers the start of a transfer. Or the first active edge of SPI_CLK triggers the start, depending on the state of SPI_CTL.CPHA bit. The interface operates in the following manner.

1. The core writes to the SPI_CTL, SPI_RXCTL, and SPI_TXCTL registers. The operation defines the mode of the serial link to be the same as the mode setup in the SPI controller.
2. To prepare for the data transfer, the core writes data to be transmitted into SPI_TFIFO.
3. Once the $\overline{\text{SPI_SS}}$ falling edge is detected, the target starts sending data on active SPI_CLK edges and sampling data on inactive SPI_CLK edges.
4. Reception or transmission continues until $\overline{\text{SPI_SS}}$ is released or until the target has received the proper number of clock cycles.

5. The target device continues to receive or transmit with each new falling edge transition on SPI_SS or active SPI_CLK edge. If the transmit buffer remains empty or the receive buffer remains full, the device operates according to the states of the SPI_TXCTL.TDU and SPI_RXCTL.RDO bits.

Configuring DMA Controller Mode

The SPI interface supports a write DMA channel and a read DMA channel. It can use these functions individually or in a lock-step manner in duplex mode (SPI_TXCTL.TTI= SPI_RXCTL.RTI=1).

1. Write to the appropriate DMA registers to enable the SPI DMA channel and to configure the necessary work units, access direction, word count, and so on.
2. Write to the SPI_SLVSEL register, setting one or more of the SPI flag select bits.
3. Write to the SPI_CLK and SPI_CTL registers, enabling the device as a controller and configuring the SPI system by specifying the appropriate word length, transfer format, baud rate, and so forth.
4. Write to SPI_RXCTL to configure SPI controller receive mode, or write to SPI_TXCTL to configure SPI controller transmit mode.
5. Finally, write to the SPI_RXCTL.REN bit to enable the receive channel, or write to SPI_TXCTL.TEN to enable the transmit channel.
6. When the SPI_RXCTL.RTI bit is enabled, a receive transfer is initiated upon enabling SPI_CTL.EN bit. If the receive word counter is enabled (SPI_RXCTL.RWCEN), then the SPI_RWC register must be non-zero for a transfer to initiate.

   ADDITIONAL INFORMATION: If enabling both receive and transmit DMA channels, but not enabling SPI_TXCTL.TTI, write to the SPI_RXCTL register after writing the SPI_CTL and SPI_TXCTL registers. In this way, a transmit underrun can be prevented for the first transfer. Subsequent transfers are initiated as the SPI reads data from the receive shift register and writes to the SPI receive FIFO. The SPI then requests a write from DMA to memory. Upon a DMA grant, the DMA engine reads a word from the SPI receive FIFO and writes to memory. New requests continue to be initiated as long as the receive FIFO does not fill up, when SPI_RWC does not become zero while SPI_RXCTL.RWCEN=1.
7. When SPI_TXCTL.TTI is enabled, the SPI controller requests DMA reads from memory when there is space for more data in the transmit pipe. Upon a DMA grant, the DMA engine reads a word from memory and writes to the transmit FIFO. As long as transmit data is available in the FIFO, and the SPI_TWC register is non-zero when SPI_TXCTL.TWCEN=1, the SPI continues to initiate transfers until disabled.
8. When both the SPI_TXCTL.TTI and SPI_RXCTL.RTI bits are enabled, the SPI controller requests a DMA read from memory. However, there must be space for more data in the transmit pipe and the number of words written into the SPI must be less than SPI_TWC if SPI_TXCTL.TWCEN=1. Upon a DMA grant, the DMA engine reads a word from memory and writes to the transmit FIFO.

   ADDITIONAL INFORMATION: As the SPI writes data from the transmit FIFO into the transmit shift register, it initiates a transfer on the SPI link. Data received from the transfer is moved from the SPI receive shift register to the receive FIFO. The SPI controller requests a write from DMA to memory. Upon a DMA grant,

the DMA engine reads a word from the receive FIFO and writes to memory. Transfer continues to be initiated when both receives and transmits can accommodate new data.

9. When the receive pipe fills up due to unavailability of DMA grants, the transmit pipe stalls until the pipe is drained. If the transmit pipe fills up, the SPI stops requesting for DMA writes. IWhen the value in SPI_RWC expires, further write-requests to DMA stop. However, data already written into the transmit FIFO is sent, and read requests to DMA continue until the receive data is read from the receive FIFO.
10. The SPI then generates the programmed clock pulses on SPI_CLK and simultaneously shifts data out of SPI_MOSI while shifting data in from SPI_MISO. For receive transfers, the value in the shift register is loaded into the SPI_RFIFO register at the end of the transfer. For transmit transfers, the value in the SPI_TFIFO register is loaded into the shift register at the start of the transfer.

Configuring DMA Target Mode Operation

This mode occurs when the SPI is enabled as a target and the DMA engine is configured to transmit or receive data. A transition of the SPI_SS signal to the active-low state triggers the start of a transfer. Or, the first active edge of SPI_CLK triggers the start of a transfer, depending on the state of the SPI_CTL.CPHA bit. The following steps illustrate the SPI receive or transmit DMA sequence in an SPI target (in response to a controller command). The SPI supports a receive DMA channel and a transmit DMA channel.

1. Write to the appropriate DMA registers to enable the SPI DMA channel and configure the necessary work units, access direction, word count, and so on.
2. Write to the SPI_CTL, SPI_RXCTL, and SPI_TXCTL registers to define the mode of the serial link to be the same as the mode configured in the SPI controller.
3. When the receive channel is enabled (SPI_RXCTL.REN is asserted), the following actions occur:
    a. Once the target select input is active, the target starts receiving and transmitting data on active SPI_CLK edges.
    b. The value in the shift register is loaded into the SPI_RFIFO register at the end of the transfer.
    c. Once SPI_RFIFO has valid data, it requests a write from DMA to memory.
    d. Upon a DMA grant, the DMA engine reads a word from the receive FIFO and writes to memory.
    e. When there is data in the receive FIFO, the SPI target continues to request a DMA write to memory. The DMA engine continues to read a word from the FIFO and writes to memory. The SPI target continues receiving words on active SPI_CLK edges when the SPI_SS input is active.
    f. When the data collected in the receive pipe breaches the set level, and the DMA engine cannot keep up with the receive rate, the target can deassert the SPI_RDY signal. This signaling throttles the controller. The receive pipe level is set according to the SPI_CTL.FCWM field. The signal is deasserted as the DMA drains the receive FIFO. Alternatively, the SPI can use the SPI_RXCTL.RDO bit to decide when the incoming data is discarded or overwritten into the receive FIFO (when SPI_CTL.FCEN is inactive).
4. When the transmit channel is enabled (SPI_TXCTL.TEN is asserted), the following actions occur:

a. The SPI requests a DMA read from memory.

b. Upon a DMA grant, the DMA engine reads a word from memory and writes to the transmit FIFO.

c. The SPI then reads DMA data from the transmit FIFO and writes to the transmit shift register, awaiting the start of the next transfer.

d. Once the target select input is active, the target starts receiving and transmitting data on active SPI_CLK edges.

e. When there is room in the transmit FIFO, the SPI target continues to request a DMA read from memory. The DMA engine continues to read a word from memory and write to the transmit FIFO. The SPI target continues transmitting words on active SPI_CLK edges as long as the SPI_SS input is active.

f. When the number of outstanding data entries in the transmit pipe breaches the level set and the DMA cannot keep up with the transmit rate, the target deasserts the SPI_RDY signal. This signaling throttles the controller. The transmit pipe level is set according to the SPI_CTL.FCWM field. The signal is deasserted as the DMA fills the transmit FIFO. Alternately, the SPI_TXCTL.TDU bit decides the state of the transmit data (when SPI_CTL.FCEN is deasserted).

5. When both receive and transmit channels are enabled, the following actions occur after the actions for each channel. Transfers continue when both receive and transmit channels can accommodate new data.

a. When the receive pipe fills up due to the unavailability of DMA grant, the SPI interface stalls the controller by asserting the SPI_RDY pin. This signal is deasserted as the DMA drains the receive FIFO. Alternately, the SPI uses the SPI_RXCTL.RDO bit to decide when the incoming data is discarded or overwritten in the receive FIFO (when SPI_CTL.FCEN is deasserted).

b. When the transmit pipe fills up, the SPI stops requesting DMA writes until the pipe clears.

c. When there is an underflow problem in the transmit pipe, the target stalls the controller by deasserting SPI_RDY while the DMA fills the transmit FIFO. Alternately, the SPI uses the SPI_TXCTL.TDU bit to decide the state of the transmit data (when SPI_CTL.FCEN is deasserted).

ADSP-SC59x SPI Register Descriptions

Serial Peripheral Interface (SPI) contains the following registers.

Table 22-16: ADSP-SC59x SPI Register List

Name	Description
SPI_CLK	Clock Rate Register
SPI_CTL	Control Register
SPI_DLY	Delay Register
SPI_ILAT	Masked Interrupt Condition Register
SPI_ILAT_CLR	Masked Interrupt Clear Register

Table 22-16: ADSP-SC59x SPI Register List (Continued)

Name	Description
SPI_IMSK	Interrupt Mask Register
SPI_IMSK_CLR	Interrupt Mask Clear Register
SPI_IMSK_SET	Interrupt Mask Set Register
SPI_MMRDH	Memory Mapped Read Header
SPI_MMTOP	SPI Memory Top Address
SPI_RFIFO	Receive FIFO Data Register
SPI_RWC	Received Word Count Register
SPI_RWCR	Received Word Count Reload Register
SPI_RXCTL	Receive Control Register
SPI_SLVSEL	Slave Select Register
SPI_STAT	Status Register
SPI_TFIFO	Transmit FIFO Data Register
SPI_TWC	Transmitted Word Count Register
SPI_TWCR	Transmitted Word Count Reload Register
SPI_TXCTL	Transmit Control Register

Clock Rate Register

The SPI_CLK register selects the baud rate for SPI data transfers, relating this rate to the SPI serial clock (SPI clock) and the system clock (CLKO6).

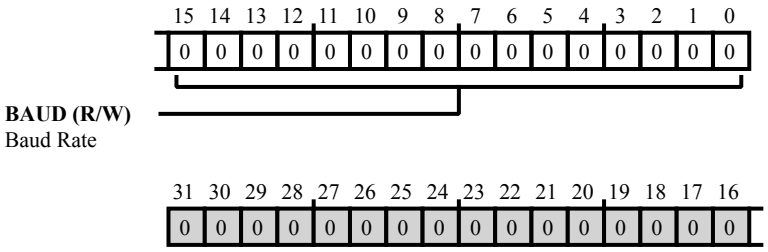


Figure 22-20: SPI_CLK Register Diagram

Table 22-17: SPI_CLK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	BAUD	Baud Rate. The SPI_CLK.BAUD bits set the SPI baud rate according to the formula: BAUD = (CLKO6 / SPI Clock) - 1

Control Register

The SPI_CTL register enables the SPI and configures settings for operating modes, communication protocols, and buffer operations.

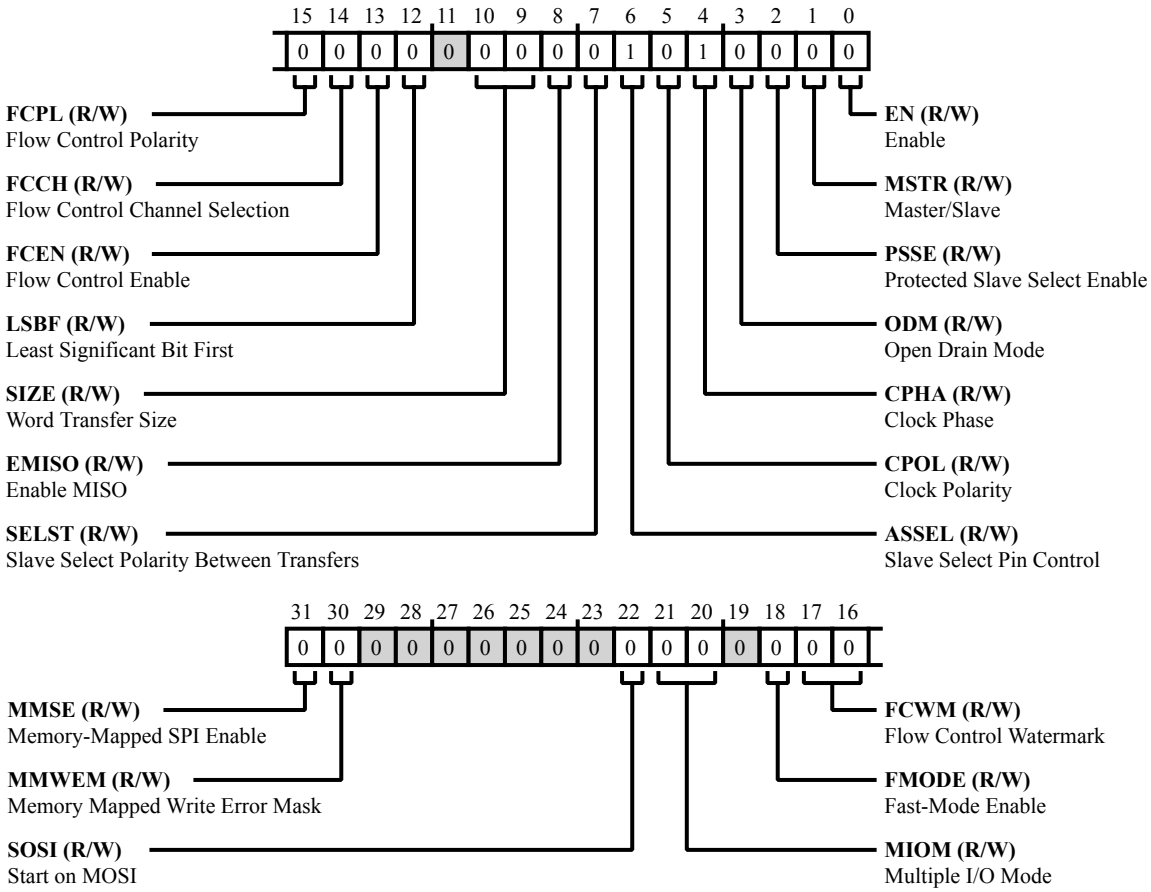


Figure 22-21: SPI_CTL Register Diagram

Table 22-18: SPI_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	MMSE	Memory-Mapped SPI Enable. When the SPI_CTL.MMSE bit is asserted, communication to an SPI memory device is automated such that the memory it contains is accessible directly through the read of processor address space assigned to it. (As far as the SPI peripheral is concerned, this includes all read accesses received by the SPI peripherals system crossbar slave port.) Note that when memory-mapped access of SPI memory is enabled, attempts to communicate with the SPI device using legacy methods are blocked and receive fabric error responses are generated. Legacy methods include any direct access made to the Tx and Rx FIFOs, whether initiated by DMA or processor MMR access.	
		0	Hardware automated access of memory-mapped SPI memory disabled.
		1	Hardware-automated access of memory-mapped SPI memory enabled.
30 (R/W)	MMWEM	Memory Mapped Write Error Mask. The SPI_CTL.MMWEM bit specifies whether an error response is returned to the fabric upon write attempts to address space reserved for memory-mapped reads of SPI memory.	
		0	Write error response returned upon write attempts to memory-mapped SPI memory
		1	Write error response masked (not returned) upon write attempts to memory-mapped SPI memory
22 (R/W)	SOSI	Start on MOSI. The SPI_CTL.SOSI bit is valid only when SPI_CTL.MIOM is enabled for either DIOM or QIOM, and this bit selects the starting pin and the bit placement on pins for these modes. In DIOM, by default, (SPI_CTL.SOSI =0) SPI sends the first bit on the SPI_MISO pin and the second bit on the SPI_MOSI pin. In QIOM, by default, the SPI sends the first bit on the SPI_D3 pin, the second bit on the SPI_D2 pin, the third bit on the SPI_MISO pin and the fourth bit on the SPI_MOSI pin. This order can be reversed by setting the SPI_CTL.SOSI bit. When this bit is set, the SPI sends the first bit on the SPI_MOSI pin. The first bit referred to here depends on the SPI_CTL.LSBF bit setting (MSB bit or LSB bit).	
		0	Start on MISO (DIOM) or start on SPI_D3
		1	Start on MOSI

Table 22-18: SPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:20 (R/W)	MIOM	Multiple I/O Mode. The SPI_CTL.MIOM bits enable SPI operation in dual I/O mode (DIOM) or quad I/O mode (QIOM). These bits can only be changed when the SPI is disabled (SPI_CTL.EN =0).	
		0	No MIOM (disabled)
		1	DIOM operation
		2	QIOM operation
		3	Reserved
18 (R/W)	FMODE	Fast-Mode Enable. The SPI_CTL.FMODE bit enables fast mode operation for SPI receive transfers. SPI transmit operations in fast mode are the same as normal mode.	
		0	Disable
		1	Enable
17:16 (R/W)	FCWM	Flow Control Watermark. The SPI_CTL.FCWM bits select the watermark level of the transmit channel (SPI_TFIFO buffer) or receive channel (SPI_RFIFO buffer) that triggers flow control operation. These bits are applicable only when the SPI is a slave (SPI_CTL.MSTR = 0) and flow control is enabled (SPI_CTL.FCEN =1). When the watermark condition is met, the SPI slave deasserts the SPI_RDY pin.	
		0	TFIFO empty or RFIFO full
		1	TFIFO 75% or more empty, or RFIFO 75% or more full
		2	TFIFO 50% or more empty, or RFIFO 50% or more full
		3	Reserved
15 (R/W)	FCPL	Flow Control Polarity. The SPI_CTL.FCPL bit selects flow control polarity for the SPI_RDY pin when flow control is enabled. When the SPI_RDY pin is active, the SPI is indicating it is ready for data transfer.	
		0	Active-low RDY
		1	Active-high RDY

Table 22-18: SPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	FCCH	Flow Control Channel Selection. The SPI_CTL.FCCH bit selects whether the SPI applies flow control to the transmit channel (SPI_TFIFO buffer) or receive channel (SPI_RFIFO buffer). This bit is applicable only when the SPI is a slave and flow control is enabled.	
		0	Flow control on RX buffer
		1	Flow control on TX buffer
13 (R/W)	FCEN	Flow Control Enable. The SPI_CTL.FCEN bit enables SPI flow control operation, which permits slow slave devices to interface with fast master devices. This bit controls the operation of the SPI_RDY pin. Note that options for flow control operation are available using the SPI_CTL.FCCH, SPI_CTL.FCPL, and SPI_CTL.FCWM bits.	
		0	Disable
		1	Enable
12 (R/W)	LSBF	Least Significant Bit First. The SPI_CTL.LSBF bit selects whether the SPI transmits/receives data as LSB first (little endian) or MSB first (big endian). This bit can only be changed when the SPI is disabled.	
		0	MSB sent/received first (big endian)
		1	LSB sent/received first (little endian)
10:9 (R/W)	SIZE	Word Transfer Size. The SPI_CTL.SIZE bits select the SPI transfer word size as 8, 16 or 32 bits. To ensure correct operation, both the master and slave must be configured with the same word size. This bit can only be changed when the SPI is disabled (SPI_CTL.EN =0).	
		0	8-bit word
		1	16-bit word
		2	32-bit word
		3	Reserved
8 (R/W)	EMISO	Enable MISO. The SPI_CTL.EMISO bit enables master-in-slave-out (MISO) mode. This SPI mode is applicable only when the SPI is a slave.	
		0	Disable
		1	Enable

Table 22-18: SPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	SELST	Slave Select Polarity Between Transfers. The SPI_CTL.SELST bit selects the state (polarity) for the $\overline{\text{SPI_SEL[n]}}$ pin between SPI transfers when the SPI is a master and hardware slave select assertion is enabled (SPI_CTL.ASSEL =1). In slave mode, this bit affects the detection of both transmit collision (SPI_STAT.TC and underrun (SPI_STAT.TUR) errors.	
		0	Deassert slave select (high)
		1	Assert slave select (low)
6 (R/W)	ASSEL	Slave Select Pin Control. The SPI_CTL.ASSEL bit selects whether the SPI hardware sets the $\overline{\text{SPI_SEL[n]}}$ pin output value (ignoring the slave select SPI_SLVSEL.SSEL1 - SPI_SLVSEL.SSEL7 bits) or whether software control of the slave select bits set the $\overline{\text{SPI_SEL[n]}}$ pin output value. This feature is applicable only when the SPI is a master. When hardware control is enabled, the $\overline{\text{SPI_SEL[n]}}$ pin output is asserted during the transfers, and the pin polarity between transfers is selected by the SPI_CTL.SELST bit. When software control is enabled, the $\overline{\text{SPI_SEL[n]}}$ pin output value is set through software control of the slave select bits, and as such, the pin may either remain asserted (low) or be deasserted between transfers.	
		0	Software slave select control
		1	Hardware slave select control
5 (R/W)	CPOL	Clock Polarity. The SPI_CTL.CPOL bit selects whether the SPI uses an active-low or active-high signal for the SPI clock (SPI_CLK). This bit works with the SPI_CTL.CPHA bit to select combinations of clock phase and polarity for the SPI_CLK pin. This bit can only be changed when the SPI is disabled.	
		0	Active-high SPI CLK
		1	Active-low SPI CLK
4 (R/W)	CPHA	Clock Phase. The SPI_CTL.CPHA bit selects whether the SPI starts toggling the signal for the SPI clock (SPI_CLK) from the start of the first data bit or from the middle of the first data bit. The SPI_CTL.CPHA bit works with the SPI_CTL.CPOL bit to select combinations of clock phase and polarity for the SPI_CLK pin. This bit can only be changed when the SPI is disabled.	
		0	SPI CLK toggles from middle
		1	SPI CLK toggles from start

Table 22-18: SPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	ODM	Open Drain Mode. The SPI_CTL.ODM bit configures the data output pins (SPI_MOSI and SPI_MISO) to behave as open drain outputs, which prevents contention and possible damage to pin drivers in multi-master or multi-slave SPI systems. When SPI_CTL.ODM is enabled and the SPI is a master, the SPI three-states the SPI_MOSI pin when the data driven out on MOSI is a logic-high. The SPI does not three-state the SPI_MOSI pin when the driven data is a logic-low. When SPI_CTL.ODM is enabled and the SPI is a slave, the SPI three-states the SPI_MISO pin when the data driven out on SPI_MISO is a logic-high. Note that an external pull-up resistor is required on both the SPI_MOSI and SPI_MISO pins when SPI_CTL.ODM is enabled.	
		0	Disable
		1	Enable
2 (R/W)	PSSE	Protected Slave Select Enable. The SPI_CTL.PSSE bit enables the $\overline{\text{SPI_SS}}$ pin to provide error detection input in a multi-master environment when the SPI is in master mode. If some other device in the system asserts the $\overline{\text{SPI_SS}}$ pin while SPI is enabled as master (and SPI_CTL.PSSE is enabled), this condition causes a mode fault error.	
		0	Disable
		1	Enable
1 (R/W)	MSTR	Master/Slave. The SPI_CTL.MSTR bit toggles the SPI between master mode and slave mode. This bit can only be changed when the SPI is disabled.	
		0	Slave
		1	Master
0 (R/W)	EN	Enable. The SPI_CTL.EN bit enables SPI operation.	
		0	Disable SPI module
		1	Enable

Delay Register

The SPI_DLY register selects a transfer delay and the lead/lag timing between slave select signals and SPI clock edge assertion/deassertion.

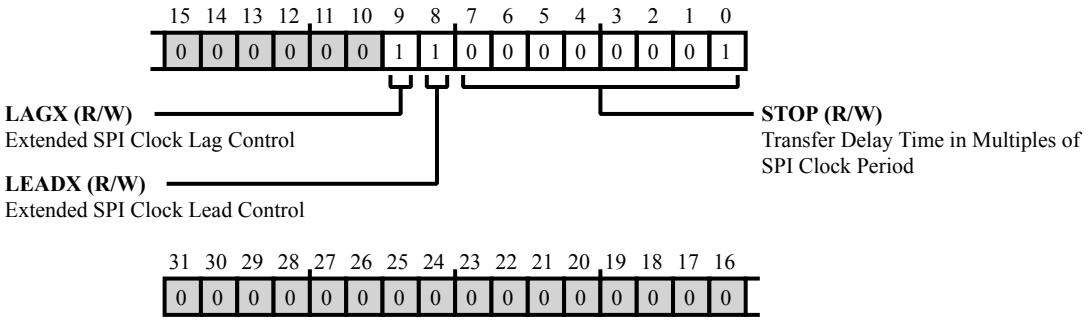


Figure 22-22: SPI_DLY Register Diagram

Table 22-19: SPI_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	LAGX	Extended SPI Clock Lag Control. The SPI_DLY.LAGX bit enables insertion of a 1-SPI_CLK cycle lag (extend lag) in the timing between the slave select (SPI_SEL[n]) assertion and first SPI clock edge.	
		0	Disable
		1	Enable
8 (R/W)	LEADX	Extended SPI Clock Lead Control. The SPI_DLY.LEADX bit enables insertion of a 1-SPI_CLK cycle lead (extend lead) in the timing between the slave select (SPI_SEL[n]) deassertion and last SPI clock edge.	
		0	Disable
		1	Enable
7:0 (R/W)	STOP	Transfer Delay Time in Multiples of SPI Clock Period. The SPI_DLY.STOP bits select a delay (number of stop bits in multiples of SPI clock duration) at the end of each SPI transfer. The default delay is the minimum value required to comply with the SPI protocol (1-bit duration). The SPI_DLY.STOP bits can be programmed with smaller delay values, resulting in continuous operation (for example, stop bits =0).	

Masked Interrupt Condition Register

The SPI_ILAT register latches interrupts, queuing the interrupt requests for service. When a condition is indicated by a bit in the SPI_STAT register and the corresponding interrupt request is unmasked in SPI_IMSK, the SPI latches the interrupt request bit in SPI_ILAT.

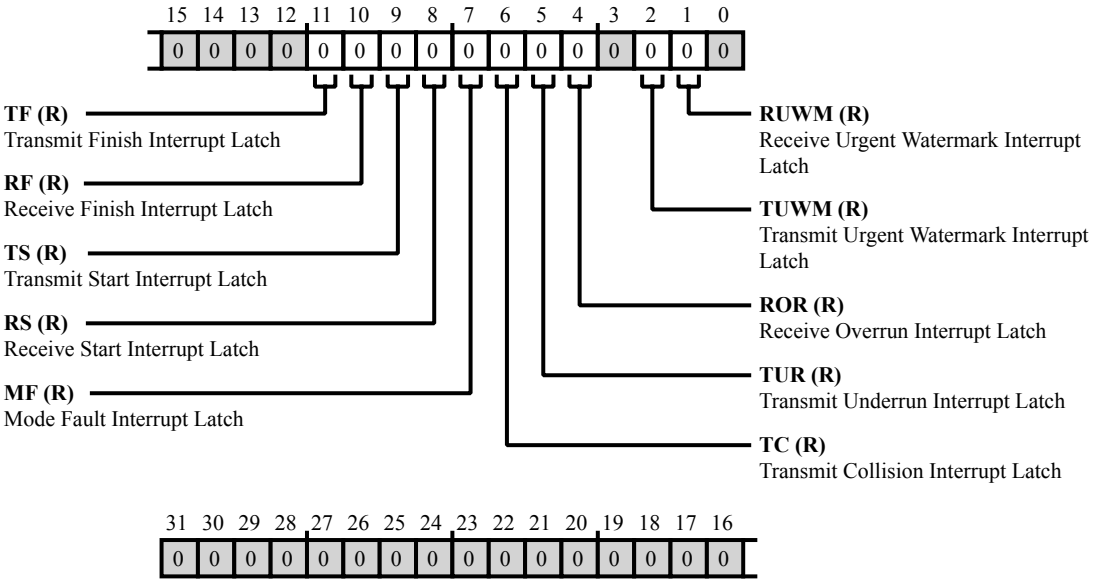


Figure 22-23: SPI_ILAT Register Diagram

Table 22-20: SPI_ILAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	TF	Transmit Finish Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
10 (R/NW)	RF	Receive Finish Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
9 (R/NW)	TS	Transmit Start Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
8 (R/NW)	RS	Receive Start Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request

Table 22-20: SPI_ILAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	MF	Mode Fault Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
6 (R/NW)	TC	Transmit Collision Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
5 (R/NW)	TUR	Transmit Underrun Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
4 (R/NW)	ROR	Receive Overrun Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
2 (R/NW)	TUWM	Transmit Urgent Watermark Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request
1 (R/NW)	RUWM	Receive Urgent Watermark Interrupt Latch.	
		0	No interrupt request
		1	Latched interrupt request

Masked Interrupt Clear Register

The SPI_ILAT_CLR register permits clearing individual mask bits in the SPI_ILAT register without affecting other bits in the register. Use write-1-to-clear on a bit in the SPI_ILAT_CLR register to clear the corresponding bit in the SPI_ILAT register.

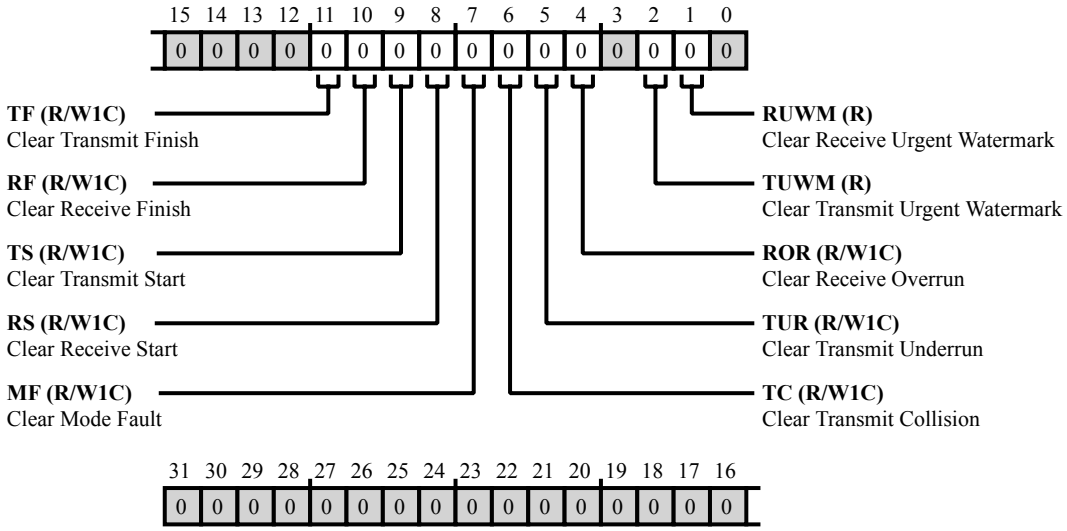


Figure 22-24: SPI_ILAT_CLR Register Diagram

Table 22-21: SPI_ILAT_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W1C)	TF	Clear Transmit Finish. The SPI_ILAT_CLR.TF bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
10 (R/W1C)	RF	Clear Receive Finish. The SPI_ILAT_CLR.RF bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
9 (R/W1C)	TS	Clear Transmit Start. The SPI_ILAT_CLR.TS bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit

Table 22-21: SPI_ILAT_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	RS	Clear Receive Start. The SPI_ILAT_CLR.RS bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
7 (R/W1C)	MF	Clear Mode Fault. The SPI_ILAT_CLR.MF bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
6 (R/W1C)	TC	Clear Transmit Collision. The SPI_ILAT_CLR.TC bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
5 (R/W1C)	TUR	Clear Transmit Underrun. The SPI_ILAT_CLR.TUR bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
4 (R/W1C)	ROR	Clear Receive Overrun. The SPI_ILAT_CLR.ROR bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit
2 (R/NW)	TUWM	Clear Transmit Urgent Watermark. The SPI_ILAT_CLR.TUWM bit clears the corresponding mask bit in the SPI_ILAT register.	
		0	No effect
		1	Clear mask bit

Table 22-21: SPI_ILAT_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	RUWM	Clear Receive Urgent Watermark. The `SPI_ILAT_CLR.RUWM` bit clears the corresponding mask bit in the `SPI_ILAT` register.	
		0	No effect
		1	Clear mask bit

Interrupt Mask Register

The SPI_IMSK register unmasks (enables) or masks (disables) SPI interrupt requests. When a condition is indicated by a bit in the SPI_STAT register and the corresponding interrupt request is unmasked in SPI_IMSK, the SPI latches the interrupt request bit in the SPI_ILAT register, queuing the interrupt request for service.

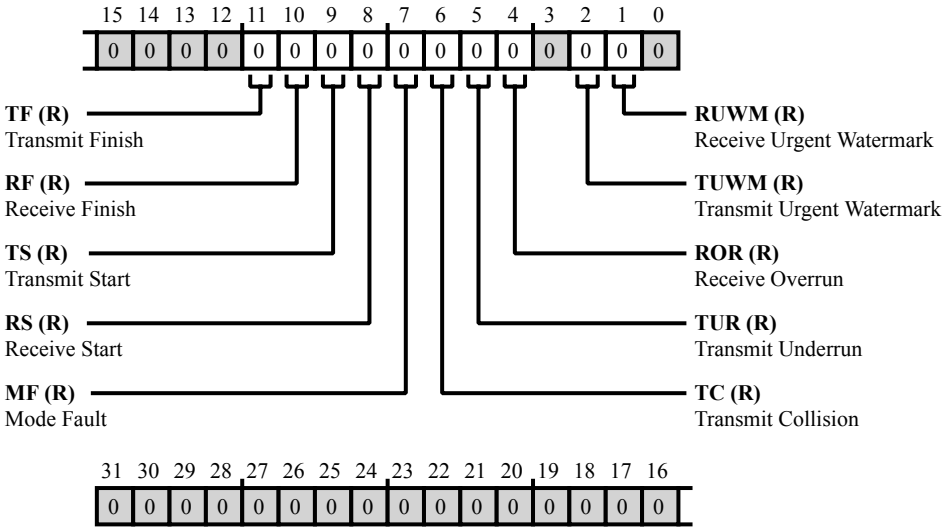


Figure 22-25: SPI_IMSK Register Diagram

Table 22-22: SPI_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	TF	Transmit Finish. The SPI_IMSK.TF bit unmasks (enables) or masks (disables) the TF interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
10 (R/NW)	RF	Receive Finish. The SPI_IMSK.RF bit unmasks (enables) or masks (disables) the RF interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
9 (R/NW)	TS	Transmit Start. The SPI_IMSK.TS bit unmasks (enables) or masks (disables) the TS interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request

Table 22-22: SPI_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	RS	Receive Start. The SPI_IMSK.RS bit unmasks (enables) or masks (disables) the RS interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
7 (R/NW)	MF	Mode Fault. The SPI_IMSK.MF bit unmasks (enables) or masks (disables) the MF interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
6 (R/NW)	TC	Transmit Collision. The SPI_IMSK.TC bit unmasks (enables) or masks (disables) the TC interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
5 (R/NW)	TUR	Transmit Underrun. The SPI_IMSK.TUR bit unmasks (enables) or masks (disables) the TUR interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
4 (R/NW)	ROR	Receive Overrun. The SPI_IMSK.ROR bit unmasks (enables) or masks (disables) the ROR interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
2 (R/NW)	TUWM	Transmit Urgent Watermark. The SPI_IMSK.TUWM bit unmasks (enables) or masks (disables) the TUWM interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request
1 (R/NW)	RUWM	Receive Urgent Watermark. The SPI_IMSK.RUWM bit unmasks (enables) or masks (disables) the RUWM interrupt.	
		0	Disable (mask) interrupt request
		1	Enable (unmask) interrupt request

Interrupt Mask Clear Register

The SPI_IMSK_CLR register permits clearing individual mask bits in the SPI_IMSK register without affecting other bits in the register. Use write-1-to-clear on a bit in the SPI_IMSK_CLR register to clear the corresponding bit in the SPI_IMSK register.

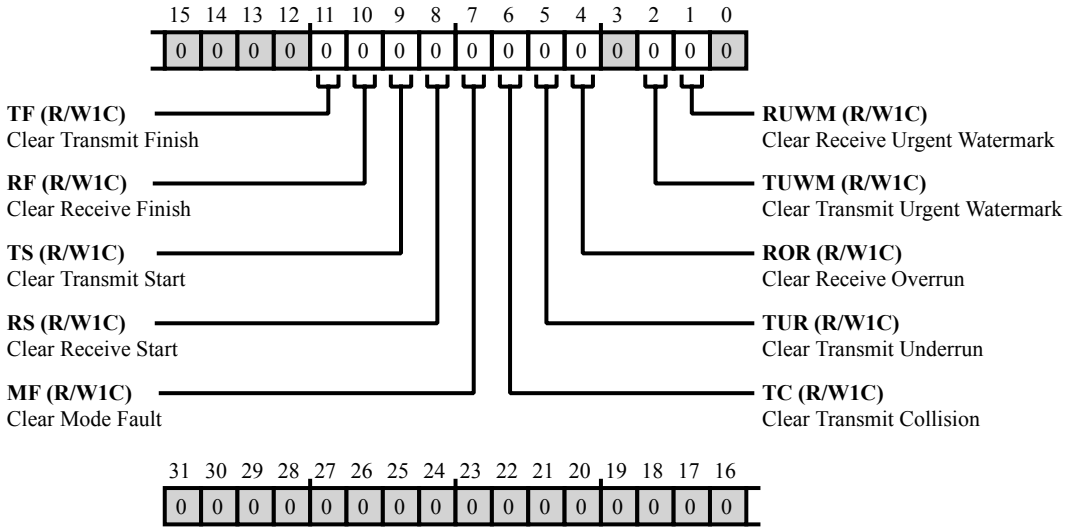


Figure 22-26: SPI_IMSK_CLR Register Diagram

Table 22-23: SPI_IMSK_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W1C)	TF	Clear Transmit Finish. The SPI_IMSK_CLR.TF bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
10 (R/W1C)	RF	Clear Receive Finish. The SPI_IMSK_CLR.RF bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
9 (R/W1C)	TS	Clear Transmit Start. The SPI_IMSK_CLR.TS bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit

Table 22-23: SPI_IMSK_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	RS	Clear Receive Start. The SPI_IMSK_CLR.RS bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
7 (R/W1C)	MF	Clear Mode Fault. The SPI_IMSK_CLR.MF bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
6 (R/W1C)	TC	Clear Transmit Collision. The SPI_IMSK_CLR.TC bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
5 (R/W1C)	TUR	Clear Transmit Underrun. The SPI_IMSK_CLR.TUR bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
4 (R/W1C)	ROR	Clear Receive Overrun. The SPI_IMSK_CLR.ROR bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit
2 (R/W1C)	TUWM	Clear Transmit Urgent Watermark. The SPI_IMSK_CLR.TUWM bit clears the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Clear mask bit

Table 22-23: SPI_IMSK_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	RUWM	Clear Receive Urgent Watermark. The `SPI_IMSK_CLR.RUWM` bit clears the corresponding mask bit in the `SPI_IMSK` register.	
		0	No effect
		1	Clear mask bit

Interrupt Mask Set Register

The SPI_IMSK_SET register permits setting individual mask bits in the SPI_IMSK register without affecting other bits in the register. Use write-1-to-set on a bit in the SPI_IMSK_SET register to set the corresponding bit in the SPI_IMSK register.

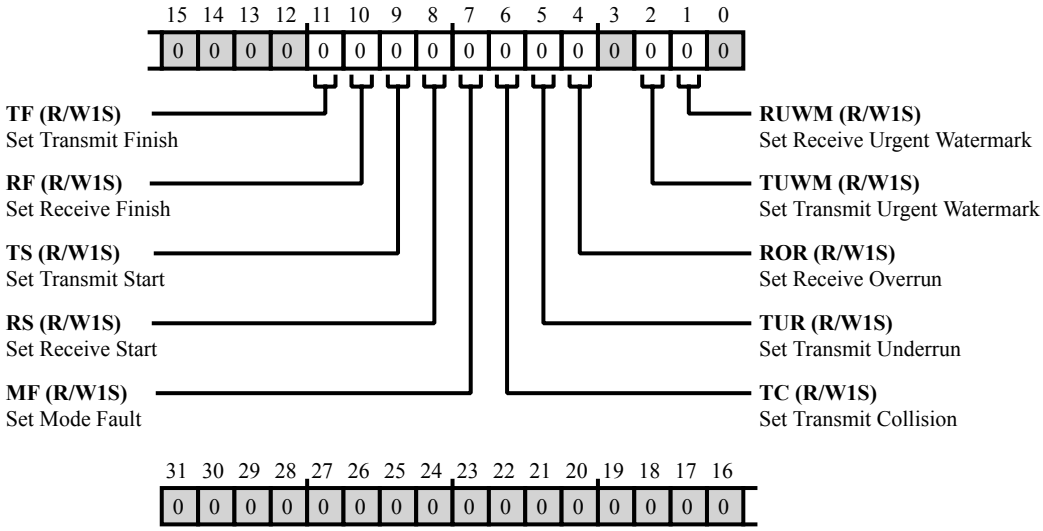


Figure 22-27: SPI_IMSK_SET Register Diagram

Table 22-24: SPI_IMSK_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W1S)	TF	Set Transmit Finish. The SPI_IMSK_SET.TF bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
10 (R/W1S)	RF	Set Receive Finish. The SPI_IMSK_SET.RF bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
9 (R/W1S)	TS	Set Transmit Start. The SPI_IMSK_SET.TS bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit

Table 22-24: SPI_IMSK_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1S)	RS	Set Receive Start. The SPI_IMSK_SET.RS bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
7 (R/W1S)	MF	Set Mode Fault. The SPI_IMSK_SET.MF bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
6 (R/W1S)	TC	Set Transmit Collision. The SPI_IMSK_SET.TC bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
5 (R/W1S)	TUR	Set Transmit Underrun. The SPI_IMSK_SET.TUR bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
4 (R/W1S)	ROR	Set Receive Overrun. The SPI_IMSK_SET.ROR bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit
2 (R/W1S)	TUWM	Set Transmit Urgent Watermark. The SPI_IMSK_SET.TUWM bit sets the corresponding mask bit in the SPI_IMSK register.	
		0	No effect
		1	Set mask bit

Table 22-24: SPI_IMSK_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1S)	RUWM	Set Receive Urgent Watermark. The `SPI_IMSK_SET.RUWM` bit sets the corresponding mask bit in the `SPI_IMSK` register.	
		0	No effect
		1	Set mask bit

Memory Mapped Read Header

The SPI_MMRDH register enables the use of memory-mapped mode. This mode allows direct memory-mapped read accesses of an SPI memory device and is primarily used to directly execute instructions from an SPI FLASH memory without using a low-level software driver. All overhead tasks such as transmission of the read header, pin turnaround timing and receive data sizing are handled in hardware.

The memory-mapped access mode is enabled by setting the SPI_CTL.MMSE bit. The features within the SPI_MMRDH register include a command skip mode, variable length byte addressing, and independent multi-pin support for command transmission, address transmission and data reception. In addition, the command opcode and mode bytes are fully programmable.

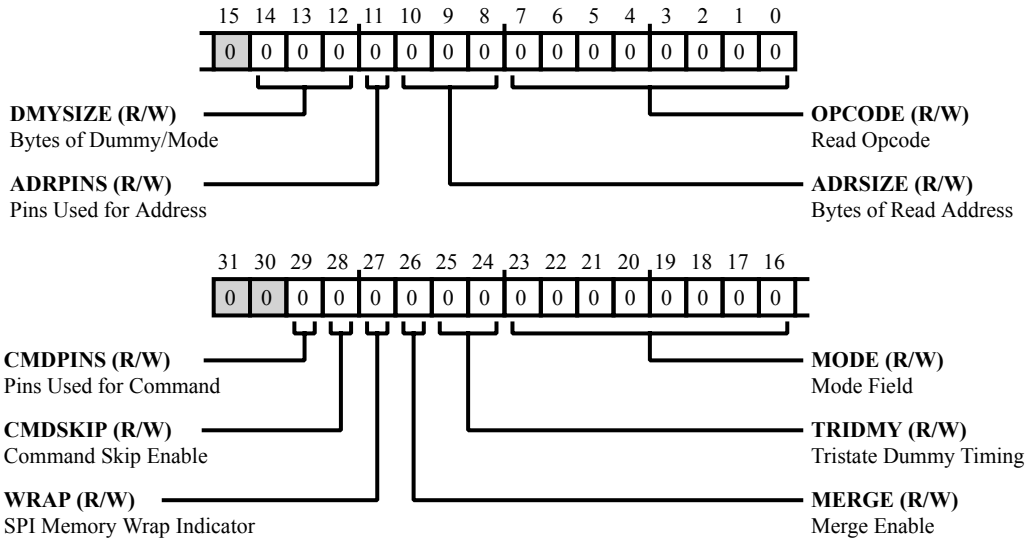


Figure 22-28: SPI_MMRDH Register Diagram

Table 22-25: SPI_MMRDH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	CMDPINS	Pins Used for Command. The SPI_MMRDH.CMDPINS bit specifies the number of pins to be used for command transmission. This bit must be set consistent with the expectations established by the read opcode. Hardware does not interpret SPI_MMRDH.OPCODE, but rather relies on this bit to specify behavior. When cleared, it overrides the SPI_CTL.MIOM bits. When set, it uses bits specified by the SPI_CTL.MIOM bit setting.	
		0	Use only one pin: MOSI (overrides SPI_CTL.MIOM bits)
		1	Use pins specified by SPI_CTL.MIOM bits

Table 22-25: SPI_MMRDH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	CMDSKIP	Command Skip Enable. The SPI_MMRDH.CMDSKIP bit enables command skip mode where the address is sent first and the OPCODE field is not sent (SPI_MMRDH.CMDSKIP bit =1). This mode is useful for supporting XIP (Execute-In-Place) operation where only the address is sent and the same read command is assumed. The SPI flash device must be primed with an initial read command, before the SPI_MMRDH.CMDSKIP bit is set.	
		0	OPCODE field is sent first followed by address
		1	OPCODE field is not sent; address is sent first
27 (R/W)	WRAP	SPI Memory Wrap Indicator. The SPI_MMRDH.WRAP bit must be set by software if software places a connected SPI memory device into a 8-byte, 16-byte or 32-byte wrap mode based on the ILINE and DLINE field setting of the cache configuration register address wrap mode. Software achieves this by transmitting a vendor specified command to the SPI memory device while the SPI_CTL.MMSE bit =0. If the SPI_MMRDH.WRAP bit =1, the SPI does not need to deassert the SPI slave select signal and resend the read header in order to wrap to the cache line base when servicing misaligned cache fill requests. Although this improves cache fill efficiency, it requires that the SPI deassert the SPI slave select pin and resend the read header whenever a DMA burst requests crosses 32 byte alignments. Setting this bit improves cache throughput but decreases DMA throughput.	
		0	SPI Memory auto increments address purely sequentially
		1	SPI Memory auto increments address but wraps within 32 Byte lines
26 (R/W)	MERGE	Merge Enable. When the SPI_MMRDH.MERGE bit is set, SPI hardware combines the two successive transfers. This increases the throughput rate when accessing a large number of sequential memory locations. For more information refer to the Merged Read Accesses section.	

Table 22-25: SPI_MMRDH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25:24 (R/W)	TRIDMY	Tristate Dummy Timing. The SPI_MMRDH.TRIDMY bits specify whether and when output pins are three-stated during the interval of time specified by the SPI_MMRDH.DMYSIZE bits. Output pins potentially three-stated include all pins which were used to transmit the address	
		0	Tristate outputs immediately
		1	Tristate outputs after 4 bits of dummy/mode are transmitted
		2	Tristate outputs after 8 bits of dummy/mode are transmitted
		3	Never tristate outputs (previously specified output state is held)
23:16 (R/W)	MODE	Mode Field. These bits specify up to a leading byte to be transmitted during the interval of time specified by the SPI_MMRDH.DMYSIZE bit field. This first byte, or a portion of it, is interpreted as mode bits when certain opcodes are used in conjunction with certain SPI memory devices. Mode bits are sent using the same number of pins which were used to transmit the address. Once sent, output pins will be held in their final resultant state until the conclusion of all dummy byte periods, unless three-stating the outputs is specified first by the SPI_MMRDH.TRIDMY bits.	
14:12 (R/W)	DMYSIZE	Bytes of Dummy/Mode. The SPI_MMRDH.DMYSIZE bit field specifies the number of bytes separating address transmission and read data return. Dummy bytes elapse assuming dummy bits are transmitted using the same number of pins which were used to transmit address.	
		0	0 Bytes
		1	1 Bytes
		2	2 Bytes
		3	3 Bytes
		4	4 Bytes
		5	5 Bytes
		6	6 Bytes
		7	7 Bytes

Table 22-25: SPI_MMRDH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	ADRPINS	Pins Used for Address. The SPI_MMRDH.ADRPINS bit specifies the number of pins to be used for address transmission. This bit must be set consistent with expectations established by read opcode. Hardware does not interpret the SPI_MMRDH.OPCODE, but rather relies on this bit to specify behavior.	
		0	Use only one pin: MOSI (overrides SPI_CTL.MIOM bits)
		1	Use pins specified by SPI_CTL.MIOM bits
10:8 (R/W)	ADRSIZE	Bytes of Read Address. The SPI_MMRDH.ADRSIZE bit field defines the number of bytes used to specify the read address. The read address is sent immediately following the transmission of opcode. Unlike opcode bits, address bits may be sent using either one or multiple pins. The number of pins is selected using the SPI_MMRDH.ADRPINS bit. The address sent to a connected SPI memory device is an echo of the read address received by the SPI peripheral slave port. The least significant bytes of address are sent when the entire address is not sent.	
		0	1 Byte
		1	1 Byte
		2	2 Bytes
		3	3 Bytes
		4	4 Bytes
7:0 (R/W)	OPCODE	Read Opcode. The SPI_MMRDH.OPCODE bit field specifies the initial bits transmitted in response to a read request of SPI memory. Although any opcode may be sent, values 0x03, 0x0B, 0x3B, 0x6B, 0xBB, and 0xEB are likely to be the most commonly used. SPI_MMRDH.OPCODE is sent by the SPI without interpretation; the states of these bits have no effect beyond specifying what is initially shifted across the SPI interface.	

SPI Memory Top Address

The SPI_MMTOP register specifies the top populated address of a connected SPI memory device.

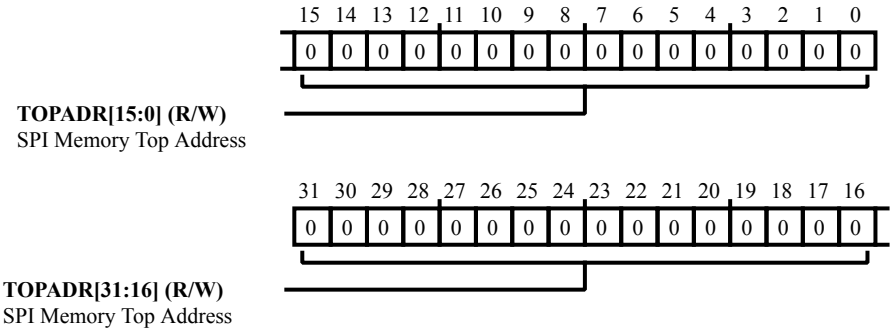


Figure 22-29: SPI_MMTOP Register Diagram

Table 22-26: SPI_MMTOP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TOPADR	SPI Memory Top Address. The SPI_MMTOP.TOPADR bit field specifies the top populated address of a connected SPI memory device. Attempts to access SPI memory are not blocked if this address is exceeded and an error is generated as part of the read response.

Receive FIFO Data Register

The SPI_RFIFO register has an interface to the receive shift register in the SPI and has an interface to the processor's data buses. The top level of the buffer is visible to programs as the 32-bit SPI_RFIFO register, but the size (number of word locations) of the receive FIFO is actually flexible with transfer word size. The size of the receive FIFO is 8 if the word size is 8-bit, or the size is 4 if the word size is 16-bit, or the size is 2 if the word size is 32-bit.

Both masters and slaves may stop or stall receive transfers based on FIFO status. When the receive FIFO is full, the SPI master stops initiating new transfers on the SPI if SPI_RXCTL.RTI is enabled. A slave may stall the SPI interface when the content of the FIFO crosses the selected watermark. If data reception continues after SPI_RFIFO is full, the data in the receive FIFO is invalid. The SPI indicates this condition with receive overrun (SPI_STAT.ROR) error. This condition is possible when SPI_RXCTL.RTI =0 and SPI_RXCTL.REN =1 for a master, or for a slave that does not exercise flow control.

Note that the receive FIFO is reset (cleared) when the SPI is disabled after being enabled.

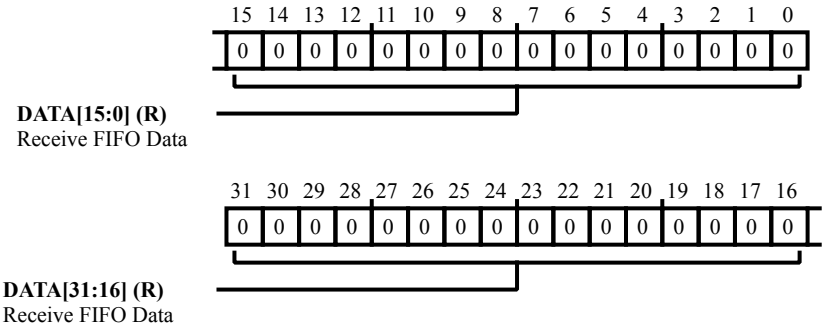


Figure 22-30: SPI_RFIFO Register Diagram

Table 22-27: SPI_RFIFO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Receive FIFO Data. The SPI_RFIFO.DATA bit field contains the FIFO receive data.

Received Word Count Register

The SPI_RWC register holds a count of the number of words remaining to be received by the SPI. To start the decrement of the word count in SPI_RWC, enable the receive word counter (SPI_RXCTL.RWCEN =1). The SPI uses the word count to control the duration of transfers and to signal the completion of a burst of transfers with the receive finish interrupt (SPI_ILAT.RF). In DMA mode, the SPI uses the SPI_RWC register to ensure that the number of frames received during a DMA transfer is equal to the number of words programmed in the DMA channel controller. The values programmed into the SPI_RWC registers should match the word count in the DMA configuration. The SPI_RWC register maintains the number of frames to be received in a transfer. The SPI_RWC should only be changed when the counter is disabled.

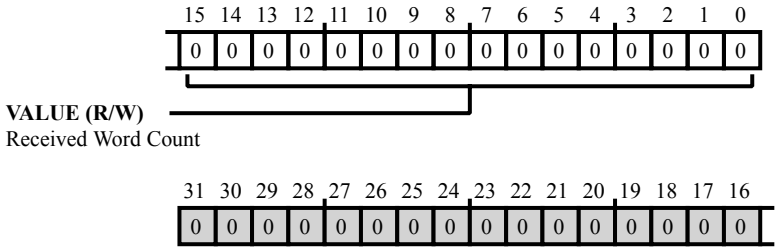


Figure 22-31: SPI_RWC Register Diagram

Table 22-28: SPI_RWC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Received Word Count. The SPI_RWC.VALUE bits hold the receive transfer word count.

Received Word Count Reload Register

The SPI_RWCR register holds the receive word count value that the SPI loads into the SPI_RWC register when the transfer count decrements to zero. To prevent the SPI from reloading the counter, use zero for the reload count value. The SPI_RWCR register should only be changed when the counter is disabled.

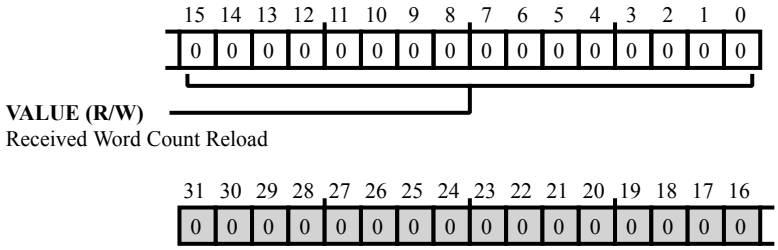


Figure 22-32: SPI_RWCR Register Diagram

Table 22-29: SPI_RWCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Received Word Count Reload. The SPI_RWCR.VALUE bits hold the receive transfer word count reload value.

Receive Control Register

The SPI_RXCTL register enables the SPI receive channel, initiates receive transfers, and configures SPI_RFIFO buffer watermark settings.

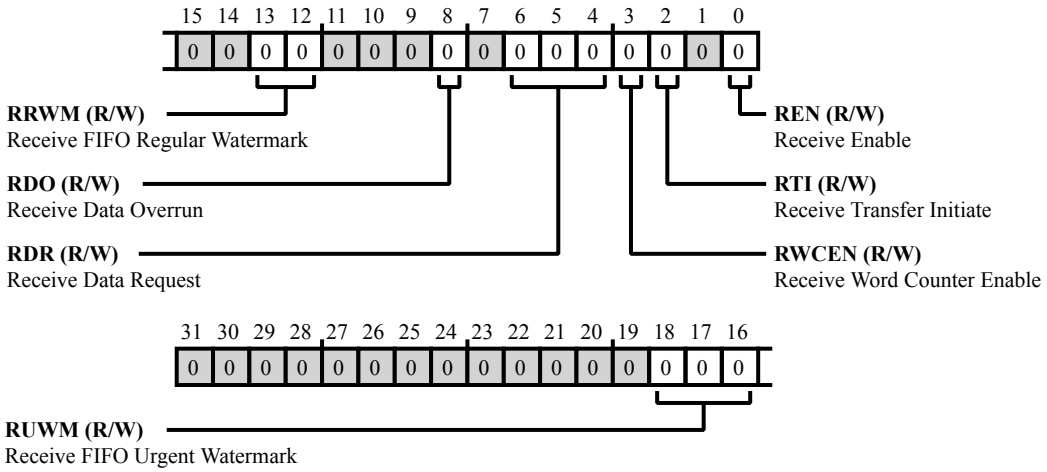


Figure 22-33: SPI_RXCTL Register Diagram

Table 22-30: SPI_RXCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18:16 (R/W)	RUWM	Receive FIFO Urgent Watermark. The SPI_RXCTL.RUWM bits select the receive FIFO (SPI_RFIFO) watermark level for urgent data bus requests. The SPI also uses this watermark level for generation of the SPI_ILAT.RUWM interrupt. When an urgent SPI_RFIFO watermark is enabled with SPI_RXCTL.RUWM, the SPI_RXCTL.RRWM selection is used as the deassertion condition for any SPI_ILAT.RUWM interrupts that are latched.	
		0	Disabled
		1	25% full RFIFO
		2	50% full RFIFO
		3	75% full RFIFO
		4	Full RFIFO
		5	Reserved
		6	Reserved
		7	Reserved

Table 22-30: SPI_RXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13:12 (R/W)	RRWM	Receive FIFO Regular Watermark. The SPI_RXCTL.RRWM bits select the receive FIFO (SPI_RFIFO) watermark level for regular data bus requests. When an urgent SPI_RFIFO watermark is enabled with SPI_RXCTL.RUWM, the SPI_RXCTL.RRWM selection is used as the deassertion condition for any SPI_ILAT.RUWM interrupts that are latched.	
		0	Empty RFIFO
		1	RFIFO less than 25% full
		2	RFIFO less than 50% full
		3	RFIFO less than 75% full
8 (R/W)	RDO	Receive Data Overrun. The SPI_RXCTL.RDO bit selects handling for receive data requests when the receive buffer (SPI_RFIFO) is full. If enabled and SPI_RFIFO is full, the SPI overwrites old data in the buffer with incoming data. If disabled and SPI_RFIFO is full, the SPI keeps old data in the buffer and discards incoming data.	
		0	Discard incoming data if SPI_RFIFO is full
		1	Overwrite old data if SPI_RFIFO is full
6:4 (R/W)	RDR	Receive Data Request. The SPI_RXCTL.RDR bits select receive FIFO (SPI_RFIFO) watermark conditions that direct the SPI to generate a receive data request.	
		0	Disabled
		1	Not empty RFIFO
		2	25% full RFIFO
		3	50% full RFIFO
		4	75% full RFIFO
		5	Full RFIFO
		6	Reserved
		7	Reserved
3 (R/W)	RWCEN	Receive Word Counter Enable. The SPI_RXCTL.RWCEN bit enables the decrement of the SPI_RWC register when the count is not zero and SPI_RXCTL.RTI is enabled. Enabling SPI_RXCTL.RWCEN prevents receive overrun errors from occurring. The SPI_RXCTL.RWCEN bit is valid only when the SPI is a master.	
		0	Disable
		1	Enable

Table 22-30: SPI_RXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	RTI	Receive Transfer Initiate. The SPI_RXCTL.RTI bit enables initiation of receive transfers if the receive FIFO (SPI_RFIFO) is not full. The bit also enables this initiation if SPI_RWC is not zero when SPI_RXCTL.RWCEN is enabled. Enabling SPI_RXCTL.RTI prevents receive overrun errors from occurring. The SPI_RXCTL.RTI bit is valid only when the SPI is a master.	
		0	Disable
		1	Enable
0 (R/W)	REN	Receive Enable. The SPI_RXCTL.REN bit enables SPI receive channel operation.	
		0	Disable
		1	Enable

Slave Select Register

The SPI_SLVSEL register enables the SPI_SEL[n] pins for output and indicates the state (high or low) of these pins when enabled.

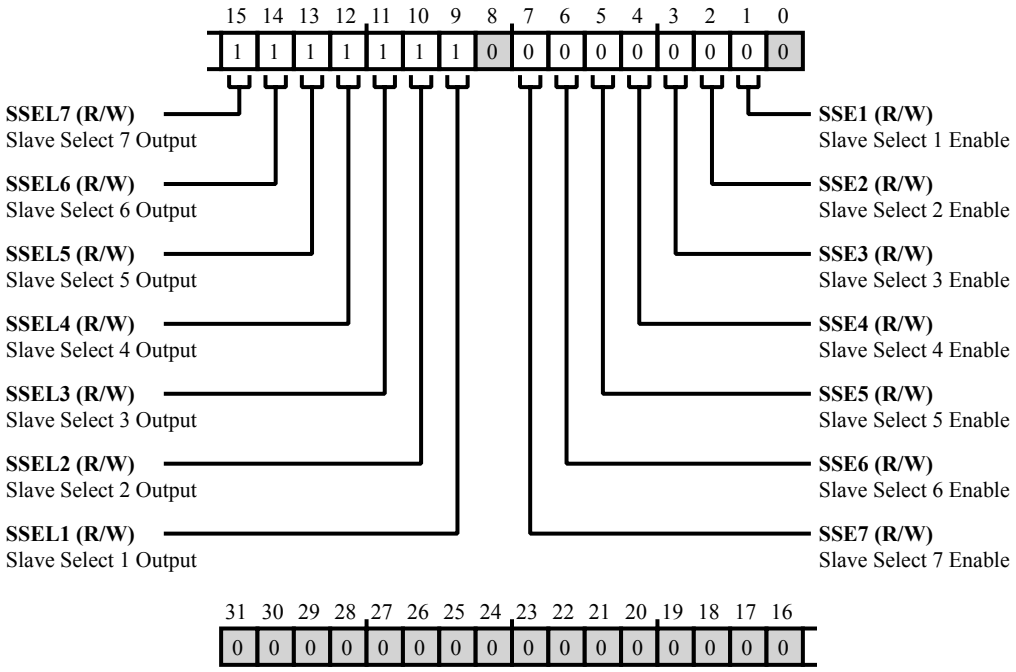


Figure 22-34: SPI_SLVSEL Register Diagram

Table 22-31: SPI_SLVSEL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	SSEL7	Slave Select 7 Output. The SPI_SLVSEL.SSEL7 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
14 (R/W)	SSEL6	Slave Select 6 Output. The SPI_SLVSEL.SSEL6 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High

Table 22-31: SPI_SLVSEL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	SSEL5	Slave Select 5 Output. The SPI_SLVSEL.SSEL5 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
12 (R/W)	SSEL4	Slave Select 4 Output. The SPI_SLVSEL.SSEL4 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
11 (R/W)	SSEL3	Slave Select 3 Output. The SPI_SLVSEL.SSEL3 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
10 (R/W)	SSEL2	Slave Select 2 Output. The SPI_SLVSEL.SSEL2 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
9 (R/W)	SSEL1	Slave Select 1 Output. The SPI_SLVSEL.SSEL1 bit state indicates the value driven on the related SPI_SEL[n] pin.	
		0	Low
		1	High
7 (R/W)	SSE7	Slave Select 7 Enable. The SPI_SLVSEL.SSE7 bit enables the related SPI_SEL[n] pin for output. If disabled, the SPI three-states the related SPI_SEL[n] pin.	
		0	Disable
		1	Enable

Table 22-31: SPI_SLVSEL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	SSE6	Slave Select 6 Enable. The SPI_SLVSEL.SSE6 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_SLVSEL.SSE7 bit description for more information.	
		0	Disable
		1	Enable
5 (R/W)	SSE5	Slave Select 5 Enable. The SPI_SLVSEL.SSE5 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_SLVSEL.SSE7 bit description for more information.	
		0	Disable
		1	Enable
4 (R/W)	SSE4	Slave Select 4 Enable. The SPI_SLVSEL.SSE4 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_SLVSEL.SSE7 bit description for more information.	
		0	Disable
		1	Enable
3 (R/W)	SSE3	Slave Select 3 Enable. The SPI_SLVSEL.SSE3 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_SLVSEL.SSE7 bit description for more information.	
		0	Disable
		1	Enable
2 (R/W)	SSE2	Slave Select 2 Enable. The SPI_SLVSEL.SSE2 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_SLVSEL.SSE7 bit description for more information.	
		0	Disable
		1	Enable
1 (R/W)	SSE1	Slave Select 1 Enable. The SPI_SLVSEL.SSE1 bit enables the related $\overline{\texttt{SPI_SEL[n]}}$ pin for output. See the SPI_SLVSEL.SSE7 bit description for more information.	
		0	Disable
		1	Enable

Status Register

The SPI_STAT register indicates SPI status including FIFO status, error conditions, and interrupt conditions. When an interrupt condition from this register is unmasked (enabled) by the corresponding bit in the SPI_IMSK register, the interrupt request is latched into the corresponding bit in the SPI_ILAT register.

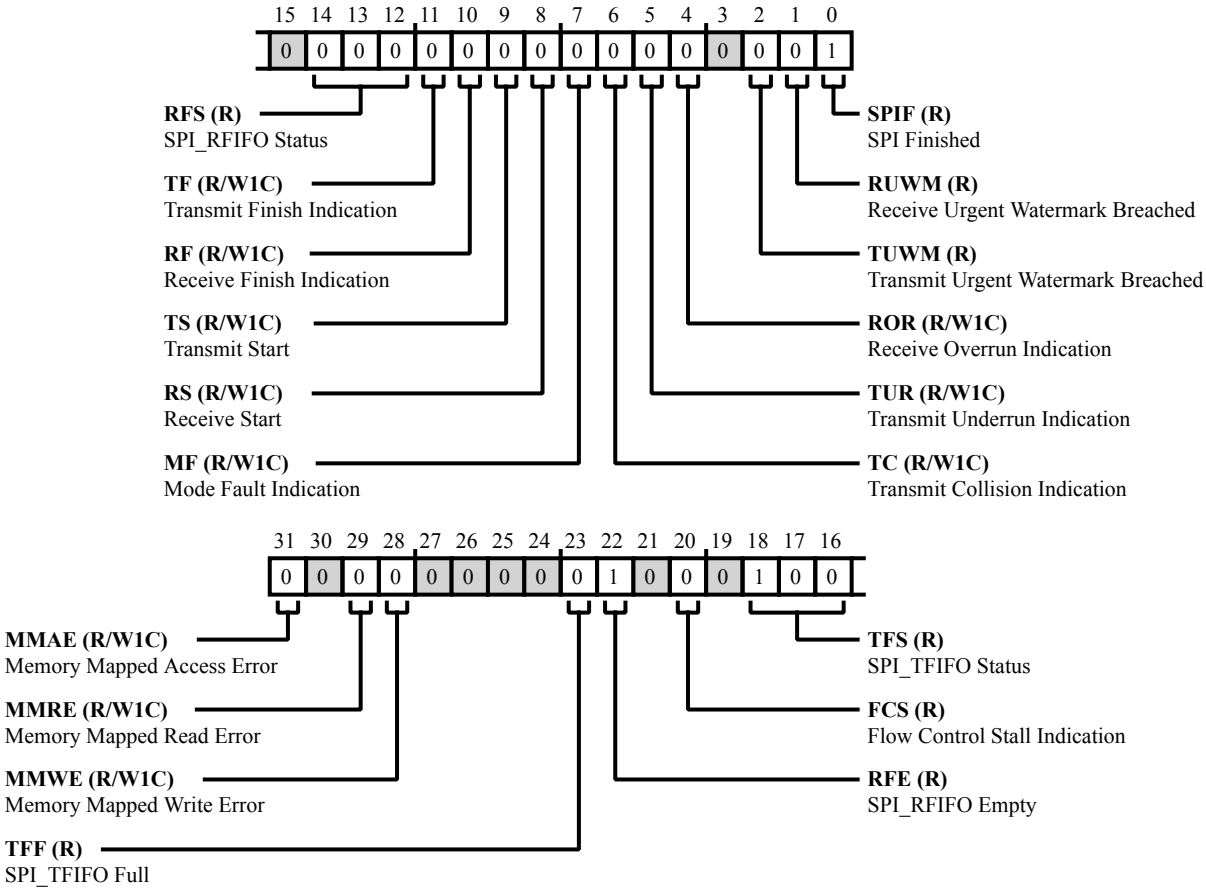


Figure 22-35: SPI_STAT Register Diagram

Table 22-32: SPI_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W1C)	MMAE	Memory Mapped Access Error. The SPI_STAT.MMAE bit =1 if an attempt is made to access either the Tx or Rx FIFO while memory-mapped access of SPI memory is enabled (see the SPI_CTL.MMSE bit). The SPI_STAT.MMAE bit =0 when a 1 is written to it. The SPI_STAT.MMAE bit is provided for software notification only. Its state has no further effect.

Table 22-32: SPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W1C)	MMRE	Memory Mapped Read Error. The SPI_STAT.MMRE bit =1 if an attempt is made to read address space reserved for memory-mapped SPI memory while memory mapping is disabled (see the SPI_CTL.MMSE bit). The SPI_STAT.MMRE bit =0 when a 1 is written to it. This bit is provided for software notification only. Its state has no further effect.	
28 (R/W1C)	MMWE	Memory Mapped Write Error. The SPI_STAT.MMWE bit =1 if an attempt is made to write address space reserved for memory-mapped SPI memory. The SPI_STAT.MMWE bit =0 when a 1 is written to it. This bit is provided for software notification only. Its state has no further effect.	
23 (R/NW)	TFF	SPI_TFIFO Full. The SPI_STAT.TFF bit indicates whether the SPI_TFIFO is full or not full.	
		0	Not full Tx FIFO
		1	Full Tx FIFO
22 (R/NW)	RFE	SPI_RFIFO Empty. The SPI_STAT.RFE bit indicates whether the SPI_RFIFO is empty or not empty.	
		0	Rx FIFO not empty
		1	Rx FIFO empty
20 (R/NW)	FCS	Flow Control Stall Indication. The SPI_STAT.FCS bit indicates whether a slave has deasserted the SPI_RDY pin to stall the SPI master while the slave is unable to service the SPI masters request. This bit is valid only when the SPI is a master (SPI_CTL.MSTR =1) and flow control is enabled (SPI_CTL.FCEN =1).	
		0	No Stall (RDY pin asserted)
		1	Stall (RDY pin deasserted)
18:16 (R/NW)	TFS	SPI_TFIFO Status. The SPI_STAT.TFS bits indicate the status of the SPI_TFIFO. The SPI uses this status when evaluating transmit watermark conditions.	
		0	Full TFIFO
		1	25% empty TFIFO
		2	50% empty TFIFO
		3	75% empty TFIFO
		4	Empty TFIFO

Table 22-32: SPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14:12 (R/NW)	RFS	SPI_RFIFO Status. The SPI_STAT.RFS bits indicate the status of the SPI_RFIFO. The SPI uses this status when evaluating receive watermark conditions.	
		0	Empty RFIFO
		1	25% full RFIFO
		2	50% full RFIFO
		3	75% full RFIFO
		4	Full RFIFO
11 (R/W1C)	TF	Transmit Finish Indication. The SPI_STAT.TF bit indicates that the SPI has detected the finish of a transmit burst transfer (the SPI_TWC count decrements to zero). This condition can only occur when SPI_TXCTL.TTI and SPI_TXCTL.TWCEN are enabled.	
		0	No status
		1	Transmit finish detected
10 (R/W1C)	RF	Receive Finish Indication. The SPI_STAT.RF bit indicates that the SPI has detected the finish of a receive burst transfer (the SPI_RWC count decrements to zero). This condition can only occur when SPI_RXCTL.RTI and SPI_RXCTL.RWCEN are enabled.	
		0	No status
		1	Receive finish detected
9 (R/W1C)	TS	Transmit Start. The SPI_STAT.TS bit indicates that the SPI has detected the start of a transmit burst transfer. A transmit bursts starts with the load of SPI_TWC from the SPI_TWCR. This condition can only occur when SPI_TXCTL.TTI and SPI_TXCTL.TWCEN are enabled.	
		0	No status
		1	Transmit start detected
8 (R/W1C)	RS	Receive Start. The SPI_STAT.RS bit indicates that the SPI has detected the start of a receive burst transfer. A receive bursts starts with the load of SPI_RWC from the SPI_RWCR. This condition can only occur when SPI_RXCTL.RTI and SPI_RXCTL.RWCEN are enabled.	
		0	No status
		1	Receive start detected

Table 22-32: SPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	MF	Mode Fault Indication. The SPI_STAT.MF bit, when SPI is a master and SPI_CTL.PSSE is enabled, indicates that multiple masters have asserted slave select inputs.	
		0	No status
		1	Mode fault occurred
6 (R/W1C)	TC	Transmit Collision Indication. The SPI_STAT.TC bit, when SPI is a slave, indicates that the load of data into the shift register has occurred too close to the first transmitting edge of the SPI clock.	
		0	No status
		1	Transmit collision occurred
5 (R/W1C)	TUR	Transmit Underrun Indication. The SPI_STAT.TUR bit, when the transmit FIFO (SPI_TFIFO) is empty, indicates that the last word in the transmit FIFO has been re-sent as transmit data. Alternately, it indicates that zero has been sent as transmit data.	
		0	No status
		1	Transmit underrun occurred
4 (R/W1C)	ROR	Receive Overrun Indication. The SPI_STAT.ROR bit, when the receive FIFO (SPI_RFIFO) is full, indicates that a word in the receive FIFO has been overwritten with incoming receive data. Alternately, it indicates that incoming receive data has been discarded.	
		0	No status
		1	Receive overrun occurred
2 (R/NW)	TUWM	Transmit Urgent Watermark Breached. The SPI_STAT.TUWM bit indicates that the transmit urgent watermark (SPI_TXCTL.TUWM) has been reached. This condition is cleared when the transmit FIFO fills enough to reach the transmit regular watermark (SPI_TXCTL.TRWM).	
		0	Tx regular watermark reached
		1	Tx urgent watermark breached
1 (R/NW)	RUWM	Receive Urgent Watermark Breached. The SPI_STAT.RUWM bit indicates that the receive urgent watermark (SPI_RXCTL.RUWM) has been reached. This condition is cleared when the receive FIFO empties enough to reach the receive regular watermark (SPI_RXCTL.RRWM).	
		0	Rx regular watermark reached
		1	Rx urgent watermark breached

Table 22-32: SPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	SPIF	SPI Finished. The `SPI_STAT.SPIF` bit indicates that a single word transfer is complete.	
		0	No status
		1	Completed single word transfer

Transmit FIFO Data Register

The SPI_TFIFO register has an interface to the transmit shift register in the SPI and has an interface to the processor's data buses. The top level of the buffer is visible to programs as the 32-bit SPI_TFIFO register, but the size (number of word locations) of the transmit FIFO is actually flexible with transfer word size. The size of the transmit FIFO is 8 if the word size is 8-bit, or the size is 4 if the word size is 16-bit, or the size is 2 if the word size is 32-bit.

Both masters and slaves may stop or stall transmit transfers based on FIFO status. When the transmit FIFO is empty, the SPI master stops initiating new transfers on the SPI if SPI_TXCTL.TTI is enabled. A slave may stall the SPI interface when the content of the FIFO crosses the selected watermark. If data transmit requests continue after SPI_TFIFO is empty, the data sent from the transmit FIFO is invalid, and the SPI indicates this condition with transmit underrun (SPI_STAT.TUR). This condition is possible when SPI_TXCTL.TTI =0 and SPI_TXCTL.TEN =1 for a master, or for a slave that does not exercise flow control.

Note that the transmit FIFO is reset (cleared) when the SPI is disabled after being enabled.

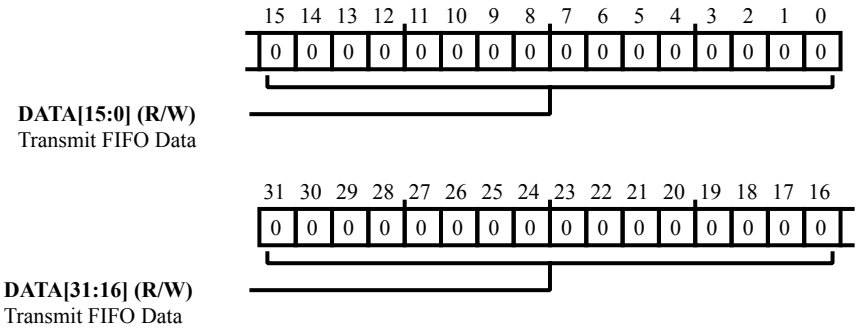


Figure 22-36: SPI_TFIFO Register Diagram

Table 22-33: SPI_TFIFO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	Transmit FIFO Data. The SPI_TFIFO.DATA bit field contains the FIFO transmit data.

Transmitted Word Count Register

The SPI_TWC register holds a count of the number of words remaining to be transmitted by the SPI. To start the decrement of the word count in SPI_TWC, enable the transmit word counter (SPI_TXCTL.TWCEN =1). The SPI uses the word count to control the duration of transfers and to signal the completion of a burst of transfers with the transmit finish interrupt request. In DMA mode, the SPI uses the SPI_TWC to ensure that the number of frames transmitted during a DMA transfer is equal to the number of words programmed in the DMA channel controller. The values programmed into the SPI_TWC registers should match the word count in the DMA configuration. The SPI_TWC maintains the number of frames to be transmitted in a transfer. The SPI_TWC should only be changed when the counter is disabled.

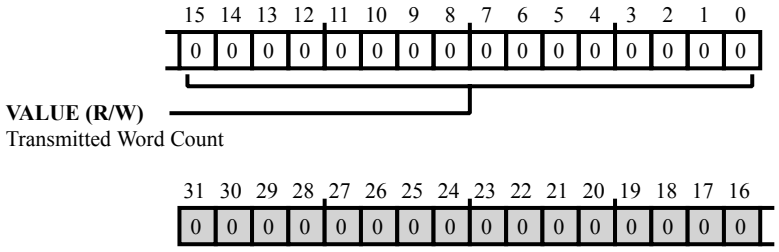


Figure 22-37: SPI_TWC Register Diagram

Table 22-34: SPI_TWC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Transmitted Word Count. The SPI_TWC.VALUE bits hold the transmit transfer word count.

Transmitted Word Count Reload Register

The SPI_TWCR register holds the transmit word count value that the SPI loads into the SPI_TWC register when the transfer count decrements to zero. To prevent the SPI from reloading the counter, use zero for the reload count value. The SPI_TWCR should only be changed when the counter is disabled.

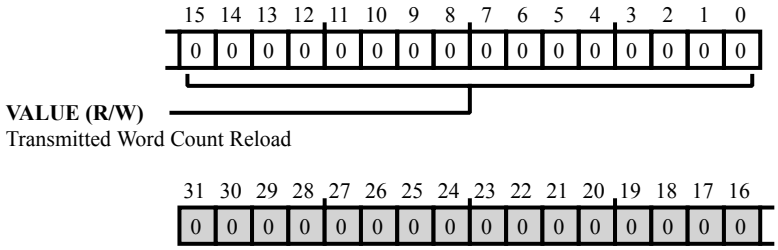


Figure 22-38: SPI_TWCR Register Diagram

Table 22-35: SPI_TWCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Transmitted Word Count Reload. The SPI_TWCR.VALUE bits hold the transmit transfer word count reload value.

Transmit Control Register

The SPI_TXCTL register enables the SPI transmit channel, initiates transmit transfers, and configures SPI_TFIFO buffer watermark settings.

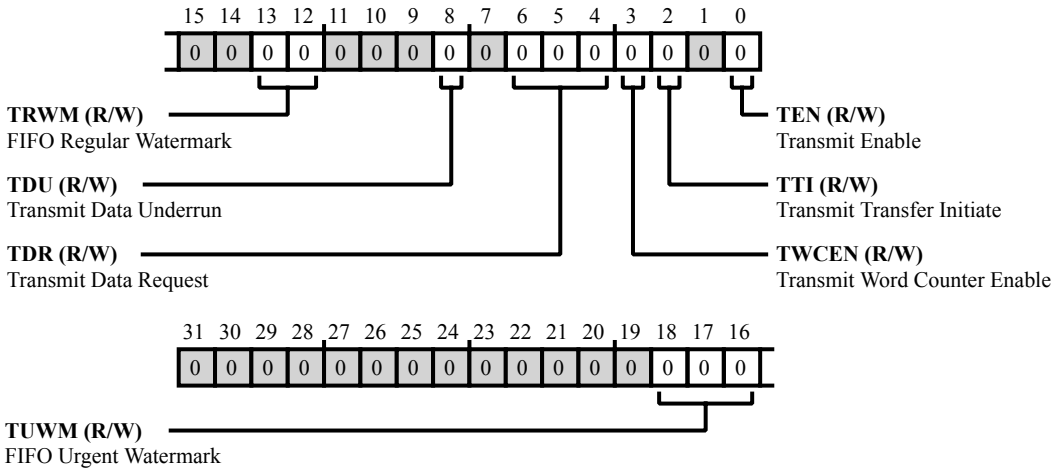


Figure 22-39: SPI_TXCTL Register Diagram

Table 22-36: SPI_TXCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18:16 (R/W)	TUWM	FIFO Urgent Watermark. The SPI_TXCTL.TUWM bits select the transmit FIFO (SPI_TFIFO) watermark level for urgent data bus requests. The SPI also uses this watermark level for generation of the SPI_ILAT.TUWM interrupt request. When an urgent SPI_TFIFO watermark is enabled with SPI_TXCTL.TUWM, the SPI_TXCTL.TRWM selection is used as the deassertion condition for any SPI_ILAT.TUWM interrupt requests that are latched.	
		0	Disabled
		1	25% empty TFIFO
		2	50% empty TFIFO
		3	75% empty TFIFO
		4	Empty TFIFO

Table 22-36: SPI_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13:12 (R/W)	TRWM	FIFO Regular Watermark. The SPI_TXCTL.TRWM bits select the transmit FIFO (SPI_TFIFO) watermark level for regular data bus requests. When an urgent SPI_TFIFO watermark is enabled with SPI_TXCTL.TUWM, the SPI_TXCTL.TRWM selection is used as the deassertion condition for any SPI_ILAT.TUWM interrupt requests that are latched.	
		0	Full TFIFO
		1	TFIFO less than 25% empty
		2	TFIFO less than 50% empty
		3	TFIFO less than 75% empty
8 (R/W)	TDU	Transmit Data Underrun. The SPI_TXCTL.TDU bit selects handling for transmit data requests when the transmit buffer (SPI_TFIFO) is empty. If enabled and SPI_TFIFO is empty, the SPI transmits zero as data. If disabled and SPI_TFIFO is empty, the SPI transmits the last word in the buffer as data.	
		0	Send last word when SPI_TFIFO is empty
		1	Send zeros when SPI_TFIFO is empty
6:4 (R/W)	TDR	Transmit Data Request. The SPI_TXCTL.TDR bits select transmit FIFO (SPI_TFIFO) watermark conditions that direct the SPI to generate a transmit status interrupt request.	
		0	Disabled
		1	Not full TFIFO
		2	25% empty TFIFO
		3	50% empty TFIFO
		4	75% empty TFIFO
		5	Empty TFIFO
3 (R/W)	TWCEN	Transmit Word Counter Enable. The SPI_TXCTL.TWCEN bit enables the decrement of the transmit word count (SPI_TWC) register when the count is not zero and SPI_TXCTL.TTI is enabled. Enabling SPI_TXCTL.TWCEN prevents transmit underrun errors from occurring. The SPI_TXCTL.TWCEN bit is valid only when the SPI is a master.	
		0	Disable
		1	Enable

Table 22-36: SPI_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	TTI	Transmit Transfer Initiate. The SPI_TXCTL.TTI bit enables initiation of transmit transfers if the transmit FIFO (SPI_TFIFO) is not empty. The bit also enables this initiation if SPI_TWC is not zero when SPI_TXCTL.TWCEN is enabled. Enabling SPI_TXCTL.TTI prevents transmit underrun errors from occurring. The SPI_TXCTL.TTI bit is valid only when the SPI is a master.	
		0	Disable
		1	Enable
0 (R/W)	TEN	Transmit Enable. The SPI_TXCTL.TEN bit enables SPI transmit channel operation.	
		0	Disable
		1	Enable

23 Octal Serial Peripheral Interface (OSPI)

The Octal Serial Peripheral Interface (OSPI) controller provides access to serial flash devices. It is designed to access flash devices with its modes enabling efficient communication, with minimum overhead on software. Standard SPI is supported along with high performance dual, quad, and octal SPI modes where data can be transferred on up to 8 data pins. Along with normal mode of operation in single transfer (STR) mode, the OSPI controller supports dual transfer rate (DTR) mode, where data, address, and commands are transferred on both edges of the serial clock. With direct access mode support, any read/write to the OSPI memory space triggers a read/write to flash memory.

Direct read accesses using core and DMA are possible, with the latter minimizing software overhead. The OSPI controller also provides Software Triggered Instruction Generation (STIG) mode, which can be used to erase the data on flash devices and access volatile and nonvolatile configuration registers, legacy flash status register, and other status/protection registers.

The OSPI provides a glueless hardware interface with SPI flash devices. The OSPI peripheral includes programmable baud rates, clock phase, clock polarity, separate dummy cycles for read and write accesses, data sampling control, programmable page size/block size, and so on. With this independent control of how many lines to use for command, address, and data allows to virtually interface any flash device with the processor through OSPI controller.

The OSPI also provides support for DQS when sampling data to improve read data capture and PHY mode support to enable high-speed data transfers.

OSPI Features

The OSPI controller supports the following features:

- Full-duplex, synchronous, serial interface
- Memory mapped direct mode operation for performing flash data transfers
- STIG mode operation to issue flash commands
- Automatic flash status polling support for flash program through direct access mode
- XIP (Execute in Place) support
- Support for single, dual, quad, or octal modes of operation
- Support for DDR mode and DTR protocol (including Octal DDR protocol)

- Up to 16-bit data transfer in each SPI clock cycle
- Programmable baud rate, clock phase, and polarity
- Programmable interrupt generation
- Programmable chip select control timing
- Programmable dummy cycles for read and write operations
- Tune data capture mechanism to improve high speed operation
- Programmable write protected regions to block system writes
- Support for DQS to sample the data to improve read data capturing
- PHY mode support to allow the high speed data transfers

OSPI Functional Description

ADSP-SC59x OSPI Register List

The OSPI is an Octal SPI controller. It contains the following registers.

Table 23-1: ADSP-SC59x OSPI Register List

Name	Description
OSPI_CTL	Octal SPI Control Register
OSPI_DLY	Device Delay Register
OSPI_DRICTL	Device Read Instruction Control Register
OSPI_DWICTL	Device Write Instruction Control Register
OSPI_DSCTL	Device Size Control Register
OSPI_DLLOB_LWR	DLL Observable Register (Lower)
OSPI_DLLOB_UP	DLL Observable Register (Upper)
OSPI_FCA	Flash Command Address Register
OSPI_FCCTL	Flash Command Control Register
OSPI_FCMCTL	Flash Command Control Memory Register
OSPI_FCRD_LWR	Flash Command Read Data Register (Lower)
OSPI_FCRD_UP	Flash Command Read Data Register (Upper)
OSPI_FCWD_LWR	Flash Command Write Data Register (Lower)
OSPI_FCWD_UP	Flash Command Write Data Register (Upper)
OSPI_IMSK	Interrupt Mask Register
OSPI_ISTAT	Interrupt Status Register

Table 23-1: ADSP-SC59x OSPI Register List (Continued)

Name	Description
OSPI_WRPROT_LWR	Lower Write Protection Register
OSPI_MBCTL	Mode Bit Control Register
OSPI_MODID	Module ID Register
OSPI_POLLEXP	Polling Expiration Register
OSPI_OE_LWR	Opcode Extension Register (Lower)
OSPI_OE_UP	Opcode Extension Register (Upper)
OSPI_PHYCTL	PHY Control Register
OSPI_PHYMCTL	PHY DLL Master Control Register
OSPI_POLSTAT	Polling Flash Status Register
OSPI_RDC	Read Data Capture Register
OSPI_REMAPADDR	Remap Address Register
OSPI_WRPROT_UP	Upper Write Protection Register
OSPI_WRPROT_CTL	Write Protection Control Register
OSPI_WCCTL	Write Completion Control Register

ADSP-SC59x OSPI Interrupt List

Table 23-2: ADSP-SC59x OSPI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
137	OSPI0_INT	OSPI0 Error	Level	

OSPI Block Diagram

The OSPI module is comprised of:

- Low level SPI protocol controller
- Internal transmit/receive FIFOs
- DAC
- STIG controller
- Register interface

The ***OSPI Controller Block Diagram*** shows the OSPI controller functional blocks.

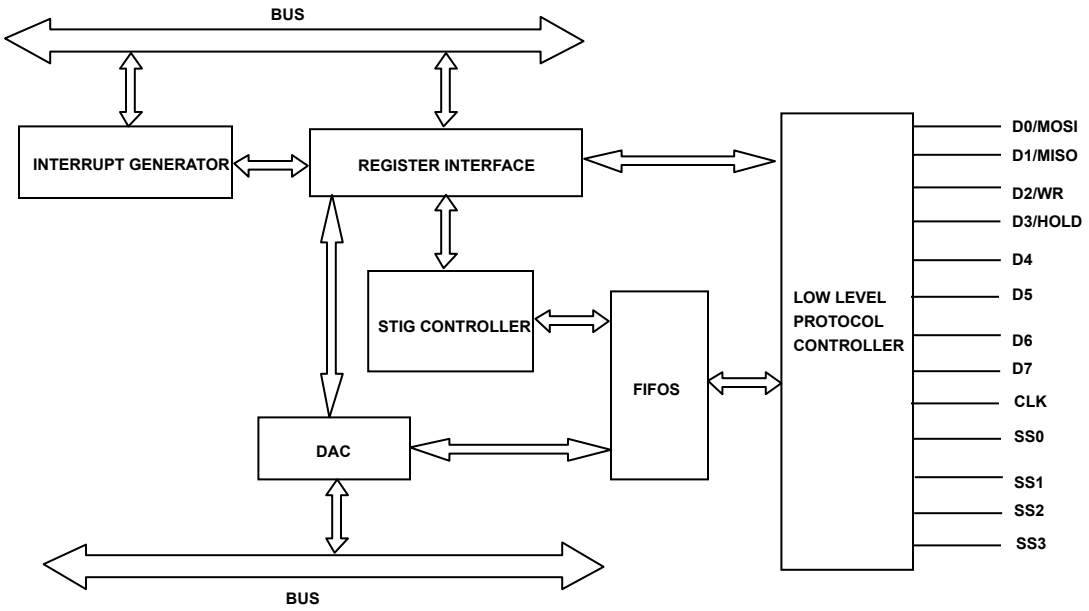


Figure 23-1: OSPI Controller Block Diagram

Architectural Concepts

There are two clock sources for the OSPI controller and one generated clock for clocking external flash device.

The SYSCLK is the main system clock used to transfer data over the AHB bus between the controller (core/DMA) and the OSPI controller. The SCLK0_0 clock is used to access the OSPI controller registers to perform basic configuration of the controller and interrupt handling. The OSPI_REFCLK, is used to serialize the data and drive the OSPI interface. The external clock driven on the OSPI_CLK pin, which is synchronous to the reference clock, is derived from the OSPI_REFCLK and divided using internal baud rate dividers. Using the OSPI_REFCLK clock, allows the core to decouple the frequency of the SPI flash device from system clocks, thereby providing a flexible clocking solution.

The OSPI controller supports two independent controllers (to support different types of flash accesses):

- Direct Access Controller (DAC)
- Software Triggered Instruction Generator (STIG)

The high-level architecture of the OSPI controller is divided into two blocks. The first is the high-level OSPI controller that interfaces with the rest of the SoC. It also queues flash transactions based on data from AHB and APB bus interfaces. The second module is the low-level SPI Protocol Controller. It is used for serializing data from high-level OSPI controller and translating them into the SPI transfer protocol where there are two data paths. The first path preserves backward compatibility with the previous SoCs. This path bypasses the PHY module. Use this path when there is no strict requirement for high performance (clock divider is greater or equal 4).

The second data path is through the PHY Module and is for dividers lower than 4. The configuration is set by software. The controller waits for a valid direct access from the AHB bus (which runs off SYSCLK) or for a software triggered read from the APB bus. When such an event occurs, the corresponding internal controller is selected

(DAC or STIG). The flash command generator block arbitrates between accesses and forwards control into the low-level SPI module. This block works on the reference clock. Transmit data is synchronized internally to the TX FIFO and then serialized to the SPI interface. When the direction of transfer changes and the device returns data to the controller, data is read into the RX FIFO and then synchronized to the internal clocks. The SPI control logic selects the source of the data path (with or without PHY) based on the register configuration. The reference clock and the associated delayed variants are relevant for this transfer type when the PHY mode is enabled.

Direct Access Controller (DAC)

Direct access refers to the operation where AHB bus accesses targeted to the OSPI memory mapped space of processor directly triggers a read/write on flash memory. It is memory mapped and used to access and directly execute the code from external flash memory. When DAC is enabled through the `OSPI_CTL.DACEN` field, any incoming AHB access is serviced by the DAC. Direct accesses can be launched by accessing the OSPI memory address space using core or via one of the MDMA channels. In memory mapped mode, communication to a flash device is automated. The flash memory is accessible directly through reads/write of the processor address space. This access allows code to execute directly from flash devices (true XIP operations). The content can be cached for good performance.

When servicing AHB reads, the DAC sends an additional access downstream apart from the one issued by the AHB in single AHB burst. This is invisible from the system interface. It is a predicted read and ensures that the SPI core operates with maximum bandwidth. The AHB burst not defined by the AMBA specification. It is defined as follows:

- The first access of the AHB burst is defined by:

  An AHB access that is non-sequential (based on address comparison).

  An AHB access that is non-sequential or sequential when the downstream modules are idle.
- The last access of the AHB burst is defined by an AHB access that is sequential and precedes a new burst.
- Size of the AHB burst is the number of AHB accesses (first access to last access).
- The number of DAC requests per AHB burst is equal to the size of the AHB burst + 1.

When AHB clock is slower than the SPI data rate, the sequential read transfer may be interrupted on the SPI side and continues after the AHB clock accepts the last data. The limiting value gets narrower as the SPI clock divider and AHB access size decrease and the number of data IOs (single, dual, quad, or octal) increase. If the system cannot meet the SPI data rate, use higher SPI clock divider. This decreases the SPI data rate but prevents OSPI accesses to external flash to break in between and pass the *Opcode-Address-Dummy* sequence each time. This increases the overall performance by keeping the data transfer continuous.

The controller is designed to work at high data rates. This predicted read results in redundant data during mixed AHB accesses on slow system data rate. Mixed accesses implies that direct AHB sequence of single non-sequential access following few sequential accesses. The predicted read cannot be dropped because of SPI transfer interruption. It is sent as a separate SPI transaction which creates redundant data on the interface for one access. To avoid this issue, drop this data by system or introduce wait states on AHB before issuing a non-sequential direct AHB access.

For bus writes, the DAC triggers a sequence of write commands that mimic the way reads are processed, though the number of DAC write requests are equal to the number of bus write requests received. The bus controller ensures

that flash bursts do not break the flash page boundary. When a page boundary is detected, only the number of byte accesses up to that boundary are forwarded. A sequential direct write request that crosses a page boundary must be detected as non-sequential, causing the controller to force the flash device to enter a self-timed page program cycle. The controller supports splitting the writes crossing the page boundaries only for word-aligned addresses.

The external bus controller supplying data for writes must guarantee write data for a particular page that can be provided to the controller in a timely fashion to avoid downstream starvation. If there is a large delay in the system issuing a sequential write, flash write cycle may be prematurely initiated, reducing the device lifetime.

Flash erase operations, which may be required before a page write, are triggered by software using the STIG. Once a page program cycle has been started, the OSPI controller automatically polls for write cycle to complete before allowing any bus accesses to complete. This is achieved by holding the subsequent AHB direct accesses in wait state.

For memory mapped accesses using the DAC, the flash memory is accessible using the address space 0x60000000 – 0x7FFFFFFF, as indicated in the processor datasheet. Since both SPI2 and OSPI controllers can accesses this space in memory mapped mode, there are control bits in the SCB5_SPI2_OSPI_REMAP register to control which controller is able to access this flash memory mapped space. Either OSPI or SPI2 can exclusively access whole of the flash memory space or there is an option to divide this space between OSPI and SPI2.

When the SCB5_SPI2_OSPI_REMAP register = 0x0, the entire flash memory mapped space is only accessible by the SPI2 controller. Do not try to access this space with the OSPI controller.

When the SCB5_SPI2_OSPI_REMAP register = 0x1, the entire flash memory mapped space is only accessible by the OSPI controller. Do not try to access this space with the SPI2 controller.

When the SCB5_SPI2_OSPI_REMAP register = 0x2, the flash memory mapped space is divided between the OSPI and SPI2 controllers. The flash memory mapped space (0x60000000 - 0x602FFFFF) is only accessible by SPI2 and the flash memory mapped space (0x60300000 - 0x7FFFFFFF) is only accessible by OSPI. This option is only valid for the ADSP-SC59x/2159x HPC package.

Software Triggered Instruction Generator (STIG)

DAC is primarily used to transfer data. To access the volatile and nonvolatile configuration registers, the legacy Flash SPI Status register, other status/protection registers and perform erase operations, a separate software controller is required.

The STIG is controlled using the OSPI_FCCTL register by setting up the command to issue to the flash device. This is a generic controller used to perform instructions that the flash device supports from the extended SPI protocol. The STIG controller command is sent to the flash device and written to the software in the OSPI_FCCTL.OPCODE field. This is different from the OSPI_DRICTL.OPCODERD and OSPI_DWICTL.OPCODEWR fields, as these fields do not impact the STIG operation.

Configuring instructions that are not compliant with the flash specifications can cause unpredicted behavior of the controller. Bit 0 is used to trigger the command, bit 1 used by software to poll the status of the command execution. For reads, when the command has been serviced (bit 1 toggles from 1 to 0), up to 8 bytes of read data is placed in the OSPI_FCRD_LWR and OSPI_FCRD_UP registers. For writes, data must be placed in the OSPI_FCWD_LWR and OSPI_FCWD_UP registers. The STIG completion request can be also checked by the corresponding interrupt.

The interrupt indicates that the controller is ready to accept a new STIG request. The STIG completion request is not equivalent to completion on the SPI side. For example, if STIG is configured to the data command only on transmit, data is taken from the corresponding STIG register fields and put into TX FIFO. As all write bytes are known, another STIG can be queued before completing serialization of the current one.

There are few commands that require more data to read than 8 bytes (for example, READ ID command). Additional STIG memory bank is implemented to accommodate this data, if required. The STIG memory bank is controlled by the OSPI_FCMCTL register. If enabled, the number of bytes to read in the STIG is extended to 16 bytes, as defined in the OSPI_FCMCTL.RDSZ field. There are few commands (excluding read arrays that are not intended to be handled in STIG mode, but in Direct Access Mode) that return more than 8 bytes to the controller.

If the number of read bytes in the STIG, as defined in the OSPI_FCMCTL.RDSZ field, exceeds the memory bank depth, the remaining data overwrites the STIG memory bank locations starting from its first address. When memory bank is enabled, the OSPI_FCRD_LWR and OSPI_FCRD_UP registers keep the last 8 bytes read from the flash device by the STIG. For example, to get a single byte from the last eight bytes of a long continuous read SPI data chain, there is no need to access the STIG memory bank as data can be obtained from the OSPI_FCRD_LWR and OSPI_FCRD_UP registers. To access more data, STIG memory bank data request must be triggered. It is controlled by the OSPI_FCMCTL register and works analogously to trigger the STIG. Bit 0 (OSPI_FCMCTL.TRIGREQ) is used to trigger the command. Bit 1 (OSPI_FCMCTL.BNKREQ) used by software to poll the status of the command execution. When bit 1 toggles from 1 to 0, the data byte (OSPI_FCMCTL.BNKDATA) from the corresponding address (OSPI_FCMCTL.BNKADDR) is valid. The address must be set before triggering the STIG memory bank access. Each consecutive STIG access overwrites the previous one so that the data in the bank fits into byte index fetched by the last STIG access configured to use the memory bank (first incoming byte equals first address of the memory bank, second one equals the second address and so on).

Servicing STIG Request

The OSPI controller determines the number of bytes in the OSPI_FCCTL register to be sent to the flash device, when there is a STIG request. The OSPI_FCCTL.OPCODE field indicates the instruction to be sent and is always pushed first. If there is an address to send, the address (size is also programmed in the same register) is sent next. The address is stored in the OSPI_FCA register. If there are dummy cycles (size of which is also programmed in the OSPI_FCCTL register), they are sent next.

For write data (size of which is also programmed in the OSPI_FCCTL register), up to 8 bytes can be sent (as stored in the OSPI_FCWD_LWR and OSPI_FCWD_UP registers). When read data is collected from the flash device, the OSPI controller stores it in the OSPI_FCRD_LWR and OSPI_FCRD_UP registers.

Arbitration Between DAC and STIG Access

When multiple controllers are simultaneously active, a simple, fixed priority arbitration scheme is used to arbitrate between each interface and to access the external flash.

The fixed priority is defined as follows, highest priority first:

1. Direct Access Write
2. STIG Access

3. Direct Access Read

When one controller is servicing the request, the other controller is back pressured while waiting to be serviced.

Auto Polling for DAC Write Access

For DAC write accesses, which initiate flash program operation, the OSPI controller provides an option to automatically poll the flash status for program completion. When a flash write operation is triggered through DAC, the OSPI controller sends the write data up to page size defined in the OSPI_DSCTL.PGSZ field. Once a page boundary is reached, the DAC stops sending data to the flash device as the flash device enters the program mode. Once the device is in program mode, the OSPI controller can automatically send the flash status read commands to check the state of the flash.

The OSPI_WCCTL register defines the setting to poll the flash device correctly. By default, it is configured to poll bit 0 of the Device STATUS register (using opcode 0x05), which is common across all devices to indicate Write in Progress (WIP). However, in some devices (such as Micron devices, where size is > 512 MB), it is required that the controller polls a different bit of a different device register. This means the controller must issue a different command during this polling phase. For example, in Micron N25Q devices, it is bit 7 of the FLAG STATUS register (opcode 0x70) instead of bit 0 of the STATUS register that must be polled. For this reason, the OSPI_WCCTL register has been provided for software to determine the bit and the opcode to use to poll for write completion, and also the number of successive valid polls that should take place.

To ensure that the bandwidth of the device is not affected by continuous read status SPI transactions, prolong the delay between the successive flash status read command by programming the OSPI_WCCTL.REPDLY field. This feature is implemented to free up the SPI bandwidth, if required. However, defining this delay is not always desired as the ready bit indication can come later what impacts the overall performance. This register must be setup while the controller is idle.

When the OSPI controller starts to service a STIG request, it sets the OSPI_FCCTL.STAT bit to indicate a command execution is in progress. When the OSPI controller is in the auto polling state, servicing a STIG request is different. Several devices are inaccessible after a program operation until the device has completed that write. Few of them have a possibility to suspend the programming page. It can be controlled by the OSPI_POLSTAT.STAT bit, which indicates active auto polling phase. After requesting a STIG, the OSPI controller issues an appropriate opcode to memory. While servicing a STIG (in auto polling phase), the OSPI_FCCTL.STAT bit remains steady and other parts of transfer such as address or dummy bits are disabled (to issue program suspend command, only opcode is needed).

SPI Command Translation

Requests issued by the DAC or STIG are translated into a sequence of byte transfers to send downstream (before serializing the flash device). These sequences depend on the requested transfer, except the following example (1-byte non-sequential read):

INSTRUCTION OPCODE > ADDRESS > MODEBYTE > DUMMY CYCLES > 1-DATA BYTE

NOTE: During data phase mode, data programmed in the OSPI_MBCTL register is driven on pins. If mode data is not enabled, the address phase is followed directly by dummy cycles. During the dummy cycle phase the

OSPI data pins (D0-D7) are three-stated (High-z). To ensure the correct signal level on the data pins during this phase, external pull-ups can be used on data pins.

For sequential accesses, an extra data byte per read is pushed to the flash device on the back of the above sequence, assuming that it can be done without any gap between each transferred byte.

The actual sequence sent to the flash device depends on the requested transfers (non-sequential or sequential), whether the device is configured in XIP mode or not, and the state of the OSPI_DRICTL and OSPI_DWICTL registers. For writes, the write enable latch (WEL) within the flash device must be high before a write sequence can be issued.

The OSPI controller automatically issues a WEL command before triggering a write command via the DAC (the program need not perform this operation). To increase flexibility and performance, turn off this feature by setting the OSPI_DWICTL.WELDIS bit. The opcode for WEL is 0x06 and common between the devices. However, the WEL operation is programmed using the OSPI_OE_LWR.XRDBYT/OSPI_OE_UP.FSTBYTWEL bits and (optionally) the OSPI_OE_LWR.XWRBYT/OSPI_OE_UP.XBYTWEL bits (flash device requires 2-byte command code for WEL).

When write requests from the DAC are not received and all outstanding requests have been sent, the flash device automatically starts the page program write cycle. Any incoming request at this time is held in wait state until the cycle is completed. The OSPI controller automatically polls the flash device legacy SPI status register to identify when the write cycle has completed. This is achieved by sending the Read Status Register (RDSR) opcode to the flash device and waiting until the device indicates that the write cycle is completed (the Write in Progress bit is cleared to zero and WEL bit is cleared to zero or device ready bit is set to one). The WREN and RDSR device instructions are only sent by the controller. For other specific instruction that the user determines must be sent to the device (for example, if the device needs to be unprotected before a write command is issued), must be separately handled by issuing flash commands via the STIG.

Hold and Reset Control

There is an option to trigger hold and reset features on the I/O of the flash device. The hold functionality is common across the devices and takes an alternative function of the DQ3 pin (applicable when device does not operate in quad SPI or DDR modes). The transfer can be held and resumed by the OSPI_CTL.HLD bit. Devices with hold feature on DQ3 need another dedicated pin for hardware reset. Devices without hold feature have an alternative reset on the DQ3 pin, which makes the additional reset pin redundant. The controller supports both variants. The OSPI_CTL.RSTCFG field allows the user to configure the hardware reset solution implemented in the device. After configuration, it is possible to trigger hold or reset features using I/O (OSPI_CTL.HLD or OSPI_CTL.RST bits).

NOTE: The OSPI controller does not support dedicated reset pin to control the reset of flash devices. Flash devices with reset functionality on DQ3 can be controlled directly via OSPI controller.

After hold activation, the controller is in waiting state. Any other operations must not be requested before de-asserting the OSPI_CTL.HLD bit. The hold feature is useful when any SPI transaction needs to be prolonged to adjust it into specific point in time. To check if the hold request is suspended, the OSPI_CTL.IDLE bit can be polled

for. If SPI is not in the idle state, the transfer is suspended. Software must reset the OSPI_CTL.HLD bit before any new SPI transaction. In case hold request is set before starting any transfer, it will be in hold state after it starts.

Hardware reset must be activated when CS is high (no valid transaction is present on SPI bus). It can be checked by polling the OSPI_CTL.IDLE bit. If the controller is in the idle state and no other transfer requests are queued, hardware reset can be triggered. Reset feature is useful when any write, program, or erase operation needs to be cancelled. No transfer request is permitted before driving the reset back to inactive. Triggering hold or reset on the DQ3 pin at the time the device is configured to work in Quad SPI or DDR mode, overwrites transfer data on the DQ3 with 0. This behavior is considered as a software error. Ensure that the flash device is introduced to suitable SPI Mode (by polling its Configuration Register) before triggering alternative DQ3 function.

Flash Instruction Type Support

To send correct read and write opcodes, software must initialize the OSPI_DRICTL and OSPI_DWICTL registers. These registers include fields to setup the required instruction opcodes that is intended to be used to access the flash (default is basic read and basic page program) as well as the instruction type, edge mode (DDR or SDR) and whether the instruction uses single, dual, quad, or octal pins for address and data transfers.

To ensure that the controller can operate from a reset state, the registers are reset to an opcode compatible with single I/O standard SPI devices. Though it is applicable for both reads and writes, the OSPI_DRICTL.INSTRTYP field appears only once (not included in the OSPI_DWICTL register). If software sets this to any value other than 0, the OSPI_DRICTL.ADDRTRNSFR/OSPI_DWICTL.ADDRTRNSFR/ OSPI_DRICTL.DATATRNSFR/OSPI_DWICTL.DATATRNSFR bits become don't care. It is made available to allow software to support less common flash instructions, where opcode, address, and data are sent on 2 or 4 data lines (opcode from several instructions are sent serially to the flash device, even for dual/quad instructions).

Dual Data Rate (DDR) Operations

The OSPI controller supports DDR mode of operation, in which some or all transfer phases happen in DDR mode, where the I/O lines are driven/sampled on both rising and falling edges of the clock.

In DDR mode, the OSPI controller supports the following modes:

- Dual Transfer Rate (DTR) mode, where command is driven in SDR mode and address/dummy cycles/data are driven in DDR mode.
- DTR protocol mode, where everything including command is driven in DDR mode.

For DTR mode, there are specific commands which are recognized by the flash as DTR commands. These are DTR Read commands which can be handled in STR Protocol (but in DTR Mode). This is achieved by setting the OSPI_CTL.DTREN bit to 0 and OSPI_DRICTL.DDREN bit to 1.

For DTR protocol mode, all normal commands can be sent in DDR fashion, provided the flash is already configured in a mode to accept the commands in DDR mode. Support for DTR protocol provides significant enhancement. This is enabled in the OSPI controller by setting the OSPI_DRICTL.DDREN bit to 1. It allows the device to work in DTR mode for each possible command (not only DTR read ones). It can also handle opcode phase in DTR mode, which improves the overall performance.

There are several devices (for example, Micron N25Q512A) that can handle read operations in DTR mode. They can issue and capture data on both rising and falling edges while working with a dedicated command type. This enables the controller to maintain same throughput twice the lower frequency of SPI_CLK for STR mode.

In MT25Q family of Micron devices, DTR protocol is implemented. It enables device to handle all commands in DTR mode. DTR Read commands detect DTR mode based on dedicated opcode. Therefore, opcode must be sent as STR. When DTR protocol is enabled, the device does not need opcode to detect edge mode, as it can recognize based on the volatile or nonvolatile bit in the flash configuration register.

The ***Read Configuration Examples*** table illustrates how software must configure the controller for selected specific read and write instruction supported by the different devices.

Table 23-3: Read Configuration Examples

Command	Opcode (Number of Lanes/ Edge Mode)	Address/ Dummy/ Mode (Number of Lanes/ Edge Mode)	Data (Number of Lanes/ Edge mode)	Instruction Type `OSPI_DRICTL.INSTRTYP`	Address Transfer Type `OSPI_DRICTL.ADDRTRNSFR`	Data Transfer Type `OSPI_DRICTL.DATATRNSFR`	DDR Bit Enable `OSPI_DRICTL.DDREN`
READ	1/SDR	1/SDR	1/SDR	0	0	0	0
FAST_READ	1/SDR	1/SDR	1/SDR	0	0	0	0
DTR FAST_READ	1/SDR	1/DDR	1/DDR	0	0	0	1
DOR (Dual output Fast Read)	1/SDR	1/SDR	2/SDR	0	0	1	0
DTR DOR (Dual output Fast Read)	1/SDR	1/DDR	2/DDR	0	0	1	1
DIOR (Dual I/O Fast Read)	1/SDR	2/SDR	2/SDR	0	1	1	0
DTR DIOR (Dual I/O Fast Read)	1/SDR	2/DDR	2/DDR	0	1	1	1
QOR (Quad output Fast Read)	1/SDR	1/SDR	4/SDR	0	0	2	0
DTR QOR (Quad output Fast Read)	1/SDR	1/DDR	4/DDR	0	0	2	1
QIOR (Quad I/O Fast Read)	1/SDR	4/SDR	4/SDR	0	2	2	0

Table 23-3: Read Configuration Examples (Continued)

Command	Opcode (Number of Lanes/ Edge Mode)	Address/ Dummy/ Mode (Number of Lanes/ Edge Mode)	Data (Number of Lanes/ Edge mode)	Instruction Type `OSPI_DRICTL.INSTRTYP`	Address Transfer Type `OSPI_DRICTL.ADDRTRNSFR`	Data Transfer Type `OSPI_DRICTL.DATATRNSFR`	DDR Bit Enable `OSPI_DRICTL.DDREN`
DTR QIOR (Quad I/O Fast Read)	1/SDR	4/DDR	4/DDR	0	2	2	1
OOR (Octal output Fast Read)	1/SDR	1/SDR	8/SDR	0	0	3	0
DTR OOR (Octal output Fast Read)	1/SDR	1/DDR	8/DDR	0	0	3	1
OIOR (Octal I/O Fast Read)	1/SDR	8/SDR	8/SDR	0	3	3	0
DTR OIOR (Octal I/O Fast Read)	1/SDR	8/DDR	8/DDR	0	3	3	1
DCFR (Dual Command Fast Read)	2/SDR	2/SDR	2/SDR	1	don't care	don't care	0
DTR DCFR (Dual Command Fast Read)	2/SDR	2/DDR	2/DDR	1	don't care	don't care	1
QCFR (Quad Command Fast Read)	4/SDR	4/SDR	4/SDR	2	don't care	don't care	0
DTR QCFR (Quad Command Fast Read)	4/SDR	4/DDR	4/DDR	2	don't care	don't care	1
OCFR (Octal Command Fast Read)	8/SDR	8/SDR	8/SDR	3	don't care	don't care	0
DTR OCFR (Octal Command Fast Read)	8/SDR	8/DDR	8/DDR	3	don't care	don't care	1

NOTE: This data is applicable for 3-byte and 4-byte address variants of the commands.

In DTR protocol, all transfer phases (including opcode) take DDR edge mode independent of the command under execution. DTR protocol is enabled by the OSPI_CTL.DTREN bit. It has higher priority than the OSPI_DRICTL.DDREN bit. Therefore, if the OSPI_CTL.DTREN bit is set, irrespective of value programmed in the OSPI_DRICTL.DDREN bit, all command, address, dummy cycles, and data are transferred in DDR mode.

Table 23-4: Write Configuration Examples

Command	Opcode (Number of Lanes)	Address/ Dummy/Mode (Number of Lanes)	Data (Number of Lanes)	Instruction **OSPI_DRICTL.INSTRTYP**	Address Transfer Type **OSPI_DWICTL.ADDRTRNSFR**	Data Transfer Type **OSPI_DWICTL.DATATRNSFR**
PP (Page Program)	1	1	1	0	0	0
DIFP (Dual Input Fast Program)	1	1	2	0	0	1
DIEFP (Dual Input Extended Fast Program)	1	2	2	0	1	1
QIFP (Quad Input Fast Program)	1	1	4	0	0	2
QIEFP (Quad Input Extended Fast Program)	1	4	4	0	2	2
OIFP (Octal Input Fast Program)	1	1	8	0	0	3
OIEFP (Octal Input Extended Fast Program)	1	8	8	0	3	3
DCPP (Dual Command Page Program)	2	2	2	1	don't care	don't care
QCPP (Quad Command Page Program)	4	4	4	2	don't care	don't care
OCPP (Octal Command Page Program)	8	8	8	3	don't care	don't care

NOTE: This data is applicable for both 3-byte or 4-byte address variants of the commands.

In DTR protocol, all transfer phases (including opcode) take DDR edge mode independent of the command under execution.

Data Sampling

The OSPI provides flexibility to configure the sampling point of read data with the SPI clock to meet setup and hold timings of the flash devices when operating at higher speed. The OSPI_RDC register provides an option to control the sampling point. In OSPI controller, data received on the OSPI pins is sampled by internal reference clock (OSPI_REFCLK). In STR mode, by default (when the OSPI_RDC.DLYRD = 0), the incoming data is on the first falling edge of the OSPI_REFCLK in second half of SPI clock. In DTR mode, it is on the first falling edge of the OSPI_REFCLK in second quarter of SPI clock.

The OSPI_RDC.SMPLEDG bit selects the edge of the OSPI_REFCLK, in which data from flash memory are sampled. This increases the sampling resolution by two times. The OSPI_RDC.DLYRD field controls the additional number of OSPI_REFCLK cycles that must be applied to the internal read data capture circuit. Large clock-to-out delay of the flash memory, trace delays, and other device delays may impose an upper limit on the flash clock frequency which is less for the flash memory to operate. To compensate this, software must set this register to a value that guarantees robust data captures.

The OSPI_RDC.SMPLEDG bit selects the edge of the SYSCLK, in which data from flash memory are sampled. This increases the sampling resolution by two times. The OSPI_RDC.DLYRD field controls the additional number of SYSCLK cycles that must be applied to the internal read data capture circuit. Large clock-to-out delay of the flash memory, trace delays, and other device delays may impose an upper limit on the flash clock frequency which is less for the flash memory to operate. To compensate this, software must set this register to a value that guarantees robust data captures.

For example, when the baud divider (OSPI_CTL.BAUD field) is programmed to 7, it results in divider of 16. Therefore, SPI_CLK: OSPI_REFCLK = 16:1 (every 1 SPI_CLK cycle has 16 OSPI_REFCLK cycles). When the OSPI_RDC.DLYRD bits are programmed to X,

In STR mode of operation, received data is captured on:

- (16/2) + X^{th} falling edge of OSPI_REFCLK if OSPI_RDC.SMPLEDG = 0
- (16/2) + X^{th} rising edge of OSPI_REFCLK if OSPI_RDC.SMPLEDG = 1

In DTR mode of operation, received data is captured on:

- (16/4) + X^{th} falling edge of OSPI_REFCLK if OSPI_RDC.SMPLEDG = 0
- (16/4) + X^{th} rising edge of OSPI_REFCLK if OSPI_RDC.SMPLEDG = 1

To improve the hold timing during transfers in DTR mode, the OSPI_RDC.DDRDLYRD delays the transmitted data by programmable number of OSPI_REFCLK cycles.

NOTE: The OSPI interface for Octal DTR mode may not work at higher frequencies for flash devices that require DQS.

PHY Module Architecture

The PHY Module is divided into the following four sub-blocks:

- Data Transmitter

  The data transmitter serializes the transmit data to assign complaint signal values into the controller output interface. The input data size is two bytes to cover the octal DDR scenario where two bytes are sent within a single reference clock cycle. The configuration interface covers selected SPI protocols (single, dual, quad, or octal) or command types (DDR, SDR). These values are necessary to calculate the number of bits that fit into a clock cycle and consequently what set value must be set by the hardware.

- Data Receiver

  The data receiver block samples data from flash device.

- Delay Line Module

  The delay line module block provides an appropriate delay for the input clock to ensure correct data transmitting and sampling. This block contains two separate delay paths for the OSPI clock, one feeding the external FLASH device and the other feeding the internal PHY sampling clock used for capturing data from the device when the PHY mode is enabled. Delay line values are configurable in software.

- Clock Arbiter

  The clock arbiter handles gating the clock logic for clocks forwarded to the DLL inputs.

The ***PHY Module Block Diagram*** shows the sub-blocks in the PHY module. The sections following the diagram describe the architecture of the sub-blocks in more detail.

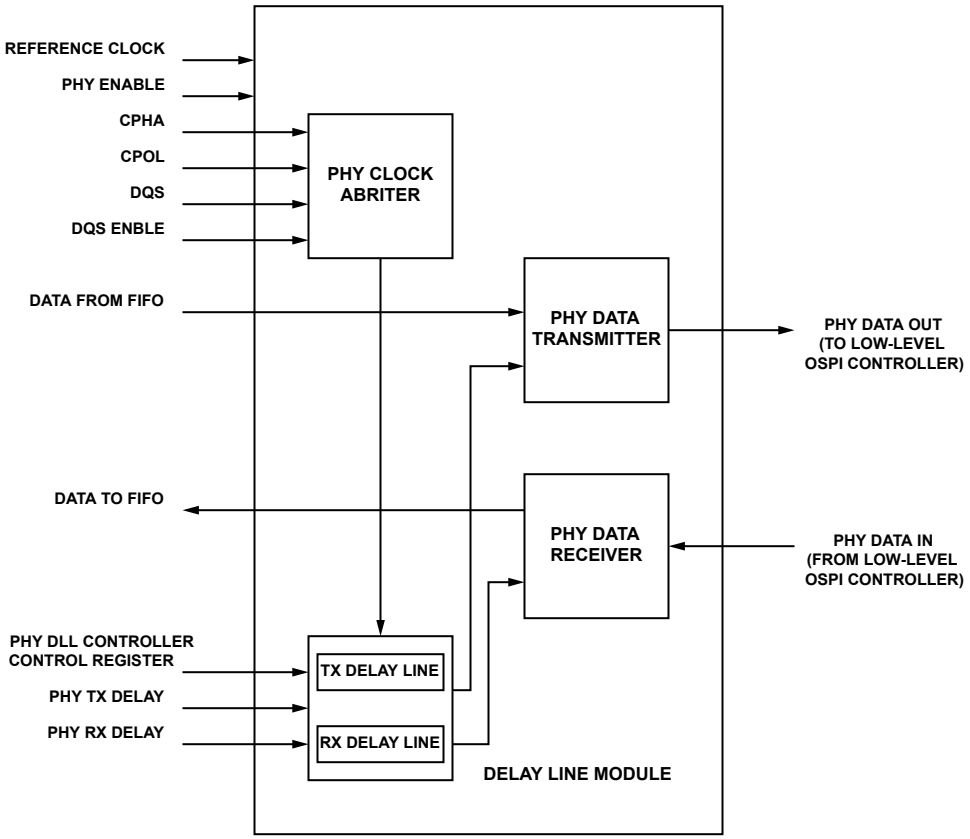


Figure 23-2: PHY Module Block Diagram

PHY Data Transmitter

The data transmitter latches queued data to transmit on both the negative and positive edges of the reference clock. Then data is chopped to fit into the output interface. With the PHY module, in the last stage of the TX data path data from both edges are connected, forwarded back to the low-level OSPI controller, and then put on the OSPI controller outputs if the PHY enable bit is asserted. Otherwise, the controller outputs generated by the low-level OSPI controller are selected.

PHY Data Receiver

In the data receiver module, data is sampled on both edges of the delayed reference clock (sampling clock) ensuring a wide range of the sampling window. Read data is gathered in shift registers and then merged into a byte-sized input FIFO. Data reception is programmable to work either with the delayed reference clock or with the DQS input signal from the flash device.

Delay Line Module

Due to the asynchronous nature of flash devices, it is important to address the timing requirements for capturing and receiving data between the controller and the flash. The PHY module contains logic to meet the timing requirements. The delay compensation circuit is designed with the following features:

- Programmable OSPI clock delay specified as a percentage of a clock cycle

- Programmable sampling clock delay specified as a percentage of a clock cycle
- Delay compensation circuit re-sync circuitry activated during refresh cycles to compensate for temperature and voltage drift

The delay compensation circuitry relies on a controller/target approach. The controller delay line is used to determine how many delay elements constitute a complete cycle. This count, along with the programmable fractional delay settings, determines the actual number of delay elements to program into the target delay lines. The controller and target delay lines are identical. This approach allows the memory controller to observe a clock and then delay other signals by a fixed percentage of that clock.

DLL Locking is controlled through the register interface. When the DLL is reset, the controller DLL performs a locking procedure starting with the DLL start point value in the `OSPI_PHYMCTL.INITDLY` bit field and the current frequency of operation. The DLL start point value must not exceed a delay of 1 and 1/4 cycles of delay, when calculated by estimating the worst case timing through a delay element at the highest frequency the DLL will be operating at. For example, if the maximum operating frequency is 200 MHz (period of 5 ns) and the worst case delay element has a 85 ps delay, then program the DLL start point to 5 / 0.085 = 58.8 = 59 elements = 0x3B.

With this setting, the controller DLL is will correctly lock for all frequencies below 200 MHz and in any process corner. If the delay provided by the delay line is enough to cover a full clock cycle, the controller DLL is in full clock mode. In this case, the DLL lock value in the `OSPI_DLLOB_LWR.LWRLOCK` bit field reports the number of delay element in one full clock cycle. The target delay line fractional setting uses this number to determine the number of delay elements to add to the target delay lines. For example, if the DLL lock value = 50 = 0x32, then the target delay line percentage = 25% = 0x20, and the target delay line delay value = 50 × 0.25 = 12.5 elements. The DLL Module will round the calculated value into an integer number of delay elements.

If the frequency of operation is such that the delay line is not long enough to accommodate a full cycle of delay, the controller DLL will automatically detect this and switch to half clock mode. In this mode, the controller DLL attempts to lock when the delay in the delay line reaches a half-clock cycle. If lock is achieved in half clock mode, the target delay lines are automatically adjusted to program a fractional delay of a full cycle. There is no need to change the target delay settings based upon the lock mode of the controller DLL. For example, if the DLL lock value = 50 = 0x32, then the target delay line percentage = 25% = 0x20, and the target delay line delay value = (50 2) × 0.25 = 25 elements. Do not work in half clock mode when using a loop-back sampling method or when the SPI clock is configured to SPI Mode 3 (CPHA =1, CPOL =1).

If the frequency of operation is such that the delay line is not long enough to capture a half-clock cycle of delay, the controller DLL will indicate lock and set the number of delays to the maximum length of the delay line. This is called saturation mode. There is no need to change the target delay settings in this mode. The target delay settings will be fixed at the fractional delay based upon the maximum delay of the delay line times 2. For example, if the DLL lock value = MAX = 128 = 0x80, and the target delay line percentage = 25% = 0x20, then the target delay line delay value = (128 × 2) × 0.25 = 64 elements.

PHY Clock Arbiter

The OSPI clock is generated only when the SPI control logic FSM indicates that an SPI transfer is ongoing. Conversely, the reference clock is continuously generated. This module generates a sampling clock that is ready to pass

through the RX delay line. It is generated when read data phase of SPI transfer is detected based on information from the SPI control module. The sampling clock is optionally sourced from DQS or loop-back input.

PHY Pipeline Mode

The PHY pipeline mode is for a direct read mode of operation. The flash command generator pipelines and puts a few expected sequential accesses into the TX FIFO causing the CS to stay active since the low-level OSPI controller controls the internal TX FIFO fill level. To correctly trigger a direct read in pipeline mode, first poll the `OSPI_CTL.IDLE` bit to determine that the TX FIFO is empty. The sequential data transfer is interrupted when non-sequential access occur or too many wait states are introduced by the SPB controller to keep the flash transaction continuous. Introducing wait states slows down the system data rate.

Enable this mode when following conditions are met:

- SYSCLK > 1.1 × OSPI_REFCLK

  NOTE: In PHY mode the reference clock is taken from the CDU CLKO10

- Accesses are only 32-bits in size

Programming Concepts

Software configures the OSPI controller before communicating with the flash device. The static configuration bits must be setup before the OSPI controller is enabled using the `OSPI_CTL.EN` field. To change the controller configuration, disable it before reconfiguring.

Configuring OSPI after Reset

The OSPI controller comes up with a state that is suitable to perform basic reads and writes using the DAC. Basic read (opcode 0x03) and write (opcode 0x02) instructions are supported by all target devices. The controller also wakes up with a baud rate divider setting of 32. When the reference clock operates at 500 MHz after reset, the effective SPI clock is 15.625 MHz. This must be slow enough to meet all timing requirements of all target devices without any further device programming.

When the target device does not use three address bytes, the `OSPI_DSCTL` register must be modified to the appropriate size. If software wants writes to the device and the number of bytes per device page is not equal to 256, the `OSPI_DSCTL` register must also be modified. The software must enable the write protect feature prior to enabling the OSPI controller. This blocks any AHB writes from taking effect. The `OSPI_WRPROT_LWR`, `OSPI_WRPROT_UP`, `OSPI_WRPROT_CTL`, and `OSPI_DSCTL.PGSZ` fields must be setup.

After reset, software can read from and write to the flash device. Enabling/disabling the controller and DAC is achieved by writing to the corresponding `OSPI_CTL` register fields. Maintain the default values of the baud rate divisor and default state of the `OSPI_CTL.CPOL`/ `OSPI_CTL.CPHA` bits. A write data value of 0x00780081 is recommended.

Programming Dummy Cycles

When programming the dummy cycles for OSPI reads (DAC and STIG-initiated reads) from the flash device, mode data should be considered as it impacts the actual number of dummy cycles seen on the bus. When mode data is not enabled, then the number of dummy cycles is that same as those set in dummy cycles field, respectively, for STIG or DAC operation.

However, when mode data is enabled, then in addition to the programmed number of dummy cycles, a few extra cycles are needed to transmit out the mode data. The number of cycles added depends on the bus mode of the OSPI. For example, when the address phase of the command is configured to be sent on a single line in STR mode, then the mode bits consume eight cycles. When the address phase of the command is configured to be sent on four lines in DTR mode, then the mode bits consume one cycle. These extra cycles should be considered when configuring the dummy cycles for the controller, so the overall dummy clock cycles match the dummy cycles provided in flash data sheet.

Configuring OSPI for Optimal Use

Software must accurately configure the controller to optimally access the flash.

To configure the controller:

1. Wait until pending STIG are completed or poll the `OSPI_CTL.IDLE` field.
2. Disable the `OSPI_CTL.DACEN` field. The OSPI controller can be completely disabled using the `OSPI_CTL.EN` field.
3. Update the `OSPI_DLY` register. This register allows the user to tweak how the chip select is driven after each flash access. This is required as each device may have different timing requirements. As the serial clock frequency is increased, these timing requirements become critical. The numbers programmed in this register are based on the period of OSPI_REFCLK.

   For example, an ATMEL device needs 50 ns minimum time before CS can be re-asserted after it has been de-asserted. By default, the controller provides a a minimum of one SCLK period. When the device is operating at 50 MHz, the SCLK period is only 20 ns. Therefore, additional 30 ns are required. As the register defines the number of OSPI_REFCLK cycles to add, if OSPI_REFCLK is running at 500 MHz (2 ns period), user must program a value of at least 15 to the `OSPI_DLY.DSRT` bits. This delay can be extended during auto-polling phase. Polling repetition delay can be defined in the `OSPI_WCCTL.REPDLY` field.
4. Update the `OSPI_DSCTL` register. The number of bytes per page is required to perform any write operation on flash. The number of bytes per device block is only required if the write protect feature is used.
5. Setup and enable the write protection registers (`OSPI_WRPROT_LWR`, `OSPI_WRPROT_UP`, and `OSPI_WRPROT_CTL`), if write protection features are required.
6. Enable the required interrupts using the `OSPI_IMSK` register.
7. Update the `OSPI_REMAPADDR` register to remap the DAC addresses to different address in flash.
8. Set the baud rate divisor in the `OSPI_CTL.BAUD` to define the required clock frequency of the target device.

9. Select the appropriate chip select signal to use through the OSPI_CTL.BAUD bits.
10. Update the OSPI_RDC register. This register delays when read data is captured and helps when read data path from the device to the controller is long and the device clock frequency is high.
11. Enable the OSPI controller and DAC using the OSPI_CTL register.

Configuring OSPI for DAC Read Operation

1. Configure the OSPI_DRICTL register:
    a. Program the desired opcode for Read command (0xBB for Dual SPI read).
    b. Configure th instruction type, address transfer type, and data transfer type as per the opcode.
    c. Configure the dummy cycles for the selected command opcode depending on the flash device.
    d. Set the Mode bit Enable if the mode data needs to be transferred during dummy cycles phase.
    e. Set the DDR enable bit if the command works in DTR mode.
2. If the operation requires dual opcode, set the OSPI_CTL.OPCODEEN bit.
3. Update the second byte for read opcode in the OSPI_OE_LWR.XRDBYT field.
4. To operate in DTR protocol mode, set the OSPI_CTL.DTREN bit.

   If this bit is set, the OSPI_DRICTL.DDREN bit has no effect and all phases of command including the op-code are done in DDR mode.
5. Update the OSPI_DSCTL.ADDRSZ field.
6. Update the OSPI_MBCTL.MODE field.
7. Enable the DAC mode using the OSPI_CTL.DACEN bit, if not enabled.
8. Start the read transfer by accessing the OSPI memory mapped space through core or MDMA accesses. Any access to address 0x60000000 will access the flash address 0x0 if address remapping is 0.
9. If using MDMA for the transfer, wait for the DMA to complete.

NOTE: For accesses through core, software need not check FIFO levels. All the FIFOs are internally maintained by DAC.

Configuring OSPI for DAC Write Operation

1. Configure the OSPI_DWICTL register:
    a. Program the desired opcode for write command like 0x02 for single SPI page program.
    b. Configure the address transfer type and data transfer type as per the opcode.
    c. Configure the dummy cycles for selected command opcode depending on the flash device.

2. Update the OSPI_DRICTL.INSTRTYP field as per the selected opcode.
3. Clear the OSPI_DWICTL.WELDIS bit to manually send the WEL (Write Enable) instruction to flash. Else, the controller automatically issues WEL to flash before sending any write command to flash.

   If WEL command for flash under use is not 0x06, update the OSPI_OE_UP.FSTBYTWEL field.
4. If the operation requires dual opcode, set the OSPI_CTL.OPCODEEN bit.
5. Update the second byte for read opcode in the OSPI_OE_LWR.XWRBYT field.
6. Update the number of address bytes the command expects in the OSPI_DSCTL.ADDRSZ field.
7. As the controller supports the automatic polling for write operation complete, update the OSPI_WCCTL register accordingly to enable the automatic polling for flash program operation complete.
   a. Enable the automatic polling by clearing the OSPI_WCCTL.DIS bit.
   b. Update Polling opcode and define polling bit index and polarity as per the flash device.
   c. Update polling count and polling repetition delay.
   d. Optionally, update the OSPI_POLLEXP register if polling expiration is enabled.
8. Enable the DAC mode using the OSPI_CTL.DACEN, if not enabled.
9. Start the read transfer by accessing the OSPI memory mapped space through core or MDMA accesses. Any access to address 0x60000000 will access the flash address 0x0 if address remapping is 0.
10. Wait for the DMA to complete, if using MDMA for the transfer.

NOTE: For accesses through core, software need not check FIFO levels. All FIFOs are internally maintained by DAC.

Issuing STIG Command

This is a typical method that software uses to access the flash device registers and perform erase operations. It can also be used to access the flash array (though only 8 data bytes can be read or written at a time), as defined in the OSPI_FCRD_LWR, OSPI_FCWD_LWR, OSPI_FCRD_UP and, OSPI_FCWD_UP registers.

To issue a STIG command:

1. Set the OSPI_IMSK.STIGREQ_MSK bit to enable the STIG command completion interrupt.
2. Update the OSPI_FCCTL.OPCODE and OSPI_FCCTL.DMY bits, as per the flash command that need to be executed via STIG.
3. Clear the OSPI_FCCTL.STIGBNKEN bit.
4. Set the OSPI_FCCTL.ADDREN bit, iwhen the command requires the address to be driven.

   When set:

a. Update the OSPI_FCCTL.ADDRSZ field according to the number of address bytes expected by the command.

b. Update the OSPI_FCA register with the desired flash address.

5. When the command expects the data to be sent to flash, set the OSPI_FCCTL.WREN bit.

   When set:

   a. Update the OSPI_FCCTL.WRSZ field according to the number of data bytes to be sent to flash.

   b. Update the OSPI_FCWD_LWR and OSPI_FCWD_UP registers with data to be sent to flash.

6. When the command expects the data to be received from the flash, set the OSPI_FCCTL.RDEN bit.

   When set, update the OSPI_FCCTL.RDSZ field according to the number of data bytes to expected from flash.

7. Update the OSPI_FCCTL.MODEEN bit, when the mode data are sent during dummy cycles.

   When set, update the OSPI_MBCTL.MODE field with desired mode data.

8. Set the OSPI_FCCTL.EXE bit to execute the STIG command.

9. Set the OSPI_ISTAT.STIGREQ bit (or wait for interrupt if enabled) indicating that the STIG command execution is completed.

10. When the command expects the data to be received from the flash, read the data from the OSPI_FCWD_LWR and OSPI_FCWD_UP registers.

NOTE: Do not set the OSPI_FCCTL.WREN and OSPI_FCCTL.RDEN bits simultaneously. A given command can either read data or send data to the flash, but not both at the same time.

With normal STIG command, only 8 bytes can be read at a time, but with support of STIG memory bank up to 16 bytes can be read in to STIG memory bank. This is controlled by the OSPI_FCCTL.STIGBNKEN bit and OSPI_FCMCTL register. Commands issued using this interface have a higher priority than all other read accesses coming from AHB, and interrupts any read commands being requested by the DAC.

To issue a STIG memory bank read command:

1. Set the OSPI_IMSK.STIGREQ_MSK bit to enable STIG command completion interrupt.

2. Update the OSPI_FCCTL.OPCODE and OSPI_FCCTL.DMY bits as per the flash command that need to be executed via STIG.

3. Set the OSPI_FCCTL.STIGBNKEN bit.

4. Set the OSPI_FCCTL.ADDREN bit, if the command requires the address to be driven.

   If set:

   a. Update the OSPI_FCCTL.ADDRSZ field according to the number of address bytes expected by the command.

b. Update the OSPI_FCA register with the desired flash address.

5. Update the OSPI_FCCTL.MODEEN bit, if the mode data are sent during dummy cycles.

   If set, update the OSPI_MBCTL.MODE field with the desired mode data.

6. Set the OSPI_FCCTL.EXE bit to execute the STIG command.
7. Set the OSPI_ISTAT.STIGREQ bit (or wait for interrupt if enabled) indicating that the STIG command execution is completed.

At this stage, 16 bytes of data read from the flash device are available in the STIG memory bank. To access this data, a STIG memory bank read must be issued. Each read can read one byte form STIG memory bank at a time. To read all data, 16 reads must be issued.

To read a data byte from STIG memory bank:

1. Set the OSPI_FCMCTL.BNKADDR field. This can be any value from 0 to 16.
2. Set the OSPI_FCMCTL.TRIGREQ bit to trigger the STIG memory bank read.
3. Poll the OSPI_FCMCTL.BNKREQ bit to clear.
4. Read the requested data byte from the OSPI_FCMCTL.BNKDATA.

Entering XIP mode

The controller supports XIP operations to minimize the latency for back-to-back reads or code execution.

If the flash device comes in XIP on power up, software cannot discover the state of XIP from POR via the Flash Status Register reads. The only operation of a flash device when XIP mode is enabled, is an XIP read. In such cases, software must be aware that if flash enters XIP from POR, the OSPI_MBCTL.MODE and OSPI_CTL.XIPIMM fields must be set. This makes the controller to enter XIP mode immediately and start communicating with the flash device in XIP mode on next read issued. Therefore, it does not require the read opcode to be transferred. To exit XIP mode, this bit must be set to 0. This takes effect in the attached device only after the next read instruction is executed. Software must ensure that at least one read instruction is requested after resetting this bit before it can be sure XIP mode in the device is exited.

OSPI_CTL.XIPRD: If it is not known that the flash device enters XIP from POR, and XIP from POR may be supported by the attached flash device, software can attempt to exit the XIP mode by issuing an XIP exit command using a STIG command. For this, software must be aware of the mode bit requirements of that device, as XIP entry and exit changes per device.

XIP mode is supported in several flash devices. However, flash manufacturers do not have a consistent standard approach. Most of them use signature bits that are sent to the device immediately following the address bytes. Few of them (such as Micron devices) use signature bits and require a flash device configuration register write to enable XIP.

The following section describes how software ensures entry into XIP mode for the flash devices compliant with the OSPI controller.

Micron N25Q, MT25Q, and MT35X Devices

XIP mode must first be enabled by setting the corresponding field of VCR within the flash device. The VCR can be written to using the OSPI_FCCTL register to issue a VCR write command.

1. Disable the DAC using the OSPI_CTL.DACEN bit to ensure that no new AHB read accesses are sent to the flash device.
2. Use STIG mode to issue a VCR write to flash memory.
3. Set XIP mode bits in the OSPI_MBCTL.MODE field to 8'b00000000.
4. Enable the OSPI controller XIP mode by setting the OSPI_CTL.XIPRD bit.
5. Re-enable the DAC.

Micron (Supporting Basic XIP Mode), Winbond, Spansion Devices

1. Disable the DAC using the OSPI_CTL.DACEN bit to ensure that no new AHB read accesses are sent to the flash device.
2. Set XIP mode bits in the OSPI_MBCTL.MODE to:
   - 8'b10000000, for Micron devices supporting basic XIP mode
   - 8'b00100000, for Winbond devices
   - 8'b10100000, for Spansion devices
3. Enable XIP mode by setting the OSPI_CTL.XIPRD bit.
4. Reenable the DAC.

Using PHY Mode

The OSPI PHY mode extends the architecture of the OSPI flash controller to allow the interfacing of high-speed flash devices and ensure the reliable data transfers at high speed. For non-PHY operations, the controller assumes that OSPI transfer clock is at least 4 times lower than the reference clock to ensure correct data transactions in SDR mode and 8 times lower for DDR transactions. This provides a comfortable regulation of data synchronization, but it is not effective from a dynamic power efficiency standpoint.

The PHY mode allows transfers at speeds that match the increasing reference clocks of emerging higher performance flash devices, including frequencies close to 200 MHz. The PHY module works with the OSPI flash controller to enable DDR or SDR transfers at the device frequencies. For example, when a given flash device operates at 200 MHz, the reference clock is also 200 MHz, eliminating the need for high speed peripheral clocks.

The default reference clock (OSPI_REFLK) for OSPI is SYSCLK, but to enable support for PHY mode the reference clock is selected from one of the options from CDU output clock 10 (Refer to the CDU chapter). By default, it is connected to SYSCLK which is suitable for non-PHY mode of operation. For PHY mode, the reference clock (OSPI_REFLK) is either SCLK0_0 or SCLK1_1.

To configure PHY mode, first implement the procedure described in the Configuring OSPI for Optimal Use section and then proceed with the following steps based on the DLL mode configuration:

DLL Bypass Mode

1. Enable the PHY mode by setting the `OSPI_CTL.PHYEN` bit and enable the DDR protocol with the `OSPI_CTL.DTREN` bit.

   NOTE: Ensure that the device is configured to work in DDR Protocol (do not confuse with DDR commands).

2. Calibrate the software using the following sub steps:

   a. Calculate how many delay elements are necessary to shift the reference clock period by 25% and program this value in the `OSPI_PHYCTL.TXDLY` bit field. This is the best case for DDR transfers from the set-up/hold timing standpoint. The delay may vary slightly in a real design.

   b. Resynchronize the DLLs by setting the `OSPI_PHYCTL.RESYNC` bit. If this has already been set by a previous resynchronization, toggle the bit to 0 and then back to 1 to trigger the re-synchronization DLL logic.

   c. Enable the PHY bypass mode by setting the `OSPI_PHYMCTL.BYPCTL` bit.

   d. Trigger a read request from a location where the value is predictable. Depending on the device, this is the parameter page, ID, status, data from the OTP region, or data from the location of the flash array.

   e. Check for data correctness by incrementing the value of the RX clock delay using the `OSPI_PHYCTL.RXDLY` bit field, resynchronizing the DLLs, triggering a valid read request, and then checking for the correct data and store information. Continue to check until the range boundary of the RX clock delay is achieved and then proceed to the next step.

3. Set the RX clock delay value with one from the middle of valid range based on the information in storage.
4. Resynchronize the DLLs
5. Set the device read instruction register for Octal Read DDR Configuration. Configure each transfer phase to work in octal mode and set the number of dummy cycles to a minimum of the number specified in the documentation of the device. More dummy cycles may be set to accommodate additional read path delays in actual systems data.
6. Enable pipeline mode by setting the `OSPI_CTL.PIPEPHYEN` bit.
7. Perform a sequential read of data consistent with the conditions indicated in the PHY Pipeline Mode section.
8. Poll for IDLE with the `OSPI_CTL.IDLE` bit and when it is asserted to high trigger the next transfer request.

DLL Controller Mode

1. Enable the PHY mode by setting the `OSPI_CTL.PHYEN` bit and enable the DDR protocol with the `OSPI_CTL.DTREN` bit.

NOTE: Ensure that the device is configured to work in DDR protocol (do not confuse with DDR commands).

2. Calibrate the software using the following sub steps:
   a. Drive the OSPI_CTL.RST bit low to reset the DLL.
   b. Calculate initial delay value for the controller DLL according to the PHY DLL controller control register bits[6:0] and write this value into the OSPI_PHYMCTL.INITDLY bit field.
   c. Set DLL reset bit (OSPI_CTL.RST) back to high. The controller delay line automatically starts the locking procedure with the configuration given once the DLL module is in reset state.
   d. Poll the OSPI_DLLOB_LWR.LOCK bit. When it is set, the lock is done.
   e. Resynchronize the DLLs by setting the OSPI_PHYCTL.RESYNC bit. If this has already been set by a previous resynchronization, toggle the bit to 0 and then back to 1 to trigger the re-synchronization DLL logic.
   f. Set the TX DLL delay bit field (OSPI_PHYCTL.TXDLY) and the RX DLL delay bit field (OSPI_PHYCTL.RXDLY) to the current percentage clock offsets. Wait 20 reference clock cycles for the new configuration to propagate before triggering the next SPI transfer.
   g. Trigger a read request from a location where the value is predictable. Depending on the device, this is the parameter page, ID, status, data from the OTP region, or data from the location of the flash array.
   h. Check for data correctness by incrementing the value of the RX clock delay using the OSPI_PHYCTL.RXDLY bit field, resynchronizing the DLLs, triggering a valid read request, and then checking for the correct data and store information. Continue to check until the range boundary of the RX clock delay is achieved and then proceed to the next step.
3. Set the RX clock delay value with one from the middle of valid range based on the information in storage.
4. Resynchronize the DLLs
5. Set the device read instruction register for octal read DDR configuration. Configure each transfer phase to work in octal mode and set the number of dummy cycles to a minimum of the number specified in the documentation of the device. More dummy cycles may be set to accommodate additional read path delays in actual systems data.
6. Enable pipeline mode by setting the OSPI_CTL.PIPEPHYEN bit.
7. Perform a sequential read of data consistent with the conditions indicated in the PHY pipeline Mode section.
8. Poll for IDLE with the OSPI_CTL.IDLE bit and when it is asserted to high trigger the next transfer request.

ADSP-SC59x OSPI Register Descriptions

Octal SPI (OSPI) contains the following registers.

Table 23-5: ADSP-SC59x OSPI Register List

Name	Description
OSPI_CTL	Octal SPI Control Register
OSPI_DLY	Device Delay Register
OSPI_DRICTL	Device Read Instruction Control Register
OSPI_DWICTL	Device Write Instruction Control Register
OSPI_DSCTL	Device Size Control Register
OSPI_DLLOB_LWR	DLL Observable Register (Lower)
OSPI_DLLOB_UP	DLL Observable Register (Upper)
OSPI_FCA	Flash Command Address Register
OSPI_FCCTL	Flash Command Control Register
OSPI_FCMCTL	Flash Command Control Memory Register
OSPI_FCRD_LWR	Flash Command Read Data Register (Lower)
OSPI_FCRD_UP	Flash Command Read Data Register (Upper)
OSPI_FCWD_LWR	Flash Command Write Data Register (Lower)
OSPI_FCWD_UP	Flash Command Write Data Register (Upper)
OSPI_IMSK	Interrupt Mask Register
OSPI_ISTAT	Interrupt Status Register
OSPI_WRPROT_LWR	Lower Write Protection Register
OSPI_MBCTL	Mode Bit Control Register
OSPI_MODID	Module ID Register
OSPI_POLLEXP	Polling Expiration Register
OSPI_OE_LWR	Opcode Extension Register (Lower)
OSPI_OE_UP	Opcode Extension Register (Upper)
OSPI_PHYCTL	PHY Control Register
OSPI_PHYMCTL	PHY DLL Master Control Register
OSPI_POLSTAT	Polling Flash Status Register
OSPI_RDC	Read Data Capture Register
OSPI_REMAPADDR	Remap Address Register
OSPI_WRPROT_UP	Upper Write Protection Register
OSPI_WRPROT_CTL	Write Protection Control Register
OSPI_WCCTL	Write Completion Control Register

Octal SPI Control Register

The OSPI_CTL register contains basic configuration fields of the controller.

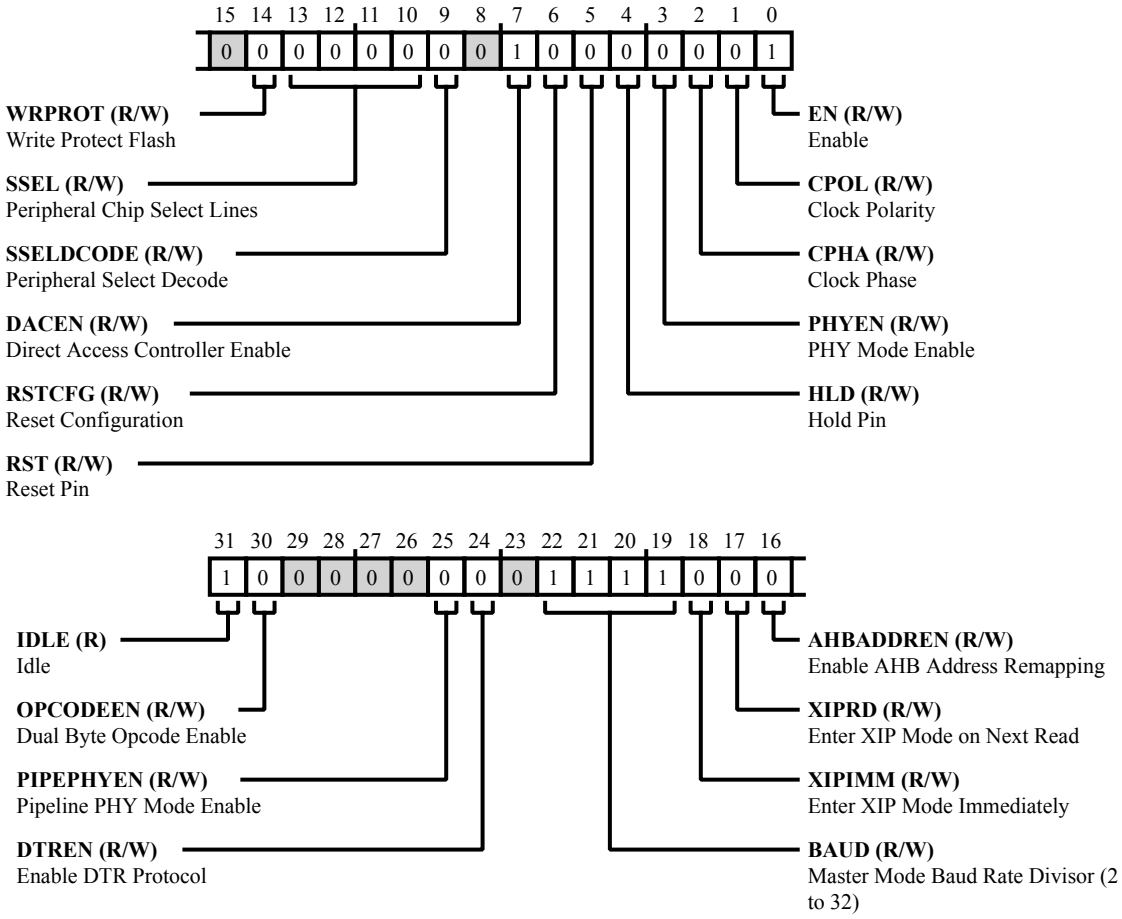


Figure 23-3: OSPI_CTL Register Diagram

Table 23-6: OSPI_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	IDLE	Idle. The OSPI_CTL.IDLE bit indicates an idle status. The serial interface and low level SPI pipeline are idle. This is a status read-only bit. Note: This is a re-timed signal. There is an inherent delay in generating this signal.

Table 23-6: OSPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/W)	OPCODEEN	Dual Byte Opcode Enable. The OSPI_CTL.OPCODEEN bit is set (=1) when the target Flash device supports dual byte opcode (Macronix MX25). It is applicable for Octal I/O mode or protocol only. It should be be cleared (=0) when the device is configured to work in another SPI mode. If enabled, the supplementing bytes are taken from the OSPI_OE_LWR and OSPI_OE_UP registers.
25 (R/W)	PIPEPHYEN	Pipeline PHY Mode Enable. The OSPI_CTL.PIPEPHYEN bit is relevant only for configuration with PHY module. It is asserted to 1 between consecutive PHY pipeline reads transfers and deasserted to 0 otherwise.
24 (R/W)	DTREN	Enable DTR Protocol. The OSPI_CTL.DTREN bit enables the DTR protocol. It must be set if the device is configured to work in DTR protocol.
22:19 (R/W)	BAUD	Master Mode Baud Rate Divisor (2 to 32). The OSPI_CTL.BAUD bit field selects the baud rate divisor. SPI Baud Rate = (Master Reference Clock)/ Baud Rate Divisor where, Baud Rate Divisor is: 4'b0000 = /2 4'b0001 = /4 4'b0010 = /6 4'b0011 = /8 4'b0100 = /10 ... 4'b1111 = /32

Table 23-6: OSPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	XIPIMM	Enter XIP Mode Immediately. When XIP is enabled, clearing the `OSPI_CTL.XIPIMM` bit (= 0) causes the controller to exit XIP mode on the next read instruction. When the `OSPI_CTL.XIPIMM` bit is set (= 1), operate the device in XIP mode immediately. Use this register when the external device wakes up in XIP mode (as per the contents of its non-volatile configuration register). The controller assumes the next read instruction is passed to the device as an XIP instruction, and therefore does not require the read opcode to be transferred. Note: To exit XIP mode, the `OSPI_CTL.XIPIMM` bit should be cleared (=0). This takes effect in the attached device only after the next read instruction is executed. Software must ensure that at least one read instruction is requested after resetting the `OSPI_CTL.XIPIMM` bit (to ensure that XIP mode is exited).	
		0	If XIP is enabled, it causes the controller to exit XIP mode on the next read instruction
		1	Operate the device in XIP mode immediately
17 (R/W)	XIPRD	Enter XIP Mode on Next Read. When XIP is enabled, clearing the `OSPI_CTL.XIPRD` bit (= 0) causes the controller to exit XIP mode on the next read instruction. When XIP is disabled, setting the `OSPI_CTL.XIPRD` bit (= 1) informs the controller that the device is ready to enter XIP on the next read instruction. The controller sends the appropriate command sequence, including mode bits to cause the device to enter XIP mode. Use this register after the controller ensures that the Flash device is configured to enter XIP mode. Note: To exit XIP mode, the `OSPI_CTL.XIPRD` bit should be cleared (= 0). This takes effect in the attached device only after the next read instruction is executed. Software must ensure that at least one read instruction is requested after resetting the `OSPI_CTL.XIPRD` bit (to ensure that XIP mode is exited).	
		0	If XIP is enabled, it cause the controller to exit XIP mode on the next read instruction
		1	If XIP is disabled, this setting informs the controller that the device is ready to enter XIP on the next read instruction
16 (R/W)	AHBADDREN	Enable AHB Address Remapping. The `OSPI_CTL.AHBADDREN` bit enables the bus address remapping (direct access mode only). When set (=1), the incoming bus address is adapted and sent to the Flash device as (address + N), where N is the value stored in the remap address register.	

Table 23-6: OSPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	WRPROT	Write Protect Flash. The `OSPI_CTL.WRPROT` bit, when set (=1), drives the write protect pin of the Flash. This is resynchronized to the generated memory clock, as necessary.	
13:10 (R/W)	SSEL	Peripheral Chip Select Lines. The `OSPI_CTL.SSEL` bit field indicates the peripheral chip select line. If `OSPI_CTL.SSELDCODE` = 0, ss[3:0] is output: ss[3:0] ..... SSEL[3:0] 4'bxxx0.....1110.....SSEL0 selected 4'bxx01.....1101.....SSEL1 selected 4'bx011.....1011.....SSEL2 selected 4'b0111.....0111.....SSEL3 selected 4'b1111.....1111.....No peripheral selected If `OSPI_CTL.SSELDCODE` = 1, ss[3:0] directly drives SSEL[3:0].	
9 (R/W)	SSELDCODE	Peripheral Select Decode. The `OSPI_CTL.SSELDCODE` bit indicates the peripheral select decode.	
		0	Only one of the 4 selects n_ss_out [3:0] is active
		1	Allows external 4:16 decode (n_ss_out = ss)
7 (R/W)	DACEN	Direct Access Controller Enable. The `OSPI_CTL.DACEN` bit enables the direct access controller. When the direct access controller and indirect access controller are disabled, all bus requests are completed with an error response.	
		0	Disable Direct Access Controller once current transfer of the data word is complete
		1	Enable Direct Access Controller
6 (R/W)	RSTCFG	Reset Configuration. The `OSPI_CTL.RSTCFG` bit indicates the reset pin configuration.	
		0	Reset Feature on DQ3 Pin of the Device
		1	Reset Feature on Dedicated Pin of the Device (Controlling the 5th bit Influences the Output)
5 (R/W)	RST	Reset Pin. The `OSPI_CTL.RST` bit, when set (=1), drives the reset pin of the Flash. When cleared (=0), it deactivates the reset pin.	

Table 23-6: OSPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	HLD	Hold Pin. The `OSPI_CTL.HLD` bit, when set (=1), drives the hold pin of the Flash. When cleared (=0), it deactivates the hold pin.	
3 (R/W)	PHYEN	PHY Mode Enable. When the `OSPI_CTL.PHYEN` bit is enabled, the controller is informed that PHY module is to be used for handling SPI transfers. This bit is relevant only for configuration with PHY module.	
2 (R/W)	CPHA	Clock Phase. The `OSPI_CTL.CPHA` bit selects the clock phase.	
		0	SPI clock is active outside the word
		1	SPI clock is inactive outside the word
1 (R/W)	CPOL	Clock Polarity. The `OSPI_CTL.CPOL` bit selects clock polarity outside SPI word.	
		0	OSPI clock is quiescent low
		1	OSPI clock is quiescent high
0 (R/W)	EN	Enable. The `OSPI_CTL.EN` bit enables OSPI operation.	
		0	Disable OSPI Module
		1	Enable OSPI Module

Device Delay Register

The OSPI_DLY register introduces relative delays in the generation of the master output signals. All timings are defined in cycles of the SPI REFERENCE CLOCK/ext_clk.

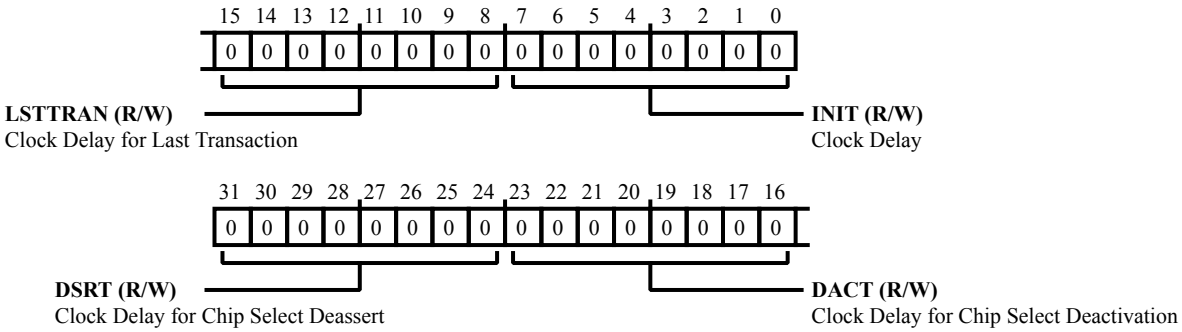


Figure 23-4: OSPI_DLY Register Diagram

Table 23-7: OSPI_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	DSRT	Clock Delay for Chip Select Deassert. The OSPI_DLY.DSRT field indicates the delay in the master reference clocks for the length that the master mode chip select outputs are deasserted between transactions. The minimum delay is always an SCLK period to ensure the chip select is not asserted again within an SCLK period.
23:16 (R/W)	DACT	Clock Delay for Chip Select Deactivation. The OSPI_DLY.DACT field indicates the delay in the master reference clocks between one chip select being deactivated and the activation of another. This is used to ensure a quiet period between the selection of two different slaves and requires the transmit FIFO to be empty.
15:8 (R/W)	LSTTRAN	Clock Delay for Last Transaction. The OSPI_DLY.LSTTRAN field indicates the delay in the master reference clocks between last bit of the current transaction and deasserting the device chip select (n_ss_out). By default, the chip select is deasserted on the cycle following the completion of the current transaction.
7:0 (R/W)	INIT	Clock Delay. The OSPI_DLY.INIT field indicates the delay in the master reference clocks between setting n_ss_out low and the first bit transfer.

Device Read Instruction Control Register

This register defines the configuration of Multiple-SPI READ instruction. This register should be setup while the controller is idle.

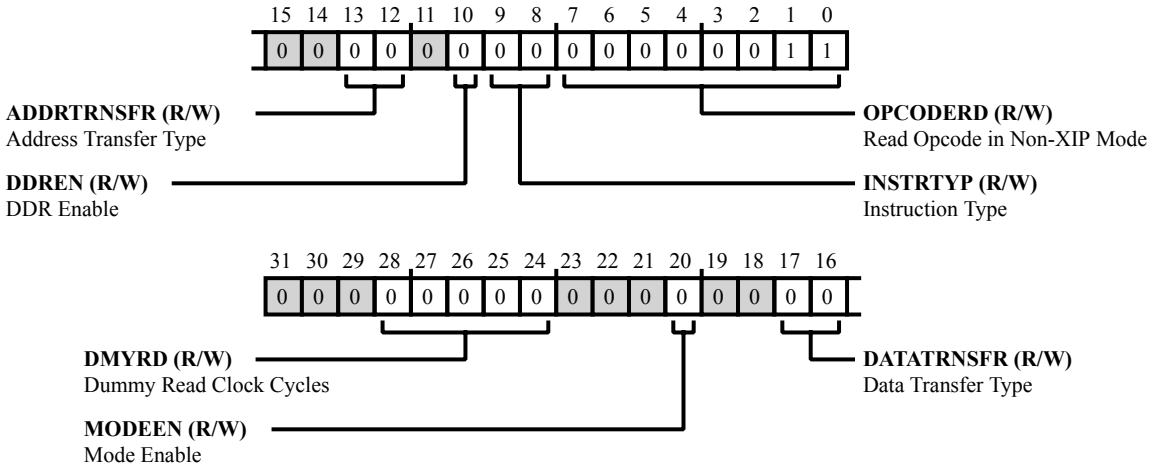


Figure 23-5: OSPI_DRICTL Register Diagram

Table 23-8: OSPI_DRICTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28:24 (R/W)	DMYRD	Dummy Read Clock Cycles. The `OSPI_DRICTL.DMYRD` field indicates the number of dummy clock cycles required by the device for a read instruction.	
20 (R/W)	MODEEN	Mode Enable. The `OSPI_DRICTL.MODEEN` field is set (=1) to ensure that the `OSPI_MBCTL.MODE` bits are sent following the address bytes.	
17:16 (R/W)	DATATRNSFR	Data Transfer Type. The `OSPI_DRICTL.DATATRNSFR` field indicates the data transfer type for standard SPI modes.	
		0	SIO Mode. Data is shifted to the device on DQ0 only and from the device on DQ1 only.
		1	Used for Dual Input/Output instructions. For data transfers, DQ0 and DQ1 are used as both inputs and outputs.
		2	Used for Quad Input/Output instructions. For data transfers, DQ0, DQ1, DQ2, and DQ3 are used as both inputs and outputs.
		3	Used for Quad Input/Output instructions. For data transfers, DQ[7:0] are used as both inputs and outputs.

Table 23-8: OSPI_DRICTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13:12 (R/W)	ADDRTRNSFR	Address Transfer Type. The `OSPI_DRICTL.ADDRTRNSFR` field indicates the address transfer type for standard SPI modes.	
		0	Addresses can be shifted to the device on DQ0 only.
		1	Addresses can be shifted to the device on DQ0 and DQ1 only.
		2	Addresses can be shifted to the device on DQ0, DQ1, DQ2, and DQ3.
		3	Addresses can be shifted to the device on DQ [7:0]
10 (R/W)	DDREN	DDR Enable. The `OSPI_DRICTL.DDREN` field indicates whether the opcode from the `OSPI_DRICTL.OPCODERD` field is compliant with one of the DDR read commands. Set the `OSPI_DRICTL.DDREN` field to 1 when opcode from bits 7 to 0 is compliant with DDR command. Master output data is issued by the controller in DDR fashion starting of the address SPI transfer phase. It is not applicable for STIG Mode.	
9:8 (R/W)	INSTRTYP	Instruction Type. The `OSPI_DRICTL.INSTRTYP` field indicates the instruction type.	
		0	Use standard SPI mode. Data is shifted to the device on DQ0 only and from the device on DQ1 only.
		1	Use DIO-SPI mode. Instruction sent on DQ0 and DQ1.
		2	Use QIO-SPI mode. Instruction sent on DQ0, DQ1, DQ2, and DQ3.
		3	Use OIO-SPI mode. Instruction sent on DQ[7:0].
7:0 (R/W)	OPCODERD	Read Opcode in Non-XIP Mode. The `OSPI_DRICTL.OPCODERD` field indicates the read opcode to use when not in XIP mode.	

Device Write Instruction Control Register

This register defines the configuration of Multiple-SPI WRITE (Program Page) instruction. This register should be setup while the controller is idle.

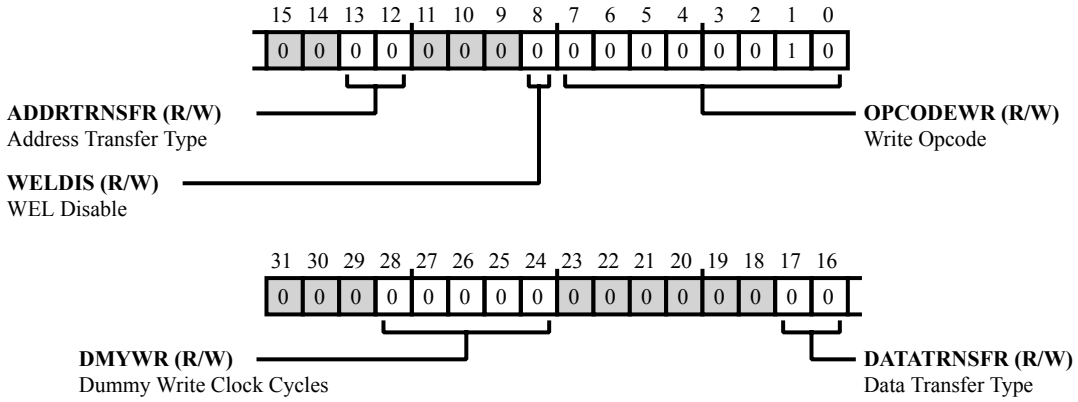


Figure 23-6: OSPI_DWICTL Register Diagram

Table 23-9: OSPI_DWICTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28:24 (R/W)	DMYWR	Dummy Write Clock Cycles. The `OSPI_DWICTL.DMYWR` field indicates the number of dummy clock cycles required by the device for write instruction.	
17:16 (R/W)	DATATRNSFR	Data Transfer Type. The `OSPI_DWICTL.DATATRNSFR` field indicates the data transfer type for standard SPI modes.	
		0	SIO Mode. Data is shifted to the device on DQ0 only and from the device on DQ1 only.
		1	Used for Dual Input/Output instructions. For data transfers, DQ0 and DQ1 are used as both inputs and outputs.
		2	Used for Octal Input/Output Instructions. For data transfers, DQ0, DQ1, DQ2, and DQ3 are used as both inputs and outputs.
		3	Used for Octal Input/Output instructions. For data transfers, DQ[7:0] are used as both inputs and outputs.

Table 23-9: OSPI_DWICTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13:12 (R/W)	ADDRTRNSFR	Address Transfer Type. The `OSPI_DWICTL.ADDRTRNSFR` field indicates the address transfer type for standard SPI modes.	
		0	Addresses can be shifted to the device on DQ0 only.
		1	Addresses can be shifted to the device on DQ0 and DQ1.
		2	Addresses can be shifted to the device on DQ0, DQ1, DQ2, and DQ3.
		3	Addresses can be shifted to the device on DQ [7:0].
8 (R/W)	WELDIS	WEL Disable. The `OSPI_DWICTL.WELDIS` bit disables the automatic issuing of the WEL command before a write operation for DAC or INDAC.	
7:0 (R/W)	OPCODEWR	Write Opcode. Write Opcode	

Device Size Control Register

This register allows the user to define the memory organization of using Flash Devices. This register should be setup while the controller is idle.

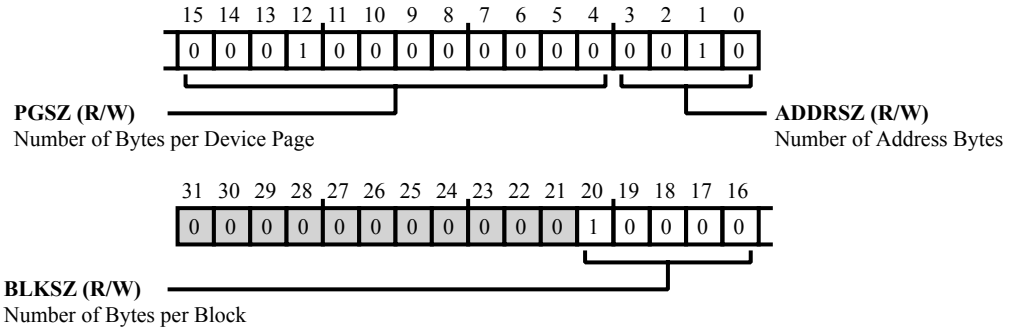


Figure 23-7: OSPI_DSCTL Register Diagram

Table 23-10: OSPI_DSCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:16 (R/W)	BLKSZ	Number of Bytes per Block. The controller uses the `OSPI_DSCTL.BLKSZ` field to perform write protection logic. The number of bytes per block must be a power of 2. 0: 1 byte 1: 2 bytes 3: 8 bytes ... 16: 65535 bytes and so on
15:4 (R/W)	PGSZ	Number of Bytes per Device Page. The controller uses the `OSPI_DSCTL.PGSZ` field to perform flash writes up to and across page boundaries.
3:0 (R/W)	ADDRSZ	Number of Address Bytes. The `OSPI_DSCTL.ADDRSZ` field indicates the address size for DAC mode. A value of 0 indicates 1 byte.

DLL Observable Register (Lower)

The OSPI_DLLOB_LWR register contains the DLL status. This register is synchronized into the APB domain by the octal-SPI flash controller and updates dynamically.

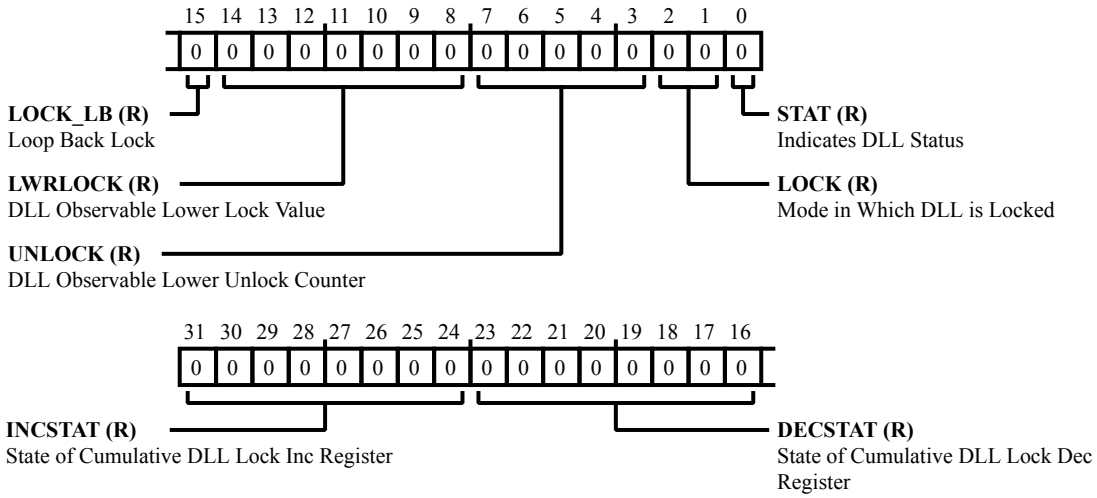


Figure 23-8: OSPI_DLLOB_LWR Register Diagram

Table 23-11: OSPI_DLLOB_LWR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	INCSTAT	State of Cumulative DLL Lock Inc Register. The OSPI_DLLOB_LWR.INCSTAT field indicates the state of the cumulative dll_lock_inc register.
23:16 (R/NW)	DECSTAT	State of Cumulative DLL Lock Dec Register. The OSPI_DLLOB_LWR.DECSTAT field indicates the state of the cumulative dll_lock_dec register.
15 (R/NW)	LOCK_LB	Loop Back Lock. The OSPI_DLLOB_LWR.LOCK_LB bit indicates that lock of loop back is done.
14:8 (R/NW)	LWRLOCK	DLL Observable Lower Lock Value. The OSPI_DLLOB_LWR.LWRLOCK field indicates the DLL encoder value from the master DLL to slave DLLs.
7:3 (R/NW)	UNLOCK	DLL Observable Lower Unlock Counter. The OSPI_DLLOB_LWR.UNLOCK field indicates the number of increments or decrements required for the master DLL to complete the locking process.
2:1 (R/NW)	LOCK	Mode in Which DLL is Locked. The OSPI_DLLOB_LWR.LOCK field indicates the mode in which the DLL has achieved the lock.

Table 23-11: OSPI_DLLOB_LWR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	STAT	Indicates DLL Status.

DLL Observable Register (Upper)

The OSPI_DLLOB_UP register contains the DLL status. This register is synchronized into the APB domain by the octal-SPI flash controller and updates dynamically.

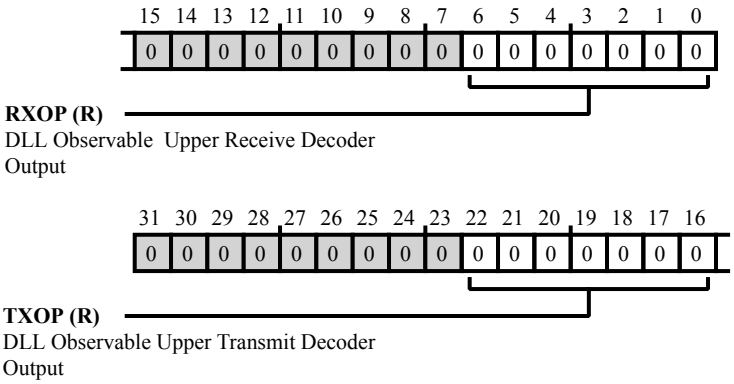


Figure 23-9: OSPI_DLLOB_UP Register Diagram

Table 23-12: OSPI_DLLOB_UP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
22:16 (R/NW)	TXOP	DLL Observable Upper Transmit Decoder Output. The OSPI_DLLOB_UP.TXOP field indicates the encoded value for the TX delay line for this slice.
6:0 (R/NW)	RXOP	DLL Observable Upper Receive Decoder Output. The OSPI_DLLOB_UP.RXOP field indicates the encoded value for the RX delay line for this slice.

Flash Command Address Register

This register allows the user to define the address of the command using by the STIG controller. This register should be setup while the controller is idle.

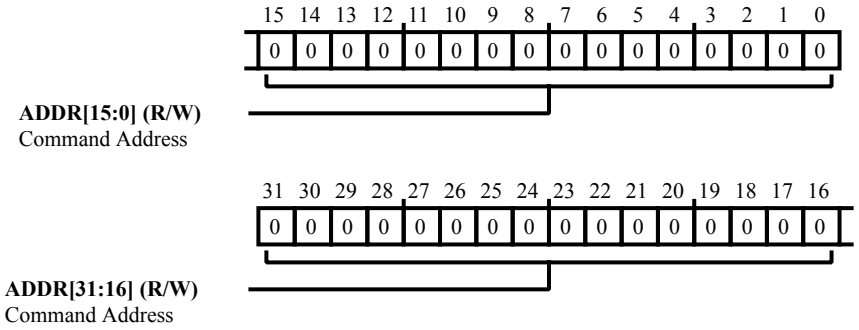


Figure 23-10: OSPI_FCA Register Diagram

Table 23-13: OSPI_FCA Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDR	Command Address. The `OSPI_FCA.ADDR` field indicates the address used by the command specified in the `OSPI_FCCTL.OPCODE` field. This must be setup before triggering the STIG command using the `OSPI_FCCTL.EXE` field.

Flash Command Control Register

This register controls SPI transactions generated by STIG. It allows the user to define corresponding SPI frame to particular command, triggering the transfer and polling for its completion.

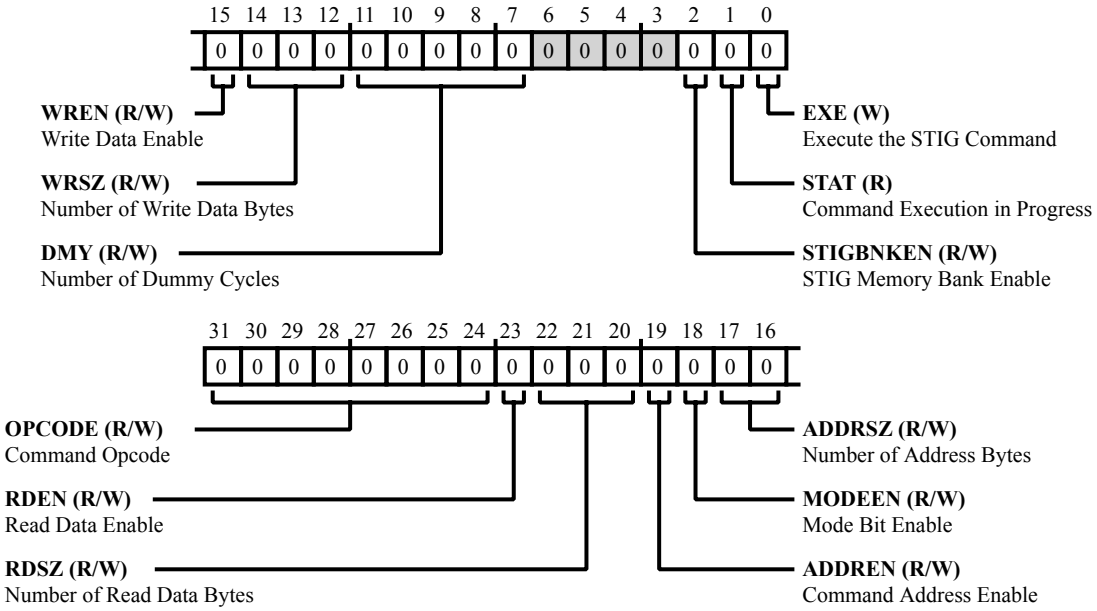


Figure 23-11: OSPI_FCCTL Register Diagram

Table 23-14: OSPI_FCCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	OPCODE	Command Opcode. The `OSPI_FCCTL.OPCODE` field must be configured before triggering the STIG command. For example, 0x20 maps to sub sector erase. Writing to the `OSPI_FCCTL.EXE` field launches the command. Note: Using this approach to issue commands to the device makes use of the instruction type of the `OSPI_DRICTL` and `OSPI_DWICTL` registers.
23 (R/W)	RDEN	Read Data Enable. When the `OSPI_FCCTL.RDEN` bit is set (=1), the command specified in the `OSPI_FCCTL.OPCODE` can read data bytes received from the device.
22:20 (R/W)	RDSZ	Number of Read Data Bytes. The `OSPI_FCCTL.RDSZ` field indicates the number of read data bytes; up to 8 data bytes can be read using this command. Configure the `OSPI_FCCTL.RDSZ` field to 0 for 1 byte and 7 for 8 bytes.

Table 23-14: OSPI_FCCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	ADDREN	Command Address Enable. When the OSPI_FCCTL.ADDREN bit is set (=1), the command specified in the OSPI_FCCTL.OPCODE field requires an address. This bit must be set before triggering the command via writing a 1 to the execute field.	
18 (R/W)	MODEEN	Mode Bit Enable. When the OSPI_FCCTL.MODEEN bit is set (=1), the OSPI_MBCTL.MODE field is sent following the address bytes.	
17:16 (R/W)	ADDRSZ	Number of Address Bytes. The OSPI_FCCTL.ADDRSZ field indicates the number of address bytes required (the address itself is programmed in the OSPI_FCA register. This field must be configured before triggering the command via bit 0 of this register.	
		0	1 address byte
		1	2 address bytes
		2	3 address bytes
		3	4 address bytes
15 (R/W)	WREN	Write Data Enable. When set (=1), the command specified in the OSPI_FCCTL.OPCODE field enables write data bytes to be sent to the device.	
14:12 (R/W)	WRSZ	Number of Write Data Bytes. The OSPI_FCCTL.WRSZ field indicates the number of write data bites; up to 8 data bytes may be written using this command. Configure to 0 for 1 byte, 7 for 8 bytes.	
11:7 (R/W)	DMY	Number of Dummy Cycles. The OSPI_FCCTL.DMY field indicates the number of dummy cycles required. This field should be configured before triggering the command via the OSPI_FCCTL.EXE field.	
2 (R/W)	STIGBNKEN	STIG Memory Bank Enable. The OSPI_FCCTL.STIGBNKEN bit must be set (=1) before triggering the flash memory bank transfer. It must be cleared (=0) before triggering any other operation. The OSPI_FCCTL.STIGBNKEN bit is set before triggering the command via OSPI_FCCTL.EXE field.	
1 (R/NW)	STAT	Command Execution in Progress.	
0 (RX/W)	EXE	Execute the STIG Command.	

Flash Command Control Memory Register

This register controls the Memory Bank accesses. It also defines the number of bytes intended to get by STIG access configured to use the STIG Memory Bank.

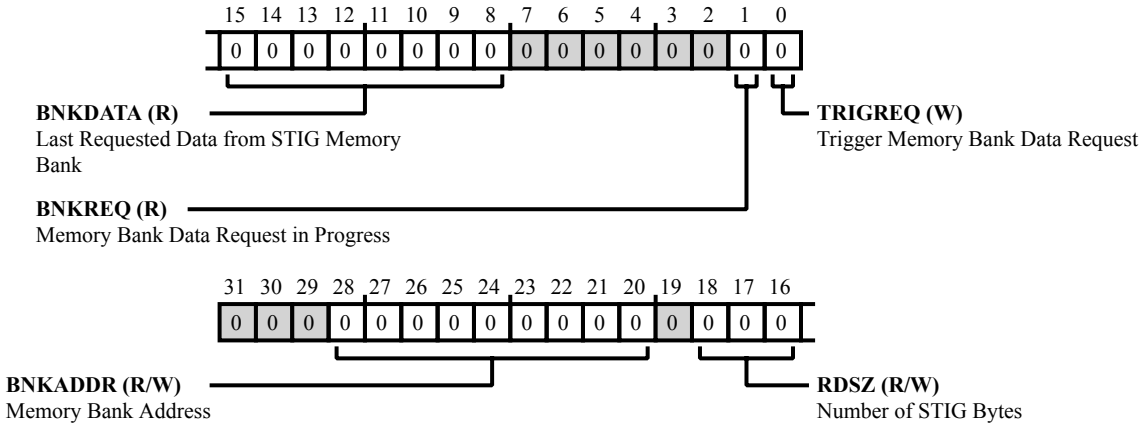


Figure 23-12: OSPI_FCMCTL Register Diagram

Table 23-15: OSPI_FCMCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:20 (R/W)	BNKADDR	Memory Bank Address. The `OSPI_FCMCTL.BNKADDR` field indicates the address of the memory bank from which data is read.
18:16 (R/W)	RDSZ	Number of STIG Bytes. The `OSPI_FCMCTL.RDSZ` field indicates the number of read bytes for the extended STIG.
15:8 (R/NW)	BNKDATA	Last Requested Data from STIG Memory Bank.
1 (R/NW)	BNKREQ	Memory Bank Data Request in Progress.
0 (RX/W)	TRIGREQ	Trigger Memory Bank Data Request. Trigger the Memory Bank data request. This bit is internally synchronized.

Flash Command Read Data Register (Lower)

This register keeps the last 4 bytes read by STIG SPI access.

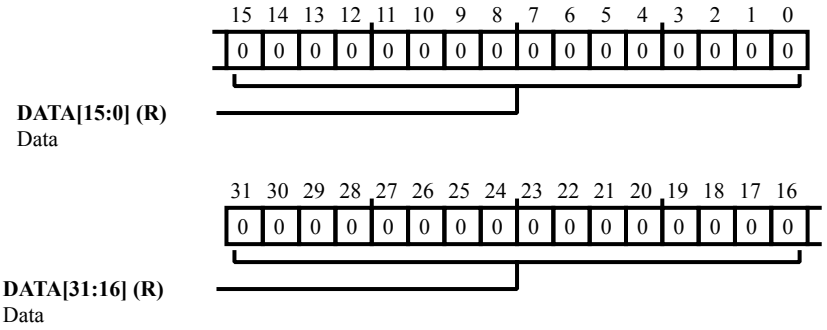


Figure 23-13: OSPI_FCRD_LWR Register Diagram

Table 23-16: OSPI_FCRD_LWR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Data. The `OSPI_FCRD_LWR.DATA` field contains data that is returned by the flash device for any status or configuration read operation carried out by triggering the STIG event in the `OSPI_FCCTL` register. The register is valid when the `OSPI_FCCTL.STAT` bit is low.

Flash Command Read Data Register (Upper)

This register keeps the last but 4 bytes read by STIG SPI access. This register in conjunction with the Flash Command Read Data Register (Lower) enables the controller to keep 8 last bytes read from the Flash Device using STIG.

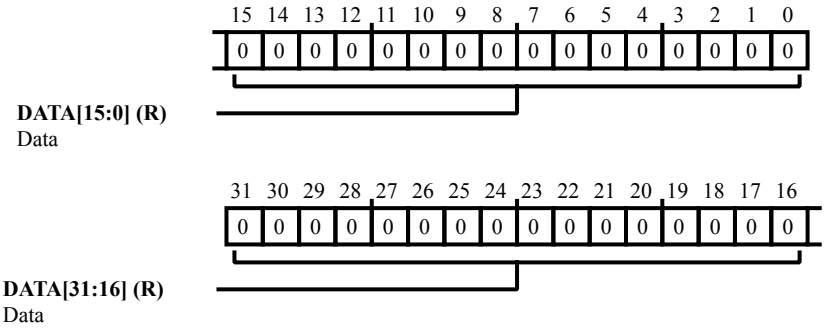


Figure 23-14: OSPI_FCRD_UP Register Diagram

Table 23-17: OSPI_FCRD_UP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Data. The `OSPI_FCRD_UP.DATA` field contains data that is returned by the flash device for any status or configuration read operation carried out by triggering the STIG event in the `OSPI_FCCTL` register. The register is valid when the `OSPI_FCCTL.STAT` bit is low.

Flash Command Write Data Register (Lower)

The OSPI_FCWD_LWR register takes the first 4 bytes to be written by STIG.

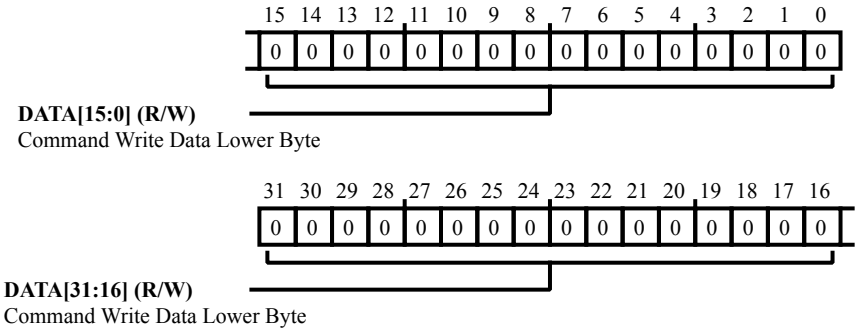


Figure 23-15: OSPI_FCWD_LWR Register Diagram

Table 23-18: OSPI_FCWD_LWR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	Command Write Data Lower Byte. The OSPI_FCWD_LWR.DATA field contains data written to the flash for any status or configuration write operation carried out by triggering the STIG event in the OSPI_FCCTL register. This field must be setup before triggering the command using the OSPI_FCCTL.EXE field.

Flash Command Write Data Register (Upper)

This register takes the bytes ranging from 5 to 8 to be written by STIG.

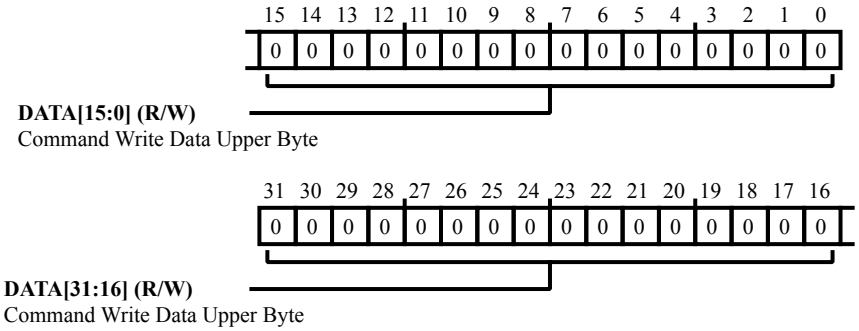


Figure 23-16: OSPI_FCWD_UP Register Diagram

Table 23-19: OSPI_FCWD_UP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	Command Write Data Upper Byte. The `OSPI_FCWD_UP.DATA` field contains data written to the flash for any status or configuration write operation carried out by triggering the event in the `OSPI_FCCTL` register. This field must be setup before triggering the command using the `OSPI_FCCTL.EXE` field.

Interrupt Mask Register

The OSPI_IMSK register allows the user to mask/unmask particular interrupt sources. This register must be setup when the controller is idle.

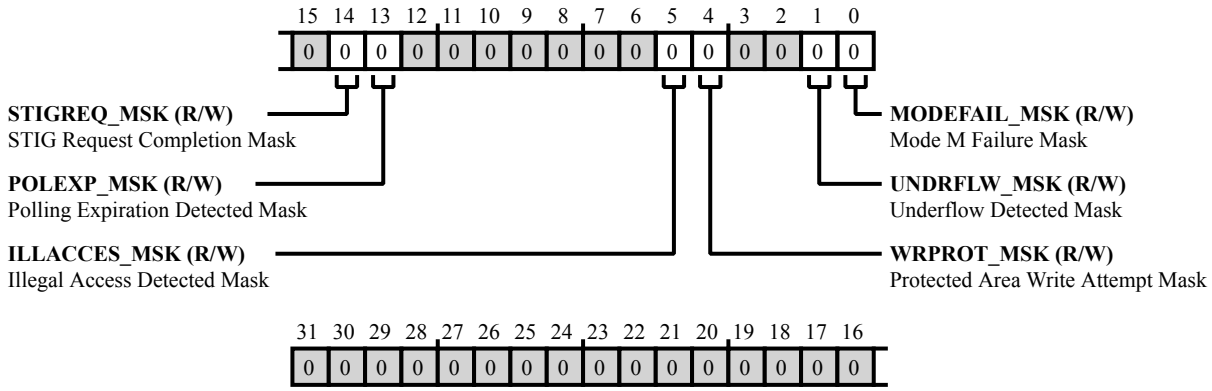


Figure 23-17: OSPI_IMSK Register Diagram

Table 23-20: OSPI_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W)	STIGREQ_MSK	STIG Request Completion Mask. 0 = the interrupt for the corresponding interrupt status register bit is disabled. 1 = the interrupt for the corresponding interrupt status register bit is enabled.
13 (R/W)	POLEXP_MSK	Polling Expiration Detected Mask. 0 = the interrupt for the corresponding interrupt status register bit is disabled. 1 = the interrupt for the corresponding interrupt status register bit is enabled.
5 (R/W)	ILLACCES_MSK	Illegal Access Detected Mask. 0 = the interrupt for the corresponding interrupt status register bit is disabled. 1 = the interrupt for the corresponding interrupt status register bit is enabled.
4 (R/W)	WRPROT_MSK	Protected Area Write Attempt Mask. 0 = the interrupt for the corresponding interrupt status register bit is disabled. 1 = the interrupt for the corresponding interrupt status register bit is enabled.
1 (R/W)	UNDRFLW_MSK	Underflow Detected Mask. 0 = the interrupt for the corresponding interrupt status register bit is disabled. 1 = the interrupt for the corresponding interrupt status register bit is enabled.
0 (R/W)	MODEFAIL_MSK	Mode M Failure Mask. 0 = the interrupt for the corresponding interrupt status register bit is disabled. 1 = the interrupt for the corresponding interrupt status register bit is enabled.

Interrupt Status Register

The status fields in the OSPI_ISTAT register are set when the described event occurs, and the interrupt is enabled in the OSPI_IMSK register. If any of these bit fields are set, the interrupt output is asserted high. The fields are each cleared by writing a 1 to the field.

Note: Bit fields [10:6] are valid only when legacy SPI mode is active.

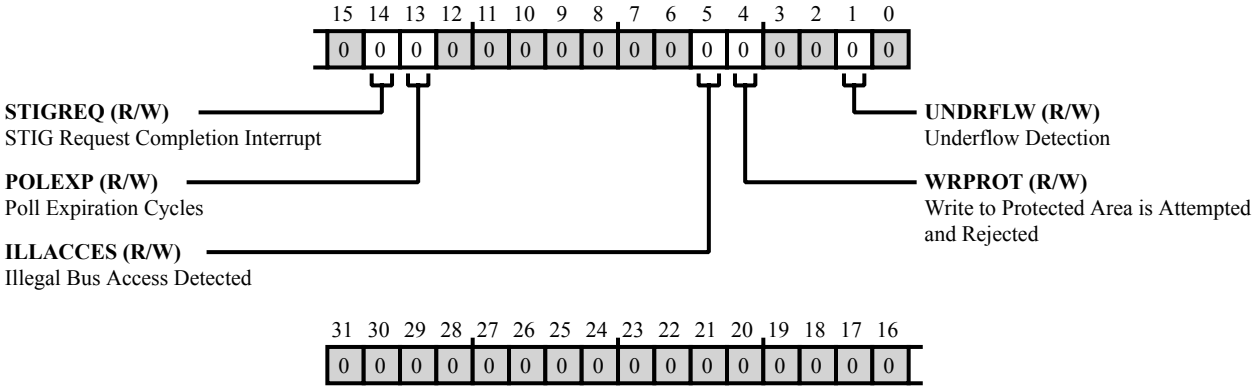


Figure 23-18: OSPI_ISTAT Register Diagram

Table 23-21: OSPI_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W)	STIGREQ	STIG Request Completion Interrupt. The OSPI_ISTAT.STIGREQ bit indicates that the controller is ready to get another STIG request.
13 (R/W)	POLEXP	Poll Expiration Cycles. The OSPI_ISTAT.POLEXP bit indicates that the maximum number of programmed polls cycles has expired.
5 (R/W)	ILLACCES	Illegal Bus Access Detected. The OSPI_ISTAT.ILLACCES bit indicates that an illegal bus access has been detected. Bus wrapping bursts and the use of SPLIT/RETRY accesses cause the error interrupt to trigger.
4 (R/W)	WRPROT	Write to Protected Area is Attempted and Rejected. Write to protected area was attempted and rejected.

Table 23-21: OSPI_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	UNDRFLW	Underflow Detection. The `OSPI_ISTAT.UNDRFLW` bit indicates that an underflow condition has been detected.	
		0	No underflow is detected.
		1	underflow is detected and an attempt to transfer data is made when the small TX FIFO is empty. This may occur when AHB write data is slowly provided to keep up with the requested write operation.

Lower Write Protection Register

The OSPI_WRPROT_LWR register allows the user to define lower boundary of the write protection area. This register should be setup while the controller is idle.

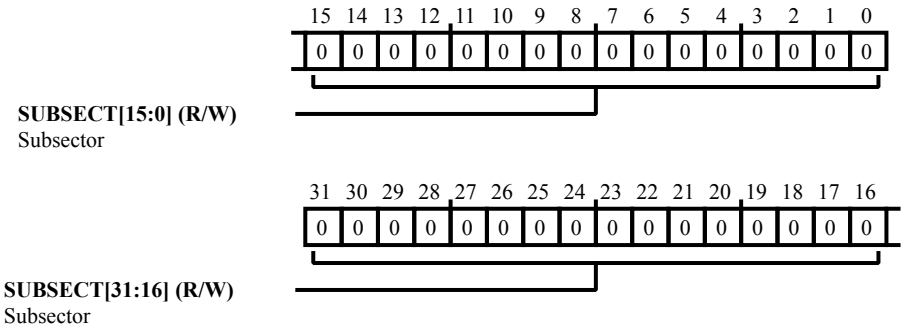


Figure 23-19: OSPI_WRPROT_LWR Register Diagram

Table 23-22: OSPI_WRPROT_LWR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	SUBSECT	Subsector. The OSPI_WRPROT_LWR.SUBSECT field defines the lower end of the address range to be protected. The address to be programmed must be OSPI memory map address (starting from 0x60000000) shifted by the value in the OSPI_DSCTL.BLKSZ. For example, if lower address of flash to be protected is 0x100 and the OSPI_DSCTL.BLKSZ= 4, the value to be programmed in this register must be 0x60000100 >> 4, which is 0x06000010.

Mode Bit Control Register

This register allows the user to define the mode bits for corresponding Flash Device. It also provides configuration for CRC aware SPI transfers. This register should be setup while the controller is idle.

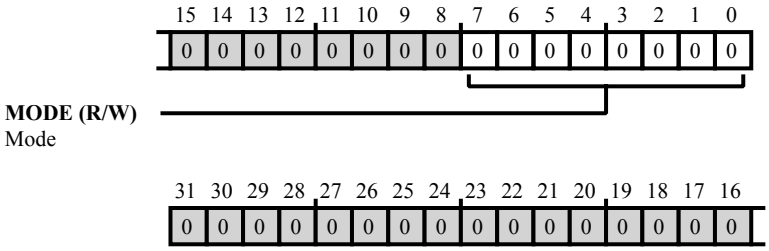


Figure 23-20: OSPI_MBCTL Register Diagram

Table 23-23: OSPI_MBCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MODE	Mode. The `OSPI_MBCTL.MODE` field is sent to the device following the address bytes when the mode bit transmission is enabled.

Module ID Register

This register provides the IP release number and the configuration data.

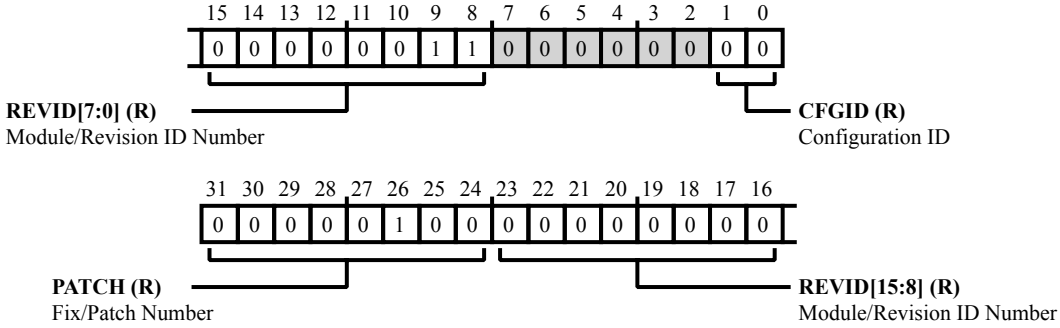


Figure 23-21: OSPI_MODID Register Diagram

Table 23-24: OSPI_MODID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	PATCH	Fix/Patch Number. The `OSPI_MODID.PATCH` field indicates the fix/patch number related to the `OSPI_MODID.REVID` field.
23:8 (R/NW)	REVID	Module/Revision ID Number. The `OSPI_MODID.REVID` field holds the module revision.
1:0 (R/NW)	CFGID	Configuration ID. The `OSPI_MODID.CFGID` field holds the configuration ID number.

Polling Expiration Register

This register defines maximum number of poll cycles. If the expected value of the bit being polled is not gotten after number defined in this register, the auto-polling is done on the next phase. The value stored in this register matters only if Write Completion Control Register bits[15] and Polling Flash Status Register bit[8] are set to "1". If auto-polling is not disabled by Write Completion Control Register bits[14], at least two auto-polling phases are queued before execution and popped after polled bit have taken expected value. This register should be setup while the controller is idle.

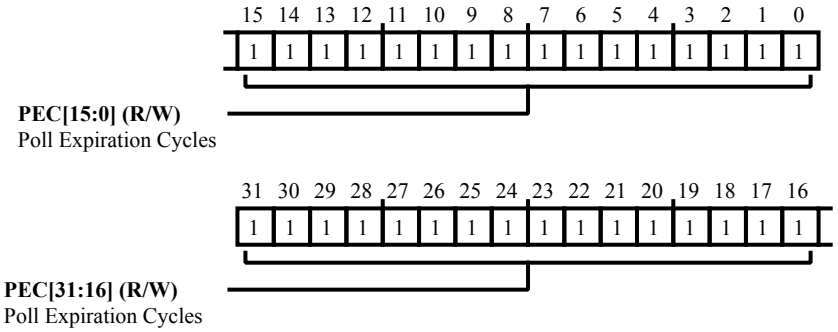


Figure 23-22: OSPI_POLLEXP Register Diagram

Table 23-25: OSPI_POLLEXP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PEC	Poll Expiration Cycles. The `OSPI_POLLEXP.PEC` field defines the number of polls cycles before expiration.

Opcode Extension Register (Lower)

This register provides the supplementing opcodes for Dual Byte Opcode Mode activated by Octal-SPI Configuration Register bit[30].

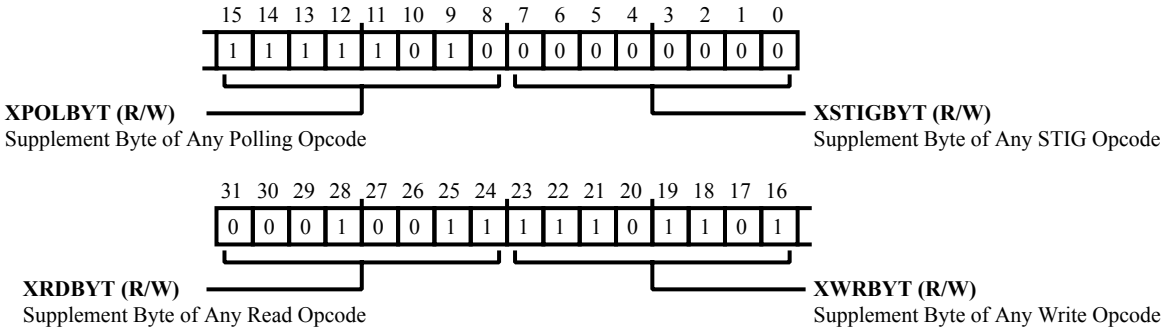


Figure 23-23: OSPI_OE_LWR Register Diagram

Table 23-26: OSPI_OE_LWR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	XRDBYT	Supplement Byte of Any Read Opcode.
23:16 (R/W)	XWRBYT	Supplement Byte of Any Write Opcode.
15:8 (R/W)	XPOLBYT	Supplement Byte of Any Polling Opcode.
7:0 (R/W)	XSTIGBYT	Supplement Byte of Any STIG Opcode.

Opcode Extension Register (Upper)

This register provides the supplementing opcodes for Dual Byte Opcode Mode activated by Octal-SPI Configuration Register bit[30]. Additionally, it allows the user to define the Write Enable Latch (WEL) command first byte opcode used for automatic WEL insertion before any AHB write access.

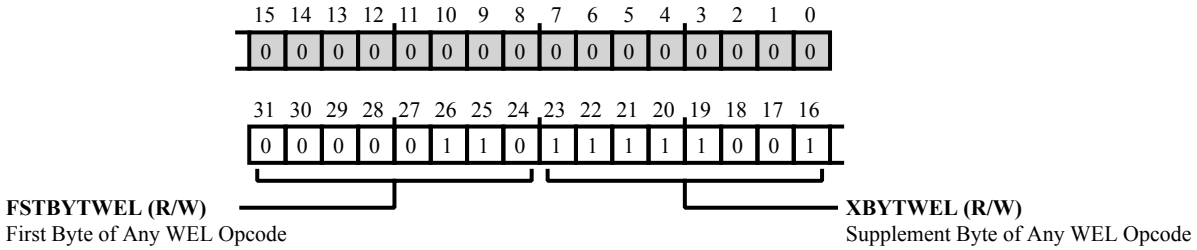


Figure 23-24: OSPI_OE_UP Register Diagram

Table 23-27: OSPI_OE_UP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	FSTBYTWEL	First Byte of Any WEL Opcode.
23:16 (R/W)	XBYTWEL	Supplement Byte of Any WEL Opcode.

PHY Control Register

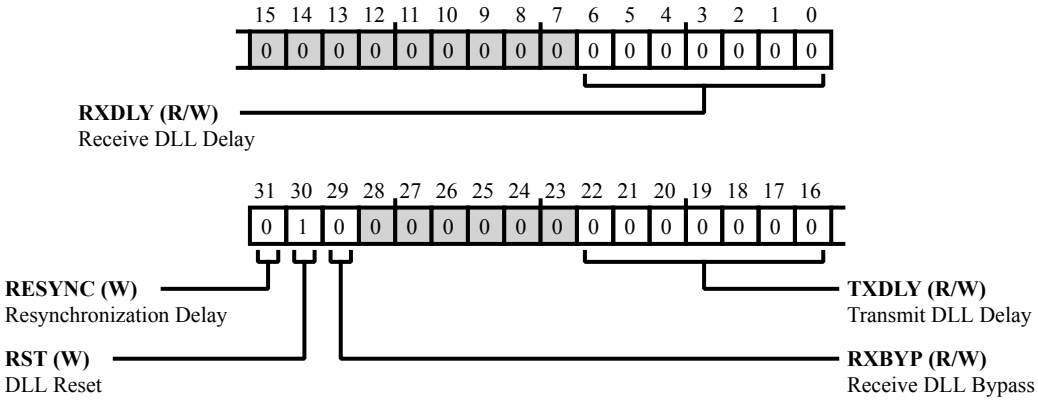


Figure 23-25: OSPI_PHYCTL Register Diagram

Table 23-28: OSPI_PHYCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (RX/W)	RESYNC	Resynchronization Delay. The `OSPI_PHYCTL.RESYNC` bit is used for resynchronization delay lines to update them with values from the `OSPI_PHYCTL.RXDLY` and `OSPI_PHYCTL.TXDLY` fields.
30 (RX/W)	RST	DLL Reset. The `OSPI_PHYCTL.RST` bit is used for reset of Delay Lines by software.
29 (R/W)	RXBYP	Receive DLL Bypass. The `OSPI_PHYCTL.RXBYP` field determines if RX DLL is bypassed.
22:16 (R/W)	TXDLY	Transmit DLL Delay. The `OSPI_PHYCTL.TXDLY` field determines the number of delay elements to insert on data path between ref_clk and spi_clk.
6:0 (R/W)	RXDLY	Receive DLL Delay. The `OSPI_PHYCTL.RXDLY` field determines the number of delay elements to insert on data path between ref_clk and rx_dll_clk.

PHY DLL Master Control Register

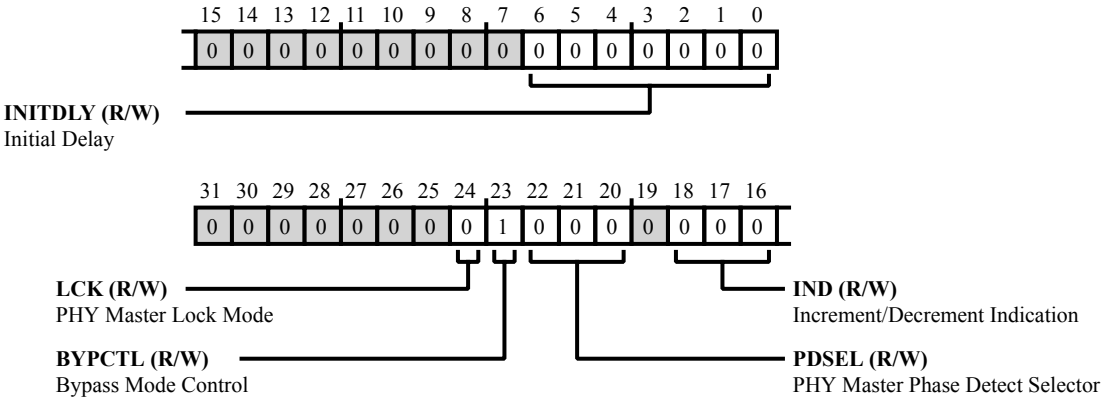


Figure 23-26: OSPI_PHYMCTL Register Diagram

Table 23-29: OSPI_PHYMCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/W)	LCK	PHY Master Lock Mode. The `OSPI_PHYMCTL.LCK` bit indicates whether the master delay line locks on a full cycle or half cycle of delay.
23 (R/W)	BYPCTL	Bypass Mode Control. The `OSPI_PHYMCTL.BYPCTL` bit indicates the bypass mode of the master and slave DLLs.
22:20 (R/W)	PDSEL	PHY Master Phase Detect Selector. The `OSPI_PHYMCTL.PDSEL` field selects the number of delay elements to be inserted between the phase detect flip-flops.
18:16 (R/W)	IND	Increment/Decrement Indication. The `OSPI_PHYMCTL.IND` field indicates the number of consecutive increment or decrement indications.
6:0 (R/W)	INITDLY	Initial Delay. The `OSPI_PHYMCTL.INITDLY` field is the initial delay value for the DLL.

Polling Flash Status Register

This register provides auto-polling data. It acts as the extension for the Write Completion Control Register where full status is not available and any action can be taken only relying on the indication of single bit being polled for.

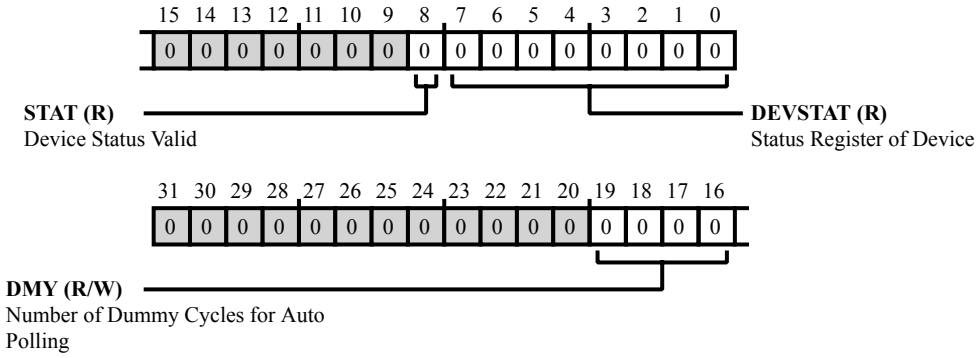


Figure 23-27: OSPI_POLSTAT Register Diagram

Table 23-30: OSPI_POLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:16 (R/W)	DMY	Number of Dummy Cycles for Auto Polling. The `OSPI_POLSTAT.DMY` field indicates user-defined additional dummy cycles for read status during the auto-polling state. It may be necessary to add some cycles if the external delay of read data path shifts it outside the defined clock cycle.
8 (R/NW)	STAT	Device Status Valid. The `OSPI_POLSTAT.STAT` bit is set (=1) when value in bits from 7 to 0 is valid.
7:0 (R/NW)	DEVSTAT	Status Register of Device. The `OSPI_POLSTAT.DEVSTAT` field indicates the actual status register of the device.

Read Data Capture Register

This register is used to adjust SPI transfer conditions in order to fetch and capture data reliably. This register should be setup while the controller is idle.

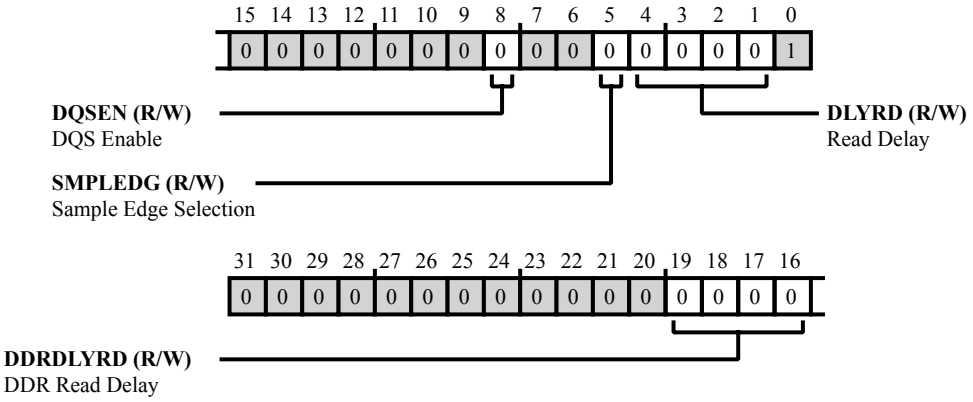


Figure 23-28: OSPI_RDC Register Diagram

Table 23-31: OSPI_RDC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/W)	DDRDLYRD	DDR Read Delay. The `OSPI_RDC.DDRDLYRD` field indicates the delay of the transmitted data by the number of ref_clk cycles. This field is relevant only when DDR read Command is executed. Otherwise, it can be ignored.	
8 (R/W)	DQSEN	DQS Enable. If the `OSPI_RDC.DQSEN` bit is set (=1), the signal from DQS input is driven into RX DLL and is used for data capturing in PHY mode instead of internally generated gated ref_clk.	
5 (R/W)	SMPLEDG	Sample Edge Selection. The `OSPI_RDC.SMPLEDG` bit selects the edge on which data outputs from flash memory is sampled.	
		0	Data outputs from Flash are sampled on falling edge of the ref_clk.
		1	Data outputs from Flash are sampled on rising edge of the ref_clk.
4:1 (R/W)	DLYRD	Read Delay. The `OSPI_RDC.DLYRD` field indicates the delay of the read data capturing logic by the programmed number of OSPI_REFCLK cycles.	

Remap Address Register

This register allows the user to define the address offset for DAC accesses. This register should be setup while the controller is idle.

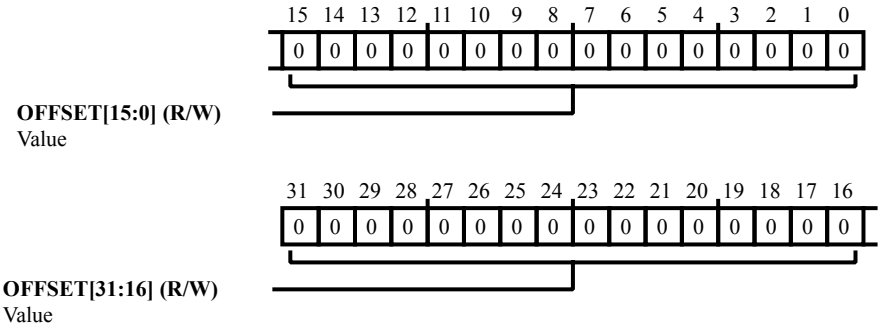


Figure 23-29: OSPI_REMAPADDR Register Diagram

Table 23-32: OSPI_REMAPADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	OFFSET	Value. The `OSPI_REMAPADDR.OFFSET` field is used to remap an incoming bus address to a different address used by the Flash.

Upper Write Protection Register

This register allows the user to define upper boundary of the write protection area. This register should be setup while the controller is idle.

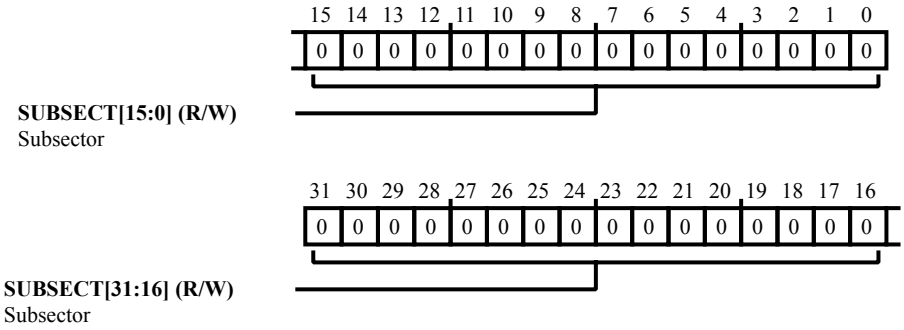


Figure 23-30: OSPI_WRPROT_UP Register Diagram

Table 23-33: OSPI_WRPROT_UP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	SUBSECT	Subsector. The `OSPI_WRPROT_UP.SUBSECT` field defines the upper end of the address range to be protected. The address to be programmed must be an OSPI memory map address (starting from 0x60000000) shifted by the value in the `OSPI_DSCTL.BLKSZ`. For example, if the upper address of flash to be protected is 0x100 and the `OSPI_DSCTL.BLKSZ`= 4, the value to be programmed in this register must be 0x60000100 >> 4, which is 0x06000010.

Write Protection Control Register

This register allows the user to define the configuration of write protection settings. This register should be setup while the controller is idle.

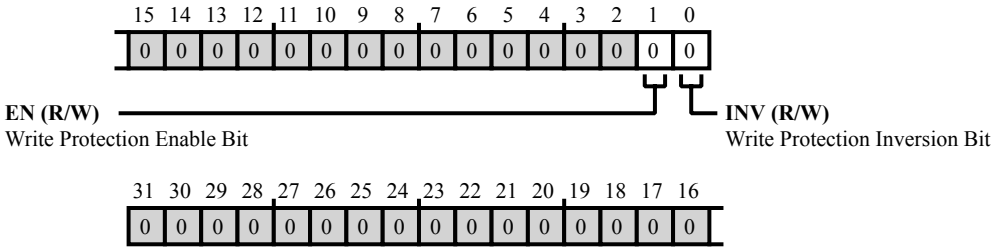


Figure 23-31: OSPI_WRPROT_CTL Register Diagram

Table 23-34: OSPI_WRPROT_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	EN	Write Protection Enable Bit. When set (=1), any AHB write access with an address within the protection region defined in the lower and upper write protection registers is rejected. An AHB error response is generated and an interrupt source triggered. When cleared (=0), the protection region is disabled.
0 (R/W)	INV	Write Protection Inversion Bit. When set (=1), the protection region defined in the lower and upper write protection registers is inverted meaning it is the region that the system is permitted to write to. When cleared (=0), the protection region defined in the lower and upper write protection registers is the region that the system is not permitted to write to.

Write Completion Control Register

The OSPI_WCCTL register defines how the controller polls the device following a write transfer.

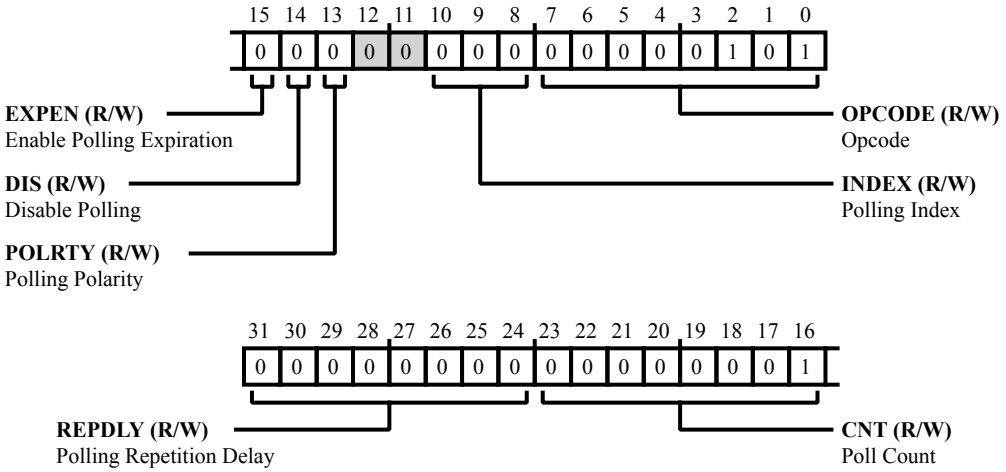


Figure 23-32: OSPI_WCCTL Register Diagram

Table 23-35: OSPI_WCCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	REPDLY	Polling Repetition Delay. The OSPI_WCCTL.REPDLY field defines the additional delay required to ensure that the Chip Select is deasserted during auto polling phase.	
23:16 (R/W)	CNT	Poll Count. The OSPI_WCCTL.CNT field defines the number of times the controller should expect to see a true result from polling in successive reads of the device register.	
15 (R/W)	EXPEN	Enable Polling Expiration. The OSPI_WCCTL.EXPEN bit is set (=1) to enable auto polling expiration.	
14 (R/W)	DIS	Disable Polling. The OSPI_WCCTL.DIS bit switches off automatic polling function.	
13 (R/W)	POLRTY	Polling Polarity. The OSPI_WCCTL.POLRTY bit defines the polling polarity.	
		0	Write transfer to the device will be complete if the polled bit is 0.
		1	Write transfer to the device will be complete if the polled bit is 1.

Table 23-35: OSPI_WCCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10:8 (R/W)	INDEX	Polling Index. The `OSPI_WCCTL.INDEX` field defines the bit index that must be polled. A value of 010 means that bit 2 of the returned data is polled for. A value of 111 means that bit 7 of the returned data is polled for.
7:0 (R/W)	OPCODE	Opcode. The `OSPI_WCCTL.OPCODE` field defines the opcode that must be issued by the controller when it is automatically polling for device program completion. This command is issued followed all device write operations. By default, this polls the standard device status register using opcode 0x05.

24 Universal Asynchronous Receiver/Transmitter (UART)

The UART module is a full-duplex peripheral compatible with PC-style industry-standard UARTs. The UART converts data between serial and parallel formats. The serial communication follows an asynchronous protocol that supports various word lengths, stop bits, bit rates, and parity-generation options. Multiple events can generate interrupts.

The UART is logically compliant to EIA-232E, EIA-422, EIA-485 and LIN standards, but usually requires external transceiver devices to meet electrical requirements. In IrDA (Infrared Data Association) mode, the UART meets the half-duplex IrDA SIR (9.6/115.2 Kbps rate) protocol. In multi-drop bus mode, the UART meets the full-duplex MDB/ICP v2.0 protocol.

The UART module supports partial modem status and control functionality to allow for hardware flow control.

The UART is a DMA-capable peripheral with separate transmit and receive DMA controller channels. The use of DMA requires minimal software intervention as the DMA engine moves the data. The UART can also use a programmed core mode of operation. The core mode requires software management of the data flow using either interrupts or polling.

The UART can use one of the peripheral timers for a hardware-assisted auto-baud detection mechanism. The timers are external to the UART.

NOTE: The UARTs (UART0, UART1, and UART2) operate in the SCLK0 domain.

UART Features

Each UART includes the following features.

- 5–8 data bits
- Programmable extra stop bit and programmable extra half-stop bit
- Even, odd, and sticky parity bit options
- Extra 8-stage receive FIFO with programmable threshold interrupt request
- Flexible transmit and receive interrupt request timing

- Three interrupt outputs for receive, transmit, and status
- Independent DMA operation for receive and transmit
- Programmable automatic request to send (RTS)/clear to send (CTS) hardware flow control
- False start bit detection
- SIR IrDA operation mode
- MDB/ICP v2.0 operation mode
- Internal loopback
- Improved bit rate granularity
- LIN break command/Inter-frame gap transmission support

Table 24-1: UART Specifications

Feature	Availability
Protocol	
Controller-Capable	Yes
Target-Capable	Yes
Transmission Simplex	Yes
Transmission Half-Duplex	Yes
Transmission Full-Duplex	Yes
Access Type	
Data Buffer	Yes
Core Data Access	Yes
DMA Data Access	Yes
DMA Channels	2 (per UART Port)
DMA Descriptor	Yes
Boot Capable	Yes (Target Mode)
Local Memory	No
Clock Operation	SCLK0/16

UART Functional Description

The following sections provide details on the UARTs functionality.

ADSP-SC59x UART Register List

The Universal Asynchronous Receiver/Transmitter module (UART) is a full-duplex peripheral compatible with PC-style industry-standard UARTs. The UART converts data between serial and parallel formats. The serial communication follows an asynchronous protocol that supports various word length, stop bit, parity, and interrupt generation options. A set of registers governs UART operations. For more information on UART functionality, see the UART register descriptions.

Table 24-2: ADSP-SC59x UART Register List

Name	Description
UART_CLK	Clock Rate Register
UART_CTL	Control Register
UART_IMSK	Interrupt Mask Register
UART_IMSK_CLR	Interrupt Mask Clear Register
UART_IMSK_SET	Interrupt Mask Set Register
UART_RBR	Receive Buffer Register
UART_RSR	Receive Shift Register
UART_RXCNT	Receive Counter Register
UART_SCR	Scratch Register
UART_STAT	Status Register
UART_TAIP	Transmit Address/Insert Pulse Register
UART_THR	Transmit Hold Register
UART_TSR	Transmit Shift Register
UART_TXCNT	Transmit Counter Register

ADSP-SC59x UART Interrupt List

Table 24-3: ADSP-SC59x UART Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
138	UART0_TXDMA	UART0 Transmit DMA	Level	20
139	UART0_RXDMA	UART0 Receive DMA	Level	21
140	UART0_STAT	UART0 Status	Level	
141	UART1_TXDMA	UART1 Transmit DMA	Level	34
142	UART1_RXDMA	UART1 Receive DMA	Level	35
143	UART1_STAT	UART1 Status	Level	
144	UART2_TXDMA	UART2 Transmit DMA	Level	37

Table 24-3: ADSP-SC59x UART Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
145	UART2_RXDMA	UART2 Receive DMA	Level	38
146	UART2_STAT	UART2 Status	Level	
147	UART3_TXDMA	UART3 Transmit DMA	Level	53
148	UART3_RXDMA	UART3 Receive DMA	Level	54
149	UART3_STAT	UART3 Status	Level	
269	UART0_TXDMA_ERR	UART0 Transmit DMA Error	Level	
270	UART0_RXDMA_ERR	UART0 Receive DMA Error	Level	
271	UART1_TXDMA_ERR	UART1 Transmit DMA Error	Level	
272	UART1_RXDMA_ERR	UART1 Receive DMA Error	Level	
273	UART2_TXDMA_ERR	UART2 Transmit DMA Error	Level	
274	UART2_RXDMA_ERR	UART2 Receive DMA Error	Level	
275	UART3_TXDMA_ERR	UART3 Transmit DMA Error	Level	
276	UART3_RXDMA_ERR	UART3 Receive DMA Error	Level	

ADSP-SC59x UART Trigger List

Table 24-4: ADSP-SC59x UART Trigger List Generators

Trigger ID	Name	Description	Sensitivity
155	UART0_TXDMA	UART0 Transmit DMA	Edge
156	UART0_RXDMA	UART0 Receive DMA	Edge
157	UART1_TXDMA	UART1 Transmit DMA	Edge
158	UART1_RXDMA	UART1 Receive DMA	Edge
159	UART2_TXDMA	UART2 Transmit DMA	Edge
160	UART2_RXDMA	UART2 Receive DMA	Edge
161	UART3_TXDMA	UART3 Transmit DMA	Edge
162	UART3_RXDMA	UART3 Receive DMA	Edge

Table 24-5: ADSP-SC59x UART Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
169	UART0_TXDMA	UART0 Transmit DMA	Pulse
170	UART0_RXDMA	UART0 Receive DMA	Pulse

Table 24-5: ADSP-SC59x UART Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
171	UART1_TXDMA	UART1 Transmit DMA	Pulse
172	UART1_RXDMA	UART1 Receive DMA	Pulse
173	UART2_TXDMA	UART2 Transmit DMA	Pulse
174	UART2_RXDMA	UART2 Receive DMA	Pulse
175	UART3_TXDMA	UART3 Transmit DMA	Pulse
176	UART3_RXDMA	UART3 Receive DMA	Pulse

ADSP-SC59x UART DMA Channel List

Table 24-6: ADSP-SC59x UART DMA Channel List

DMA ID	DMA Channel Name	Description
DMA20	UART0_TXDMA	UART0 Transmit DMA
DMA21	UART0_RXDMA	UART0 Receive DMA
DMA34	UART1_TXDMA	UART1 Transmit DMA
DMA35	UART1_RXDMA	UART1 Receive DMA
DMA37	UART2_TXDMA	UART2 Transmit DMA
DMA38	UART2_RXDMA	UART2 Receive DMA
DMA53	UART3_TXDMA	UART3 Transmit DMA
DMA54	UART3_RXDMA	UART3 Receive DMA

UART Block Diagram

The ***UART Block Diagram*** figure shows a simplified block diagram of one UART module and how it interconnects to the processor system.

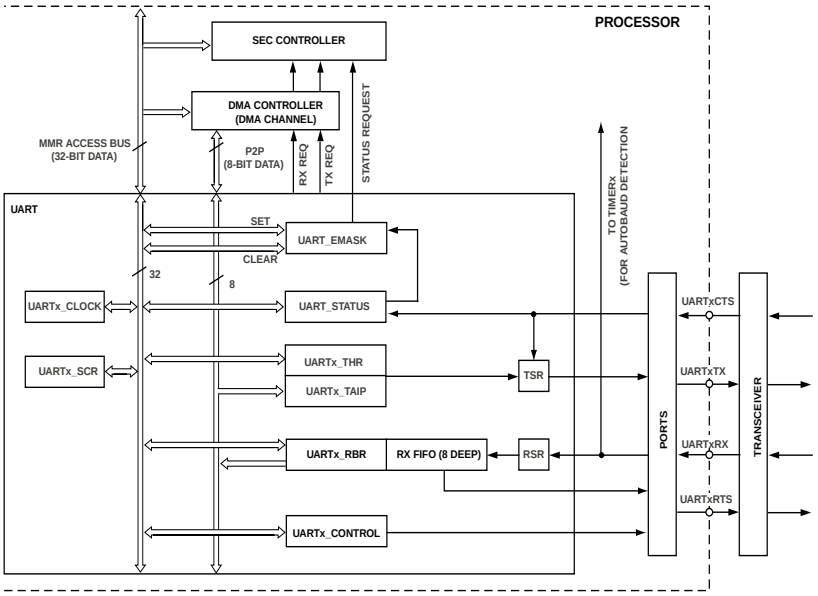


Figure 24-1: UART Block Diagram

UART Architectural Concepts

The following sections provide information about the UART architecture.

Internal Interface

The UART is a DMA-capable peripheral with support for separate transmit and receive DMA controller channels. It operates in either DMA or programmed core modes. The core mode requires software management of the data flow using either interrupts or polling. The DMA method requires minimal software intervention, as the DMA engine itself moves the data. The `UART_RBR` and `UART_THR` registers also connect to one of the peripheral DMA buses.

All UART registers are 32 bits, and the registers connect to the peripheral MMR bus. Not all MMRs can be used, and unused bits are zero-filled. The UART has three interrupt outputs described as follows.

- The transmit and receive request outputs can function as DMA requests and connect to the DMA controller. Therefore, if the DMA is not enabled, the DMA controller simply forwards the request to the system event controller (SEC).
- The status interrupt output connects directly to the SEC . On many processors, the alternative capture input () of one of the GP timers also senses the $\overline{\texttt{UART_RX}}$ pin. When configured in capture mode, the processor can then use the GP timer to detect the bit rate of the received signal.

External Interface

Each UART features a $\overline{\texttt{UART_RX}}$ (receive) pin and a $\overline{\texttt{UART_TX}}$ (transmit) pin available through the general-purpose ports. These two pins usually connect to an external transceiver device that meets the electrical requirements of full-duplex or half-duplex standards. For example, EIA-232, EIA-422, 4-wire EIA-485 for full-duplex or 2-wire EIA-485, LIN for half-duplex. Additionally, the UART features a pair of clear-to-send, input pins ($\overline{\texttt{UART_CTS}}$),

and request-to-send, output pins (UART_RTS) for hardware flow control. UART signals are multiplexed with other functions at the pin level.

Hardware Flow Control

To prevent the UART transmitter from sending data while the receiving counterpart is not ready, the UART features a UART_RTS/UART_CTS hardware flow control mechanism. The UART_RTS signal is an output that connects to the UART_CTS input of the communication partner. If data transfer is bidirectional, the figure shows the ***UART Hardware Flow Control*** handshake.

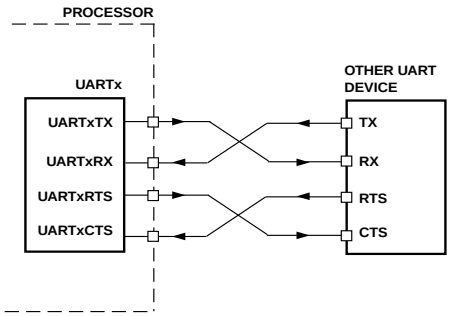


Figure 24-2: UART Hardware Flow Control

In both DMA and core mode, the receiver can deassert the UART_RTS signal to indicate that its receive buffer is almost full. Continued data transfers can cause an overrun error. The transmitter pauses when the UART_CTS input is in a deasserted state. In this state, the transmitter completes transmission of the data currently held in the transmit shift register (UART_TSR) but it does not continue with the data in the transmit hold register (UART_THR). If the UART_CTS pin is asserted again, the transmitter resumes and loads the content of UART_THR register into the UART_TSR register.

Bit Rate Generation

The peripheral clock (SCLK0) and the 16-bit divisor in the UART_CLK register characterize the sample clock. The UART uses the UART_CTL.EN bit to enable the clock. By default, every serial bit is oversampled 16 times. The bit clock is 1/16th of the sample clock. If not in IrDA mode, the bit clock can equal the sample clock if the UART_CLK.EDBO bit is set, so that the following equation applies:

$$\text{Bit Rate} = \text{SCLK0}/(16^{(1-\text{EDBO})} \times \text{Divisor})$$

The ***UART Bit Rate Examples with 100 MHz SCLK*** table provides example divide factors required to support standard baud rates at an SCLK0 of 100 MHz.

Table 24-7: UART Bit Rate Examples with 100 MHz SCLK0

Bit Rate (bits/sec)	EDBO = 0			EDBO = 1		
	DL	Actual	% Error	DL	Actual	% Error
2400	2604	2400.15	0.006	41667	2399.98	0.001
4800	1302	4800.31	0.006	20833	4800.08	0.002

Table 24-7: UART Bit Rate Examples with 100 MHz SCLK0 (Continued)

Bit Rate (bits/sec)	EDBO = 0			EDBO = 1		
	DL	Actual	% Error	DL	Actual	% Error
9600	651	9600.61	0.006	10417	9599.69	0.003
19200	326	19171.78	0.147	5208	19201.23	0.006
38400	163	38343.56	0.147	2604	38402.46	0.006
57600	109	57339.45	0.452	1736	57603.69	0.006
115200	54	115740.74	0.469	868	115207.37	0.006
921600	7	892857.14	3.119	109	917431.19	0.452
1500000	4	1562500	4.167	67	1492537.31	0.498
3000000	2	3125000	4.167	33	3030303.03	1.01
6250000	1	6250000	0	16	6250000	0

NOTE: Carefully select the SCLK0 frequencies. Even multiples of the bit rate can decrease the error percentage.

Setting the bit clock equal to the sample clock (`UART_CLK.EDBO` =1) improves the bit rate granularity and the bit clock matches with the bit rate of the communication partner. Use `UART_CLK.EDBO` mode only when bit rate accuracy is not acceptable in the `UART_CLK.EDBO`=0 mode.

The `UART_CLK.EDBO`=1 mode is not intended to increase the speed of operation beyond the electrical limitations of the UART transfer protocol.

Autobaud Detection

At the chip level, the $\overline{\texttt{UART_RX}}$ pin is typically routed to an alternate capture input (`TIMER_ACI[nn]`) of a general-purpose timer. When working in width capture mode, the processor uses this general-purpose timer to detect the bit rate applied to the $\overline{\texttt{UART_RX}}$ pin automatically by an external device. It often uses the capture capabilities of the timer to supervise the bit rate at run time. If the UART communicates with any device supplied by a weak clock oscillator that drifts over time, the processor can then readjust its UART bit rate dynamically, as required.

Often, the processor uses autobaud detection for initial bit rate negotiations where it is most likely a completer device waiting for the host to send a predefined autobaud character. This situation is common for UART booting. Do not enable the `UART_CTL.EN` bit while autobaud detection is in-process, to prevent the UART from starting a receive operation with incorrect bit rate matching. Alternatively, set the `UART_CTL.LOOP_EN` bit to disconnect the UART from its $\overline{\texttt{UART_RX}}$ pin.

A software routine can detect the pulse widths of serial stream bit cells. The sample base of the timer is synchronous with the UART operation (all derived from the same SCLK0). The UART uses pulse widths to calculate the bit rate divider as follows:

Divisor = `TIMER_TMR[n]_WID`/($16^{(1-\text{EDBO})}$ × Number of captured UART bits)

To increase the number of timer counts and the resolution of the captured signal, do not measure just the pulse width of a single bit. Instead, enlarge the pulse of interest over more bits. Traditionally, a NULL character (ASCII 0x00) is used in autobaud detection, as shown in the ***Autobaud Detection*** figure.

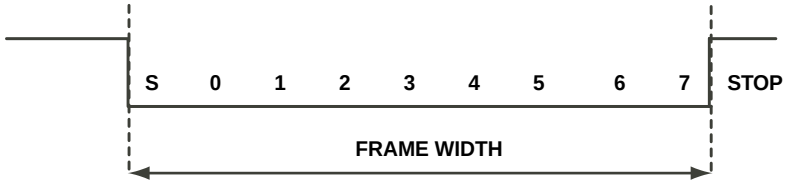


Figure 24-3: Autobaud Detection

Because the example frame encloses 8 data bits and 1 start bit, apply the following formula:

Divisor = `TIMER_TMR[n]_WID`/($16^{(1-EDBO)}$ × 9)

NOTE: For processor-specific mapping of timer alternate capture inputs to the UARTs of the processor, see "Width Capture (WIDCAP) Mode" in the "*General-Purpose Timer (TIMER)*" chapter.

Real receive signals often have asymmetrical falling and rising edges, and the sampling logic level is not exactly in the middle of the signal voltage range. At higher bit rates, such pulse-width-based autobaud detection does not always return adequate results without extra conditioning of the analog signal. Measure signal periods to work around this issue.

For example, predefine the ASCII character “@” (0x40) as the autobaud detection character and measure the period between two subsequent falling edges. As shown in the ***Autobaud Detection Character 0x40*** figure, measure the period between the falling edge of the start bit and the falling edge after bit 6. Since this period encloses 8 bits, apply the following formula:

Divisor = `TIMER_TMR[n]_PER`/($16^{(1-EDBO)}$ × 8)

or:

- Divisor = `TIMER_TMR[n]_PER`>> 7, if `UART_CLK.EDBO`=0
- Divisor = `TIMER_TMR[n]_PER`>> 3, if `UART_CLK.EDBO`=1

The ***Autobaud Detection Character 0x40*** figure shows the ASCII “@” (0x40) detection character.

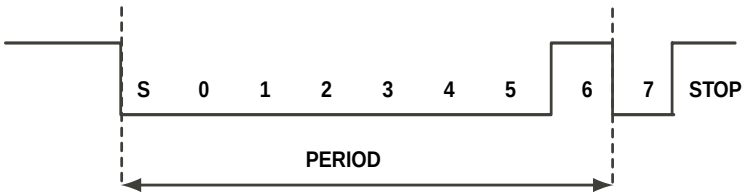


Figure 24-4: Autobaud Detection Character 0x40

UART Debug Features

The UART can automatically calculate and transmit a parity bit. The ***UART Parity*** table summarizes parity behavior assuming 8-bit data words (`UART_CTL.WLS`=b#11).

Table 24-8: UART Parity

PEN	STP	EPS	Data (hex)	Data (binary, LSB first)	Parity
0	x	x	x	x	None
1	0	0	0x60	0000 0110	1
1	0	0	0x57	1110 1010	0
1	0	1	0x60	0000 0110	0
1	0	1	0x57	1110 1010	1
1	1	0	x	x	1
1	1	1	x	x	0

The two force error bits, UART_CTL.FPE and UART_CTL.FFE, are intended for test purposes. They are useful for debugging software, especially in loopback mode.

The UART can be set to internal loopback mode (UART_CTL.LOOP_EN=1). Loopback mode disconnects the input of the receiver from the receive pin and internally redirects the transmit output to the receiver. The transmit pin remains active and continues to transmit data externally as well. Loopback mode also forces the UART_RTS pin to deassert, disconnects the UART_STAT.CTS bit from the UART_CTS input pin, and connects the internal version of UART_RTS to the UART_STAT.CTS bit.

Additionally, the UART_TX pin can be forced to zero asynchronously using the UART_CTL.SB bit.

UART Operating Modes

The following sections describe the main operating modes of the UART.

- UART Mode
- IrDA SIR Mode
- Multi-Drop Bus Mode

UART Mode

The UART mode follows an asynchronous serial communication protocol with these options:

- 1 start bit
- 5–8 data bits
- Address bit (available in MDB mode only)
- None, even, odd or sticky parity
- One, 1½, or two stop bits (1½ stop bits valid only in 5-bit word length)

The UART_CTL register controls the format of received and transmitted character frames. Data is always transmitted and received with the least significant bit (LSB) first.

The ***Bit Stream on a UART TX Pin Transmitting an "S" Character (0x53)*** figure shows a typical physical bit stream measured on a UART_TX pin.

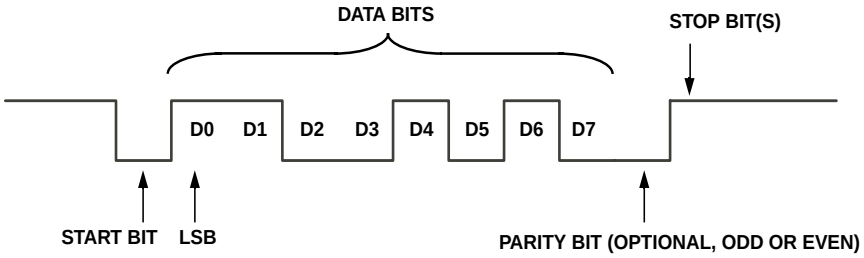


Figure 24-5: Bit Stream on a UART TX Pin Transmitting an "S" Character (0x53)

IrDA SIR Mode

The UART also supports serial data communication by way of infrared signals, according to the recommendations of the Infrared Data Association (IrDA). The physical layer known as IrDA SIR (9.6/115.2 Kbps rate) is based on return-to-zero-inverted (RZI) modulation. The UART does not support pulse position modulation.

Using the 16x data rate clock, RZI modulation is achieved by inverting and modulating the non-return-to-zero (NRZ) code normally transmitted by the UART. On the receive side, the UART uses a 16x clock to determine an IrDA pulse sample window, from which it recovers the RZI modulated NRZ code.

NOTE: The UART_CLK.EDBO bit is not valid in IrDA mode. Clear (=0) this bit in this mode.

Multi-Drop Bus Mode

The UART uses a protocol for point-to-point connections as well as in networks where the EIA-485 standard is representative of UART-based bus systems. The EIA-232 standard defines point-to-point connections. In such networks, node addressing is important.

In a multidrop bus (MDB) network, for example, an address bit enhances the UART frame. The address bit is inserted between the data bits and the optional parity bit. To configure the UART for MDB mode, set the mode of operation bits (UART_CTL.MOD [5:4]) to 01.

By convention, the address bit is transmitted low for regular data bytes. When set, it marks special address bytes that require the attention of all nodes on the network.

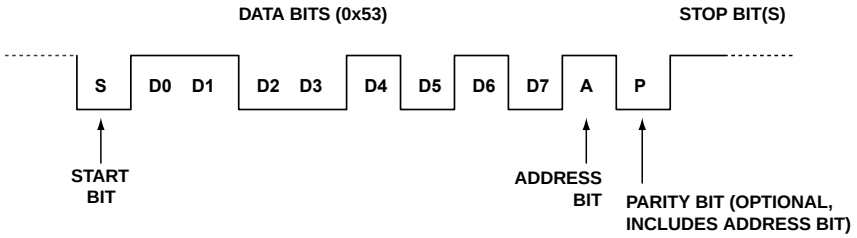


Figure 24-6: UART Frame with Address Bit

All transmit operations are processed through the transmit buffer register (UART_THR), so all DMA data transmissions clear the address bit. If data is written to the transmit address or insert pulse register (UART_TAIP) instead, the same transmit operation is initiated with the only exception that the address bit is sent high.

The UART uses the UART_STAT.ADDR bit of the receiver to signal whether the previously received frame had the address bit set or not. Hardware updates it every time a new frame is received. When the enable address word interrupt bit (UART_IMSK.EAWI) is set, the reception of an address byte triggers a special status interrupt request.

The address sticky bit (UART_STAT.ASTKY) is the sticky version of the UART_STAT.ADDR bit. Hardware sets it whenever the UART_STAT.ADDR bit is set. Software can clear the UART_STAT.ASTKY bit with a W1C operation.

In MDB mode, only address bytes progress to the receive FIFO by default. Data bytes are gated unless the UART_STAT.ASTKY bit is set. The receiver ignores all traffic on the UART bus. This way, the processor can go into low-power mode and interrupt activity does not load the processor every time a frame is transmitted on the UART bus. If, however, an address frame is transmitted, the receiver immediately samples all further traffic. A software routine can analyze the received data, decide whether it was of relevance for the local network node, and W1C the UART_STAT.ASTKY bit if it was not.

Software can overrule of the hardware address frame detection by setting the UART_STAT.ADDR bit and (indirectly) the UART_STAT.ASTKY bit with a W1S operation.

The MDB mode follows an asynchronous serial communication protocol with the following options.

- 1 start bit
- 5–8 data bits
- Address bit
- None, even, odd or sticky parity
- One, 1½, or two stop bits (1½ stop bits are valid only in 5-bit word length)

NOTE: If the address bit and parity bit are both enabled, the parity check and generation includes the address bit in its parity calculation.

UART Data Transfer Modes

The UART can transfer data using both the core and DMA. Receive and transmit paths operate independently except that the bit rate and the frame format are identical for both transfer directions. Transmit and receive channels are both buffered. The UART_THR register buffers the transmit shift register (UART_TSR) and the UART_RBR register buffers the receive shift register (UART_RSR).

UART Mode Transmit Operation (Core)

In core mode, the processor core moves data to and from the UART. A write to the UART_THR register initiates the transmit operation. If no former operation is pending, the UART_THR register passes the data immediately to the

UART_TSR register. There, it is shifted out at the bit rate characterized by the UART_CLK register, with start, stop, and parity bits appended as defined by the UART_CTL register.

The UART_THR register and the UART_TSR register can be modeled as a two-stage transmit buffer. The least significant bit (LSB) always transmits first. This bit is bit 0 of the value written to the UART_THR register.

UART Mode LIN Break Command

Some UART-based protocols demand synchronization methods that are not native to standard UART implementations. For example, the Local Interconnect Network (LIN) protocol requires a low-pulse of well-defined transmit length as a prologue to every multi-byte message. Its length must be at least 13 bit-times.

With previous UARTs, there were two options to implement this protocol:

- A null byte is transmitted with a temporarily lowered bit rate, or
- A software counter generates the period, and the asynchronous set break (SB) mechanisms pull the transmit pin low

Since both methods have their disadvantages, the newer UART introduces a new inter-frame gap technique.

The feature is not available in MDB or IrDA operating modes. However, in standard UART mode (bits UART_CTL.MOD [5:4] =00), a write to the UART_TAIP register initiates the transmission of an inter-frame pulse. If the transmit buffer is not empty, the UART first transmits all bytes in the queue. It only initiates with pulse generation after the last stop bit of the last byte has been shifted out.

The value written into the UART_TAIP register defines the nature and the duration of the transmitted pulse. Bits [6:0] control the duration in bit-times and bit [7] controls the value (duration = UART_TAIP[6:0] / UART_CLK[15:0]). If UART_TAIP[7] is set, and an active high pulse is issued, the number of stop bits is extended. If UART_TAIP[7] is cleared, a low pulse is generated. Invert the polarity using the UART_CTL.FCPOL bit. Writing a value of 13 into the UART_TAIP register generates the break command as required by the LIN protocol.

NOTE: If the UART_CTL.TPOLC bit is enabled, an inverted most-significant bit can be transmitted.

NOTE: If another transmission is pending (in the UART_TSR register), the UART_TAIP initiated pulse is queued until after all pending operations have finished and all stop bits are transmitted.

The transmission of break command/inter-frame gap precedes transmission of the number of stop bits as set in the UART_CTL.STB and UART_CTL.STBH bit fields.

The UART receiver can detect break commands through the break indicator (UART_STAT.BI) flag. This flag reports that an entire UART frame has been received in low state. It does not report whether the duration of the received low pulse was exact or at least 13 bit-times as LIN controllers transmit. Typically, the break indicator meets LIN requirements. The processor can use GP timers to determine the pulse width more precisely, if necessary.

Each $\overline{\text{UART_RX}}$ pin is also routed to a GP timer through its alternate capture input (TACI). This functionality is not only useful for bit rate detection (*autobaud*) but also helps to measure the pulse widths precisely on the $\overline{\text{UART_RX}}$ input. Additionally, the GP timers can issue an interrupt request or a fault condition when the received

pulse width is shorter than a bit time or longer than the worst-case break condition. The windowed watchdog width mode of the GP timers controls this functionality.

UART Mode Receive Operation (Core)

The receive operation uses the same data format as the transmit configuration except that one valid stop bit is always sufficient. The UART_CTL.STB and UART_CTL.STBH bits have no impact on the receiver.

The UART receiver senses the falling edges of the receive input. When it detects an edge, the receiver starts sampling the input according to settings in the UART_CLK register. The receiver samples the start bit (majority sampling) close to its midpoint. If sampled low, it assumes a valid start condition. Otherwise, it discards the detected falling edge.

After detection of the start bit, the received word is shifted into the UART_RSR register.

After the corresponding stop bit is received, the content of the UART_RSR register is transferred to the 8-deep receive FIFO and is accessible by reading the UART_RBR register.

The receive FIFOs and the UART_RBR register act as a 9-stage receive buffer. If the stop bit of the ninth word is received before software reads the UART_RBR register, an overrun error is reported. Overruns protect data in the UART_RBR register and the receive FIFO from being overwritten by further data until the software clears the UART_STAT.OE bit. However, the data in the UART_RSR register is immediately destroyed as soon as the overrun occurs.

The sampling clock is 16 times faster than the bit clock. The receiver oversamples every bit 16 times and makes a majority-decision based on the middle three samples. This functionality improves immunity against noise and hazards on the line. The receiver disregards spurious pulses of less than two times the sampling clock period.

Normally, the receiver samples every incoming bit at exactly the 7th, 8th, and 9th sample clock. If, however, the UART_CLK.EDBO bit is set to 1, the receiver samples bits approximately at 7/16th, 8/16th, and 9/16th of their period. This configuration achieves better bit rate granularity and accuracy as required at high operation speeds. Hardware design must ensure that the incoming signal is stable between 6/16th and 10/16th of the nominal bit period.

Reception starts when the UART receiver detects a falling edge on the $\overline{\text{UART_RX}}$ input pin. The receiver attempts to see a start bit. The data is shifted into the UART_RSR register. After the ninth sample of the first, the receiver processes the stop bit and copies the received data to the 8-stage receive FIFO. The UART_RSR recovers for further data reception.

The receiver samples data bits close to their midpoint. Because the receiver clock is typically asynchronous to the data rate of the transmitter, the sampling point can drift relative to the center of the data bits. The sampling point is synchronized again with each start bit, so the error accumulates only over the length of a single word. The polarity of received data is selectable, using the UART_CTL.RPOLC bit.

NOTE: The receiver checks for only a single stop bit. After the third sample of the first stop bit has been received (at time 9/16th of the stop bit duration), the receiver immediately acts (status update). It then prepares for new falling edge detection (start detection).

IrDA Transmit Operation

To generate the IrDA pulse transmitted by the UART, the normal NRZ output of the transmitter is first inverted if the UART_CTL.TPOLC bit is configured for active-low operation. In this configuration, a zero is transmitted as a high pulse of 16 UART clock periods and a one is transmitted as a low pulse for 16 UART clock periods. Then, six UART clock periods delay the leading edge of the pulse. Similarly, eight UART clock periods truncate the trailing edge of the pulse. For a 16-cycle UART clock period, this operation results in the final representation of the original zero as a high pulse of only 3/16 clock periods. The ***IrDA Transmit Pulse*** figure shows how the pulse is centered around the middle of the bit time. The final IrDA pulse is fed to the off-chip infrared driver.

This modulation approach ensures a pulse width output from the UART of three cycles high out of every 16 UART clock cycles. As shown in the ***IrDA Transmit Pulse*** figure, the error terms associated with the bit rate generator are small and well within the tolerance of most infrared transceiver specifications.

NOTE: In IrDA mode, writes to the UART_TAIP register are equivalent to writes to the UART_THR register.

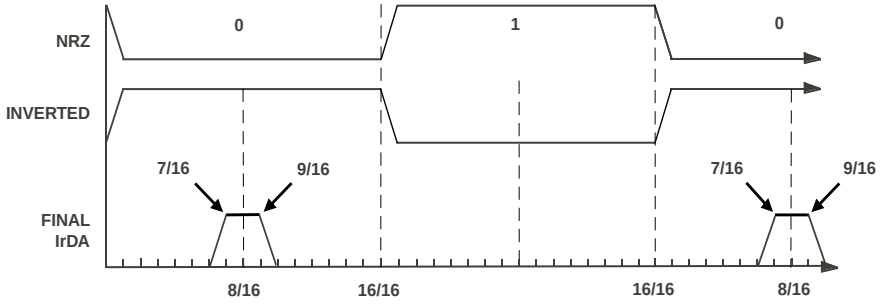


Figure 24-7: IrDA Transmit Pulse

IrDA Receive Operation

The IrDA receiver function is more complex than the transmit function. The receiver must discriminate the IrDA pulse and reject noise. The receiver looks for the IrDA pulse in a narrow window centered around the middle of the expected pulse.

Glitch filtering is accomplished by counting 16 system clocks from the time the receiver detects an initial pulse. If the pulse is absent when the counter expires, the receiver interprets it as a glitch. Otherwise, the receiver interprets it as a zero. This assessment is acceptable because glitches originating from on-chip capacitive cross-coupling typically do not last for more than a fraction of the system clock (SCLK0) period. Appropriate shielding avoids sources outside of the chip and not part of the transmitter. The only other source of a glitch is the transmitter itself. The processor relies on the transmitter to perform within specification. If the transmitter violates the specification, unpredictable results can occur. The 4-bit counter adds an extra level of protection at a minimal cost.

NOTE: Because SCLK0 can change across systems, the longest glitch tolerated is inversely proportional to the SCLK0 frequency.

A counter that is clocked at the 16x bit-time sample clock determines the receive sampling window. The sampling window is resynchronized with each start bit by centering the sampling window around the start bit.

The polarity of receive data is selectable, using the UART_CTL.RPOLC bit. The ***IrDA Receiver Pulse Detection*** figure provides examples of each polarity type.

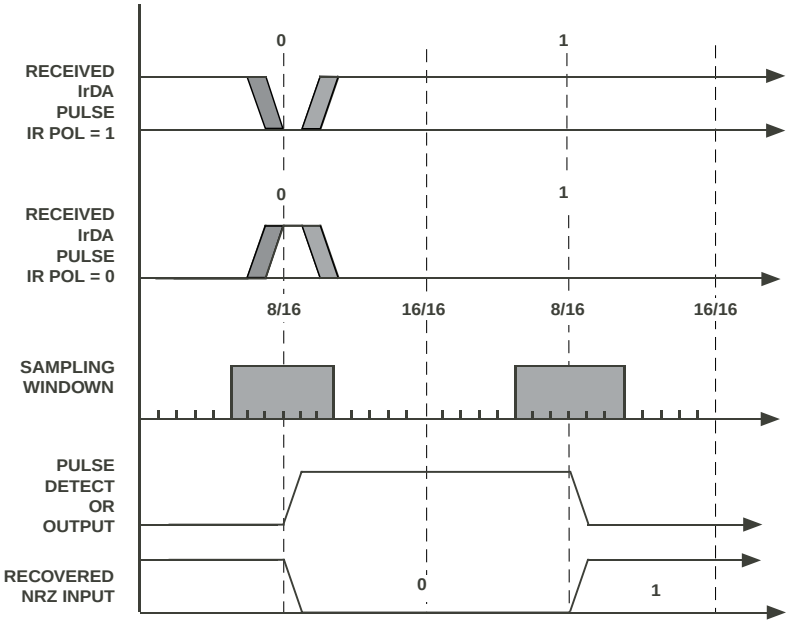


Figure 24-8: IrDA Receiver Pulse Detection

MDB Transmit Operation

In MDB mode, receive and transmit paths operate independently from each other, except for sharing bit rate and frame formats for both transfer directions.

Transmit operation is initiated by writing the UART_THR or UART_TAIP registers. A write to the UART_THR register transmits the written word with the appending address bit set low. A write to the UART_TAIP register transmits the written word with the appended address bit set high. The data is moved into the UART_TSR register, where it is shifted out at the bit rate programmed by the UART_CLK register, with start, stop, address, and parity bits appended, as required.

If DMA is enabled, the DMA engine always writes the data into the UART_THR register, and the written word is transmitted with the appending address bit set low.

The polarity of transmit data is selectable, using the UART_CTL.TPOLC bit.

MDB Receive Operation

Receive operations use the same data format as the transmit configuration, except that the number of stop bits is always assumed to be 1. After detection of the start bit, the received word is shifted into the UART_RSR register at the programmed bit.

Normally, the receiver samples every incoming bit at exactly the 7th, 8th, and 9th sample clock. If, however, the UART_CLK.EDBO bit is set, the receiver samples the bits approximately at 7/16th, 8/16th, and 9/16th of their period. This configuration achieves better bit rate granularity and accuracy needed at high operation speeds. Hardware design must ensure that the incoming signal is stable between 6/16th and 10/16th of the nominal bit period.

After the appropriate number of bits (including address, parity, and stop bits) is received, the UART_RSR register is transferred to the receive FIFO and accessible through the UART_RBR register.

The polarity of receive data is selectable, using the UART_CTL.RPOLC bit.

DMA Mode

In DMA mode, separate receive and transmit DMA channels move data between the UART and memory. The software does not have to move data; it just must set up the appropriate transfers either through the descriptor mechanism or through autobuffer mode.

DMA channels provide a 4-deep FIFO, resulting in total buffer capabilities of 6 words at the transmit side and 9 words at the receive side. In DMA mode, the bus activity and arbitration mechanism determine the latency. The processor loading and interrupt priorities do not determine the latency.

To enable UART DMA, first set up the system DMA control registers. Then, enable the UART_IMSK.ERBFI or UART_IMSK.ETBEI interrupts. This sequence is necessary because these interrupt request lines double as DMA request lines. With DMA enabled, once these requests are received, the DMA control unit generates a direct memory access. If DMA is not enabled, the UART interrupt is passed on to the system interrupt handling unit. The status interrupt for the UART goes directly to the system event controller (SEC), bypassing the DMA unit completely.

For transmit DMA, programs must set the DMA_CFG.SYNC bit. With this bit set, interrupt generation is delayed until the entire DMA FIFO is drained to the UART module. The UART transmit DMA interrupt service routine can disable the DMA or to clear the UART_IMSK.ETBEI control bit only when the DMA_CFG.SYNC bit is set. Otherwise, up to four data bytes can be lost.

When the UART_IMSK.ETBEI bit is set, an initial transmit DMA request is issued immediately. The program then clears the UART_IMSK.ETBEI bit through the DMA service routine.

In DMA transmit mode, the UART_IMSK.ETBEI bit enables the peripheral request to the DMA FIFO. The DMA_CFG.EN bit enables the strobe on the memory side. If the DMA count is less than the DMA FIFO depth, which is 4, then the DMA interrupt can be requested before the UART_IMSK.ETBEI bit is set. If this behavior is unwanted, set the DMA_CFG.SYNC bit.

Regardless of the DMA_CFG.SYNC setting, the DMA stream has not left the UART transmitter completely at the time the interrupt request is generated. Transmission can abort in the middle of the stream, causing data loss, when the UART clock was disabled without extra synchronization with the UART_STAT.TEMT bit.

The UART provides functionality to avoid resource-consuming polling of the UART_STAT.TEMT bit. The UART_IMSK_SET.EDTPTI bit enables the UART_STAT.TEMT bit to trigger a DMA interrupt. To delay the DMA completion interrupt until the last data word of a STOP DMA has left the UART, keep the DMA_CFG.INT bit cleared and set the UART_IMSK_SET.EDTPTI bit instead. Then, the normal DMA completion interrupt is suppressed. Later, the UART_STAT.TEMT event triggers a DMA interrupt after the last word of the DMA has left the UART transmit buffers. If DMA_CFG.INT and UART_IMSK.EDTPTI are set, when finishing STOP mode, the DMA requests two interrupts.

The DMA of the UART module supports 8-bit and 16-bit operation, but not 32-bit operation. It does not support sign-extension.

Mixing DMA and Core Modes

Switching from DMA mode to core operation dynamically requires some consideration, especially for transmit operations. By default, the interrupt timing of the DMA is synchronized with the memory side of the DMA FIFOs. Normally, the transmit DMA completion interrupt is generated after the last byte is copied from the memory into the DMA FIFO. The transmit DMA interrupt service routine is not yet permitted to disable the DMA_CFG.EN bit. The interrupt is requested when the DMA_STAT.IRQDONE bit is set. The DMA_STAT.RUN bit, however, remains set until the data has completely left the transmit DMA FIFO.

When planning to switch from a DMA to core mode, set the DMA_CFG.SYNC bit in the word of the last descriptor or work unit before handing over control. Then, after the interrupt request occurs, software can write new data into the UART_THR register as soon as the UART_STAT.THRE bit permits. If the DMA_CFG.SYNC bit cannot be set, software can poll the DMA_STAT.RUN bit instead. Alternatively, using the UART_IMSK.EDTPTI bit can avoid expensive status bit polling.

When switching from core to DMA operation, ensure that the first DMA request is issued properly. If the DMA is enabled while the UART is still transmitting, no precaution is required. If, however, the DMA is enabled after the UART_STAT.TEMT bit is high, pulse the UART_IMSK.ETBEI bit to initiate DMA transmission.

Setting Up Hardware Flow Control

The following steps show how to set up UART hardware flow control:

1. Configure automatic or manual hardware flow control for the receiver through the UART_CTL.ARTS bit, or the transmitter through the UART_CTL.ACTS bit.
2. Configure $\overline{\text{UART_CTS}}$ and $\overline{\text{UART_RTS}}$ polarity through the UART_CTL.FCPOL bit.

On reset, when the UART is not yet enabled and the port multiplexing has not been programmed, the $\overline{\text{UART_RTS}}$ pin is not driven. Some applications require a resistor to pull the UART_RTS signal to either state during reset.

UART Event Control

Status flags in the UART_STAT register are available to signal data reception, parity, and error conditions, if necessary.

DMA and Interrupt Multiplexing

See the *Direct Memory Access (DMA)* chapter for information on DMA multiplexing. Several interrupts and DMA channels in the UART can be multiplexed.

NOTE: To operate in interrupt mode without using DMA channels, set the UART_IMSK.ELSI bit. This configuration redirects receive and transmit requests to the status interrupt output. The status interrupt goes directly to the SEC without going through the DMA controller.

Interrupt Masks

Each UART features a set of interrupt mask registers: UART_IMSK, UART_IMSK_SET, and UART_IMSK_CLR. The UART_IMSK register supports read/write operations. Writing ones to the UART_IMSK_SET register enables interrupts, writing ones to the UART_IMSK_CLR register disables them. Reads from either register return the enabled bits. This way, different interrupt service routines can control transmit, receive, and status interrupt requests independently and easily.

The UART module uses the UART_IMSK registers to enable requests for system handling of empty or full states of data registers. Unless polling is used as a means of action, the UART_IMSK.ERBFI and UART_IMSK.ETBEI bits in this register are normally set.

Each UART module has three interrupt outputs. It uses one for transmission, one for reception, and one for reporting status events. The UART module routes transmit and receive requests through the DMA controller. The status request goes directly to the system event controller (SEC).

If the associated DMA channel is enabled, the request functions as a DMA request. If the DMA channel is disabled, it simply forwards the request to the SEC. A DMA channel must be associated with the UART module to enable transmit and receive interrupts. Otherwise, transmit and receive requests cannot be forwarded.

NOTE: To operate in interrupt mode without using DMA channels, set the UART_IMSK.ELSI bit. This configuration redirects receive and transmit requests to the status interrupt request output. The status interrupt goes directly to the SEC without going through the DMA controller.

Interrupt Servicing

Interrupt service routines (ISRs) perform UART writes and reads. Separate interrupt lines are provided for transmit, receive, and status. The UART_IMSK register group enables the independent interrupts individually. To enable UART transmit interrupts, set the UART_CTL.EN bit.

The ISRs evaluate the status bits in the UART_STAT register to determine the signaling interrupt source. The system event controller for the processor assigns and unmasks interrupts. The ISRs must clear the interrupt latches explicitly. To reduce interrupt frequency on the receive side in core mode, use the UART_IMSK.ERFCI status interrupt as an alternative to the regular UART_IMSK.ERBFI receive interrupt. Hardware must ensure that at least two (if UART_CTL.RFIT=0) or four (if UART_CTL.RFIT=1) words are available in the receive buffer by the time the interrupt is requested.

Transmit Interrupts

The UART module uses the UART_IMSK_SET.ETBEI bit to enable transmit interrupt requests.

The UART_THR and UART_TAIP registers are the same physical register, and both affect the signaling of the UART_STAT.TEMT, UART_STAT.TFI, and UART_STAT.THRE bits similarly.

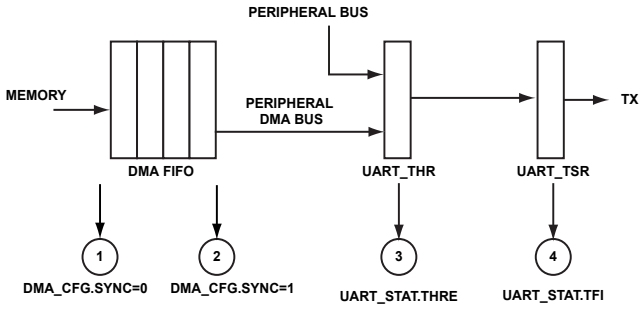


Figure 24-9: Transmit Interrupts

The UART module asserts the transmit request along with the `UART_STAT.THRE` bit, indicating that the transmit buffer is ready for new data. The `UART_STAT.THRE` bit resets to 1. When the `UART_IMSK_SET.ETBEI` bit is set, the UART module immediately issues an interrupt or DMA request. This way, no special handling of the first character is required when transmission of a string is initiated. Set the `UART_IMSK_SET.ETBEI` bit and let the interrupt service routine load the first character from memory and write it to the `UART_THR` register in the normal manner. ISRs can clear the `UART_IMSK.ETBEI` bit through the `UART_IMSK_CLR` register when the string transmission has completed.

Hardware clears the `UART_STAT.THRE` bit when new data is written to the `UART_THR` register. These write operations also clear the transmit interrupt request. However, they also initiate further transmission. If continued transmission is undesirable, the UART module can alternatively clear the transmit request through the `UART_IMSK_CLR.ETBEI` bit register. Transfers of data from the `UART_THR` register to the `UART_TSR` register reset this status flag in the `UART_STAT` register.

ISRs can interrogate the `UART_STAT.TEMT` bit to discover any ongoing transmission. The sticky counterpart of the `UART_STAT.TEMT` bit, `UART_STAT.TFI`, indicates when the transmit buffer has drained and can trigger a status interrupt. When data is pending in either one of these registers, the `UART_STAT.TEMT` flag is low. As soon as all data has left the `UART_TSR` register, the `UART_STAT.TEMT` bit goes high again and indicates that all pending transmit operations (including stop bits) have finished. Then, it is safe to disable the `UART_CTL.EN` bit or to three-state off-chip line drivers. Then, the UART module can generate an interrupt either through the status interrupt channel when the `UART_IMSK.ETFI` bit is set, or through the DMA controller when enabled by the `UART_IMSK.EDTPTI` bit.

When enabled by the `UART_IMSK.ETBEI` bit, the `UART_STAT.THRE` flag requests data along the peripheral command lines to the DMA controller (referred to as TXREQ). This signal is routed through the DMA controller. If the associated DMA channel is enabled, the TXREQ signal functions as a DMA request, otherwise the DMA controller simply forwards it to the SEC. Alternatively the `UART_IMSK.ETXS` bit can redirect the transmit interrupts to the UART status interrupt.

With interrupts disabled, the UART module can poll the status flags to determine when data is ready to move. Because polling is processor intensive; it is not typically used in real-time signal processing environments. Since read operations from `UART_STAT` registers have no side effects, different software threads can interrogate these registers without mutual impacts.

Polling the SEC_SSTAT[n] register without enabling the interrupts by the SEC_CCTL[n] register is an alternate method of operation to consider. Software can write up to two words into the UART_THR register before enabling the UART clock. As soon as the UART_CTL.EN bit is set, the UART module sends those two words.

Receive Interrupts

The UART module uses the UART_IMSK_SET.ERBFI bit to enable receive interrupt requests. If set, the UART_STAT.DR flag requests an interrupt on the dedicated RXREQ output, indicating that new data is available in the UART_RBR register. This signal is routed through the DMA controller. If the associated DMA channel is enabled, the RXREQ signal functions as a DMA request; otherwise the DMA controller simply forwards it to the SEC. Alternatively, if no DMA channel is assigned to the UART, the UART_IMSK.ERXS bit can redirect the receive interrupts to the UART status interrupt. When software reads the UART_RBR register, hardware clears the UART_STAT.DR bit again, which, in turn, clears the receive interrupt request.

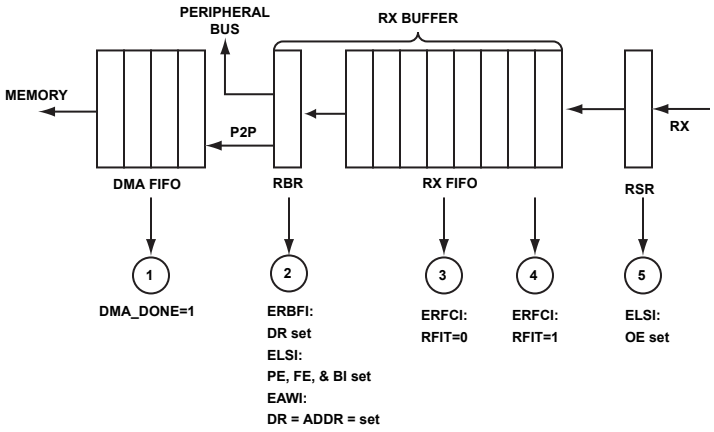


Figure 24-10: Receive Interrupts

Hardware updates the following:

- UART_STAT.DR bits
- UART_STAT.ADDR bits
- UART_STAT.ASTKY bits
- UART_STAT.PE bits
- UART_STAT.FE bits
- UART_STAT.BI bits
- UART_RBR register

The UART_STAT.OE bit is updated as soon as an overflow condition occurs (for example when a stop bit for a frame is received and the receive FIFO is full). When software does not read the UART_RBR register in time, the received data is protected from being overwritten by new data until software clears the UART_STAT.OE bit. Only the content of the UART_RSR register can be overwritten in the overrun case.

The UART module uses the UART_STAT.RFCS bit to monitor the state of the 8-deep receive FIFO. It uses the UART_CTL.RFIT bit to control the behavior of the buffer. If UART_CTL.RFIT is zero, the UART_STAT.RFCS bit is set when the receive buffer holds four or more words. If UART_CTL.RFIT is set, the UART_STAT.RFCS bit is set when the receive buffer holds seven or more words. Hardware clears the UART_STAT.RFCS bit when a core or DMA reads the UART_RBR register and when the buffer is flushed below the level of four (UART_CTL.RFIT=0) or seven (UART_CTL.RFIT=1). If the associated interrupt bit UART_IMSK.ERFCI is enabled, a status interrupt request is reported when the UART_STAT.RFCS bit is set.

If errors are detected during reception, an interrupt can be requested from the status interrupt output. This status interrupt request goes directly to the SEC. The bit enables status interrupt requests.

The controller detects the following error conditions, shown with their associated bits in the UART_STAT register.

- Overrun error (UART_STAT.OE bit)
- Parity error (UART_STAT.PE bit)
- Framing error or invalid stop bit (UART_STAT.FE bit)
- Break indicator (UART_STAT.BI bit)

Status Interrupts

The UART module uses status interrupt channels for the following purposes:

- Line status interrupt requests
- Flow control interrupt requests
- Receive FIFO threshold interrupt requests
- Transmission finished interrupt request

The UART module uses the UART_IMSK.ELSI bit to enable the line status interrupts. If set, the status interrupt request is asserted with one of the UART_STAT.BI, UART_STAT.FE, UART_STAT.PE, or UART_STAT.OE receive errors bits. A W1C operation to the UART_STAT register clears the error bits. Once all error conditions are cleared, the interrupt request deasserts.

The UART module uses the UART_IMSK_SET.ERFCI bit to enable the receive FIFO count interrupt. If set, a status interrupt request is generated when the UART_STAT.RFCS is active. The UART_STAT.RFCS bit indicates a receive buffer threshold level. If the UART_CTL.RFIT bit is cleared, software can safely read two words out of the UART_RBR register by the time the UART_STAT.RFCS interrupt occurs.

If the UART_CTL.RFIT bit is set, software can safely read four words. The interrupt request and the UART_STAT.RFCS bit are cleared when the UART_RBR is read enough of times, so that the receive buffer drains below the threshold of two (UART_CTL.RFIT=0) or four (UART_CTL.RFIT=1). Because in DMA mode a status service routine may not be permitted to read UART_RBR, this interrupt is only recommended in core mode. In DMA mode, use this functionality for error recovery only.

The UART module uses the UART_IMSK_SET.EDSSI bit to enable the flow control interrupts. If active, a status interrupt is generated when the sticky UART_STAT.SCTS bit register is set, indicating that the UART_CTS input of the transmitter been reasserted. A W1C operation to the UART_STAT.SCTS bit clears the interrupt request.

The UART module uses the UART_IMSK_SET.ETFI bit to enable the transmission finished interrupt. If active, a status interrupt request is asserted when the UART_STAT.TFI bit is set. The UART_STAT.TFI is the sticky version of the UART_STAT.TEMT bit, indicating that a byte that started transmission has finished. A W1C operation to the UART_STAT.TFI bit clears the interrupt request.

Multi-Drop Bus Events

Several status bits and interrupt features in the UART_STAT and UART_IMSK registers facilitate efficient data handling in multi-drop bus mode. These features include the address (UART_STAT.ADDR) bit, the address sticky (UART_STAT.ASTKY) bit and the enable address word interrupt (UART_IMSK.EAWI). One of the key features of the multi-drop bus protocol is its address bit. The address bit signifies to the completer that the requester is transmitting an address word (all read it) or a data word (only the addressed target reads its). The UART hardware provides an efficient method of handling the situation described with the use of UART_STAT.ASTKY bit.

NOTE: The UART module uses the UART_STAT.ASTKY bit in multi-drop bus mode to indicate when an address operation for a peripheral is occurring. The UART_STAT.ASTKY bit is a sticky version of the UART_STAT.ADDR bit. Hardware sets the bit whenever the UART_STAT.ADDR bit is set. Only software clears it with a W1C operation. With the UART_STAT.ASTKY bit set, words are received iregardless of the mode bit or address bit setting. With the UART_STAT.ASTKY bit cleared, only address words (mode bit =1) are received and words with mode bit =0 are ignored in MDB mode. This bit does not affect reception in non-MDB modes. (Words with mode bit =0 are not moved from the UART_RSR register to the receive FIFO.)

UART Programming Model

The following sections provide basic procedures for configuring various UART operations.

Detecting Autobaud

To detect Autobaud:

1. Ensure that the timer is disabled.
2. Configure the following bits: UART_CTL.MOD =00, UART_CTL.LOOP_EN =1, UART_CTL.WLS =11 (8-bit data), and UART_CTL.EN =1
3. Configure the following bits: TIMER_TMR[n]_CFG.TMODE =1101, TIMER_TMR[n]_CFG.OUTDIS =1, TIMER_TMR[n]_CFG.IRQMODE =10 and enable the timer.
4. Send test data through the host device and wait for the timer interrupt and disable the timer.

   The bit rate can be derived from the timer period register value according to the formula provided in the Autobaud Detection section.

Using Common Initialization Steps

When using the core or the DMA to execute transfers, the following steps are common to all UART modes.

1. All UART signals are multiplexed and compete with other functions at pin level. First, program the port registers according to the guidelines in the PORT chapter.
2. Program the UART_CLK register. Refer to Bit Rate Generation topic.
3. Program the UART_CTL register and enable the UART clock.

Using Core Transfers

Write data into the UART_THR register, when the UART_STAT.THRE bit is set, to initiate a core transmit operation. If the UART_STAT.DR bit is set, received data can be read from the UART_RBR register.

Using DMA Transfers

1. Make sure that the UART_IMSK.ETBEI or the UART_IMSK.ERBFI bits are cleared before configuring the DMA.
2. Configure the dedicated DMA channel.
3. Set the UART_IMSK.ETBEI or UART_IMSK.ERBFI bits to start the transfer.

Using Interrupts

Each UART features three interrupt signal outputs.

1. Enable individual interrupts in the system event controller (SEC)
2. Register IRQ handlers.
3. Use the interrupts mask registers to enable specific IRQ events.

Setting Up Hardware Flow Control

1. Configure automatic or manual hardware flow control for the receiver through the UART_CTL.ARTS bit, or the transmitter through the UART_CTL.ACTS bit.
2. Configure $\overline{\text{UART_CTS}}$ and $\overline{\text{UART_RTS}}$ polarity through the UART_CTL.FCPOL bit.

ADSP-SC59x UART Register Descriptions

UART (UART) contains the following registers.

Table 24-9: ADSP-SC59x UART Register List

Name	Description
UART_CLK	Clock Rate Register

Table 24-9: ADSP-SC59x UART Register List (Continued)

Name	Description
UART_CTL	Control Register
UART_IMSK	Interrupt Mask Register
UART_IMSK_CLR	Interrupt Mask Clear Register
UART_IMSK_SET	Interrupt Mask Set Register
UART_RBR	Receive Buffer Register
UART_RSR	Receive Shift Register
UART_RXCNT	Receive Counter Register
UART_SCR	Scratch Register
UART_STAT	Status Register
UART_TAIP	Transmit Address/Insert Pulse Register
UART_THR	Transmit Hold Register
UART_TSR	Transmit Shift Register
UART_TXCNT	Transmit Counter Register

Clock Rate Register

The UART_CLK register divides the system clock (SCLK0) down to the bit clock.

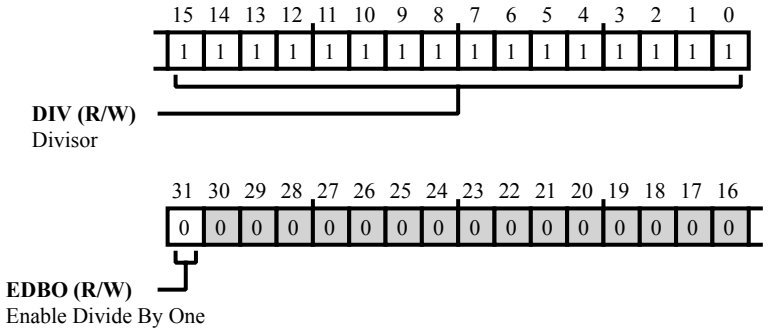


Figure 24-11: UART_CLK Register Diagram

Table 24-10: UART_CLK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EDBO	Enable Divide By One. The UART_CLK.EDBO bit enables the bypassing of the divide-by-16 prescaler in bit clock generation. This functionality improves bit rate granularity, especially at high bit rates. Do not set this bit in IrDA mode. Note: Properly select the SCLK frequencies. Even multiples of bit rate decrease the error percentage. Setting the bit clock equal to the sample clock (UART_CLK.EDBO =1) improves the bit rate granularity and the bit clock matches with the bit rate of the communication partner. The disadvantage is that the power dissipation is higher and sample points are not always accurate. Therefore, use UART_CLK.EDBO=1 mode only when bit rate accuracy is not acceptable in the UART_CLK.EDBO=0 mode. The UART_CLK.EDBO=1 mode is not intended to increase the speed of operation beyond the electrical limitations of the UART transfer protocol.	
		0	Bit clock prescaler = 16
		1	Bit clock prescaler = 1
15:0 (R/W)	DIV	Divisor. The UART_CLK.DIV provides the divisor for the UART's clock bit rate calculation. The bit rate is defined by: Bit Rate = SCLK0 / ($16^{(1-EDBo)}$ x UART_CLK.DIV)	

Control Register

The UART_CTL register provides enable and disable control for UART and IrDA mode of operation. It also provides UART line control, permitting selection of the format of received and transmitted character frames. Modem feature control also is available from this register, including partial modem functionality to allow for hardware flow control and loopback mode.

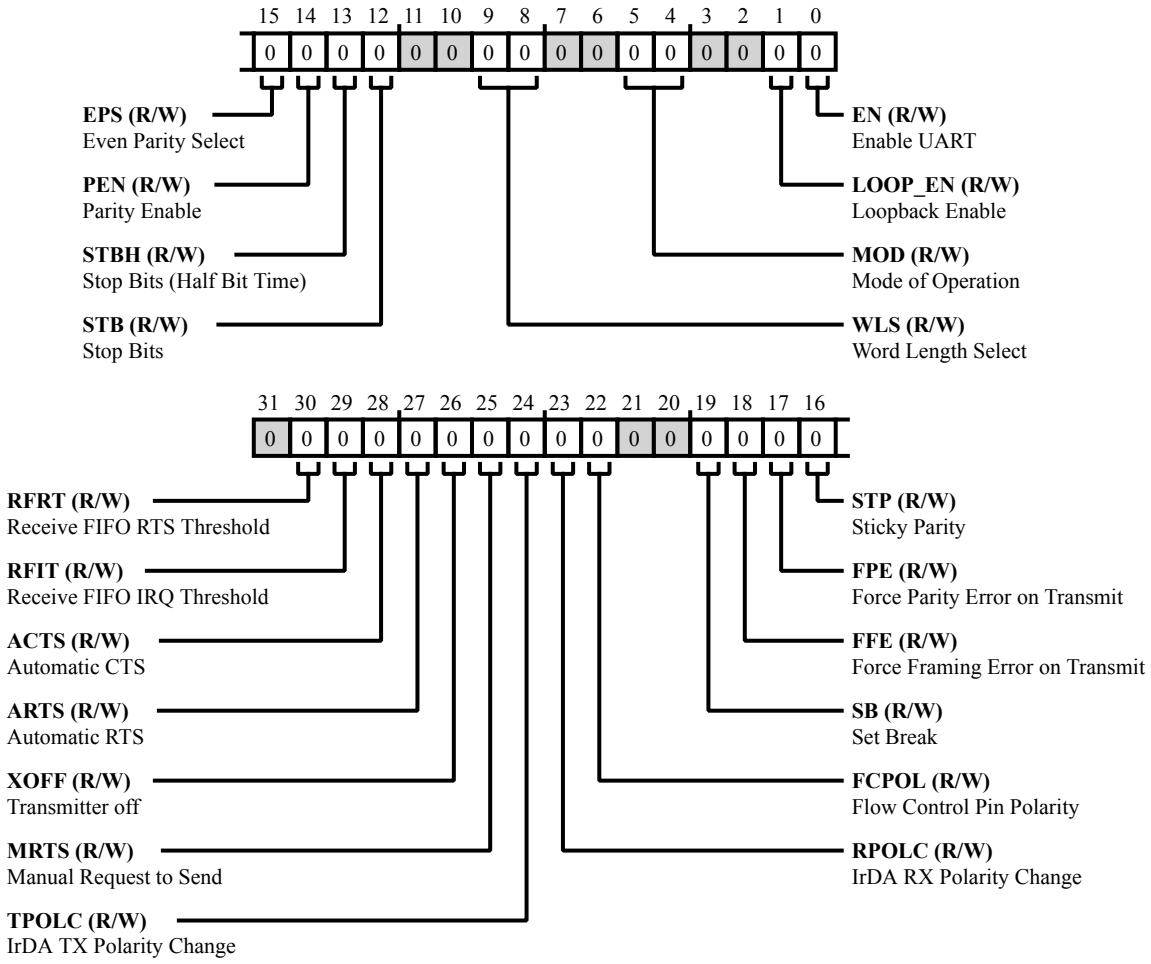


Figure 24-12: UART_CTL Register Diagram

Table 24-11: UART_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	RFRT	Receive FIFO RTS Threshold. The UART_CTL.RFRT bit controls UART_RTS pin assertion and deassertion timing. This bit is ignored if UART_CTL.ARTS is cleared. If set, the UART_RTS pin is deasserted when the receive buffer already holds seven words and an eighth start bit is detected. It is reasserted when the FIFO contains seven words or less. If cleared, the UART_RTS pin is deasserted when the RX buffer already holds four words and a fifth start bit is detected. The UART_RTS pin is reasserted when the RX buffer contains no more than 4 words.	
		0	Deassert RTS if RX FIFO word count > 4; assert if <= 4
		1	Deassert RTS if RX FIFO word count > 7; assert if <= 7
29 (R/W)	RFIT	Receive FIFO IRQ Threshold. The UART_CTL.RFIT bit controls the timing of the UART_STAT.RFCS bit. If UART_CTL.RFIT is cleared, the receive threshold is two. If UART_CTL.RFIT is set, the threshold is four words in the receive buffer.	
		0	Set RFCS=1 if RX FIFO count >= 4
		1	Set RFCS=1 if RX FIFO count >= 7
28 (R/W)	ACTS	Automatic CTS. The UART_CTL.ACTS bit must be set to enable the UART_CTS input pin for UART_TX handshaking. If enabled, the UART_STAT.CTS bit holds the value (if UART_CTL.FCPOL is set) or complement value (if UART_CTL.FCPOL is cleared) of the UART_CTS input pin. The UART_STAT.CTS bit can be used to determine whether the external device is ready to receive data (if UART_STAT.CTS is set) or whether it is busy (if UART_STAT.CTS is cleared). If UART_CTL.ACTS is cleared, the UART_TX handshaking protocol is disabled, and the UART_TX line transmits data whenever there is data to send, regardless of the value of UART_CTS. Software can pause ongoing transmission by setting the UART_CTL.XOFF bit.	
		0	Disable TX handshaking protocol
		1	Enable TX handshaking protocol
27 (R/W)	ARTS	Automatic RTS. The UART_CTL.ARTS bit must be set to enable the UART_RTS input pin for UART_TX handshaking. If set, the hardware guarantees a minimal UART_RTS pin deassertion pulse width of at least the number of data bits defined by the UART_CTL.WLS bit field. If cleared, the UART_RTS pin is not generated automatically by hardware. The UART_RTS pin can still be manually controlled by the UART_CTL.MRTS bit, and software is responsible for UART_RTS pulse width control (if needed).	
		0	Disable RX handshaking protocol.
		1	Enable RX handshaking protocol.

Table 24-11: UART_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	XOFF	Transmitter off. The UART_CTL.XOFF bit (if set) turns off transmission (XOFF) by preventing the content of THR from being continued to TSR. When set, this bit turns on transmission (XON). The state of the UART_CTL.XOFF bit is ignored if the UART_CTL.ACTS bit is set.	
		0	Transmission ON, if ACTS=0
		1	Transmission OFF, if ACTS=0
25 (R/W)	MRTS	Manual Request to Send. The UART_CTL.MRTS bit controls the state of the UART_RTS output pin when the UART_CTL.ARTS bit is cleared. When UART_CTL.MRTS is cleared, the UART deasserts the UART_RTS pin, signaling to the external device that the UART is not ready to receive. When UART_CTL.MRTS is set, the UART asserts the UART_RTS pin, signaling to the external device that the UART is ready to receive.	
		0	Deassert RTS pin when ARTS=0
		1	Assert RTS pin when ARTS=0
24 (R/W)	TPOLC	IrDA TX Polarity Change. The UART_CTL.TPOLC bit selects the active low or high polarity for IrDA communications. This bit is effective only in IrDA mode. If set, in IrDA mode, the UART_TX pin idles high. In UART or MDB mode, it is inverted-NRZ. If cleared, in IrDA mode, the UART_TX pin idles low. In UART or MDB mode, it is NRZ.	
		0	Active-low TX polarity setting
		1	Active-high TX polarity setting
23 (R/W)	RPOLC	IrDA RX Polarity Change. The UART_CTL.RPOLC bit selects the active low or high polarity for IrDA communications. This bit is effective only in IrDA mode. If set, in IrDA mode, the UART_RX pin idles high. In UART or MDB mode, it is inverted-NRZ. If cleared, in IrDA mode, the UART_RX pin idles low. In UART or MDB mode, it is NRZ.	
		0	Active-low RX polarity setting
		1	Active-high RX polarity setting

Table 24-11: UART_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	FCPOL	Flow Control Pin Polarity. The UART_CTL.FCPOL bit selects the polarities of the UART_CTS and UART_RTS pins. When the UART_CTL.FCPOL bit is cleared, the UART_RTS and UART_CTS pins are active low, and the UART is halted when the UART_RTS and UART_CTS pin state is high. When UART_CTL.FCPOL bit is set, the UART_RTS and UART_CTS pins are active high, and the UART is halted when the UART_RTS and UART_CTS pin state is low.	
		0	Active low CTS/RTS
		1	Active high CTS/RTS
19 (R/W)	SB	Set Break. If set, the UART_CTL.SB bit forces the UART_TX pin to low asynchronously, regardless of whether or not data is currently transmitted. This bit functions even when the UART clock is disabled. Because the UART_TX pin normally drives high, it can be used as a flag output pin, if the UART is not used. (For example, if UART_CTL.TPOLC is cleared, drive UART_TX pin low; or if UART_CTL.TPOLC is set, drive UART_TX pin high.)	
		0	No force
		1	Force TX pin to 0
18 (R/W)	FFE	Force Framing Error on Transmit. The UART_CTL.FFE bit is intended for test purposes. This bit is useful for debugging software, especially in loopback mode.	
		0	Normal operation
		1	Force error
17 (R/W)	FPE	Force Parity Error on Transmit. The UART_CTL.FPE bit is intended for test purposes. This bit is useful for debugging software, especially in loopback mode.	
		0	Normal operation
		1	Force parity error

Table 24-11: UART_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	STP	Sticky Parity. The UART_CTL.STP bit controls whether the parity is generated by hardware based on the data bits or whether it is set to a fixed value. If this bit is cleared, the hardware calculates the parity bit value based on the data bits. Then, the EPS bit determines whether odd or even parity mode is chosen. If this bit is set, odd parity is used. That means that the total count of logical-1 data bits including the parity bit must be an odd value. Even parity is chosen by UART_CTL.STP cleared and UART_CTL.EPS set. Then, the count of logical-1 bits must be a even value. If the UART_CTL.STP bit is set, hardware parity calculation is disabled. In this case, the sent and received parity equals the inverted UART_CTL.EPS bit.	
		0	No forced parity
		1	Force (Stick) parity to defined value (if PEN=1)
15 (R/W)	EPS	Even Parity Select.	
		0	Odd parity
		1	Even parity
14 (R/W)	PEN	Parity Enable. The UART_CTL.PEN bit enables parity transmission and parity check. The UART_CTL.PEN bit inserts one additional bit between the most significant data bit and the first stop bit. The polarity of this so-called parity bit depends on data and the UART_CTL.STP and UART_CTL.EPS control bits. Both transmitter and receiver calculate the parity value. The receiver compares the received parity bit with the expected value and issues a parity error if they do not match. If the UART_CTL.PEN bit is cleared, the UART_CTL.STP and the UART_CTL.EPS bits are ignored.	
		0	Disable
		1	Enable parity transmit and check
13 (R/W)	STBH	Stop Bits (Half Bit Time).	
		0	0 half-bit-time stop bit
		1	1 half-bit-time stop bit
12 (R/W)	STB	Stop Bits. The UART_CTL.STB bit controls how many stop bits are appended to transmitted data.	
		0	1 stop bit
		1	2 stop bits

Table 24-11: UART_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	WLS	Word Length Select. The UART_CTL.WLS field determines whether the transmitted and received UART word consists of 5, 6, 7, or 8 data bits.	
		0	5-bit word
		1	6-bit word
		2	7-bit word
		3	8-bit word
5:4 (R/W)	MOD	Mode of Operation. The UART_CTL.MOD selects the UART operation mode (UMOD).	
		0	UART mode
		1	MDB mode
		2	IrDA SIR mode
1 (R/W)	LOOP_EN	Loopback Enable. The UART_CTL.LOOP_EN bit enables UART loopback mode. When set, this bit disconnects the input of the receiver from the UART_RX pin, and internally redirects the transmit output to the receiver. The UART_TX pin remains active and continues to transmit data externally as well. Loopback mode also forces the UART_RTS pin to its deassertive state, disconnects the UART_CTS bit from the UART_CTS input pin, and directly connects the UART_CTL.MRTS bit to the UART_STAT.CTS bit. In loopback mode, setting the UART_CTL.MRTS bit sets the UART_STAT.CTS bit and enables the transmitter of the UART. Clearing the UART_CTL.MRTS bit clears the UART_STAT.CTS bit and disables the transmitter of the UART.	
		0	Disable
		1	Enable
0 (R/W)	EN	Enable UART. The UART_CTL.EN enables the UART clocks. This bit also resets the state machine and control registers when cleared. Using this bit to disable the UART -- when not used -- reduces power consumption.	
		0	Disable
		1	Enable

Interrupt Mask Register

The UART_IMSK register indicates the interrupt mask status (unmasked, if set, or masked, if cleared) of the UART status interrupt requests. This register is not a data register. Instead, it is controlled by the UART_IMSK_SET and UART_IMSK_CLR register pair. Writing ones to the UART_IMSK_SET register enables (unmasks) interrupt requests, and writing ones to the UART_IMSK_CLR register disables (masks) them. Reads from either register return the enabled bits.

The UART_IMSK register is used to enable requests for system handling of empty or full states of UART data registers. Unless polling is used as a means of action, the UART_IMSK.ERBFI and UART_IMSK.ETBEI bits are normally set. Setting this register without enabling system DMA causes the UART to notify the processor of the data inventory state using interrupts. For proper operation in this mode, system interrupts must be enabled, and appropriate interrupt handling routines must be present.

Each UART features three separate interrupt channels to handle the data transmit, data receive, and line status events independently, regardless of whether DMA is enabled or not. If no DMA channels are assigned to the UART, set the UART_IMSK.ELSI bit to reroute the transmit and receive interrupts to the status interrupt request output.

With system DMA enabled, the UART uses DMA to transfer data to or from the processor. Dedicated DMA channels are available for receive and transmit operations. Line error handling can be configured independently from the receive or transmit setup.

The DMA of the UART is enabled by first setting up the system DMA control registers and then enabling the UART_IMSK.ERBFI and UART_IMSK.ETBEI interrupts. This configuration is because the interrupt request lines double as DMA request lines. Depending on whether DMA is enabled or not, upon receiving these requests, the DMA control unit either generates a direct memory access or passes the UART interrupt on to the system interrupt handling unit(s). However, the error interrupt for the UART goes directly to the system interrupt handling unit(s), bypassing the DMA unit completely.

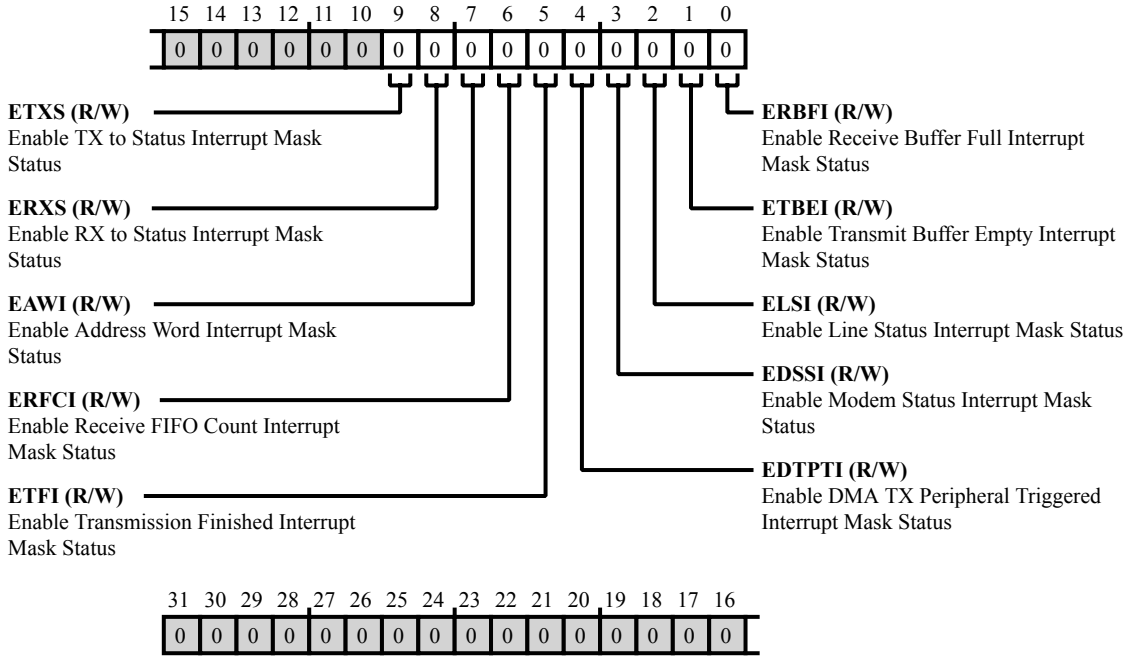


Figure 24-13: UART_IMSK Register Diagram

Table 24-12: UART_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	ETXS	Enable TX to Status Interrupt Mask Status. If set (interrupt unmasked), the `UART_IMSK.ETXS` bit indicates re-direction of the TX interrupt requests to status interrupt output. If cleared, TX interrupt requests are routed to normal interrupt outputs.	
		0	Interrupt is masked
		1	Interrupt is unmasked
8 (R/W)	ERXS	Enable RX to Status Interrupt Mask Status. If set (interrupt unmasked), the `UART_IMSK.ERXS` bit indicates re-direction of RX interrupt requests to status interrupt output. If cleared, RX interrupt requests are routed to normal interrupt outputs.	
		0	Interrupt is masked
		1	Interrupt is unmasked

Table 24-12: UART_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	EAWI	Enable Address Word Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.EAWI bit indicates generation of a status interrupt request when an Address word in MDB-mode is present in the UART_RBR. A received word is an address word if the UART_STAT.ADDR bit is set.	
		0	Interrupt is masked
		1	Interrupt is unmasked
6 (R/W)	ERFCI	Enable Receive FIFO Count Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.ERFCI bit indicates enabling of the receive buffer threshold interrupt request if signaled by the UART_STAT.RFCS bit. Read the UART_RBR register sufficient times to clear the interrupt request.	
		0	Interrupt is masked
		1	Interrupt is unmasked
5 (R/W)	ETFI	Enable Transmission Finished Interrupt Mask Status. If set (interrupt unmasked) the UART_IMSK.ETFI bit indicates enabling of interrupt generation on the status interrupt channel when the transmit buffer register, the transmit address register, and the transmit shift register are all empty as indicated by the UART_STAT.TFI. The UART_IMSK.ETFI interrupt can be used to avoid expensive polling of the UART_STAT.TEMT bit, when the UART clock or line drivers should be disabled after transmission has completed. W1C the UART_STAT.TFI bit to clear the interrupt request. In DMA operation, the UART_IMSK.ETFI bits functionality might be preferred.	
		0	Interrupt is masked
		1	Interrupt is unmasked
4 (R/W)	EDTPTI	Enable DMA TX Peripheral Triggered Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.EDTPTI bit indicates enabling of the DMA completion interrupt request to be delayed until the data has left the UART completely. This bit is required for DMA transmit operation only. If set, the UART can generate a DMA interrupt request by the time the UART_STAT.TEMT bit goes high after the last DMA data word is transmitted. When UART_IMSK.EDTPTI is set, usually the DMA_CFG.INT field is cleared to 00 in a STOP mode DMA. This set up suppresses the normal completion interrupt request, and the UART_STAT.TEMT event is signaled through the DMA controller and triggers the DMA interrupt. If both (DMA_CFG.INT not 00 and UART_IMSK.EDTPTI set), two interrupts are requested at the end of a STOP mode DMA.	
		0	Interrupt is masked
		1	Interrupt is unmasked

Table 24-12: UART_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	EDSSI	Enable Modem Status Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.EDSSI bit indicates enabling of a modem status interrupt request on the same status interrupt channel when the UART_STAT.SCTS bit is set. This indicates UART_CTS pin re-assertion. Write-1-to-clear (W1C) the UART_STAT.SCTS bit to clear the interrupt request.	
		0	Interrupt is masked
		1	Interrupt is unmasked
2 (R/W)	ELSI	Enable Line Status Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.ELSI bit indicates that redirection of TX and RX interrupt requests to the status interrupt output of the UART by OR'ing them with the UART_STAT.OE, UART_STAT.PE, UART_STAT.FE, and UART_STAT.BI interrupt requests. Set this bit when no DMA channel is associated with the UART. Enabling UART_IMSK.ELSI disables the RX/TX interrupt channels and negates the UART_IMSK.EDTPTI bit.	
		0	Interrupt is masked
		1	Interrupt is unmasked
1 (R/W)	ETBEI	Enable Transmit Buffer Empty Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.ETBEI bit indicates generation of a TX interrupt request if the UART_STAT.THRE bit is set.	
		0	Interrupt is masked
		1	Interrupt is unmasked
0 (R/W)	ERBFI	Enable Receive Buffer Full Interrupt Mask Status. If set (interrupt unmasked), the UART_IMSK.ERBFI indicates generation of an RX interrupt request if the UART_STAT.DR bit is set.	
		0	Interrupt is masked
		1	Interrupt is unmasked

Interrupt Mask Clear Register

The UART_IMSK indicates interrupt mask status (unmasked if set, masked if cleared) of UART status interrupts. This register is not a data register. Instead it is controlled by the UART_IMSK_SET and UART_IMSK_CLR register pair. Writing ones to UART_IMSK_SET enables (unmasks) interrupt requests, and writing ones to UART_IMSK_CLR disables (masks) them. Reads from either register return the enabled bits. For more information, see the UART_IMSK register description.

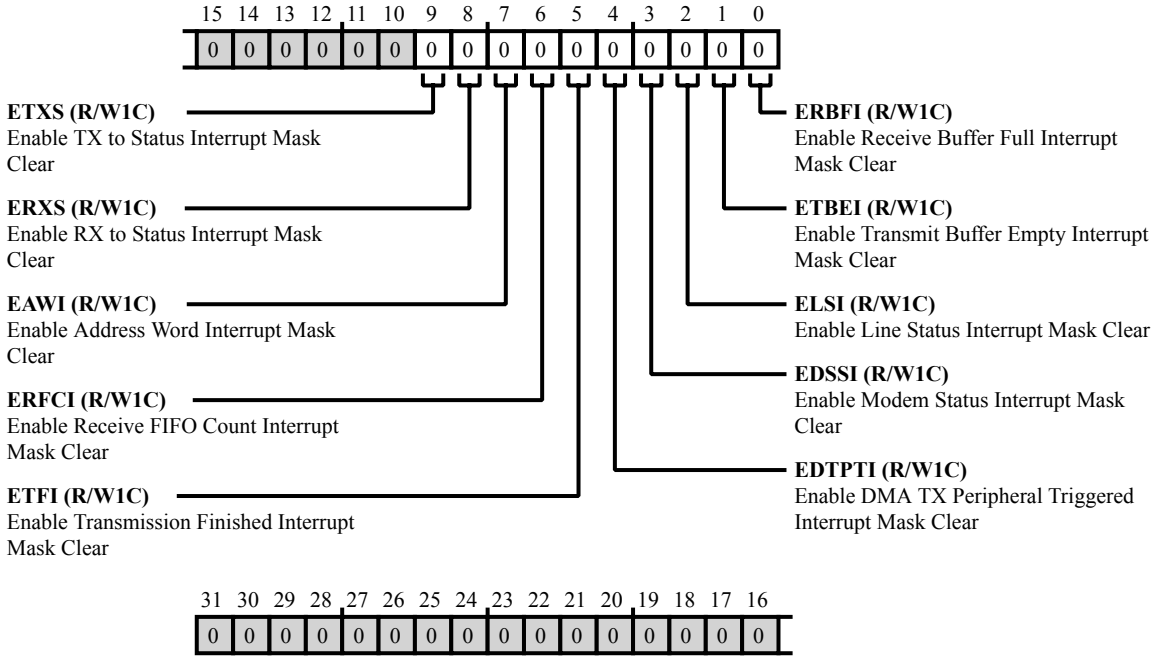


Figure 24-14: UART_IMSK_CLR Register Diagram

Table 24-13: UART_IMSK_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W1C)	ETXS	Enable TX to Status Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
8 (R/W1C)	ERXS	Enable RX to Status Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
7 (R/W1C)	EAWI	Enable Address Word Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt

Table 24-13: UART_IMSK_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1C)	ERFCI	Enable Receive FIFO Count Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
5 (R/W1C)	ETFI	Enable Transmission Finished Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
4 (R/W1C)	EDTPTI	Enable DMA TX Peripheral Triggered Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
3 (R/W1C)	EDSSI	Enable Modem Status Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
2 (R/W1C)	ELSI	Enable Line Status Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
1 (R/W1C)	ETBEI	Enable Transmit Buffer Empty Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt
0 (R/W1C)	ERBFI	Enable Receive Buffer Full Interrupt Mask Clear.	
		0	No action
		1	Mask interrupt

Interrupt Mask Set Register

The UART_IMSK indicates interrupt request mask status (unmasked if set, masked if cleared) of UART status interrupts. This register is not a data register. Instead it is controlled by the UART_IMSK_SET and UART_IMSK_CLR register pair. Writing ones to UART_IMSK_SET enables (unmasks) interrupt requests, and writing ones to UART_IMSK_CLR disables (masks) them. Reads from either register return the enabled bits. For more information, see the UART_IMSK register description.

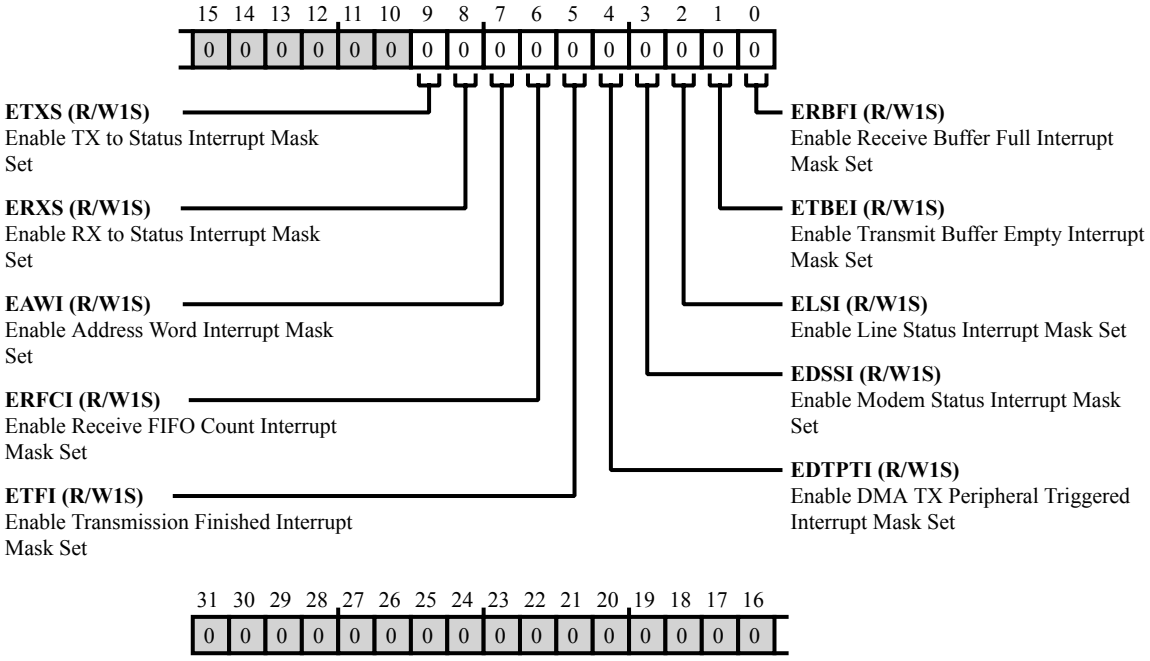


Figure 24-15: UART_IMSK_SET Register Diagram

Table 24-14: UART_IMSK_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W1S)	ETXS	Enable TX to Status Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
8 (R/W1S)	ERXS	Enable RX to Status Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
7 (R/W1S)	EAWI	Enable Address Word Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt

Table 24-14: UART_IMSK_SET Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1S)	ERFCI	Enable Receive FIFO Count Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
5 (R/W1S)	ETFI	Enable Transmission Finished Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
4 (R/W1S)	EDTPTI	Enable DMA TX Peripheral Triggered Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
3 (R/W1S)	EDSSI	Enable Modem Status Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
2 (R/W1S)	ELSI	Enable Line Status Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
1 (R/W1S)	ETBEI	Enable Transmit Buffer Empty Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt
0 (R/W1S)	ERBFI	Enable Receive Buffer Full Interrupt Mask Set.	
		0	No action
		1	Unmask interrupt

Receive Buffer Register

The read-only UART_RBR register is the UART's receive buffer. It is updated when there is pending data in the receive FIFO. Newly available data is signaled by the UART_STAT.DR bit.

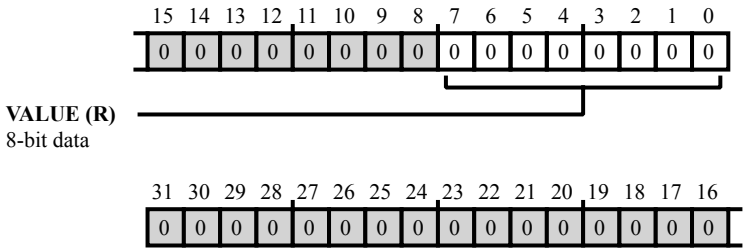


Figure 24-16: UART_RBR Register Diagram

Table 24-15: UART_RBR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VALUE	8-bit data.

Receive Shift Register

The read only UART_RSR register which returns the content of the UART's receive shift register.

The frame data is moved into this shift register after polarity inversion, if any (including the native polarity inversion in the IrDA case).

In the case of the longest frame (MDB, with parity mode, and 8 bit data word-length), the start bit may be shifted out and not available for reading at the end of the frame reception. This register is NOT reset at the start of frame. If read, in the middle of a frame reception, data corresponding the previous frame may not have entirely shifted out (for example, the read data that have been read may NOT correspond entirely to the frame being received).

Because the UART is receiving only 1 stop bit, the UART_RSR contains only 1 stop bit even if more than one stop bit is present in the actual transfer. This register may be considered as storing the 10 most recently received bits (taking into consideration the stop bit receive limitation above).

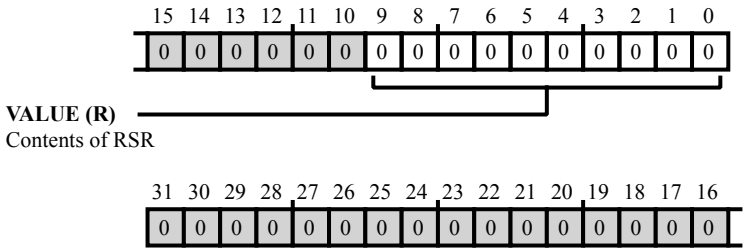


Figure 24-17: UART_RSR Register Diagram

Table 24-16: UART_RSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	VALUE	Contents of RSR.

Receive Counter Register

The UART_RXCNT register returns the content of 16-bit counter in the UART receiver. This count is used for baud rate clock generation (the lower [15:0] is the count data).

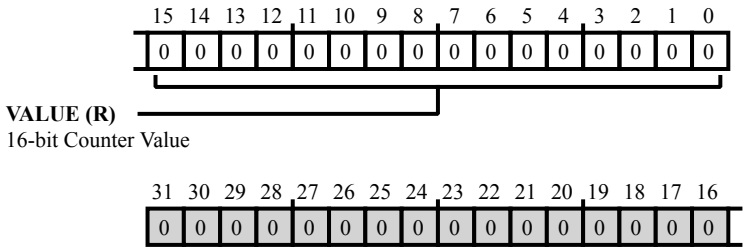


Figure 24-18: UART_RXCNT Register Diagram

Table 24-17: UART_RXCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	VALUE	16-bit Counter Value.

Scratch Register

The UART_SCR registers contain 8-bit scratch pad data. These registers are used for general purpose data storage and do not control the UART hardware in any way.

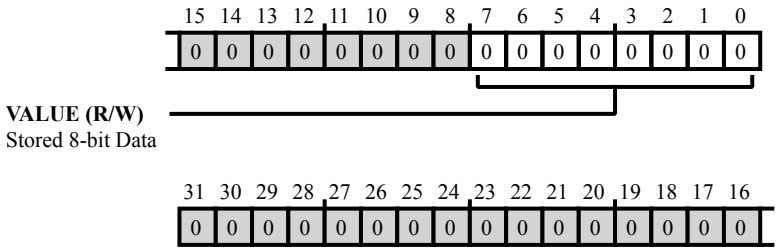


Figure 24-19: UART_SCR Register Diagram

Table 24-18: UART_SCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	Stored 8-bit Data.

Status Register

The UART_STAT register contains the UART line status and UART modem status, as indicated by the current states of the UART's UART_CTS pin and internal receive buffers. Writes to this register can perform write-one-to-clear (W1C) operations on most status bits. Reading this register has no side effects.

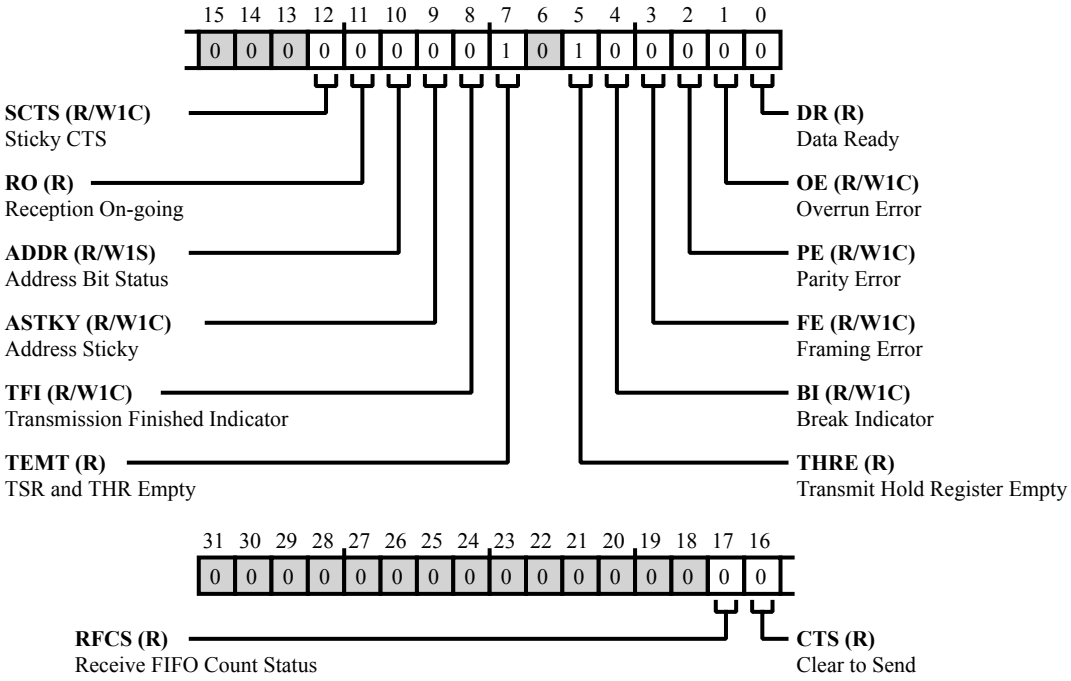


Figure 24-20: UART_STAT Register Diagram

Table 24-19: UART_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/NW)	RFCS	Receive FIFO Count Status. The UART_STAT.RFCS bit is set when the receive buffer holds more or equal entries than a certain threshold. The threshold is controlled by the UART_CTL.RFIT bit. If UART_CTL.RFIT is cleared, the threshold is four entries. If UART_CTL.RFIT is set, the threshold is seven entries. The UART_STAT.RFCS bit is cleared when the UART_RBR register is read sufficient times until the buffer is drained below the threshold. The UART_STAT.RFCS bit can trigger a status interrupt request if enabled by the UART_IMSK_SET.ERFCI bit.	
		0	RX FIFO has less than 4 (7) entries when RFIT=0 (1)
		1	RX FIFO has at least 4 (7) entries when RFIT=0 (1)

Table 24-19: UART_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	CTS	Clear to Send. The UART_STAT.CTS bit holds the value (if UART_CTL.FCPOL set) or the complement value (if UART_CTL.FCPOL cleared) of the UART_CTS input pin. The UART_CTL.ACTS bit must be set to enable this feature. The core can read the value of the UART_STAT.CTS bit to determine whether the external device is ready to receive (UART_STAT.CTS set) or if it is busy (UART_STAT.CTS cleared). If UART_CTL.ACTS is cleared, the UART_TX handshaking protocol is disabled, and the UART transmits data as long as there is data to transmit, regardless of the value of UART_STAT.CTS. When UART_CTL.ACTS is cleared, the software can pause transmission temporarily by setting the XOFF bit. Note that in loopback mode (UART_CTL.LOOP_EN set), the UART_STAT.CTS bit is disconnected from the UART_CTS input pin. Instead, the bit is directly connected to the UART_CTL.MRTS bit.	
		0	Not clear to send (External device not ready to receive)
		1	Clear to send (External device ready to receive)
12 (R/W1C)	SCTS	Sticky CTS. The UART_STAT.SCTS bit is a sticky bit that is set when UART_STAT.CTS transitions from 0 to 1. The UART_STAT.SCTS bit is cleared by software with a W1C operation. This bit can trigger a line status interrupt request if enabled by the UART_IMSK_SET.EDSSI bit.	
		0	CTS has not transitioned from low to high
		1	CTS has transitioned from low to high
11 (R/NW)	RO	Reception On-going.	
		0	No data reception in progress
		1	Data reception in progress
10 (R/W1S)	ADDR	Address Bit Status. The UART_STAT.ADDR bit is used to mirror the address bit of the word in UART_RBR in multi-drop bus protocol, and is enabled only in MDB mode. The UART_STAT.ADDR bit is updated by hardware upon detecting a received word with the address bit in UART_RBR set or cleared. Additionally, software can set the ADDR bit with a write-1-to-set (W1S) operation.	
		0	Address bit is low
		1	Address bit is high

Table 24-19: UART_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W1C)	ASTKY	Address Sticky. The UART_STAT.ASTKY bit is used in multi-drop bus mode to indicate whether a peripheral is currently being addressed. This bit is a sticky version of the UART_STAT.ADDR bit and is set by hardware when setting the UART_STAT.ADDR bit. The UART_STAT.ASTKY bit can only be cleared by software with a write-one-to-clear (W1C) operation. With the UART_STAT.ASTKY bit set, words will be received irrespective of the UART_CTL.MOD bit or UART_STAT.ADDR bit selection. With the UART_STAT.ASTKY bit cleared, only address words (UART_CTL.MOD bit set) will be received and words with UART_CTL.MOD bit cleared are ignored (not moved from the RSR to the RX FIFO) in MDB mode. The UART_STAT.ASTKY bit does not affect reception in non-MDB modes.	
		0	ADDR bit has not been set
		1	ADDR bit has been set
8 (R/W1C)	TFI	Transmission Finished Indicator. The UART_STAT.TFI bit is a sticky version of the UART_STAT.TEMT bit. While UART_STAT.TEMT is automatically cleared by hardware when new data is written to the UART_THR register, the sticky UART_STAT.TFI bit remains set, until it is cleared by software (W1C). The UART_STAT.TFI bit enables more flexible transmit interrupt request timing.	
		0	TEMT did not transition from 0 to 1
		1	TEMT transition from 0 to 1
7 (R/NW)	TEMT	TSR and THR Empty. The UART_STAT.TEMT bit indicates that the UART_THR and UART_TAIP registers and the UART_TSR register are empty. In this case, the program is permitted to write to the UART_THR and UART_TAIP registers twice without losing data. The UART_STAT.TEMT bit can also be used as indicator that pending UART transmission is completed. At that time, it is safe to disable the UART_CTL.EN bit or to three-state the off-chip line driver.	
		0	Not empty TSR/THR
		1	TSR/THR Empty

Table 24-19: UART_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	THRE	Transmit Hold Register Empty. The UART_STAT.THRE bit indicates that the UART transmit channel is ready for new data and software can write to the UART_THR and UART_TAIP registers. Writes to the UART_THR and UART_TAIP registers clear the UART_STAT.THRE. The bit is set again when the UART_THR and UART_TAIP registers are empty and ready to accept data.	
		0	Not empty THR/TAIP
		1	Empty THR/TAIP
4 (R/W1C)	BI	Break Indicator. The UART_STAT.BI bit indicates that the first stop bit is sampled low and the entire data word, including parity bit, consists of low bits only. (This condition indicates that UART_RX was held low for more than the maximum word length.) The UART_STAT.BI bit is updated simultaneously with the UART_STAT.DR bit, that is, by the time the first stop bit is received or when data is loaded from the receive FIFO to the UART_RBR register. The bit is sticky and can be cleared by W1C operations.	
		0	No break interrupt
		1	Break interrupt this indicates UARTxRX was held low(RPOLC=0) / high (RPOLC=1) for more than the maximum word length
3 (R/W1C)	FE	Framing Error. The UART_STAT.FE bit indicates that the first stop bit is sampled. This bit is updated simultaneously with the UART_STAT.DR bit, that is, by the time the first stop bit is received or when data is loaded from the receive FIFO to the UART_RBR register. The UART_STAT.FE bit is sticky and can be cleared by W1C operations. Note that invalid stop bits can be simulated by setting the UART_CTL.FFE bit.	
		0	No error
		1	Invalid stop bit error
2 (R/W1C)	PE	Parity Error. The UART_STAT.PE bit indicates that the received parity bit does not match the expected value. This bit is updated simultaneously with the UART_STAT.DR bit, that is, by the time the first stop bit is received or when data is loaded from the receive FIFO to the UART_RBR register. The UART_STAT.PE bit is sticky and can be cleared by W1C operations. Note that invalid parity bits can be simulated by setting the UART_CTL.FPE bit.	
		0	No parity error
		1	Parity error

Table 24-19: UART_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	OE	Overrun Error. The UART_STAT.OE bit indicates that further data is received while the internal receive buffer was full. This bit is set when sampling the stop bit of the sixth data word. To avoid overruns, read the UART_RBR register in time. In DMA receive mode, overruns are very unlikely to happen ever. After an overrun occurs, the UART_RBR and receive FIFO are protected from being overwritten by new data until the UART_STAT.OE bit is cleared by software. The content of the UART_RSR register is lost as soon as the overrun occurs. The UART_STAT.OE bit is sticky and can be cleared by W1C operations.	
		0	No overrun
		1	Overrun error
0 (R/NW)	DR	Data Ready. The UART_STAT.DR bit indicates that data is available in the receiver and can be read from the UART_RBR register. The bit is set by hardware when the receiver detects the first valid stop bit. The bit is cleared by hardware when the UART_RBR register is read.	
		0	No new data
		1	New data in RBR

Transmit Address/Insert Pulse Register

The UART_TAIP register and the UART_THR register share the same physical register, but UART_TAIP has different effect than the UART_THR register when UART_TAIP is written to in MDB and UART modes.

In MDB mode, data written to the UART_TAIP register is transmitted as an address frame (as with the UART_CTL.MOD bit set).

In UART mode, a write to UART_TAIP causes a pulse of value UART_TAIP [7] for a duration of UART_TAIP [6:0] x bit time. (There is additional inversion if the UART_CTL.TPOLC bit is set).

Bit time is defined by the UART_CLK register. The transmission of the pulse is followed by stop bit transmission as specified by the UART_CTL.STB and UART_CTL.STBH bits. This could be used for supporting line break command and inter-frame gap.

In IrDA mode, writes to UART_TAIP is treated the same as writes to UART_THR.

Accesses to the UART_TAIP register have the same affects as the UART_THR register with respect to the UART_STAT.THRE, UART_STAT.TEMT, and UART_STAT.TFI flags.

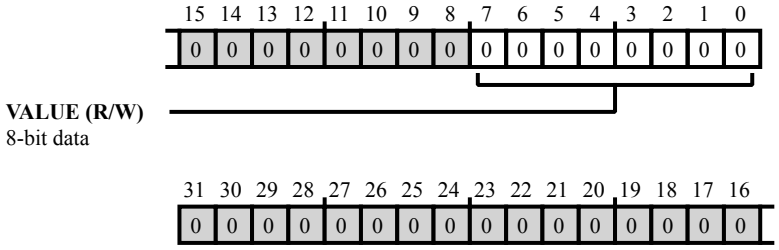


Figure 24-21: UART_TAIP Register Diagram

Table 24-20: UART_TAIP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	8-bit data.

Transmit Hold Register

The write-only UART_THR register is the UART's transmit buffer. The UART_STAT.THRE bit indicates whether data can be written to UART_THR. Writes to this register automatically propagate to the internal UART_TSR register as soon as UART_TSR is ready. Then, transmit operation is initiated immediately.

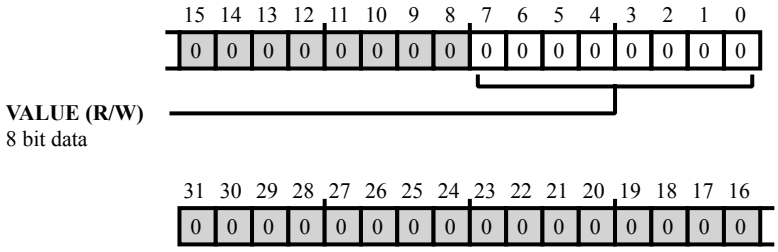


Figure 24-22: UART_THR Register Diagram

Table 24-21: UART_THR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	8 bit data.

Transmit Shift Register

The read only UART_TSR register which returns the content of the UART's transmit shift register.

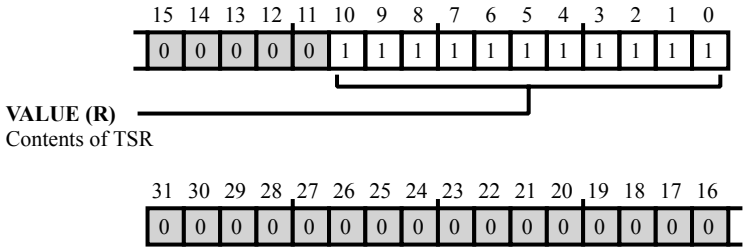


Figure 24-23: UART_TSR Register Diagram

Table 24-22: UART_TSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	VALUE	Contents of TSR.

Transmit Counter Register

The UART_TXCNT read only register returns the content of 16-bit counter in the UART transmitter. This count is used for baud rate clock generation (the lower [15:0] is the count data).

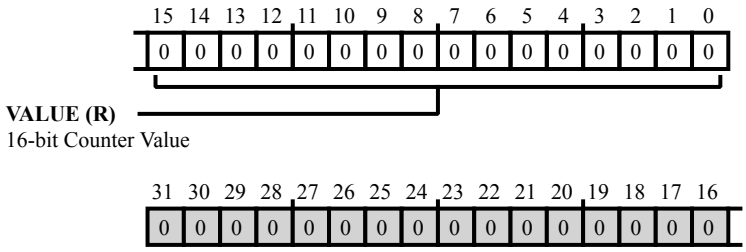


Figure 24-24: UART_TXCNT Register Diagram

Table 24-23: UART_TXCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	VALUE	16-bit Counter Value.

25 Enhanced Parallel Peripheral Interface (EPPI)

The Enhanced Parallel Peripheral Interface (EPPI) is a half-duplex, bidirectional port with a dedicated clock pin and three frame sync (FS) pins. It can support direct connections to active TFT LCDs, parallel A/D and D/A converters, video encoders and decoders, image sensor modules and other general-purpose peripherals. Each EPPI has two DMA channels associated with it. Moreover, in some modes, an EPPI can use an extra DMA channel.

EPPI Features

The EPPI module supports the following features.

- Programmable data length from 8 bits to up to 24 bits per clock cycle (depending on the product model)
- Bidirectional and half-duplex port
- Internal or external clock source
- Clock gating by an external device asserting the clock gating control signal
- Various framed and non-framed operating modes, as well as internal or external frame syncs
- Various general-purpose modes with 0, 1, 2, and 3 frame sync modes for both receive and transmit
- Ignores premature external frame syncs for data consistency
- SMPTE274M and SMPTE 296M high definition format support
- ITU-656, SMPTE 296M and SMPTE 274M status word error detection and correction for ITU-656 receive modes
- ITU-656, SMPTE 296M and SMPTE 274M receive modes – active video only, vertical blanking only, and entire field
- ITU-656, SMPTE 296M and SMPTE 274M preamble and status word decode
- Optional packing and unpacking of data to or from 32 bits from or to 8, 16 bits and 24 bits. If packing or unpacking is enabled, endianness can be altered to change the order of packing or unpacking of bytes/words
- Optional sign extension or zero-fill and alternate even or odd data sample filter for receive modes

- RGB888 to RGB666 or RGB565 conversion for transmit modes
- 4:2:2 YCrCb data Tx/Rx interleaving or de-interleaving modes
- Configurable LCD data enable (DEN) output available on frame sync 3
- Delayed start of PPI frame syncs
- Data clipping and mirroring
- Horizontal and vertical windowing for general purpose 2 and 3 frame sync modes
- Preamble, blanking and stripping support
- Multiplexing dual input

EPPI Functional Description

The EPPI has the following functionality.

RGB data formats

For transmit modes, the EPPI can convert RGB888 data in memory to either RGB565 or RGB666 at the output using bits in the Control register.

Data clipping

The EPPI contains two registers to define the lower and upper limits for the Luma and Chroma components. This functionality is used for clipping data values during 8-bit, 10-bit, 12-bit, or 16-bit transmit modes.

Data mirroring

A data mirroring feature is available which mirrors the EPPI data bits 15–0. This functionality is available in both transmit and receive modes.

Windowing

The EPPI supports windowing for general-purpose input modes.

Preamble, blanking and stripping support

The EPPI can embed blanking information and clip active data to be transmitted. This functionality is available for single channel data, interleaved data, and parallel data and supports data lengths equal to 16 bits, 20 bits or 24 bits.

ADSP-SC59x EPPI Register List

The EPPI is a half-duplex, bidirectional parallel port. It comprises a clock pin, 3 frame sync pins, and a set of data pins. For more information on EPPI functionality, see the EPPI register descriptions.

Table 25-1: ADSP-SC59x EPPI Register List

Name	Description
EPPI_CLKDIV	Clock Divide Register
EPPI_CTL	Control Register
EPPI_CTL2	Control Register 2 Register
EPPI_EVENCLIP	Clipping Register for EVEN (Luma) Data Register
EPPI_FRAME	Lines Per Frame Register
EPPI_FS1_DLY	Frame Sync 1 Delay Value Register
EPPI_FS1_PASPL	FS1 Period Register / EPPI Active Samples Per Line Register
EPPI_FS1_WLHB	FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register
EPPI_FS2_DLY	Frame Sync 2 Delay Value Register
EPPI_FS2_PALPF	FS2 Period Register / EPPI Active Lines Per Field Register
EPPI_FS2_WLVB	FS2 Width Register / EPPI Lines Of Vertical Blanking Register
EPPI_HCNT	Horizontal Transfer Count Register
EPPI_HDLY	Horizontal Delay Count Register
EPPI_IMSK	Interrupt Mask Register
EPPI_LINE	Samples Per Line Register
EPPI_ODDCLIP	Clipping Register for ODD (Chroma) Data Register
EPPI_STAT	Status Register
EPPI_VCNT	Vertical Transfer Count Register
EPPI_VDLY	Vertical Delay Count Register

ADSP-SC59x EPPI Interrupt List

Table 25-2: ADSP-SC59x EPPI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
218	EPPI0_CH0_DMA	EPPI0 Channel 0 DMA	Level	28
219	EPPI0_CH1_DMA	EPPI0 Channel 1 DMA	Level	29
220	EPPI0_STAT	EPPI0 Status	Level	
279	EPPI0_CH0_DMA_ERR	EPPI0 DMA Channel 0 Error		

Table 25-2: ADSP-SC59x EPPI Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
280	EPPI0_CH1_DMA_ERR	EPPI0 DMA Channel 1 Error		

ADSP-SC59x EPPI Trigger List

Table 25-3: ADSP-SC59x EPPI Trigger List Generators

Trigger ID	Name	Description	Sensitivity
29	EPPI0_CH0_DMA	EPPI0 Channel 0 DMA	Edge
30	EPPI0_CH1_DMA	EPPI0 Channel 1 DMA	Edge

Table 25-4: ADSP-SC59x EPPI Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
19	EPPI0_CH0_DMA	EPPI0 Channel 0 DMA	Pulse
20	EPPI0_CH1_DMA	EPPI0 Channel 1 DMA	Pulse

RGB Data Formats

For transmit modes, the EPPI can convert RGB888 data in memory to RGB666 at the output when the EPPI_CTL.RGBFMTEN bit is set and the EPPI_CTL.DLEN value is equal to 18 bits. Similarly, the EPPI can convert RGB888 data in memory to RGB565 at the output when the EPPI_CTL.RGBFMTEN bit is set and EPPI_CTL.DLEN is equal to 16 bits.

This conversion is performed as follows:

- If EPPI_CTL.PACKEN =1, the EPPI first unpacks according to the EPPI_CTL.SWAPEN bit setting, and the three 32-bit data words from the DMA are broken into four 24-bit data words to be transmitted out, as described earlier.
- If EPPI_CTL.PACKEN =0, the EPPI takes the lower 24 bits of the 32-bit DMA as the data to be transmitted. Then, the EPPI truncates this 24-bit data word to the required data width. It removes the lower 2 bits of G and the lower 2 bits or 3 bits of R and B.

Data Clipping

The EPPI contains two registers to define the lower and upper limits for the Luma and Chroma components. It uses these registers for clipping data values during 8-bit, 10-bit, 12-bit, or 16-bit transmit modes. All data values for odd samples which are less than the value in the EPPI_ODDCLIP.LOWODD bit field are replaced with the value in the EPPI_ODDCLIP.LOWODD field. All data values for even samples which are less than the value in the EPPI_EVENCLIP.LOWEVEN field are replaced with the value in the EPPI_EVENCLIP.LOWEVEN field.

In the same manner, all data values for odd samples which are more than the value in the EPPI_ODDCLIP.HIGHODD bit field are replaced with the value in the EPPI_ODDCLIP.HIGHODD field. All data values for even samples which are more than the values in the EPPI_EVENCLIP.HIGHEVEN field are replaced with the values in the EPPI_EVENCLIP.HIGHEVEN field.

Depending on the programmed EPPI length, only the corresponding bits (least aligned) are considered for clipping. For example, if the EPPI is programmed in 10-bit mode, bits 9:0 and bits 25:16 constitute the clipping thresholds. The higher bits are ignored. The EPPI supports 8-bit, 10-bit, 12-bit, and 16-bit clipping thresholds.

For the 4:2:2 YCrCb color space, Luma and Chroma typically have different lower and upper thresholds. Separate thresholds can be required for even and odd data samples. For monochrome (Y only) or some non-video clipping applications, the value in the EPPI_ODDCLIP.LOWODD field can be the same as the value in the EPPI_EVENCLIP.LOWEVEN field. The value in the EPPI_ODDCLIP.HIGHODD field can be the same as the value in the EPPI_EVENCLIP.HIGHEVEN field.

In GP 0 FS mode with internal blanking generation, clipping is valid only for the active video part of the transmitted data. ITU-R 656 preambles, status words, and blanking data bypass the clipping logic.

When the EPPI is programmed in 16-bit or 24-bit mode with the EPPI_CTL.SPLTWRD bit set, the YDATA (luma data) gets the clipping threshold levels of the EPPI_EVENCLIP register. The CDATA (chroma data) gets the clipping threshold levels of the EPPI_ODDCLIP register.

The clipping registers are ignored when the EPPI_CTL.RGBFMTEN bit is set.

Data Mirroring

To increase the pin multiplexing options for the EPPI data pins, a data mirroring feature is available which mirrors the EPPI data bits 15:0. This feature is available in both transmit and receive modes. It is enabled by setting the EPPI_CTL.DMIRR bit.

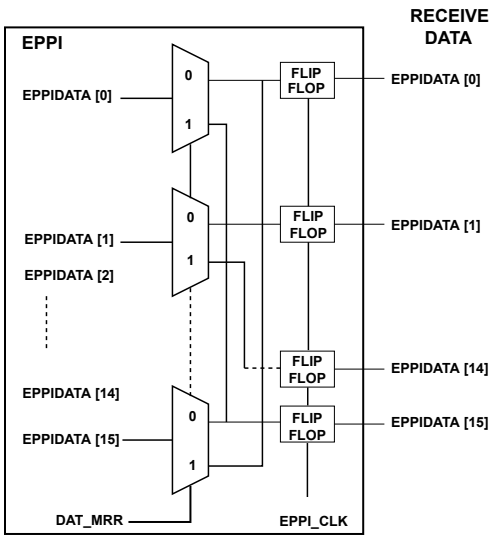


Figure 25-1: Data Mirroring Receive

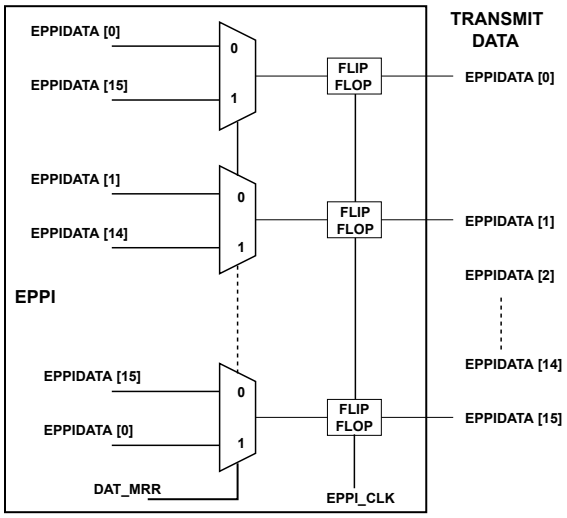


Figure 25-2: Data Mirroring Transmit

Windowing

The EPPI supports windowing for general-purpose input modes. The module can be configured to bring in a region of interest instead of the entire frame of data, which helps reduce bandwidth requirements.

Preamble, Blanking and Stripping Support

The EPPI supports embedding blanking information and clipping of active data for transmission. This functionality is available for single-channel data, interleaved data, and parallel data and supports a data length (`EPPI_CTL.DLEN`) of 16 bits or 24 bits.

Support of preamble generation or detection and stripping of blanking information is also provided for ITU656 mode and the SMPTE 274M and 296M HD formats. The ***Video Mode Comparison*** table shows the SMPTE standards in comparison with the ITU656 modes. Preambles for SMPTE and ITU656 modes are identical. Extension of preamble to 12 bits is also supported.

Table 25-5: Video Mode Comparison

Video Mode	Frame Rate	Frame Resolution	Active Video Resolution	Sampling Frequency (MHz)	Remarks
ITU656 (NTSC)	30	1716x525	720x480	27.02	Y-C interleaved
ITU656 (PAL)	25	1728x625	720x576	27.00	Y-C interleaved
SMPTE 296M	30	3300x750	1280x720	74.25	Y,C separate
	60	1650x750	1280x720	74.25	Y,C separate

Table 25-5: Video Mode Comparison (Continued)

Video Mode	Frame Rate	Frame Resolution	Active Video Resolution	Sampling Frequency (MHz)	Remarks
SMPTE 274M	30	2200x1125	1920x1080	74.25	Y,C separate
	60	2200x1125	1920x1080	148.50	Y,C separate
	25	2640x1125	1920x1080	74.25	Y,C separate
	50	2640x1125	1920x1080	148.50	Y,C separate
	24	2750x1125	1920x1080	74.25	Y,C separate

See the clock operating conditions section of the data sheet for the maximum sampling frequency for this product.

EPPI Definitions

The following definitions are helpful when using the EPPI module.

ITU-R BT.-656

Description of a digital video protocol for interfaces and data stream format required to send uncompressed PAL or NTSC standard definition TV (525 or 625 lines) signals.

YUV422

YUV is a color space where luminance (Y) and chrominance (UV) components define the pixels. The suffix signifies how the chrominance components have been decimated and provide formatting information. In this case, the YUV422 format has the chrominance decimated by two, meaning only half of each chrominance component is available. Typical YUV422 formatting interleaves the luminance and chrominance (for example, U1Y1V1Y2U2Y3V2Y4).

RGB888

RGB is a color space where three color values, one red (R), one green (G) and one blue (B), define the pixels. The suffix signifies the bit widths for these color components. In this case, RGB888 means that each red, green, and blue value is 8 bits.

RGB565

RGB is a color space where three color values, one red (R), one green (G) and one blue (B) define the pixels. The suffix signifies the bit widths for these color components. In this case, RGB565 means that the red (R) and blue (B) are 5 bits each while the green (G) is 6 bits. When packed together, each RGB565 pixel can be represented in a 16-bit data word. LCD display panels commonly use this format.

SMPTE 274M

An HD standard defining the spatial resolution (image sample structure) and frame rates for 1920x1080.

SMPTE 296M

An HD standard for defining the spatial resolution (image sample structure) and frame rates for 1280x720.

EPPI Block Diagram

The ***EPPI Block Diagram*** figure shows the functional blocks within the EPPI.

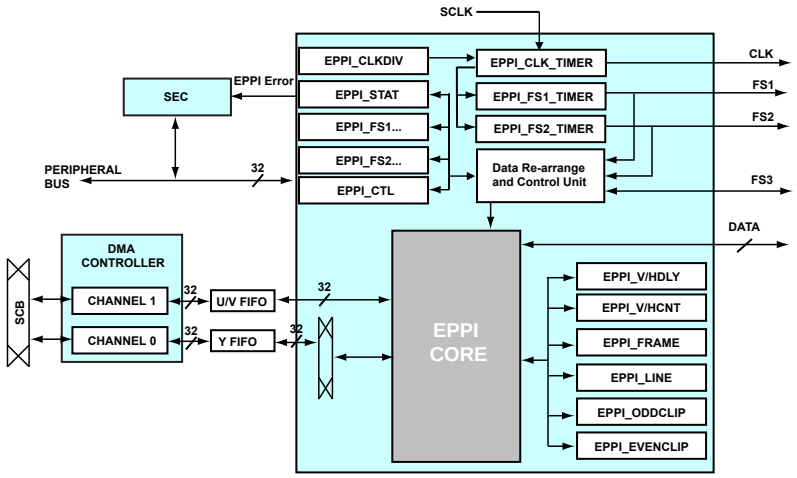


Figure 25-3: EPPI Block Diagram

EPPI Architectural Concepts

The following sections describe the architectural concepts.

- Reset Operation
- Frame Sync Polarity and Sampling Edge
- Direct Memory Access (DMA)
- EPPI Clock

Reset Operation

On a hardware reset, the entire EPPI is reset. All MMRs return to their default values. EPPI interrupt and DMA requests become inactive and internally generated `EPPI_CLK` and frame syncs are aborted.

In software, write 0 to the `EPPI_CTL.EN` bit to reset and reconfigure the EPPI. When disabled in this manner, only the `EPPI_STAT` register is cleared to its reset value. Interrupts and DMA requests become inactive and internally generated clock and frame syncs are aborted.

Frame Sync Polarity and Sampling Edge

The EPPI_CTL.POLS and EPPI_CTL.POLC bits provide a mechanism to select the active level of the frame syncs and the sampling or driving edge of the EPPI clock, respectively. This functionality allows the EPPI to connect to data sources and receivers with a wide array of control signal polarities. Often, the remote data source or receiver also offer configurable signal polarities. In these cases, the EPPI_CTL.POLS and EPPI_CTL.POLC bits add flexibility.

Table 25-6: Frame Sync Polarity Selections and Frame Sync Pin States

Bit Setting	Frame Sync 2	Frame Sync 1
POLS = b#00	Active high	Active high
POLS = b#01	Active high	Active low
POLS = b#10	Active low	Active high
POLS = b#11	Active low	Active low

Table 25-7: Clock Polarity Selections and Receive/Transmit Pin States

	Receive		Transmit	
Bit Setting	Sample Data	Sample/Drive Syncs	Drive Data	Sample/Drive Syncs
POLC = b#00	Falling edge	Falling edge	Rising edge	Rising edge
POLC = b#01	Falling edge	Rising edge	Rising edge	Falling edge
POLC = b#10	Rising edge	Falling edge	Falling edge	Rising edge
POLC = b#11	Rising edge	Rising edge	Falling edge	Falling edge

Direct Memory Access (DMA)

The EPPI has a native DMA controller with two channels. A local arbiter arbitrates between these channels and requests are forwarded to the system crossbar. The EPPI has one connection to the fabric.

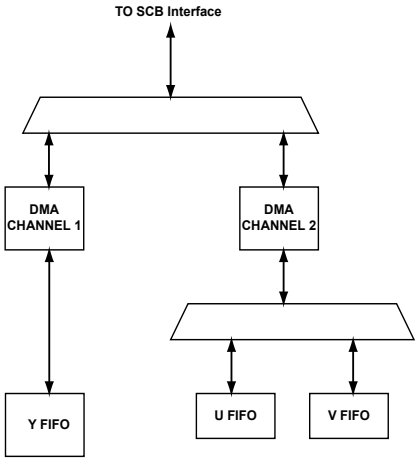


Figure 25-4: EPPI DMA Interface

The EPPI must be used with DMA. Configuring the EPPI DMA channels is a necessary step toward using the EPPI interface. The channels can be configured for either transmit or receive operation, plus have a maximum throughput of (EPPI_CLK) x (32 bits/transfer). In modes where data lengths permit, packing can increase transfer bandwidth.

The DMA engine generates interrupts at the completion of a row, frame, or partial-frame transfer. The DMA engine also coordinates the source or destination point for the data that is transferred through the EPPI.

The 2D DMA capability allows the processor to be interrupted at the end of a line or after a frame of video is transferred, or if a DMA error occurs. The DMA_XCNT and DMA_YCNT registers allow for flexible data interrupt points. For example, assume the DMA_XMOD = DMA_YMOD =1. If a data frame contains 320 × 240 bytes (240 rows of 320 bytes each), the following conditions hold.

- Setting DMA_XCNT =320, DMA_YCNT = 240, and DMA_CFG.INT =1 interrupts on every row transferred, for the entire frame.
- Setting DMA_XCNT =320, DMA_YCNT =240, and DMA_CFG.INT =2 interrupts only on the completion of the frame (when 240 rows of 320 bytes have been transferred).
- Setting DMA_XCNT =38,400 (320 x 120), DMA_YCNT =2, and DMA_CFG.INT =1 causes an interrupt when half of the frame is transferred, and again when the whole frame is transferred.

The following is the general procedure for setting up DMA operation with the EPPI.

1. Configure the DMA registers as appropriate for the desired DMA operating mode.
2. Enable the DMA channel for operation.
3. Configure appropriate EPPI registers.
4. Enable the EPPI by writing 1 to the EPPI_CTL.EN bit.

EPPI Clock

The EPPI can be supplied with an external clock, or the clock can be generated internally and supplied to external devices. For information on the maximum PPI_CLK specification in internal and external clock modes, see the product-specific data sheet.

When using an external EPPI_CLK, there can be up to two cycles latency before valid data is received or transmitted.

The internal clock can be generated from SCLK0 when the EPPI_CTL.ICLKGEN bit is set. The value in the EPPI_CLKDIV register determines the generated clock frequency. The internally generated EPPI clock frequency is:

$f_{PCLK} = f_{SCLK0}/(EPPI_CLKDIV + 1)$

where:

f_{PCLK} – frequency of internally generated EPPI clock

f_{SCLK} – frequency of SCLK0

EPPI_CLKDIV – Clock division value programmed in the EPPI_CLKDIV register.

The ***Relationship Between CLKDIV and the Ratio of SCLK0 to EPPI Clock*** table gives a few examples.

Table 25-8: Relationship Between CLKDIV and the Ratio of SCLK0 to EPPI Clock

CLKDIV15–0	EPPI/SCLK0 Clock Ratio
0x0002	1:3
0x0003	1:4
0x0004	1:5
0x0005	1:6
...	...

EPPI Operating Modes

The EPPI supports various receive and transmit modes of operation which include the detection and generation of preamble data. Specifically, the EPPI supports data formats described in the specifications ITU656, SMPTE 274M and SMPTE 296M. In addition to these modes, the EPPI also supports general-purpose receive and transmit using up to three frame syncs (FS).

The control register (EPPI_CTL) includes most of the bits used for configuring operating modes. The “Register Descriptions” section of this chapter provides complete descriptions of these bits.

ITU-R 656 Modes

The EPPI supports three input modes and one output mode for ITU-R 656 framed data. This section describes these modes.

ITU-R 656 Background

In ITU-R 656 mode, the horizontal (H), vertical (V), and field (F) signals are sent as an embedded part of the video data stream. The signals are sent in a series of bytes that form a control word. ITU-R 656 was formerly known as CCIR-656.

The letter H is used to distinguish between the *start of active video* (SAV) and *end of active video* (EAV) signals. These signals indicate the beginning and end of active video data in each line. The SAV occurs on a 1-to-0 transition of H, and EAV occurs on a 0-to-1 transition of H. The space between EAV and SAV is filled with horizontal blanking data. Therefore, H = 1 during the horizontal blanking portion of the data stream, and H = 0 during the active video portion of the data stream.

The letter V is used to denote the vertical blanking portion of the data stream. A transition in V can occur only in the EAV sequence. When V = 1, the data stream contains vertical blanking data, and when V = 0, the data stream contains active video data.

The letter F is used to distinguish Field 1 from Field 2. Interlaced video has two fields in a frame of data. It requires each field to be handled uniquely, and alternate rows of each field combined to create the actual video image.

For interlaced video, F = 0 represents Field 1 (*Odd Field*) and F = 1 represents Field 2 (*Even Field*). Progressive video makes no distinction between Field 1 and Field 2, and F is always 0 for progressive video. Interlaced video requires each field to be handled uniquely because alternate rows of each field combine to create the actual video image.

An entire field of video is comprised of active video plus horizontal blanking (the space between an EAV and SAV code) and vertical blanking (the space where V = 1). A field of video commences on a transition of the F bit.

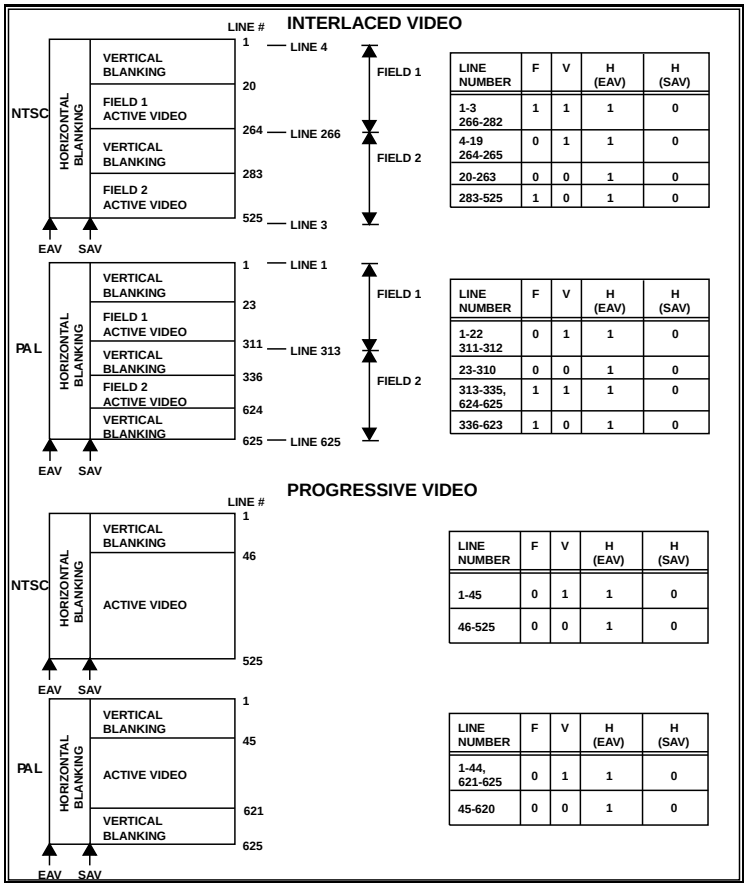


Figure 25-5: Typical Video Frame Partitioning for NTSC/PAL Systems in Interlaced and Progressive ITU-R BT.656 Systems

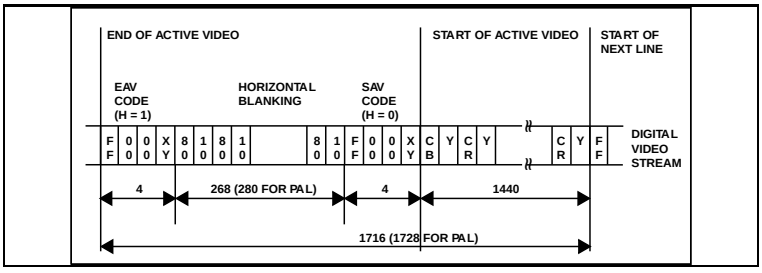


Figure 25-6: ITU-R 656 8-Bit Parallel Data Stream from NTSC (PAL) Systems

NOTE: Refer to the ***Control Sequences for 8-Bit and 10-Bit ITU-R 656 Video*** table. There is a defined preamble of three data elements (for example, in the case of 8-bit video: 0xFF, 0x00, 0x00), followed by the XY status word. The status word contains four protection bits for error detection and correction excluding the

F (field), V (vertical blanking), and H (horizontal blanking) bits. F and V are only allowed to change as part of EAV sequences (that is, transition from H = 0 to H = 1).

The bit definitions are as follows:

- F = 0 for field 1
- F = 1 for field 2
- V = 1 during vertical blanking
- V = 0 when not in vertical blanking
- H = 0 at SAV
- H = 1 at EAV
- P3 = V XOR H
- P2 = F XOR H
- P1 = F XOR V
- P0 = F XOR V XOR H

P3–P0 are protection bits that enable 1-bit and 2-bit error detection, and 1-bit error correction at the receiver. The EPPI corrects the error if it detects 1-bit errors in F, V, or H. Errors in the protection bits themselves are detected but not corrected.

Table 25-9: Control Sequences for 8-Bit and 10-Bit ITU-R 656 Video

	8-Bit Data								10-Bit Data	
	D9 (MSB	D8	D7	D6	D5	D4	D3	D2	D1	D0
Preamble	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0
Control Byte	1	F	V	H	P3	P2	P1	P0	0	0

The EPPI_STAT register contains 2 bits, EPPI_STAT.ERRDET and EPPI_STAT.ERRNCOR, that are used to report the status of error detected and error not corrected, respectively.

The EPPI_STAT.ERRDET bit is set whenever an error is detected in the status word. However, this bit does not generate an interrupt. The EPPI_STAT.ERRNCOR bit is set when more than a 1-bit error is detected in the status word. An interrupt is generated when the EPPI_STAT.ERRNCOR bit is set. It can be cleared by clearing the EPPI_STAT.ERRNCOR and EPPI_STAT.ERRDET bits. Both bits are sticky and W1C.

In many applications, video streams other than the standard NTSC/PAL formats (for example, CIF, QCIF) can be employed. The processor interface is flexible enough to accommodate different row and field lengths. In general, as

long as the incoming video has the proper EAV/SAV codes, the EPPI can read it in. A CIF image could be formatted to be 656-compliant, where EAV and SAV values define the range of the image for each line. The V and F codes are used to delimit fields and frames.

The following sections provide descriptions of EPPI operations.

Table 25-10: Operating Modes and Generic EPPI Operation

		How to configure	Useful for	How to configure in ITU R 656 Tx Mode
ITU-R BT.656 Rx	Entire field	DIR= 0 XFRTYPE = b#01		
	Active video	DIR = 0 XFRTYPE = b#00		
	Blanking only	DIR = 0 XFRTYPE = b#10		
GP 0 FS	Tx	DIR = 1 XFRTYPE = b#11 FSCFG = b#00	Applications where periodic frame syncs are not used to frame the data	BLANKGEN = 1 DLEN = (b#000, b#001 or b#100)
	Rx	DIR = 0 XFRTYPE = b#11 FSCFG = b#00		
GP 1 FS	Tx	DIR = 1 XFRTYPE = b#11 FSCFG =b#01	Interfacing with ADCs, DACs, and other general-purpose devices	BLANKGEN = 1 DLEN = (b#000, b#001 or b#100)
	Rx	DIR = 0 XFRTYPE = b#11 FSCFG = b#01		
GP 2 FS	Tx	DIR = 1 XFRTYPE = b#11 FSCFG = b#10	Video applications that use two hardware synchronization signals, HSYNC and VSYNC	BLANKGEN = 1 DLEN = (b#000, b#001 or b#100)
	Rx	DIR = 0 XFRTYPE = b#11 FS_CFG = b#10		
GP 3 FS	Tx	DIR = 1 XFRTYPE = b#11 FSCFG = b#11	Video applications that use three hardware sync signals, HSYNC, VSYNC, and FIELD	BLANKGEN = 1 DLEN = (b#000, b#001 or b#100)
	Rx	DIR = 0 XFRTYPE = b#11 FSCFG = b#11		

ITU-R 656 Input Modes

In the ITU-R 656 input modes, the video source provides the clock or the system supplies it externally.

As shown in the ***ITU-R 656 Input Submodes*** figure and described in the following sections, there are three submodes supported for ITU-R 656 inputs: entire field, active video only, and vertical blanking interval only.

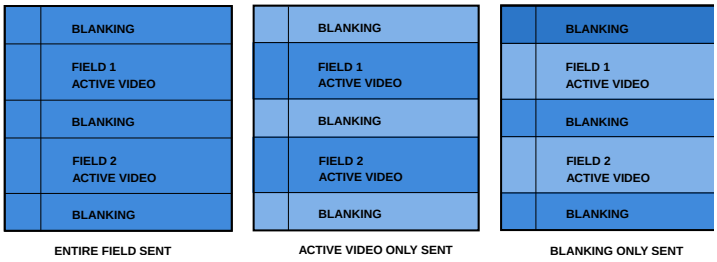


Figure 25-7: ITU-R 656 Input Submodes

Entire Field

In this mode, the EPPI reads the entire incoming bit stream. This stream includes active video as well as control byte sequences and ancillary data that can be embedded in horizontal and vertical blanking intervals.

Data transfer starts immediately after Field 1 synchronization occurs. The transfer does not include the first EAV code that contains the F = 0 assignment for interlaced video or the V = 0 assignment for progressive video.

Active Video

The EPPI uses this mode when only the active video portion of a field is of interest. The EPPI ignores (does not read in) all data between EAV and SAV, as well as all data present when V = 1. Furthermore, the control byte sequences are not stored to memory. The EPPI filters the sequences. After the start of Field 1 synchronizes, the EPPI ignores incoming samples until it sees an SAV.

In active video mode, programs must specify the number of total (active plus vertical blanking) lines per frame in the `EPPI_FRAME` register. Programs must specify the number of total (active plus horizontal blanking plus 8) samples per line in the `EPPI_LINE` register.

In this mode, any input data sequence that is considered part of the preamble is not sent to memory such as in 8-bit ITU mode. If 0xFF or 0x00 appear in the input data stream, these values are considered part of the preamble. The part of the preamble can appear individually and not be tagged along with the preamble sequence FF, 00, 00. This functionality also applies to vertical blanking interval mode.

Vertical Blanking Interval (VBI)

In this mode, data transfer is only active while V = 1 is in the control byte sequence. This functionality indicates that the video source is in the midst of the vertical blanking interval (VBI), which is sometimes used for ancillary data transmission. The ITU-R 656 recommendation specifies the format for these ancillary data packets, but the EPPI is not equipped to decode the packets themselves. Software must handle this task. Horizontal blanking data is logged where it coincides with the rows of the VBI.

The VBI is split into two regions within each field. The EPPI considers these two separate regions as one contiguous space. However, frame synchronization begins at the start of Field 1, which does not necessarily correspond to the

start of vertical blanking. For instance, in 525/60 systems, the start of Field 1 (F = 0) corresponds to line 4 of the VBI.

In VBI mode, the program must specify the number of total (active plus vertical blanking) lines per frame in the EPPI_FRAME register. The program must specify the number of total (active plus horizontal blanking plus 8) samples per line in the EPPI_LINE register.

In this mode, any input data sequence that is considered as part of the preamble is not sent to memory such as in 8-bit ITU mode. If 0xFF or 0x00 appears in the input data stream, these values are considered part of the preamble. The part of the preamble can appear individually, and not be tagged along with the preamble sequence FF, 00, 00. This functionality applies to active video mode too.

ITU-R 656 Output in General-Purpose Transmit Modes

In GP transmit mode, the EPPI frames an ITU-R 656 output stream with the proper preambles and blanking intervals by setting the EPPI_CTL.BLANKGEN bit. The EPPI fetches active data from memory through the DMA channel, saving DMA bandwidth. The EPPI generates and embeds the proper preamble, status word (EAV and SAV sequences), and blanking data along with the active video from memory. Program the EPPI_FS1_PASPL, EPPI_FS2_WLVB, EPPI_FS2_PALPF, and EPPI_FS1_WLHB registers to perform the desired functions. The EPPI can also drive out the frame syncs using the EPPI_CTL.FSCFG bit setting.

The ***16-Bit Transmit with Internal Blanking Generation*** figure shows the bit stream format in 16-bit transmit modes with blanking generation (EPPI_CTL.BLANKGEN enabled). Each 16-bit data sample consists of 8-bit luma (Y) and 8-bit chroma (Cr or Cb) components. During transmission, the chroma data and blanking bytes of value 0x80 are placed on the upper half (MSBs) of the data lines. The luma data and blacking bytes of value 0x10 are placed on the lower half (LSBs) of the data lines.

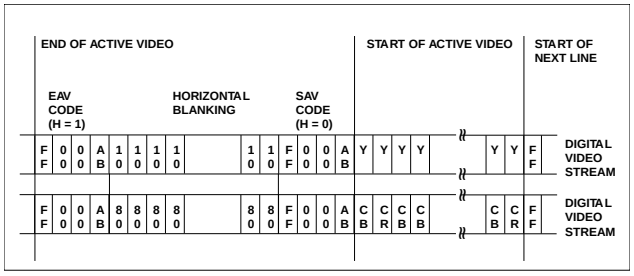


Figure 25-8: 16-Bit Transmit with Internal Blanking Generation

The ***Generated Blanking Preamble Sequence*** figure shows the data transmitted by the EPPI in this mode. After the EPPI is enabled, and if the EPPI FIFO is not empty, the transmission starts by sending out an EAV sequence for a vertical blanking line. For interlaced video, F starts at 1. For progressive video, F is always 0.

NOTE: Internal blanking generation functionality is valid only when the data length is 8, 10, or 16 bits and when the EPPI is in GP transmit modes. The EPPI_CTL.BLANKGEN bit generates preambles even in GP 2FS mode.

The internal blanking generation functionality of the ITU-R 656 output mode can also be bypassed by clearing the EPPI_CTL.BLANKGEN bit. (For example, if sending ancillary data in the blanking interval). The EPPI_CTL.BLANKGEN bit generates preambles even in GP 2FS mode.

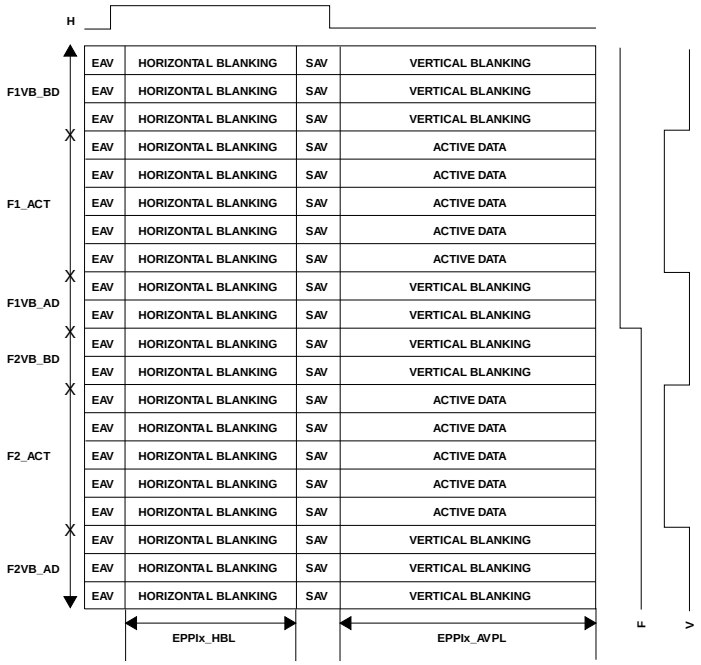


Figure 25-9: Generated Blanking Preamble Sequence

Frame Synchronization in ITU-R 656 Modes

For interlaced video, the start of frame synchronization occurs when a high-to-low transition is detected in F, the field indicator. For progressive video, the start of frame synchronization occurs when a high-to-low transition is detected in V, the vertical blanking indicator. These transitions in F and V can occur only in the EAV sequence. A start of line is detected on a low-to-high transition in H, the horizontal blanking indicator, which occurs in the EAV sequence as well.

For interlaced video, the start of frame corresponds to the start of field 1. Therefore, up to two fields can be ignored before the EPPI receives data. (For example, if field 1 started before the EPPI-to-camera channel was established). For progressive video, the start of frame corresponds to the start of active video.

Because all H and V signaling is embedded in the data stream in ITU-R 656 modes, the EPPI ignores the count registers (`EPPI_HCNT`, `EPPI_VCNT`). However, the EPPI still uses the `EPPI_FRAME` register to check for synchronization errors. Therefore, program this MMR with the number of lines expected in each frame of video.

The EPPI monitors the number of EAV-to-SAV transitions that occur from the start of a frame until it decodes the end of frame condition. (For example, a transition from F = 1 to F = 0 for interlaced video and a transition from V = 1 to V = 0 for progressive video).

At the end of frame condition, the actual number of lines processed is compared against the value in `EPPI_FRAME`. If there is a mismatch, a frame track error is asserted in the `EPPI_STAT` register. For example, if an SAV transition was missed, the current field only has NUM_ROWS – 1 rows. But resynchronization occurs at the start of the next frame. When the EPPI receives the entire field, the field status bit is toggled in the `EPPI_STAT` register. This way, an interrupt service routine (ISR) can discern which field was previously read in.

General-Purpose EPPI Modes

The general-purpose (GP) EPPI modes accommodate a wide variety of data capture and transmission applications.

Each EPPI has three bidirectional frame sync pins. The EPPI internally generates frame syncs, or an external device communicating with the EPPI generates them.

GP modes differ based on the number of frame syncs used and the EPPI supports GP 0 FS—GP 3 FS modes.

All the GP modes, except 0 FS mode, support horizontal windowing. GP modes with 2 and 3 frame syncs also support vertical windowing.

For GP transmit modes with internal clock or frame syncs, the EPPI starts generating the clock or frame syncs only when the EPPI FIFO is full for the first time. For GP 0 FS transmit mode, the EPPI only starts transmitting when the EPPI FIFO is full for the first time.

General-Purpose 0 Frame Sync Mode

This mode is useful for applications where periodic frame syncs are not used to frame the data.

After the initial trigger, the EPPI receives or transmits data samples on every clock cycle. However, if the `EPPI_CTL.SKIPEN` bit is set for receive mode, the EPPI receives only alternate data samples.

The `EPPI_LINE`, `EPPI_FRAME`, `EPPI_HCNT`, `EPPI_HDLY`, `EPPI_VCNT`, and `EPPI_VDLY` registers are not valid for GP 0 FS mode. Therefore, windowing is not possible in this mode. Also, line and frame track errors are not applicable in this mode.

GP 0 FS receive mode is further divided into two submodes; internal trigger (`EPPI_CTL.FLDSEL` bit =0) and external trigger (`EPPI_CTL.FLDSEL` bit =1). The submodes are based on how the processor initiates data transmission or reception. GP 0 FS transmit mode is always internally triggered. DMA handles all subsequent data manipulation.

- ***Frame synchronization in GP 0 FS external trigger mode.*** When the EPPI is programmed in external trigger mode, it does not generate the `EPPI_FS1` signal and the external device must provide a trigger. The EPPI starts receiving the data as soon as an `EPPI_FS1` signal assertion is detected. After that, the DMA handles all subsequent data manipulation and any activity on `EPPI_FS1` is ignored.
- ***Frame synchronization in GP 0 FS internal trigger mode.*** When the EPPI is programmed in internal trigger mode, it starts receiving or transmitting data as soon as the EPPI clock is enabled and synchronized. There can be up to four PPI clock cycles of latency before valid data is received or transmitted.

General-Purpose 1 Frame Sync Mode

This mode is useful for interfacing the EPPI with analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and other general-purpose devices. This mode works for both transmit and receive.

The `EPPI_FRAME`, `EPPI_VDLY`, and `EPPI_VCNT` registers have no effect in GP 1 FS mode. As a result, frame track errors and vertical windowing are not available.

General-Purpose 2 Frame Sync Mode

This mode is useful for video applications that use two hardware synchronization signals, HSYNC and VSYNC. The HSYNC signal can be connected to EPPI_FS1 and the VSYNC signal can be connected to EPPI_FS2.

Data Enable in General-Purpose 2 Frame Sync Transmit Mode

The EPPI_FS3 pin functions as a data enable (DEN) pin, when EPPI is configured in GP 2 FS transmit mode and generating the frame sync internally. The bits EPPI_CTL.MUXSEL and EPPI_CTL.CLKGATEN are not enabled. The functionality of the DEN pin is described in the following two cases.

Case 1

Blanking generation is configured using the EPPI_CTL.BLANKGEN bit. EPPI data length (EPPI_CTL.DLEN bit) is configured for 8, 10, or 16-bit transfers. The EPPI_FS3 pin asserts during the *active data* regions, aligned with EPPI_CLK according to the clock polarity (EPPI_CTL.POLC bit) settings. For this mode, the pin EPPI_FS3 is driven based on the EPPI_CTL.POLC setting. The pin EPPI_FS3 is driven out on the same EPPI clock edge that drives out data. The frame sync polarity (EPPI_CTL.POLS) setting does not apply here—EPPI_FS3 is always active high in this mode.

Case 2

Blanking generation (EPPI_CTL.BLANKGEN =0) is disabled. Or blanking generation is enabled, but the EPPI data length (EPPI_CTL.DLEN bit) is configured for a transfer size other than 8, 10, or 16 bits. The EPPI_FS3 pin asserts at the start of the active data region on each line, aligned with EPPI_CLK according to the EPPI_CTL.POLC bit settings. For this mode, the pin EPPI_FS3 is driven based on the EPPI_CTL.POLC setting. The EPPI_FS3 signal is driven out on the same EPPI clock edge that drives out data.

The EPPI_CTL.POLS bit setting does not apply for case 2. The EPPI_FS3 signal is always active high in this mode. Once asserted, EPPI_FS3 stays asserted for the number of clock cycles per line configured in the EPPI_HCNT register, then it deasserts. This behavior on each line continues for the total number of lines programmed in the EPPI_VCNT register per frame. The behavior repeats at the start of subsequent video frames.

In case 2, if transmission of valid data is held off due to delays programmed in the EPPI_HDLY or EPPI_VDLY registers, the assertion of EPPI_FS3 is also held off. The delay is on a per-line or per-frame basis.

General-Purpose 3 Frame Sync Mode

This mode is useful for video applications that use three synchronization signals for hardware: HSYNC, VSYNC, and FIELD. The HSYNC connects to the EPPI_FS1 pin, VSYNC connects to the EPPI_FS2 pin, and FIELD connects to the EPPI_FS3 pin.

GP 3 FS mode is similar in operation to GP 2 FS mode. However, the start of frame synchronization in GP 3 FS also considers the state of the EPPI_FS3 pin. All the windowing register settings (EPPI_FRAME, EPPI_LINE, EPPI_HDLY, EPPI_HCNT, EPPI_VDLY, and EPPI_VCNT registers), as well as data reception or transmission and error generation are the same as for GP 2 FS mode. In addition, for GP 3 FS mode with internal frame syncs, the EPPI_CTL.FLDSEL bit setting specifies the condition under which the transfer begins.

The EPPI generates the EPPI_FS3 signal and toggles during every assertion of EPPI_FS2 or a combination of EPPI_FS2 and EPPI_FS1. The toggle depends on the EPPI_CTL.FLDSEL bit setting. The EPPI skips an EPPI_FS2 signal when the EPPI_FS3 value is high. Because of this condition, program the EPPI_FS2 period value to half of the total number of pixels in the frame as in GP 3 FS mode. When in GP 2 FS mode, program the EPPI_FS2 period with the value equal to the number of pixels per frame.

Supported Data Formats

The following sections describe EPPI receive and transmit data formats.

Receive Data Formats

The ***EPPI Receive Data Formats*** table provides information about EPPI configuration for specific use models for receive data.

Table 25-11: EPPI Receive Data Formats

Input Data Width	Use Model	Splitting/Packing Options
8	NTSC/PAL data	EPPI_CTL.SPLTEO =1 EPPI_CTL.SUBSPLTODD =1 if necessary to separate chroma components
	RGB sensor	No splitting possible. EPPI_CTL.PACKEN =1 – Four EPPI words are packed to 32-bit DMA data. EPPI_CTL.PACKEN =0 – Each EPPI word is sent as 8-bit data on the 32-bit DMA bus. This transfer consumes 4 times the DMA bandwidth of the 8-bit case with EPPI_CTL.PACKEN =1;
	ADCs	Gives I (in phase) and Q (quadrature) components. EPPI_CTL.SPLTEO =1 EPPI_CTL.SUBSPLTODD =0 since there are only two components.

Table 25-11: EPPI Receive Data Formats (Continued)

Input Data Width	Use Model	Splitting/Packing Options
10	NTSC/PAL data	Each EPPI word is zero filled or sign extended to 16 bits. EPPI_CTL.SPLTEO =1. EPPI_CTL.SUBSPLTODD =1 if necessary to separate chroma components.
	RGB sensor	No splitting possible. EPPI_CTL.PACKEN =1. Two EPPI words are zero filled or sign extended to 16 bits and packed to 32-bit DMA data. EPPI_CTL.PACKEN =0. Each EPPI word can be zero filled or sign extended to 16 bits and sent as a 16-bit data on the 32-bit DMA bus. This transfer consumes double the bandwidth of the 10-bit case with EPPI_CTL.PACKEN =1;
	ADCs	Each EPPI word is zero filled or sign extended to 16 bits. EPPI_CTL.SPLTEO =1 SEPPI_CTL.SUBSPLTODD = 0 since there are only two components.
12	RGB sensor	No splitting possible. EPPI_CTL.PACKEN =1. Two EPPI words are zero filled or sign extended to 16 bits and packed to 32-bit DMA data. EPPI_CTL.PACKEN =0. Each EPPI word can be zero filled or sign extended to 16 bits and sent as a 16-bit data on the 32-bit DMA bus. This transfer consumes double the bandwidth of the 12-bit case with EPPI_CTL.PACKEN =1;
	ADCs	Each EPPI word is zero filled or sign extended to 16 bits. EPPI_CTL.SPLTEO =1 EPPI_CTL.SUBSPLTODD =0 since there are only two components.
14	ADCs	Each EPPI word is zero filled or sign extended to 16 bits. EPPI_CTL.SPLTEO =1 EPPI_CTL.SUBSPLTODD =0 since there are only two components.

Table 25-11: EPPI Receive Data Formats (Continued)

Input Data Width	Use Model	Splitting/Packing Options
16	8-bit luma/chroma pair for NTSC or HD	EPPI_CTL.SPLTEO =1, EPPI_CTL.SPLTWRD =1, EPPI_CTL.SUBSPLTODD =1 if necessary to separate chroma components.
	16-bit luma/chroma pair for NTSC or HD	EPPI_CTL.SPLTEO =1, EPPI_CTL.SPLTWRD =0, EPPI_CTL.SUBSPLTODD =1 if necessary to separate chroma components.
	RGB565 sensor	No splitting possible. EPPI_CTL.PACKEN =1. Two EPPI words are packed to a 32-bit DMA data. EPPI_CTL.RGBFMTEN is valid only in transmit modes. So, RGB565 cannot be byte aligned in memory. EPPI_CTL.PACKEN =0. Each EPPI word is sent as a 16-bit data on the 32-bit DMA bus. This transfer consumes double the bandwidth of the 16-bit case with EPPI_CTL.PACKEN =1
	8-bit ADCs I/Q pair	EPPI_CTL.SPLTEO =1, EPPI_CTL.SPLTWRD =1, EPPI_CTL.SUBSPLTODD =0.
	16-bit ADCs I/Q pair	EPPI_CTL.SPLTEO =1, EPPI_CTL.SPLTWRD =0, EPPI_CTL.SUBSPLTODD =0.

Transmit Data Formats

The ***EPPI Transmit Data Formats*** table provides information about EPPI configuration for specific use models for transmit data.

Table 25-12: EPPI Transmit Data Formats

Output Data Width	Use Model	Splitting/Packing Options
8	NTSC/PAL data	EPPI_CTL.SPLTEO =1 EPPI_CTL.SUBSPLTODD =1 if the chroma components (U and V) come in separate DMA words.
	Serial RGB for lower-resolution LCDs	No splitting possible. EPPI_CTL.PACKEN =1. The 32-bit DMA data is unpacked to drive four EPPI words. EPPI_CTL.PACKEN =0. The lowest 8 bits of the DMA data is driven on the EPPI data and the rest of the 24 bits are discarded. This transfer consumes 4 times the DMA bandwidth of the 8-bit case with EPPI_CTL.PACKEN =1.
10	NTSC/PAL data	EPPI_CTL.SPLTEO =1 EPPI_CTL.SUBSPLTODD =1 if the chroma components (U and V) come in separate DMA words.
	DACs	EPPI_CTL.SPLTEO =1, EPPI_CTL.SUBSPLTODD =0.

Table 25-12: EPPI Transmit Data Formats (Continued)

Output Data Width	Use Model	Splitting/Packing Options
12	DACs	EPPI_CTL.SPLTEO =1, EPPI_CTL.SUBSPLTODD =0.
14	DACs	EPPI_CTL.SPLTEO =1, EPPI_CTL.SUBSPLTODD =0.
16	8-bit luma/chroma pair for NTSC or HD	EPPI_CTL.SPLTEO =1, EPPI_CTL.SPLTWRD =1, EPPI_CTL.SUBSPLTODD =1 if the chroma components (U and V) come in separate DMA words.
	16-bit luma/chroma pair for NTSC or HD	EPPI_CTL.SPLTEO =1, EPPI_CTL.SPLTWRD =0, EPPI_CTL.SUBSPLTODD =1 if the chroma components (U and V) come in separate DMA words.
	RGB565 LCD	No splitting possible. EPPI_CTL.RGBFMTEN =1. Takes RGB888 data from the memory and drops the LSBs from each component to drive out RGB565 data.
	8-bit ADCs I/Q pair	EPPI_CTL.SPLTEO =1, EPPI_CTL.SPLTWRD =1, EPPI_CTL.SUBSPLTODD = 0
	16-bit ADCs I/Q pair	EPPI_CTL.SPLTEO =1, EPPI_CTL.SPLTWRD =0, EPPI_CTL.SUBSPLTODD =1
18	RGB666 LCD	No splitting possible. EPPI_CTL.RGBFMTEN =1. Takes RGB888 data from the memory and drops the 2 LSBs from each component to drive out RGB666 data.

Data Transfer Modes

The following sections describe EPPI data transfer modes, including receive or transmit data packing, sign extension, zero fill, receive or transmit split modes, clock gating, delayed start, and data consistency management.

Data Packing for Receive Modes

For receive modes, if the EPPI_CTL.PACKEN bit =1 and the DMA is 32 bits, the EPPI packs the incoming data into 32-bit words based on the EPPI_CTL.DLEN and EPPI_CTL.SWAPEN bit settings. When EPPI_CTL.SWAPEN =0, the EPPI puts the first data in the least significant bits and when EPPI_CTL.SWAPEN =1, the EPPI puts the first data in the most significant bits. The packing options for the EPPI_CTL.DLEN bits are as follows.

- When EPPI_CTL.DLEN =8, four 8-bit words can be packed into one 32-bit word.
- When EPPI_CTL.DLEN =16, two 16-bit words can be packed into one 32-bit word.
- For EPPI_CTL.DLEN values that are more than 8 bits but less than 16 bits, two such words are either sign-extended or zero-filled to 16 bits and packed into one 32-bit word.
- When EPPI_CTL.DLEN =18, the EPPI sign-extends or zero-fills the 18-bit data to 24 bits and packs four 24-bit words into three 32-bit words.

- When EPPI_CTL.DLEN =24, the EPPI packs four 24-bit words into three 32-bit words.

When EPPI_CTL.PACKEN =0, the EPPI receives the incoming data and sends it on the bus as-is. If EPPI_CTL.DLEN is less than or equal to 16 bits, the DMA is a 16-bit DMA; otherwise it is a 32-bit DMA.

Data Packing for Transmit Modes

For transmit modes, if the EPPI_CTL.DLEN bit =1 and the DMA is a 32-bit DMA, the EPPI unpacks the 32-bit word according to the EPPI_CTL.DLEN and EPPI_CTL.SWAPEN bit settings.

If EPPI_CTL.SWAPEN =1, the EPPI transmits the most significant bits as the first data, and if EPPI_CTL.SWAPEN =0, the EPPI transmits the least significant bits as the first data. The unpacking options for the EPPI_CTL.DLEN bits are as follows.

- When EPPI_CTL.DLEN =8, the EPPI transmits one 32-bit word from memory as four 8-bit data words.
- For EPPI_CTL.DLEN values greater than 8 bits but less than or equal to 16 bits, the EPPI transmits one 32-bit word from memory as two 16-bit data words.
- When EPPI_CTL.DLEN =18 or 24, the EPPI transmits three 32-bit words from memory as four data words.

Sign-Extended and Zero-Filled Data

The following list describes the bit settings and functionality for sign-extending and zero-filling data.

- For EPPI_CTL.DLEN equal to 10, 12 or 14, data is zero-filled or sign-extended to 16 bits.
- For EPPI_CTL.DLEN equal to 18 bits, data is zero-filled or sign-extended to 24 bits if packing is enabled, and zero-filled or sign-extended to 32 bits if packing is disabled.
- For EPPI_CTL.DLEN equal to 24 bits, data is zero-filled or sign-extended to 32 bits if packing is disabled.
- For EPPI_CTL.DLEN equal to 8 bits, data is zero-filled or sign-extended to 16 bits if packing is disabled.
- If EPPI_CTL.SIGNEXT =1, then the data is sign-extended, otherwise it is zero-filled.

Split Receive Modes

The control register has three control bits for split receive modes: EPPI_CTL.SPLTEO, EPPI_CTL.SUBSPLTODD, and EPPI_CTL.DMACFG. Packing is not valid in split modes.

- If EPPI_CTL.SPLTEO =1, the EPPI splits the incoming data stream into two substreams, an even stream, and an odd stream, and packs them separately.
- The EPPI_CTL.SUBSPLTODD bit is available only when EPPI_CTL.SPLTEO =1. When EPPI_CTL.SUBSPLTODD =1, the EPPI subsplits the odd substream, and packs the streams separately.
- The EPPI_CTL.DMACFG bit is also available only if EPPI_CTL.SPLTEO =1. If EPPI_CTL.DMACFG =1, the EPPI uses two DMA channels and if EPPI_CTL.DMACFG =0, the EPPI uses only one DMA channel.

Split Transmit Modes

The EPPI_CTL register has three control bits for split transmit modes: EPPI_CTL.SPLTEO, EPPI_CTL.SUBSPLTODD, and EPPI_CTL.DMACFG. The DMA is always a 32-bit DMA. Packing is not valid in split modes.

- If EPPI_CTL.SPLTEO =1, the EPPI receives the Luma (Y3Y2Y1Y0) and interleaved Chroma (Cr1Cb1Cr0Cb0) data as 32 bits from the DMA channel. The EPPI interleaves the data to form a 4:2:2 YCrCb data stream to transmit.
- The EPPI_CTL.SUBSPLTODD bit is available only when EPPI_CTL.SPLTEO =1. In this case, if EPPI_CTL.SUBSPLTODD =1, the EPPI receives the Luma (Y3Y2Y1Y0) and deinterleaved Chroma (Cb3Cb2Cb1Cb0 and Cr3Cr2Cr1Cr0). The EPPI interleaves the data to form a 4:2:2 YCrCb data stream to transmit. (The EPPI does not decimate the chroma data when formatting it into 4:2:2.)
- The EPPI_CTL.DMACFG bit is also valid only if EPPI_CTL.SPLTEO =1. If EPPI_CTL.DMACFG =1, the EPPI uses two DMA channels and if EPPI_CTL.DMACFG =0, the EPPI uses only one DMA channel.

Clock Gating

In ITU-R BT.656 and GP 0/1/2 FS modes, EPPI_FS3 becomes a clock-gating input. This functionality is valid for both internally and externally sourced EPPI_CLK, in both receive and transmit modes. This clock gating signal must be synchronous with EPPI_CLK. The external device on the rising edge of EPPI_CLK must drive the clock gating signal. Its function is to hold the sync and data lines in their current state until EPPI_FS3 is driven low. There are no additional latency cycles upon coming out of clock gating mode.

When clock gating is not required, the EPPI_FS3 pin must either be tied to ground or configured to operate as another of its multiplexed functions.

In GP 2 FS transmit mode with internally generated frame syncs, the EPPI_FS3 pin functions as a data enable signal.

Support for Delayed Start of EPPI Frame Syncs

The EPPI supports a delayed start of the EPPI_FS1 and EPPI_FS2 frame syncs. The EPPI_FS1_DLY and EPPI_FS2_DLY registers are programmable registers corresponding to EPPI_FS1 (HSYNC) and EPPI_FS2 (VSYNC).

The delay programmed in these registers applies to the first active edge of the internally generated frame sync. The delay starts from the first EPPI_CLK edge. The delay counter runs only for the first time and then shuts off until the EPPI is reenabled. (The delay counter is the period counter itself, since they do not run together.) Program the delay registers prior to the first EPPI_CLK edge (similar to the width and period registers). The ***EPPI Delayed Frame Sync Generation*** figure shows the functioning of EPPI_FS1 and EPPI_FS2.

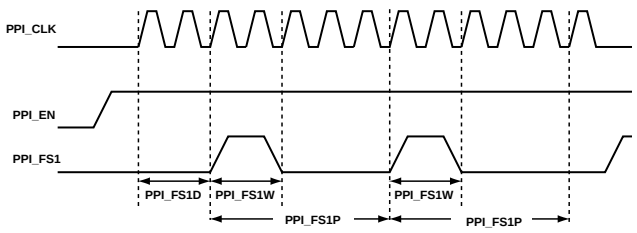


Figure 25-10: EPPI Delayed Frame Sync Generation

Ignoring Premature External Frame Syncs for Data Consistency

Once a frame has started with a VSYNC followed by an HSYNC (or both coming together), a line is tracked. When the count expires, the state machine waits at the end of line for an HSYNC to come. With the arrival of the HSYNC, the state machine starts tracking the next line, and so on.

The number of lines tracked is counted separately. Once the end of a frame is reached, the state machine waits there for the next VSYNC/HSYNC combination. The next frame starts once they are sampled. Unfortunately, every incoming FS (VSYNC or HSYNC) resets the respective counters and the tracking starts all over (even if the FS signals are premature). The result is incomplete data (or frames) to enter memory through the PxP interface.

To correct this problem, the EPPI waits for a frame or line completion before considering any incoming FS as valid.

- Single FS mode and line tracking in dual FS mode – When a line is in progress, when HSYNC is detected prematurely, it is ignored. A line track underflow event is generated.
- Dual FS mode – If a VSYNC is received when a frame is in progress, it is ignored. A frame track underflow error (`EPPI_STAT.FTERRUNDR`) is generated.

Ignoring the FS ensures that once a frame starts, the amount of data that goes into the memory/PxP interface corresponds exactly to the programmed data size in a frame.

NOTE: Even when the premature FS is a valid FS, the state machine loses at most one frame, and it recovers in the subsequent FS. The FS to number of data going into the memory relationship is always maintained as programmed.

When data underflow errors occur at the DMA interface, the EPPI does the following.

- If a premature line sync is detected, an LT underflow error is generated (`EPPI_STAT.LTERRUNDR` =1). All further line track errors are ignored until the EPPI detects the next valid line sync.
- If a premature frame sync is detected, an FT underflow error is generated (`EPPI_STAT.FTERRUNDR` =1). All further frame track and line track errors are ignored until the EPPI detects the next valid frame sync.

EPPI Event Control

The following sections describe how EPPI manages events.

EPPI Status, Error, and Interrupt Signals

The EPPI generates error interrupts (flagged in the EPPI_STAT register) when any one of the following error conditions occur.

- EPPI_STAT.YFIFOERR (YFIFO underflow or overflow)
- EPPI_STAT.CFIFOERR (CFIFO underflow or overflow)
- EPPI_STAT.LTERROVR (line track overflow error)
- EPPI_STAT.LTERRUNDR (line track underflow error)
- EPPI_STAT.FTERROVR (frame track overflow error)
- EPPI_STAT.FTERRUNDR (frame track underflow error)
- EPPI_STAT.ERRNCOR (ITU preamble error not corrected)

A W1C (write-1-to-clear) operation clears the error conditions. Each of the individual conditions which cause an EPPI error interrupt can be masked. The interrupt mask register (EPPI_IMSK) allows the masking of individual conditions which cause error interrupts.

There is only one interrupt line from each EPPI so all interrupts are internally OR'ed and sent as a single interrupt to the core. The EPPI_STAT register must then be read to discover specific errors. The following sections describe these errors in detail.

Frame and Line Track Errors

In external frame sync mode, the EPPI uses line track error (EPPI_STAT.LTERROVR and EPPI_STAT.LTERRUNDR) and frame track error (EPPI_STAT.FTERROVR and EPPI_STAT.FTERRUNDR) status bits to monitor the line and frame synchronization errors. The EPPI updates the bits when there is a mismatch detected in the HSYNC and VSYNC as compared to the programmed values in EPPI_LINE and EPPI_FRAME count registers.

Line Track Errors

The line track overflow (EPPI_STAT.LTERROVR) and underflow errors (EPPI_STAT.LTERRUNDR) generate a maskable interrupt as soon as the EPPI identifies them and not at the next frame sync.

- When the frame sync has not arrived when the EPPI_LINE counter expires, then the EPPI_STAT.LTERROVR error is generated.
- When the EPPI_LINE counter is running and a frame sync is detected, the EPPI_STAT.LTERRUNDR error is generated. A W1C operation clears both interrupts.

Frame Track Errors

The frame track overflow (EPPI_STAT.FTERROVR) and underflow errors (EPPI_STAT.FTERRUNDR) generate a maskable interrupt as soon as the EPPI identifies them. When the EPPI_FRAME.VALUE counter expires, the EPPI_STAT.FTERROVR error is reported before the next frame sync arrives.

When the EPPI_FRAME counter is running and a frame sync is detected, then an EPPI_STAT.FTERRUNDR is reported.

Both errors generate an error interrupt. Perform a W1C operation to clear the interrupts at their respective locations in the status register.

A premature frame sync results in a frame track under run error. But the error is logged (register bit set) only after the subsequent blanking period (if any) elapses.

Preamble Error Not Corrected Error

The EPPI supports data embedded frame syncs in ITU and SMPTE formats. In these formats, the module can receive an erroneous preamble which is not correctable. The EPPI_STAT.ERRNCOR error signals when this event occurs.

EPPI Programming Model

The following sections describe programming techniques, including receiving or transmitting ITU-R 656 frames; configuring transfers in GP0, GP1, GP2, and GP3 modes; and managing EPPI mode configurations.

Receiving ITU-R 656 Frames

The EPPI supports the reception of ITU-R 656 compliant frames.

1. Configure the EPPI to receive either full ITU-R 656 frame, active video, or blanking information by configuring the EPPI_CTL.XFRTYPE bits.
2. In both active video mode and in VBI (vertical blanking information) mode, specify the number of total (active plus vertical blanking) lines per frame in the EPPI_FRAME register. Specify the number of total (active plus horizontal blanking plus 8) samples per line in the EPPI_LINE register.
3. Configure DMA descriptors to move the data to memory.
4. Enable DMA.
5. Enable the EPPI.
6. To program the EPPI in internal clock mode, follow the procedure above with the EPPI_CTL.ICLKGEN bit =0. After enabling the EPPI, add a delay of 200 SCLK0 cycles (worst case) to ensure the EPPI FIFO becomes full. Then switch to internal clock mode by setting the EPPI_CTL.ICLKGEN bit =1.

Depending on the EPPI configuration, either the full ITU-R 656 frame is moved to memory or only the active video or only the blanking information.

Transmitting ITU-R 656 Frames in GP Transmit Modes

The EPPI can take active video from memory and generate the proper preambles and blanking information to produce valid ITU-R 656 video frames for transmission.

1. Provide active data frame in memory.
2. Set the EPPI_CTL.BLANKGEN bit so the EPPI generates blanking information.
3. Configure the EPPI_FS1_WLHB, EPPI_FS1_PASPL, EPPI_FS2_WLVB, EPPI_FS2_PALPF registers accordingly.
4. Configure the rest of the EPPI settings.
5. Configure DMA to fetch active frame data from memory buffers.
6. Enable DMA.
7. Enable the EPPI.
8. To program the EPPI in internal clock mode, follow the procedure above with the EPPI_CTL.ICLKGEN bit =0. After enabling the EPPI, add a delay of 200 SCLK0 cycles (worst case) to ensure the EPPI FIFO becomes full. Then switch to internal clock mode by setting the EPPI_CTL.ICLKGEN bit =1.

The EPPI takes the active data from memory, generates the blanking information, and transmits an ITU-R 656 frame

Configuring Transfers in GP 0 FS Mode

The EPPI can be configured to not use periodic frame syncs to frame the data.

1. Configure the EPPI to operate in GP 0 FS mode by setting EPPI_CTL.XFRTYPE = b#11 and EPPI_CTL.FSCFG = b#00.
2. When receiving, configure the EPPI to trigger on internally or externally by setting the EPPI_CTL.FLDSEL field appropriately. When transmitting, the EPPI always generates a trigger internally.
3. Configure DMA to move the data to or from memory.
4. Enable DMA.
5. Enable EPPI.
6. To program the EPPI in internal clock mode, follow the procedure above with the EPPI_CTL.ICLKGEN bit =0. After enabling the EPPI, add a delay of 200 SCLK0 cycles (worst case) to ensure the EPPI FIFO becomes full. Then switch to internal clock mode by setting the EPPI_CTL.ICLKGEN bit =1.

The DMA descriptions control the amount of data transferred. The frame syncs from the EPPI do not control the amount.

Configuring Transfers in GP 1 FS Mode

The GP 1 FS mode is useful for interfacing the EPPI with analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and other general-purpose devices. This mode works for both transmit and receive.

NOTE: The EPPI_FRAME, EPPI_VDLY, and EPPI_VCNT registers have no effect in GP 1 FS mode. As a result, frame track errors and vertical windowing are not possible in this mode.

1. Configure GP 1 FS mode by setting the EPPI_CTL.XFRTYPE bit =b#11 and the EPPI_CTL.FSCFG bit =b#01. An external device can provide the frame syncs or the EPPI can source the frame syncs.
2. Program the EPPI_LINE register to contain the number clock cycles expected between two assertions of the EPPI_FS1 signal to monitor the line track errors. Program the EPPI_LINE register before the EPPI_HCNT register.
3. Program the EPPI_HDLY register to contain the number of clock cycles to wait after the assertion of EPPI_FS1. For example, the start of frame.
4. Program the EPPI_HCNT register to contain the number of data samples to receive or transmit for each frame.
5. Configure DMA to move the data to or from memory.
6. Enable DMA.
7. Enable the EPPI.
8. To program the EPPI in internal clock mode, follow the procedure above with the EPPI_CTL.ICLKGEN bit =0. After enabling the EPPI, add a delay of 200 SCLK0 cycles (worst case) to ensure the EPPI FIFO becomes full. Then, switch to internal clock mode by setting the EPPI_CTL.ICLKGEN bit =1.

Data moves in or out of memory. A frame sync frames the data for every line.

Configuring Transfers in GP 2 FS Mode

GP 2 FS mode is useful for video applications that use two hardware synchronization signals, HSYNC and VSYNC. The HSYNC connects to the EPPI_FS1 signal and VSYNC connects to the EPPI_FS2 signal.

1. Configure the EPPI to operate in GP 2 FS mode by setting the EPPI_CTL.XFRTYPE bit =b#11 and the EPPI_CTL.FSCFG bit =b#10. An external device can provide the frame syncs or the EPPI can source the frame syncs.
2. Program the EPPI_FRAME register to contain the number of expected lines per frame. The value can be equal to the number of EPPI_FS1 signal assertions expected between the start of each frame sync. The EPPI uses the value to monitor frame track errors. Program the EPPI_FRAME register before the EPPI_VCNT register.
3. Program the EPPI_LINE register to contain the number of clock cycles expected between two assertions of the EPPI_FS1 signal to monitor line track errors. Program the EPPI_LINE register before the EPPI_HCNT register.
4. Program the EPPI_HDLY register to configure the number of clock cycles to wait after the assertion of the EPPI_FS1 signal. (For example, the start of the line).
5. Program the EPPI_HCNT register to contain the number of data samples to receive or transmit for each line.

6. Program the EPPI_VDLY register to contain the number of lines to wait after the start of frame is detected.
7. Program the EPPI_VCNT register to contain the number of lines to receive or transmit.
8. If setting up the EPPI for transmit, the data enable (DEN) pin behaves according to the enabling of the blanking generation and the data length setting (DLEN). See Data Enable in General-Purpose 2 Frame Sync Transmit Mode for more details.
9. Enable DMA.
10. Enable the EPPI.
11. To program the EPPI in internal clock mode, follow the procedure above with the EPPI_CTL.ICLKGEN bit =0. After enabling the EPPI, add a delay of 200 SCLK0 cycles (worst case) to ensure the EPPI FIFO becomes full. Then switch to internal clock mode by setting the EPPI_CTL.ICLKGEN bit =1.

Data moves in or out of memory. A frame sync frames the data for every line and frame.

Configuring Transfers in GP 3 FS Mode

GP 3 FS mode is useful for video applications that use three synchronization signals for hardware: HSYNC, VSYNC, and FIELD. The HSYNC connects to EPPI_FS1, VSYNC connects to EPPI_FS2, and FIELD connects to EPPI_FS3.

1. Configure the EPPI to operate in GP 3 FS mode by setting the EPPI_CTL.XFRTYPE bit =b#11 and the EPPI_CTL.FSCFG bit =b#11. An external device can provide the frame syncs or the EPPI can source the frame syncs.
2. Configure the windowing registers according to steps in GP 2 FS mode.
3. Enable DMA.
4. Enable the EPPI.
5. To program the EPPI in internal clock mode, follow the procedure above with the EPPI_CTL.ICLKGEN bit =0. After enabling the EPPI, add a delay of 200 SCLK0 cycles (worst case) to ensure the EPPI FIFO becomes full. Then switch to internal clock mode by setting the EPPI_CTL.ICLKGEN bit =1.

Data moves in or out of memory. A frame sync frames the data for every line and frame. Operation and results are like the operation in GP 2 FS mode, but the EPPI also uses the EPPI_FS3 signal.

Configuring the EPPI to Use the Windowing Feature

Windowing is a useful feature for applications where the region of interest is smaller than the active video stream (for example, sensor calibration, auto-focusing, and others). It can result in significant DMA bandwidth reduction. The EPPI supports windowing for GP input modes.

1. Program the EPPI_FRAME register with the number of lines the frame contains.

2. Program the EPPI_LINE register with the number of samples per line in the frame.
3. Program the EPPI_VDLY register with the number of lines to wait after the start of a new frame before starting to read or transmit data.
4. Program the EPPI_VCNT register with the number of lines to read in or write out after EPPI_VDLY number of lines from the start of the frame.
5. Program the EPPI_HDLY register with the number of clock cycles to delay after the assertion of EPPI_FS1 is detected for the start of a new line.
6. Program the EPPI_HCNT register with the number of samples to read in or write out after EPPI_HDLY number of cycles have expired since the assertion of EPPI_FS1.

EPPI Mode Configuration

This section describes EPPI mode configurations, including support for all EPPI transmit and receive modes.

Configuring 8-Bit Receive Mode

For 8-bit non-split receive mode and if EPPI_CTL.PACKEN =1, the EPPI packs 4 bytes of incoming data into a 32-bit word. Alternate even or odd samples can be skipped based on the EPPI_CTL.SKIPEN and EPPI_CTL.SKIPEO bits. The first incoming data can be placed either in the least significant bit positions or in the most significant bit positions, based on the EPPI_CTL.SWAPEN bit setting.

Table 25-13: 8-Bit Receive Mode with Packing Enabled

Pin Data (8 bits)	DMA DATA SKIPEN=0 SKIPEO =X SWAPEN=0 SIGNEXT=X	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=1 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=0 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=0 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=1 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=1 SIGNEXT=X
0x11						
0x22						
0x33						
0x44	0x4433 2211	0x1122 3344				
0x55						
0x66						
0x77			0x7755 3311		0x1133 5577	
0x88	0x8877 6655	0x5566 7788		0x8866 4422		0x2244 6688
0x99						
0xAA						
0xBB						

Table 25-13: 8-Bit Receive Mode with Packing Enabled (Continued)

Pin Data (8 bits)	DMA DATA SKIPEN=0 SKIPEO =X SWAPEN=0 SIGNEXT=X	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=1 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=0 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=0 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=1 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=1 SIGNEXT=X
0xCC	0xCCBB AA99	0x99AA BBCC				
0xDD						
0xEE						
0xFF			0xFFDD BB99		0x99BB DDFF	
0x00	0x00FF EEDD	0xDDE EFF00		0x00EE CCAA		0xAACC EE00

If `EPPI_CTL.PACKEN`=0, the DMA is a 16-bit DMA and the EPPI either sign-extends or zero-fills the bytes of incoming data into a 16-bit word. The `EPPI_CTL.SWAPEN` bit has no effect if `EPPI_CTL.PACKEN`=0.

Table 25-14: 8-Bit Receive Mode with Packing Disabled

Pin Data (8 bits)	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=X SIGNEXT=0	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=X SIGNEXT=1	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=X SIGNEXT=0	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=X SIGNEXT=1
0x44	0x0044	0x0044	0x0044	
0x55	0x0055	0x0055		0x0055
0x66	0x0066	0x0066	0x0066	
0x77	0x0077	0x0077		0x0077
0x88	0x0088	0xFF88	0x0088	
0x99	0x0099	0xFF99		0xFF99
0xAA	0x00AA	0xFFAA	0x00AA	
0xBB	0x00BB	0xFFBB		0xFFBB

Configuring 10/12/14-Bit Receive Modes

For 10, 12, or 14-bit non-split receive modes, the EPPI first either zero-fills or sign-extends the incoming data into a 16-bit word. The action depends on the setting of the `EPPI_CTL.SIGNEXT` bit. If `EPPI_CTL.PACKEN` =1, the EPPI then packs two of these words into one 32-bit word. Alternate even or odd samples can be skipped based on the `EPPI_CTL.SKIPEN` and `EPPI_CTL.SKIPEO` bits. The first incoming data can be placed either in the least significant bit positions or in the most significant bit positions, based on the `EPPI_CTL.SWAPEN` bit setting.

Table 25-15: 10-Bit Receive Mode with Sign Extension, with Packing Enabled

Pin Data (10 bits)	MSB	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=0 SIGNEXT=1	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=1 SIGNEXT=1	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=0 SIGNEXT=1
0x111	0			
0x222	1	0xFE22 0111	0x0111 FE22	
0x333	1			0xFF33 0111
0x044	0	0x0044 FF33	0xff33 0044	
0x155	0			
0x266	1	0xFE66 0155	0x0155 FE66	
0x377	1			0xFF77 0155
0x088	0	0x0088 FF77	0xFF77 0088	

Table 25-16: 10-Bit Receive Mode with Sign Extension, with Packing Enabled

Pin Data (10 bits)	MSB	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=0 SIGNEXT=1	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=1 SIGNEXT=1	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=1 SIGNEXT=1
0x111	0			
0x222	1			
0x333	1		0x0011 FF33	
0x044	0	0x0044 FE22		0xFE22 0044
0x155	0			
0x266	1			
0x377	1		0x0155 FF77	
0x088	0	0x0088 FE66		0xFE66 0088

Table 25-17: 10-Bit Receive Mode, with Zero-Fill, with Packing Enabled

Pin Data (10 bits)	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=0 SIGNEXT=0	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=1 SIGNEXT=0	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=0 SIGNEXT=0	DMA DATA SKIP_EN=1 SKIP_EO=0 SWAPEN=0 SIGNEXT=0	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=1 SIGNEXT=0	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=1 SIGNEXT=0
0x111						
0x222	0x0222 0111	0x0111 0222				
0x333			0x0333 0111		0x0011 0333	
0x044	0x0044 0333	0x0333 0044		0x0044 0222		0x0222 0044
0x155						
0x266	0x0266 0155	0x0155 0266				
0x377			0x0377 0155		0x0155 0377	
0x088	0x0088 0377	0x0377 0088		0x0088 0266		0x0266 0088

The ***10-bit Receive Mode with Packing Disabled*** table shows a 10-bit receive mode example when `EPPI_CTL.PACKEN` =0:

Table 25-18: 10-bit Receive Mode with Packing Disabled

Pin Data (10 bits)	MSB	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=X SIGNEXT=1	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=X SIGNEXT=0	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=X SIGNEXT=1	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=X SIGNEXT=0
0x111	0	0x0111	0x0111	0x0111	
0x222	1	0xFE22	0x0222		0x0222
0x333	1	0xFF33	0x0333	0xFF33	
0x044	0	0x0044	0x0444		0x0444
0x155	0	0x0155	0x0155	0x0155	
0x266	1	0xFE66	0x0266		0x0266
0x377	1	0xFF77	0x0377	0xFF77	
0x088	0	0x0088	0x0088		0x088

Configuring 16-Bit Receive Mode

For 16-bit non-split receive mode, if `EPPI_CTL.PACKEN` =1, the EPPI packs two 16-bit incoming data into one 32-bit word. Alternate even or odd samples can be skipped based on the `EPPI_CTL.SKIPEN` and

EPPI_CTL.SKIPEO bits. The first incoming data can be placed either in the least significant bit positions or in the most significant bit positions, based on the EPPI_CTL.SWAPEN bit setting.

Table 25-19: 16-Bit Receive Mode with Packing Enabled

Pin Data (16 bits)	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=0 SIGNEXT=X	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=1 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=0 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=0 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=1 SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=1 SIGNEXT=X
0x1111						
0x2222	0x2222 1111	0x1111 2222				
0x3333			0x3333 1111		0x1111 3333	
0x4444	0x4444 3333	0x3333 4444		0x4444 2222		0x2222 4444
0x5555						
0x6666	0x6666 5555	0x5555 6666				
0x7777			0x7777 5555		0x5555 7777	
0x8888	0x8888 7777	0x7777 8888		0x8888 6666		0x6666 8888

Table 25-20: 16-bit Receive Mode with Packing Disabled

Pin Data (16 bits)	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=X SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=X SIGNEXT=X	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=X SIGNEXT=X
0x1111	0x1111	0x1111	
0x2222	0x2222		0x2222
0x3333	0x3333	0x3333	
0x4444	0x4444		0x4444
0x5555	0x5555	0x5555	
0x6666	0x6666		0x6666
0x7777	0x7777	0x7777	
0x8888	0x8888		0x8888

Configuring 18-Bit Receive Mode

For 18-bit non-split receive mode, if EPPI_CTL.PACKEN =0, the EPPI zero-fills or sign-extends the incoming data into a 32-bit word. If EPPI_CTL.PACKEN =1, the EPPI first zero-fills or sign-extends the incoming data to 24 bits, and then packs four such 24-bit data words into three 32-bit words. Alternate even or odd samples can be

skipped based on the EPPI_CTL.SKIPEN and EPPI_CTL.SKIPEO bits. The EPPI_CTL.SWAPEN bit has no effect.

Table 25-21: 18-bit Receive Mode with Packing Disabled

Pin Data (18 bits)	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=X SIGNEXT=0	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=X SIGNEXT=0	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=X SIGNEXT=0
0x0 6666	0x0000 6666	0x0000 6666	
0x1 7777	0x0001 7777		0x0001 7777
0x2 8888	0x0002 8888	0x0002 8888	
0x3 9999	0x0003 9999		0x0003 9999

Table 25-22: 18-bit Receive Mode with Packing Enabled

Pin Data (18 bits)	DMA DATA SKIPEN=0 SKIPEO=X SWAPEN=X SIGNEXT=0	DMA DATA SKIPEN=1 SKIPEO=1 SWAPEN=X SIGNEXT=0	DMA DATA SKIPEN=1 SKIPEO=0 SWAPEN=X SIGNEXT=0
0x0 1122			
0x1 3344	0x4400 1122		
0x2 5566	0x5566 0133	0x6600 1122	
0x3 7788	0x0377 8802		0x8801 3344
0x0 99AA		0x99AA 0255	
0x1 BBCC	0xCC00 99AA		0xBBCC 0377
0x2 DDEE	0xDDEE 01BB	0x02DD EE00	
0x3 FF12	0x03FF 122D		0x03FF 1201

Configuring 8-Bit Split Receive Mode

For 8-bit split receive mode, the EPPI_CTL.PACKEN and EPPI_CTL.SIGNEXT bits are not valid. The EPPI always packs 4 bytes of data into one 32-bit word.

Table 25-23: 8-bit Split Receive Mode with SKIPEN = 0 and SWAPEN = 0

Pin Data (8 bits)	SPLTEO=1 SUBSPLTODD= 0 SWAPEN=0 SKIPEN=0 SKIPEO=X			SPLTEO=1 SUBSPLTODD= 1 SWAPEN=0 SKIPEN=0 SKIPEO=X		
	DMACFG=1		DMACFG=0	DMACFG=1		DMACFG=0
	Primary DMA Channel	Secondary DMA Channel	Primary DMA Channel	Primary DMA Channel	Secondary DMA Channel	Primary DMA Channel
V_0						
Y_0						
U_0						
Y_1						
V_1						
Y_2						
U_1		$U_1V_1U_0V_0$	$U_1V_1U_0V_0$			
Y_3	$Y_3Y_2Y_1Y_0$		$Y_3Y_2Y_1Y_0$	$Y_3Y_2Y_1Y_0$		$Y_3Y_2Y_1Y_0$
V_2						
Y_4						
U_2						
Y_5						
V_3					$V_3V_2V_1V_0$	$V_3V_2V_1V_0$
Y_6						
U_3		$U_3V_3U_2V_2$	$U_3V_3U_2V_2$		$U_3U_2U_1U_0$	
Y_7	$Y_7Y_6Y_5Y_4$		$Y_7Y_6Y_5Y_4$	$Y_7Y_6Y_5Y_4$		$Y_7Y_6Y_5Y_4$
V_4						$U_3U_2U_1U_0$

Table 25-24: 8-bit Split Receive Mode with SKIPEN = 0 and SWAPEN = 1

Pin Data (8 bits)	SPLTEO=1 SUBSPLTODD=0 SWAPEN=1 SKIPEN=0 SKIPEO=X			SPLTEO=1 SUBSPLTODD=1 SWAPEN=1 SKIPEN=0 SKIPEO=X		
	DMACFG=1		DMACFG=0	DMACFG=1		DMACFG=0
	PRIMARY DMA CHANNEL	SECONDARY DMA CHANNEL	PRIMARY DMA CHANNEL	PRIMARY DMA CHANNEL	SECONDARY DMA CHANNEL	PRIMARY DMA CHANNEL
V_0						
Y_0						
U_0						
Y_1						
V_1						
Y_2						
U_1		$V_0U_0V_1U_1$	$V_0U_0V_1U_1$			
Y_3	$Y_0Y_1Y_2Y_3$		$Y_0Y_1Y_2Y_3$	$Y_0Y_1Y_2Y_3$		$Y_0Y_1Y_2Y_3$
V_2						
Y_4						
U_2						
Y_5						
V_3					$V_0V_1V_2V_3$	$V_0V_1V_2V_3$
Y_6						
U_3		$V_2U_2V_3U_3$	$V_2U_2V_3U_3$		$U_0U_1U_2U_3$	
Y_7	$Y_4Y_5Y_6Y_7$		$Y_4Y_5Y_6Y_7$	$Y_4Y_5Y_6Y_7$		$Y_4Y_5Y_6Y_7$
V_4						$U_0U_1U_2U_3$

When the bits settings are `EPPI_CTL.SPLTEO` =1, `EPPI_CTL.SUBSPLTODD` =1 and `EPPI_CTL.DMACFG` =0, the EPPI packs the second Chroma component sent over the DMA bus completely before the Luma component. However, it is intentionally held until that previous word is moved out. This functionality allows the separation of Luma and Chroma values into individual buffers when using 2D-DMA. The second Chroma component is U0U1U2U3 in the ***8-bit Split Receive Mode with SKIPEN = 0 and SWAPEN = 0*** and ***8-bit Split Receive Mode with SKIPEN = 0 and SWAPEN = 1*** tables. The Luma component is Y4Y5Y6Y7 in the ***8-bit Split Receive Mode with SKIPEN = 0 and SWAPEN = 1*** table.

Configuring 10/12/14/16-Bit Split Receive Mode with SPLTWRD=0

For 16-bit split receive mode, the `EPPI_CTL.PACKEN` bit is not valid. The EPPI always packs two 16-bit words into one 32-bit word. For 10, 12, or 14-bit split receive modes, the EPPI first either sign-extends or zero-fills the incoming data into a 16-bit word. The EPPI then packs two of these words into one 32-bit word to send to the DMA.

Table 25-25: 16-bit Split Receive Mode with SPLTWRD = 0, SKIPEN = 0 and SWAPEN = 0

Pin Data (16 bits)	**SPLTEO=1 SUBSPLTODD=0 SWAPEN=0 SKIPEN=0 SKIPEO=X**			**SPLTEO=1 SUBSPLTODD=1 SWAPEN=0 SKIPEN=0 SKIPEO=X**		
	DMACFG=1		**DMACFG=0**	**DMACFG=1**		**DMACFG=0**
	Primary DMA Channel	**Secondary DMA Channel**	**Primary DMA Channel**	**Primary DMA Channel**	**Secondary DMA Channel**	**Primary DMA Channel**
V_0						
Y_0						
U_0		U_0V_0	U_0V_0			
Y_1	Y_1Y_0		Y_1Y_0	Y_1Y_0		Y_1Y_0
V_1					V_1V_0	V_1V_0
Y_2						
U_1		U_1V_1	U_1V_1		U_1U_0	
Y_3	Y_3Y_2		Y_3Y_2	Y_3Y_2		Y_3Y_2
V_2						U_1U_0

Table 25-26: 16-bit Split Receive Mode with SPLTWRD = 0, SKIPEN = 0 and SWAPEN = 1

Pin Data (16 bits)	SPLTEO=1 SUBSPLTODD=0 SWAPEN=1 SKIPEN=0 SKIPEO=X			SPLTEO=1 SUBSPLTODD=1 SWAPEN=1 SKIPEN=0 SKIPEO=X		
	DMACFG=1		DMACFG=0	DMACFG=1		DMACFG=0
	PRIMARY DMA CHANNEL	SECONDARY DMA CHANNEL	PRIMARY DMA CHANNEL	PRIMARY DMA CHANNEL	SECONDARY DMA CHANNEL	PRIMARY DMA CHANNEL
V_0						
Y_0						
U_0		V_0U_0	V_0U_0			
Y_1	Y_0Y_1		Y_0Y_1	Y_0Y_1		Y_0Y_1
V_1					V_0V_1	V_0V_1
Y_2						
U_1		V_1U_1	V_1U_1		U_0U_1	
Y_3	Y_2Y_3		Y_2Y_3	Y_2Y_3		Y_2Y_3
V_2						U_0U_1

Configuring 16-Bit Split Receive Mode with SPLTWRD=1

For 16-bit split receive mode, the `EPPI_CTL.PACKEN` bit is not valid. The EPPI always packs two 16-bit words into one 32-bit word. The `EPPI_CTL.SPLTWRD` bit is only valid when the `EPPI_CTL.DLEN` bit =16 bits.

Table 25-27: 16-bit Split Receive Mode with SPLTWRD = 1, SKIPEN = 0 and SWAPEN = 0

Pin Data (16 bits)	SPLTEO=1 SUBSPLTODD=0 SWAPEN=0 SKIPEN=0 SKIPEO=X			SPLT_EVEN_ODD=1 SUBSPLTODD=1 SWAPEN=0 SKIPEN=0 SKIPEO=X		
	DMACFG=1		DMACFG=0	DMACFG=1		DMACFG=0
	Primary DMA Channel	Secondary DMA Channel	Primary DMA Channel	Primary DMA Channel	Secondary DMA Channel	Primary DMA Channel
V_0Y_0						

Table 25-27: 16-bit Split Receive Mode with SPLTWRD = 1, SKIPEN = 0 and SWAPEN = 0 (Continued)

Pin Data (16 bits)	SPLTEO=1 SUBSPLTODD=0 SWAPEN=0 SKIPEN=0 SKIPEO=X			SPLT_EVEN_ODD=1 SUBSPLTODD=1 SWAPEN=0 SKIPEN=0 SKIPEO=X		
	DMACFG=1		DMACFG=0	DMACFG=1		DMACFG=0
	Primary DMA Channel	Secondary DMA Channel	Primary DMA Channel	Primary DMA Channel	Secondary DMA Channel	Primary DMA Channel
U_0Y_1						
V_1Y_2						
U_1Y_3	$Y_3Y_2Y_1Y_0$	$U_1V_1U_0V_0$	$Y_3Y_2Y_1Y_0$	$Y_3Y_2Y_1Y_0$		$Y_3Y_2Y_1Y_0$
V_2Y_4			$U_1V_1U_0V_0$			
U_2Y_5						
V_3Y_6					$V_3V_2V_1V_0$	$V_3V_2V_1V_0$
U_3Y_7	$Y_7Y_6Y_5Y_4$	$U_3V_3U_2V_2$	$Y_7Y_6Y_5Y_4$	$Y_7Y_6Y_5Y_4$	$U_3U_2U_1U_0$	$Y_7Y_6Y_5Y_4$
V_4Y_8			$U_3V_3U_2V_2$			$U_3U_2U_1U_0$

Configuring 8-Bit Transmit Mode

For 8-bit non-split transmit mode, if the `EPPI_CTL.PACKEN` bit =1, the DMA is a 32-bit DMA and the EPPI unpacks the 32-bit word from memory into 4 bytes to transmit. The EPPI transmits either the MSBs or the LSBs as the first data, depending on the `EPPI_CTL.SWAPEN` bit setting. If `EPPI_CTL.PACKEN` =0, the DMA is a 16-bit DMA and the EPPI transmits the lower 8 bits. The `EPPI_CTL.SWAPEN` bit has no effect when `EPPI_CTL.PACKEN` =0.

Table 25-28: 8-bit Transmit Mode with Packing Enabled

DMA Data (32 bits)	Pin Data when SWAPEN=0	Pin Data when SWAPEN=1
0x11223344	0x44	0x11
0x55667788	0x33	0x22
	0x22	0x33
	0x11	0x44
	0x88	0x55
	0x77	0x66
	0x66	0x77

Table 25-28: 8-bit Transmit Mode with Packing Enabled (Continued)

DMA Data (32 bits)	Pin Data when SWAPEN=0	Pin Data when SWAPEN=1
	0x55	0x88

Table 25-29: Data Sent in 8-bit Transmit Mode with Packing Disabled

DMA Data (16 bits)	Pin Data SWAPEN=X
0x1234	0x34
0x2345	0x45
0x3456	0x56

Configuring 10/12/14-Bit Transmit Modes

For 10, 12, or 14-bit non-split transmit modes, if the `EPPI_CTL.PACKEN` bit =1, the DMA is a 32-bit DMA. The EPPI unpacks the 32-bit word from memory into two 16-bit data words. The EPPI then transmits the required LSBs from each data word. The EPPI transmits either the most significant word or the least significant word as the first data, depending on the `EPPI_CTL.SWAPEN` bit setting. If `EPPI_CTL.PACKEN` =0, the DMA is a 16-bit DMA and the EPPI transmits the required LSBs. The `EPPI_CTL.SWAPEN` bit has no effect when the `EPPI_CTL.PACKEN` bit =0.

Table 25-30: 10-bit Transmit Mode with Packing Enabled

DMA Data (32 bits)	Pin Data when SWAPEN=0	Pin Data when SWAPEN=1
0x1111 2222	0x222	0x111
0x3333 4444	0x111	0x222
	0x044	0x333
	0x333	0x044

Table 25-31: 10-bit Transmit Mode with Packing Disabled

DMA Data (16 bits)	Pin Data SWAPEN=X
0x1234	0x234
0x2345	0x345
0x3456	0x056
0x4567	0x167

Configuring 16-Bit Transmit Mode

For 16-bit non-split transmit mode, if the `EPPI_CTL.PACKEN` bit =1, the DMA is a 32-bit DMA. The EPPI unpacks the 32-bit word from memory into two 16-bit data words to transmit. The EPPI transmits either the MSBs or the LSBs as the first data, depending on the `EPPI_CTL.SWAPEN` bit setting. If the `EPPI_CTL.PACKEN` bit =0, the DMA is a 16-bit DMA and the EPPI transmits the data as is. The `EPPI_CTL.SWAPEN` has no effect when `EPPI_CTL.PACKEN` bit =0.

Table 25-32: 16-bit Transmit Mode with Packing Enabled

DMA Data (32 bits)	Pin Data when SWAPEN=0	Pin Data when SWAPEN=1
0x1111 2222	0x2222	0x1111
0x3333 4444	0x1111	0x2222
	0x4444	0x3333
	0x3333	0x4444

Table 25-33: 16-bit Transmit Mode with Packing Disabled

DMA Data (16 bits)	Pin Data SWAPEN=X
0x1234	0x1234
0x2345	0x2345
0x3456	0x3456

Configuring 18-Bit Transmit Mode

For 18-bit transmit mode, if the `EPPI_CTL.PACKEN` bit =1, the DMA is a 32-bit DMA and the EPPI unpacks the 32-bit word from memory. In this mode, when `EPPI_CTL.RGBFMTEN` is set, the least significant 2 bits of R, G, and B are dropped.

Table 25-34: 18-bit Transmit Mode with Packing Enabled

DMA Data	Pin Data (18-bits)	
	RGBFMTEN=0	RGBFMTEN=1
0x0123 4567	0x3 4567	0x0 8459
0x89AB CDEF	0x1 EF01	0x3 3EC0
0x0123 4567	0x3 89AB	0x1 98AA
	0x1 2345	0x0 0211

Table 25-35: 18-bit Transmit Mode with Packing Disabled

DMA Data	Pin Data (18-bits)	
	RGBFMTEN=0	RGBFMTEN=1
0x0123 4567	0x3 4567	0x0 8459
0x89AB CDEF	0x3 CDEF	0x2 ACFB
0x0123 4567	0x3 4567	0x0 8459

Configuring 8-Bit Split Transmit Mode

For 8-bit split transmit mode, the `EPPI_CTL.PACKEN` bit is not valid. The EPPI always unpacks the 32-bit DMA data into 4 bytes to transmit.

Table 25-36: 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=0 and SWAPEN=0

DMACFG=1			DMACFG=0	
DMA0 DATA (32 bits)	**DMA1 DATA (32 bits)**	**Pin Data (8 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (8 bits)**
$Y_3Y_2Y_1Y_0$	$U_1V_1U_0V_0$	V_0	$U_1V_1U_0V_0$	V_0
$Y_7Y_6Y_5Y_4$	$U_3V_3U_2V_2$	Y_0	$Y_3Y_2Y_1Y_0$	Y_0
		U_0	$U_3V_3U_2V_2$	U_0
		Y_1	$Y_7Y_6Y_5Y_4$	Y_1
		V_1		V_1
		Y_2		Y_2
		U_1		U_1
		Y_3		Y_3
		V_2		V_2
		Y_4		Y_4
		U_2		U_2
		Y_5		Y_5
		V_3		V_3
		Y_6		Y_6
		U_3		U_3
		Y_7		Y_7

Table 25-37: 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=1 and SWAPEN=0

DMACFG=1			DMACFG=0	
DMA0 DATA (32 bits)	**DMA1 DATA (32 bits)**	**Pin Data (8 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (8 bits)**
$Y_3Y_2Y_1Y_0$	$V_3V_2V_1V_0$	V_0	$V_3V_2V_1V_0$	V_0
$Y_7Y_6Y_5Y_4$	$U_3U_2U_1U_0$	Y_0	$Y_3Y_2Y_1Y_0$	Y_0
	$V_7V_6V_5V_4$	U_0	$U_3U_2U_1U_0$	U_0
	$U_7U_6U_5U_4$	Y_1	$Y_7Y_6Y_5Y_4$	Y_1
		V_1		V_1
		Y_2		Y_2
		U_1		U_1
		Y_3		Y_3
		V_2		V_2

Table 25-37: 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=1 and SWAPEN=0 (Continued)

DMACFG=1			DMACFG=0	
DMA0 DATA (32 bits)	**DMA1 DATA (32 bits)**	**Pin Data (8 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (8 bits)**
		Y_4		Y_4
		U_2		U_2
		Y_5		Y_5
		V_3		V_3
		Y_6		Y_6
		U_3		U_3
		Y_7		Y_7

Table 25-38: 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=0 and SWAPEN=1

DMACFG=1			DMACFG=0	
DMA0 DATA (32 bits)	**DMA1 DATA (32 bits)**	**Pin Data (8 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (8 bits)**
$Y_3Y_2Y_1Y_0$	$U_1V_1U_0V_0$	U_1	$U_1V_1U_0V_0$	U_1
$Y_7Y_6Y_5Y_4$	$U_3V_3U_2V_2$	Y_3	$Y_3Y_2Y_1Y_0$	Y_3
		V_1	$U_3V_3U_2V_2$	V_1
		Y_2	$Y_7Y_6Y_5Y_4$	Y_2
		U_0		U_0
		Y_1		Y_1
		V_0		V_0
		Y_0		Y_0
		U_3		U_3
		Y_7		Y_7
		V_3		V_3
		Y_6		Y_6
		U_2		U_2
		Y_5		Y_5
		V_2		V_3
		Y_4		Y_4

Table 25-39: 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=1, and SWAPEN=1

DMACFG=1			DMACFG=0	
DMA0 DATA (32 bits)	**DMA1 DATA (32 bits)**	**Pin Data (8 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (8 bits)**
$Y_3Y_2Y_1Y_0$	$V_3V_2V_1V_0$	V_3	$V_3V_2V_1V_0$	V_3
$Y_7Y_6Y_5Y_4$	$U_3U_2U_1U_0$	Y_3	$Y_3Y_2Y_1Y_0$	Y_3
	$V_7V_6V_5V_4$	U_3	$U_3V_3U_2V_2$	U_3
	$U_7U_6U_5U_4$	Y_2	$Y_7Y_6Y_5Y_4$	Y_2
		V_2		V_2
		Y_1		Y_1
		U_2		U_2
		Y_0		Y_0
		V_1		V_1
		Y_7		Y_7
		U_1		U_1
		Y_6		Y_6
		V_0		V_0
		Y_5		Y_5
		U_0		U_0
		Y_4		Y_4

Configuring 10/12/14/16-Bit Transmit Mode with SPLTWRD=0

For 16-bit split transmit mode, the `EPPI_CTL.PACKEN` bit is not valid. The EPPI always unpacks the 32-bit DMA data into two 16-bit words to transmit. For 10, 12, or 14-bit split transmit modes, the EPPI first unpacks the data in the same way as for 16-bit transmit mode. But, the EPPI transmits only the required number of LSBs.

Table 25-40: 16-bit Split Transmit Mode with SPLTEO = 1, SUBSPLTODD = 0, and SWAPEN = 0

DMACFG = 1			DMACFG = 0	
DMA0 DATA (32 bits)	**DMA1 DATA (32 bits)**	**Pin Data (16 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (16 bits)**
Y_1Y_0	U_0V_0	V_0	U_0V_0	V_0
Y_3Y_2	U_1V_1	Y_0	Y_1Y_0	Y_0
		U_0	U_1V_1	U_0
		Y_1	Y_3Y_2	Y_1
		V_1		V_1

Table 25-40: 16-bit Split Transmit Mode with SPLTEO = 1, SUBSPLTODD = 0, and SWAPEN = 0 (Continued)

DMACFG = 1			DMACFG = 0	
DMA0 DATA (32 bits)	DMA1 DATA (32 bits)	Pin Data (16 bits)	DMA0 DATA (32 bits)	Pin Data (16 bits)
		Y_2		Y_2
		U_1		U_1
		Y_3		Y_3

Table 25-41: 16-bit Split Transmit Mode with SPLTEO = 1, SUBSPLTODD = 1, and SWAPEN = 0

DMACFG = 1			DMACFG = 0	
DMA0 DATA (32 bits)	DMA1 DATA (32 bits)	Pin Data (16 bits)	DMA0 DATA (32 bits)	Pin Data (16 bits)
Y_1Y_0	V_1V_0	V_0	V_1V_0	V_0
Y_3Y_2	U_1U_0	Y_0	Y_1Y_0	Y_0
	V_3V_2	U_0	U_1U_0	U_0
	U_3U_2	Y_1	Y_3Y_2	Y_1
		V_1		V_1
		Y_2		Y_2
		U_1		U_1
		Y_3		Y_3

Table 25-42: 16-bit Split Transmit Mode with SPLTEO = 1, SUBSPLTODD = 0, and SWAPEN = 1

DMACFG = 1			DMACFG = 0	
DMA0 DATA (32 bits)	DMA1 DATA (32 bits)	Pin Data (16 bits)	DMA0 DATA (32 bits)	Pin Data (16 bits)
Y_1Y_0	V_0U_0	V_0	V_0U_0	V_0
Y_3Y_2	V_1U_1	Y_1	Y_1Y_0	Y_1
		U_0	V_1U_1	U_0
		Y_0	Y_3Y_2	Y_0
		V_1		V_1
		Y_3		Y_3
		U_1		U_1
		Y_2		Y_2

Table 25-43: 16-bit Split Transmit Mode with SPLTEO = 1, SUBSPLTODD = 1, and SWAPEN = 1

DMACFG = 1			DMACFG = 0	
DMA0 DATA (32 bits)	**DMA1 DATA (32 bits)**	**Pin Data (16 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (16 bits)**
Y_1Y_0	V_1V_0	V_1	V_1V_0	V_1
Y_3Y_2	U_1U_0	Y_1	Y_1Y_0	Y_1
	V_3V_2	U_1	U_1U_0	U_1
	U_3U_2	Y_0		Y_0
		V_0		V_0
		Y_3		Y_1
		U_0		U_0
		Y_2		Y_2

Configuring 16-Bit Split Transmit Mode with SPLTWRD=1

For 16-bit split transmit mode, the `EPPI_CTL.PACKEN` bit is not valid. The EPPI always unpacks the 32-bit DMA data into two 16-bit words to transmit. The `EPPI_CTL.SPLTWRD` bit is only valid when the `EPPI_CTL.DLEN` bit =16 bits.

Table 25-44: 16-bit Split Transmit Mode with SPLTWRD = 1, SUBSPLTODD = 0, and SWAPEN = 0

DMACFG = 1			DMACFG = 0	
DMA0 DATA (32 bits)	**DMA1 DATA (32 bits)**	**Pin Data (16 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (16 bits)**
$Y_3Y_2Y_1Y_0$	$U_1V_1U_0V_0$	V_0Y_0	$U_1V_1U_0V_0$	V_0Y_0
$Y_7Y_6Y_5Y_4$	$U_3V_3U_2V_2$	U_0Y_1	$Y_3Y_2Y_1Y_0$	U_0Y_1
		V_1Y_2	$U_3V_3U_2V_2$	V_1Y_2
		U_1Y_3	$Y_7Y_6Y_5Y_4$	U_1Y_3
		V_2Y_4		V_2Y_4
		U_2Y_5		U_2Y_5
		V_3Y_6		V_3Y_6
		U_3Y_7		U_3Y_7

Table 25-45: 16-bit Split Transmit Mode with SPLTWRD = 1, SUBSPLTODD = 1, and SWAPEN = 0

DMACFG = 1			DMACFG = 0	
PRIMARY DMA DATA (32 bits)	**SECONDARY DMA DATA (32 bits)**	**Pin Data (16 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (16 bits)**
$Y_3Y_2Y_1Y_0$	$V_3V_2V_1V_0$	V_0Y_0	$V_3V_2V_1V_0$	V_0Y_0

Table 25-45: 16-bit Split Transmit Mode with SPLTWRD = 1, SUBSPLTODD = 1, and SWAPEN = 0 (Continued)

DMACFG = 1			DMACFG = 0	
PRIMARY DMA DATA (32 bits)	**SECONDARY DMA DATA (32 bits)**	**Pin Data (16 bits)**	**DMA0 DATA (32 bits)**	**Pin Data (16 bits)**
$Y_7Y_6Y_5Y_4$	$U_3U_2U_1U_0$	U_0Y_1	$Y_3Y_2Y_1Y_0$	U_0Y_1
	$V_7V_6V_5V_4$	V_1Y_2	$U_3U_2U_1U_0$	V_1Y_2
	$U_7U_6U_5U_4$	U_1Y_3	$Y_7Y_6Y_5Y_4$	U_1Y_3
		V_2Y_4		V_2Y_4
		U_2Y_5		U_2Y_5
		V_3Y_6		V_3Y_6
		U_3Y_7		U_3Y_7

EPPI Programming Concepts

This section provides information on SMPTE programming.

SMPTE Modes Programming

The programming model of SMPTE modes is like ITU Modes. All programming modes pertaining to ITU modes such as XFRTYPE, FSCFG, FLDSEL, and BLANKGEN also hold true for SMPTE modes The only difference is that because ITU modes use Y-Cr/Cb interleaved data and SMPTE use parallel Y-Cr/Cb data, SPLTWRD could be set while operating in SMPTE modes. The ***Programming Modes for SMPTE Formats*** table describes the programming modes for different SMPTE formats.

Table 25-46: Programming Modes for SMPTE Formats

SMPTE Format	SMPTE Channel Width	EPPI Input Bit Width	EPPI Mode	Remarks
296M	8	16 Cr/Cb - [15:8] Y - [7:0]	DLEN = 16 bits SPLTWRD = 1	SIGNEXT not supported
	8	16 Cr/Cb - [15:8] Y - [7:0]	DLEN = 16 bits SPLTWRD = 1	SIGNEXT not supported

ADSP-SC59x EPPI Register Descriptions

Enhanced Parallel Peripheral Interface (EPPI) contains the following registers.

Table 25-47: ADSP-SC59x EPPI Register List

Name	Description
EPPI_CLKDIV	Clock Divide Register
EPPI_CTL	Control Register
EPPI_CTL2	Control Register 2 Register
EPPI_EVENCLIP	Clipping Register for EVEN (Luma) Data Register
EPPI_FRAME	Lines Per Frame Register
EPPI_FS1_DLY	Frame Sync 1 Delay Value Register
EPPI_FS1_PASPL	FS1 Period Register / EPPI Active Samples Per Line Register
EPPI_FS1_WLHB	FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register
EPPI_FS2_DLY	Frame Sync 2 Delay Value Register
EPPI_FS2_PALPF	FS2 Period Register / EPPI Active Lines Per Field Register
EPPI_FS2_WLVB	FS2 Width Register / EPPI Lines Of Vertical Blanking Register
EPPI_HCNT	Horizontal Transfer Count Register
EPPI_HDLY	Horizontal Delay Count Register
EPPI_IMSK	Interrupt Mask Register
EPPI_LINE	Samples Per Line Register
EPPI_ODDCLIP	Clipping Register for ODD (Chroma) Data Register
EPPI_STAT	Status Register
EPPI_VCNT	Vertical Transfer Count Register
EPPI_VDLY	Vertical Delay Count Register

Clock Divide Register

The EPPI_CLKDIV register provides the divisor for EPPI internal clock generation. The generated clock frequency is given by following formula:

EPPI_CLK = (SCLK0) / (EPPI_CLKDIV + 1)

Note that a value of 0xFFFF is invalid for the EPPI_CLKDIV register.

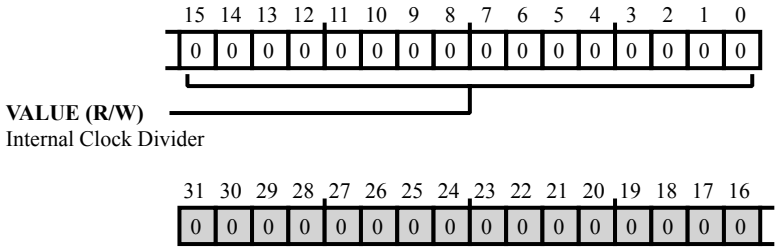


Figure 25-11: EPPI_CLKDIV Register Diagram

Table 25-48: EPPI_CLKDIV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Internal Clock Divider.

Control Register

The EPPI_CTL register configures the EPPI for operating mode, control signal polarities, and data width of the port.

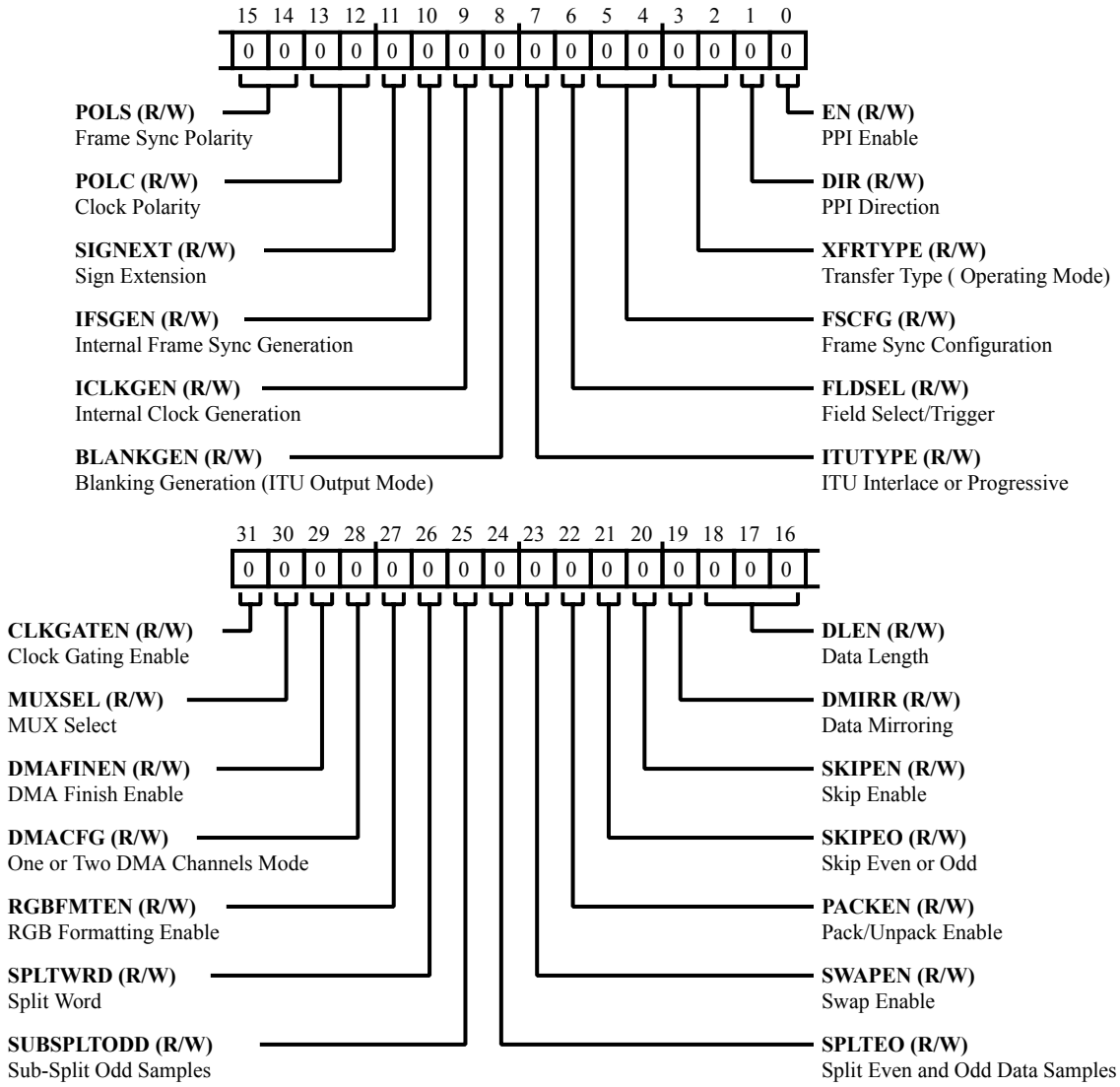


Figure 25-12: EPPI_CTL Register Diagram

Table 25-49: EPPI_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKGATEN	Clock Gating Enable. The EPPI_CTL.CLKGATEN bit enables using the EPPI_FS3 pin as a clock gating pin. When EPPI_CTL.CLKGATEN is set, the EPPI_FS3 pin acts as a clock gating signal, and both the internal and external clock are gated. Note that the EPPI_FS3 pin gating signal is active low, and the EPPI_CTL.CLKGATEN selection is ignored if EPPI_CTL.MUXSEL is set or EPPI_CTL.FSCFG equals 0x3.	
		0	Disable
		1	Enable
30 (R/W)	MUXSEL	MUX Select. The EPPI_CTL.MUXSEL bit enables multiplexing of a primary and alternate camera using the EPPI main data and clock lines. For more information on this feature, see the EPPI functional description.	
		0	Normal Operation
		1	Multiplexed Operation
29 (R/W)	DMAFINEN	DMA Finish Enable. The EPPI_CTL.DMAFINEN bit selects whether or not the EPPI sends a finish command (010) through the DDE COMMAND line soon after a frame/line is received completely.	
		0	No Finish Command
		1	Enable Send Finish Command
28 (R/W)	DMACFG	One or Two DMA Channels Mode. The EPPI_CTL.DMACFG bit is valid only if EPPI_CTL.SPLTEO is set. If EPPI_CTL.DMACFG is set, the EPPI uses two DMA channels. And, if EPPI_CTL.DMACFG is cleared, the EPPI uses only one DMA channel.	
		0	PPI Uses One DMA Channel
		1	PPI Uses Two DMA Channels
27 (R/W)	RGBFMTEN	RGB Formatting Enable. For 16- or 18-bit transmit modes only, the EPPI_CTL.RGBFMTEN bit enables conversion of RGB888 from memory into RGB666 output data (18-bit transmit) or enables conversion of RGB888 from memory into RGB565 output data (16-bit transmit). Note that EPPI_CTL.SPLTEO and EPPI_CTL.RGBFMTEN should never be set simultaneously.	
		0	Disable RGB Formatted Output
		1	Enable RGB Formatted Output

Table 25-49: EPPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	SPLTWRD	Split Word. The EPPI_CTL.SPLTWRD bit selects split word data placement when the data length (EPPI_CTL.DLEN) selects 16-, 20-, or 24-bit data words. For all other EPPI_CTL.SPLTWRD values, the set or clear selections for EPPI_CTL.SPLTWRD produce the same result (act as though EPPI_CTL.SPLTWRD is cleared). For EPPI_CTL.SPLTWRD set, the EPPI_CTL.DLEN values below result in the following combinations of split words: DLEN Cr/Cb data Y data 16 PPI_DATA[15:8] PPI_DATA[7:0] 20 PPI_DATA[19:10] PPI_DATA[9:0] 24 PPI_DATA[23:12] PPI_DATA[11:0]	
		0	PPI_DATA has (DLEN-1) bits of Y or Cr or Cb
		1	PPI_DATA Contains 2 Elements per Word
25 (R/W)	SUBSPLTODD	Sub-Split Odd Samples. The EPPI_CTL.SUBSPLTODD bit is valid only if EPPI_CTL.SPLTEO is set. If EPPI_CTL.SUBSPLTODD is set, the EPPI sub-splits the odd sub-stream, and packs them separately.	
		0	Disable
		1	Enable
24 (R/W)	SPLTEO	Split Even and Odd Data Samples. If EPPI_CTL.SPLTEO is set, the EPPI splits the incoming data stream into two sub-streams, an even stream and an odd stream, and packs them separately.	
		0	Do Not Split Samples
		1	Split Even/Odd Samples
23 (R/W)	SWAPEN	Swap Enable. The EPPI_CTL.SWAPEN selects whether or not to swap the order of the first data (most-significant bits versus least-significant bits) of the DMA word. For receive modes, the EPPI puts the first data in the most significant bits (if set) or puts the first data in the least significant bits (if cleared) of the DMA word. For transmit modes, the EPPI transmits the most significant bits in the DMA word as the first data (if set) or transmits the least significant bits in the DMA word as the first data (if cleared).	
		0	Disable
		1	Enable

Table 25-49: EPPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	PACKEN	Pack/Unpack Enable. The EPPI_CTL.PACKEN select whether or not packing is enabled (for receive modes) and unpacking is enabled (for transmit modes). When this bit is set, EPPI transfer DMA is 32-bits wide. When this bit is cleared and the EPPI_CTL.DLEN is less than or equal to 16 bits, EPPI transfer DMA is 16-bits wide. For receive modes, if this bit is set, then the EPPI packs the incoming data into 32-bit words. If this bit is cleared, then the EPPI does not do any packing. For transmit modes, if this bit is set, then the EPPI always unpacks the 32-bit data from DMA. If this bit is not set, the EPPI does not do any unpacking.	
		0	Disable
		1	Enable
21 (R/W)	SKIPEO	Skip Even or Odd. The EPPI_CTL.SKIPEO bit selects whether even (if set) or odd (if cleared) samples are skipped if sample skipping is enabled (EPPI_CTL.SKIPEN is set). This feature only is useful for receive modes.	
		0	Skip Odd Samples
		1	Skip Even Samples
20 (R/W)	SKIPEN	Skip Enable. The EPPI_CTL.SKIPEN bit enables skipping alternate samples. This feature only is useful for receive modes.	
		0	No Samples Skipping
		1	Skip Alternate Samples
19 (R/W)	DMIRR	Data Mirroring. The EPPI_CTL.DMIRR field enables mirroring (bit reversing) of the data coming in or going out on the EPPI data pins. Pin Data (DAT_MRR=0) PPI Data (DAT_MRR=1) 15 15 0 14 14 1 .... 1 1 14 0 0 15	
		0	No Data Mirroring
		1	Data Mirroring
18:16 (R/W)	DLEN	Data Length. The EPPI_CTL.DLEN bits select the data length for the EPPI. Note that the 20 bits data length selection is valid only for SMPTE modes (EPPI_CTL.SPLTWRD set).	

Table 25-49: EPPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	8 bits
		1	10 bits
		2	12 bits
		3	14 bits
		4	16 bits
		5	18 bits
		6	20 bits
		7	24 bits
15:14 (R/W)	POLS	Frame Sync Polarity. The `EPPI_CTL.POLS` selects whether the frame syncs' polarity is active low versus active high.	
		0	FS1 and FS2 are active high
		1	FS1 is active low. FS2 is active high
		2	FS1 is active high. FS2 is active low
		3	FS1 and FS2 are active low
13:12 (R/W)	POLC	Clock Polarity. The `EPPI_CTL.POLC` selects the rising versus falling edge for sampling data and sampling/driving syncs.	
		0	Clock/Sync Polarity Mode 0. For receive mode: Sample data on falling edge and sample/drive syncs on falling edge. For transmit mode: Drive data on rising edge and sample/drive syncs on rising edge.
		1	Clock/Sync Polarity Mode 1. For receive mode: Sample data on falling edge and sample/drive syncs on rising edge. For transmit mode: Drive data on rising edge and sample/drive syncs on falling edge.
		2	Clock/Sync Polarity Mode 2. For receive mode: Sample data on rising edge and sample/drive syncs on falling edge. For transmit mode: Drive data on falling edge and sample/drive syncs on rising edge.
		3	Clock/Sync Polarity Mode 3. For receive mode: Sample data on rising edge and sample/drive syncs on rising edge. For transmit mode: Drive data on falling edge and sample/drive syncs on falling edge.

Table 25-49: EPPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	SIGNEXT	Sign Extension. The EPPI_CTL.SIGNEXT select whether (for receive modes when EPPI_CTL.DLEN selecting 16 bit data length) the data is sign extended or zero filled. Not that EPPI_CTL.SPLTWRD is removed from this shared bit.	
		0	Zero Filled
		1	Sign Extended
10 (R/W)	IFSGEN	Internal Frame Sync Generation. The EPPI_CTL.IFSGEN bit selects whether the frame syncs are generated internally or are supplied by an external device.	
		0	External Frame Sync
		1	Internal Frame Sync
9 (R/W)	ICLKGEN	Internal Clock Generation. The EPPI_CTL.ICLKGEN bit selects whether the EPPI_CLK is generated internally or is supplied by an external device.	
		0	External Clock
		1	Internal Clock
8 (R/W)	BLANKGEN	Blanking Generation (ITU Output Mode). The EPPI_CTL.BLANKGEN enables ITU output with internal blanking. In GP 8, 10 transmit bit modes (when EPPI_CTL.SPLTWRD is cleared) and 16-, 20-, and 24-bit transmit modes (when EPPI_CTL.SPLTWRD is set), EPPI_CTL.BLANKGEN selects whether or not the EPPI generates blanking and generates preamble and insertion with active data from memory.	
		0	Disable
		1	Enable
7 (R/W)	ITUTYPE	ITU Interlace or Progressive. The EPPI_CTL.ITUTYPE selects interlaced or progressive operation for ITU656 mode. This selection is valid for both TX and RX modes.	
		0	Interlaced
		1	Progressive

Table 25-49: EPPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	FLDSEL	Field Select/Trigger. The EPPI_CTL.FLDSEL bits configure the EPPI field and trigger selection. These are valid for GP modes (EPPI_CTL.XFRTYPE =0x3) and ITU656 active video mode (EPPI_CTL.XFRTYPE cleared).	
		0	Field Mode 0. Read field 1 (for ITU656 active video mode). Set internal trigger (for GP RX mode). FS3 is toggled on FS2 assertion followed by FS1 assertion (when the EPPI_CTL.FSCFG bit selects sync mode 3 and the EPPI_CTL.IFSGEN bit selects internal frame sync).
		1	Field Mode 1 Read field 1 and field 2 (ITU656 active video mode). Set external trigger (GP RX mode). FS3 is toggled on FS2 assertion (when the EPPI_CTL.FSCFG bit selects sync mode 3 and the EPPI_CTL.IFSGEN bit selects internal frame sync).
5:4 (R/W)	FSCFG	Frame Sync Configuration. The EPPI_CTL.FSCFG bits configure the EPPI frame syncs. These are valid only for GP modes (EPPI_CTL.XFRTYPE =0x3). The output of the frames syncs also depends on whether the EPPI transfer direction is transmit and the EPPI is in ITU output mode (EPPI_CTL.BLANKGEN is set).	
		0	Sync Mode 0. FS0 driven in GP mode. FS0 not driven in ITU output mode.
		1	Sync Mode 1. FS1 driven in GP mode. HSYNC driven on FS1 in ITU output mode.
		2	Sync Mode 2. FS2 driven in GP mode. HSYNC driven on FS1 and VSYNC driven on FS1 in ITU output mode.
		3	Sync Mode 3. FS3 driven in GP mode. HSYNC driven on FS1, VSYNC driven on FS2, and FIELD driven on FS3 in ITU output mode.

Table 25-49: EPPI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	XFRTYPE	Transfer Type (Operating Mode). The EPPI_CTL.XFRTYPE bits select the EPPI operating mode. In receive mode (EPPI_CTL.DIR cleared), the EPPI modes include ITU656 active video only mode, ITU656 entire field mode, ITU656 vertical blanking only mode, and non-ITU656 mode (GP mode). For transmit mode (EPPI_CTL.DIR set), the EPPI_CTL.XFRTYPE bits have no effect, and the EPPI (in transmit mode) is always in GP mode.	
		0	ITU656 Active Video Only Mode
		1	ITU656 Entire Field Mode
		2	ITU656 Vertical Blanking Only Mode
		3	Non-ITU656 Mode (GP Mode)
1 (R/W)	DIR	PPI Direction. The EPPI_CTL.DIR bit selects whether the EPPI is in receive mode (if cleared) or in transmit mode (if set).	
		0	Receive Mode
		1	Transmit Mode
0 (R/W)	EN	PPI Enable. The EPPI_CTL.EN bit enables or disables the EPPI.	
		0	Disable
		1	Enable

Control Register 2 Register

The EPPI_CTL2 register controls HSYNC finish signal generation.

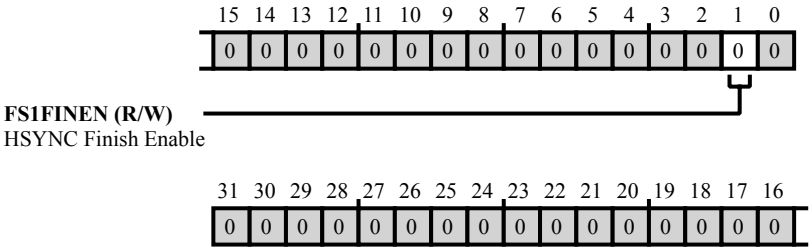


Figure 25-13: EPPI_CTL2 Register Diagram

Table 25-50: EPPI_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	FS1FINEN	HSYNC Finish Enable. The EPPI_CTL2.FS1FINEN bit selects whether (if set) the EPPI sends a finish command (010) through the DDE COMMAND line soon after a LINE is received completely or (if cleared) the EPPI sends a finish command (010) through the DDE COMMAND line soon after a FRAME is received completely. Note that the EPPI_CTL.DMAFINEN bit must be set for the EPPI to generate either of the finish commands.	
		0	Finish sent after frame RX done. PPI sends a finish command (010) through the DDE COMMAND line soon after a FRAME is received completely
		1	Finish sent after frame/line RX done. PPI sends a finish command (010) through the DDE COMMAND line soon after a frame/line is received completely.

Clipping Register for EVEN (Luma) Data Register

The EPPI_EVENCLIP register selects the clipping threshold for luma data, which provides clipping of individual video components.

The high even and low even spaces in EPPI_EVENCLIP are 16-bits wide and (depending on the EPPI_CTL.DLEN bit selection) only the corresponding video component bits are considered for clipping.

For example, if the EPPI is programmed in 10-bit mode, bits [9:0] and bits [25:16] constitute the clipping thresholds. The higher bits are (in this case) ignored.

Using the this method, 8-, 10-, 12- and 16-bit clipping thresholds can be set.

Note that when the EPPI is programmed in 16-, 20-, or 24-bit mode with the EPPI_CTL.SPLTWRD bit set, the luma data gets the clipping threshold levels of the EPPI_EVENCLIP register, and the chroma data gets the clipping threshold levels of the EPPI_ODDCLIP register.

Also, note that the EPPI_EVENCLIP and EPPI_ODDCLIP registers are ignored when the EPPI_CTL.RGBFMTEN bit is set.

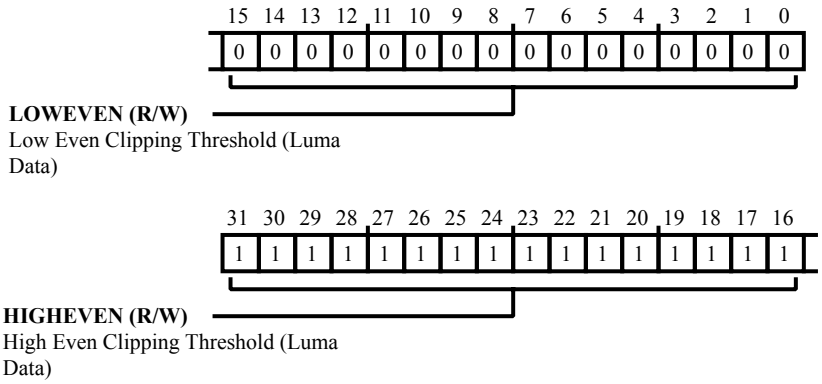


Figure 25-14: EPPI_EVENCLIP Register Diagram

Table 25-51: EPPI_EVENCLIP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	HIGHEVEN	High Even Clipping Threshold (Luma Data). The EPPI_EVENCLIP.HIGHEVEN bit field selects the clipping threshold for luma data. The high even spaces are 16-bits wide and (depending on the EPPI_CTL.DLEN selection) only the corresponding video component bits are considered for clipping.
15:0 (R/W)	LOWEVEN	Low Even Clipping Threshold (Luma Data). The EPPI_EVENCLIP.LOWEVEN bit field selects the clipping threshold for luma data. The low even spaces are 16-bits wide and (depending on the EPPI_CTL.DLEN selection) only the corresponding video component bits are considered for clipping.

Lines Per Frame Register

The EPPI_FRAME register tracks the frame track overflow and underflow errors. This register should be programmed with the number of lines expected per frame. Any write to the EPPI_FRAME register will also write the same value to the EPPI_VCNT register. But, any write to the EPPI_VCNT register does not affect the EPPI_FRAME register value. So the EPPI_FRAME register should be programmed before the EPPI_VCNT register.

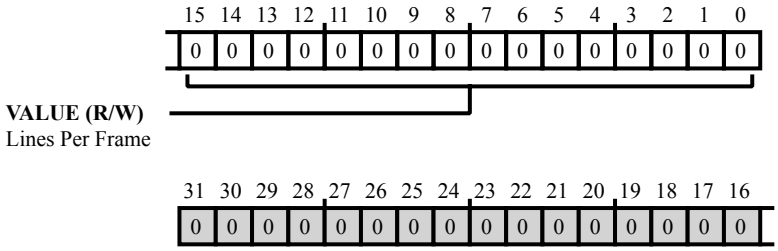


Figure 25-15: EPPI_FRAME Register Diagram

Table 25-52: EPPI_FRAME Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Lines Per Frame. The EPPI_FRAME.VALUE holds the number of lines expected per frame of data.

Frame Sync 1 Delay Value Register

The EPPI_FS1_DLY register selects the delay count (based on the period of the EPPI_CLK clock) between the first rising edge of EPPI_CLK after the EPPI is enabled and the first active edge of the associated frame sync when the internal frame sync is used.

Note that if the EPPI_FS1_DLY or EPPI_FS2_DLY registers are programmed with value 0, the EPPI operates as though 0 value is 1, and the first frame sync transition occurs after the completion of one period value of the respective counters.

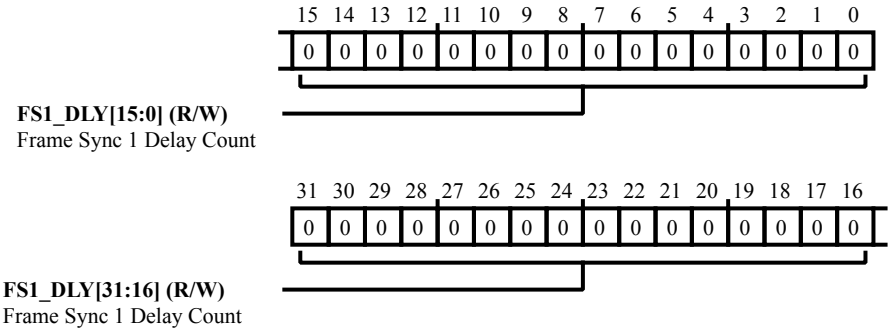


Figure 25-16: EPPI_FS1_DLY Register Diagram

Table 25-53: EPPI_FS1_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	FS1_DLY	Frame Sync 1 Delay Count. The EPPI_FS1_DLY.FS1_DLY bit field selects the delay count.

FS1 Period Register / EPPI Active Samples Per Line Register

The EPPI_FS1_PASPL register content varies depending on whether the EPPI is in GP1/2/3 FS modes or in GP transmit mode.

In GP 1, 2, or 3 FS modes, the EPPI_FS1_PASPL register is used for the generation of frame sync 1. The register contains the period required for EPPI_FS1 based on the EPPI_CLK clock.

In GP transmit mode with the EPPI_CTL.BLANKGEN bit set, this register contains the number of samples of active video or vertical blanking samples per line. When used for blanking generation, only the lower 16 bits are valid.

Note that a value of 0 for this register is illegal. If programmed as 0, the EPPI regards the EPPI_FS1_PASPL register as containing 1.

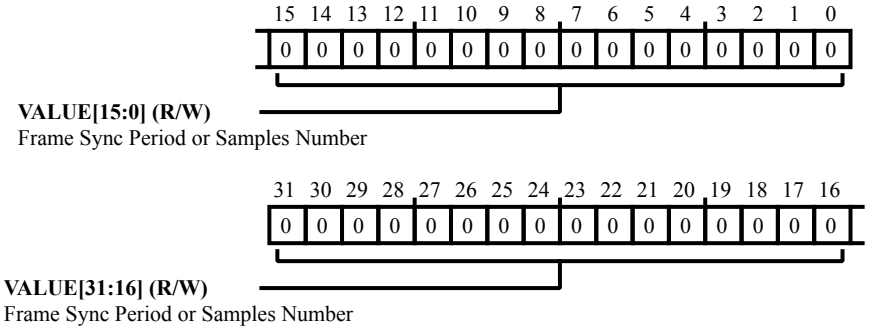


Figure 25-17: EPPI_FS1_PASPL Register Diagram

Table 25-54: EPPI_FS1_PASPL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Frame Sync Period or Samples Number. In GP 1, 2, or 3 FS modes, the EPPI_FS1_PASPL.VALUE bit field is used for the generation of frame sync 1 and contains the period required for EPPI_FS1 based on the EPPI_CLK clock. In GP transmit mode with the EPPI_CTL.BLANKGEN bit set, this bit field contains the number of samples of active video or vertical blanking samples per line.

FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register

The EPPI_FS1_WLHB register's content varies depending on whether the EPPI is in GP1/2/3 FS modes or in GP transmit mode.

In GP 1, 2 or 3 FS modes, EPPI_FS1_WLHB is used for the generation of frame sync 1. The register contains the width required for EPPI_FS1 based on the EPPI_CLK clock.

In GP transmit mode with the EPPI_CTL.BLANKGEN bit set, this register contains the number of samples of horizontal blanking per line. When used for blanking generation, only the lower 16 bits are valid.

Note that a value of 0 for the EPPI_FS1_WLHB register is illegal. If programmed as 0, the EPPI regards the EPPI_FS1_WLHB register as containing 1.

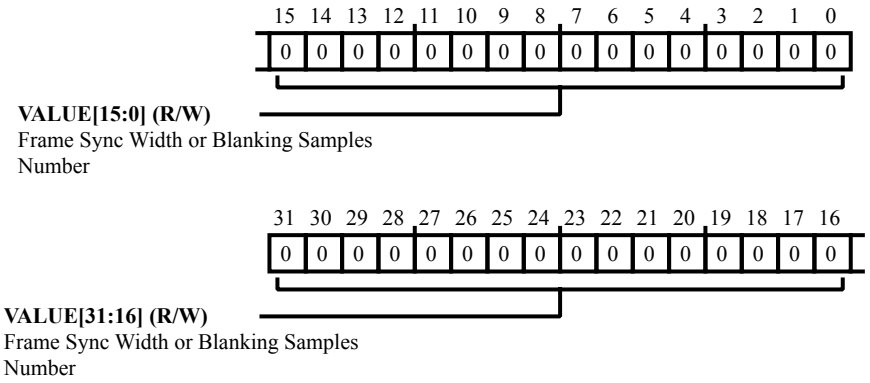


Figure 25-18: EPPI_FS1_WLHB Register Diagram

Table 25-55: EPPI_FS1_WLHB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Frame Sync Width or Blanking Samples Number. The EPPI_FS1_WLHB.VALUE bit field content varies depending on whether the EPPI is in GP1/2/3 FS modes or in GP transmit mode. In GP 1, 2 or 3 FS modes, the EPPI_FS1_WLHB.VALUE bit field is used for the generation of frame sync 1. The register contains the width required for EPPI_FS1 based on the EPPI_CLK clock. In GP transmit mode with EPPI_CTL.BLANKGEN set, this bit field contains the number of samples of horizontal blanking per line.

Frame Sync 2 Delay Value Register

The EPPI_FS2_DLY register selects the delay count (based on the period of the EPPI_CLK clock) between the first rising edge of EPPI_CLK after EPPI enabled and the first active edge of the associated frame sync when the internal frame sync is used.

Note that if the EPPI_FS1_DLY or EPPI_FS2_DLY registers are programmed with the value 0, the EPPI operates as though 0 value is 1, and the first frame sync transition occurs after the completion of one period value of the respective counters.

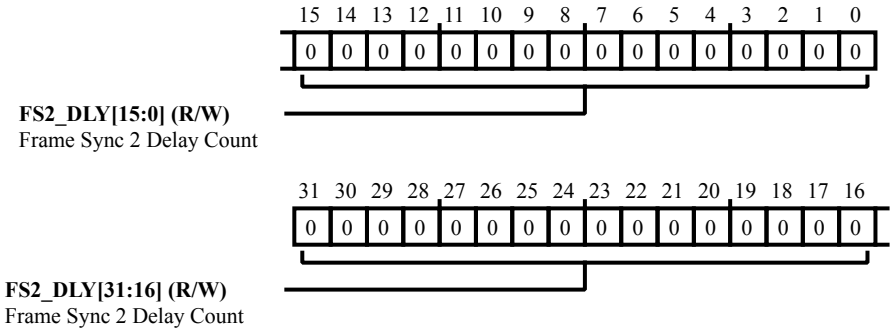


Figure 25-19: EPPI_FS2_DLY Register Diagram

Table 25-56: EPPI_FS2_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	FS2_DLY	Frame Sync 2 Delay Count. The EPPI_FS2_DLY.FS2_DLY bit field selects the delay count.

FS2 Period Register / EPPI Active Lines Per Field Register

The EPPI_FS2_PALPF register content varies depending on whether the EPPI is in GP2/3 FS modes or in GP transmit mode.

In GP 2 or 3 FS modes, EPPI_FS2_PALPF is used for the generation of frame sync 2. This register contains the period required for EPPI_FS2 based on the EPPI_CLK clock.

In GP transmit mode with the EPPI_CTL.BLANKGEN bit set, this register contains the number of lines of active video per field.

Note that a value of 0 for the EPPI_FS2_PALPF.F1ACT or EPPI_FS2_PALPF.F2ACT bits is illegal. If either is programmed as 0, the EPPI regards the 0 value fields as containing 1.

Also, note that for progressive video, the EPPI_FS2_PALPF.F2ACT bit is ignored.

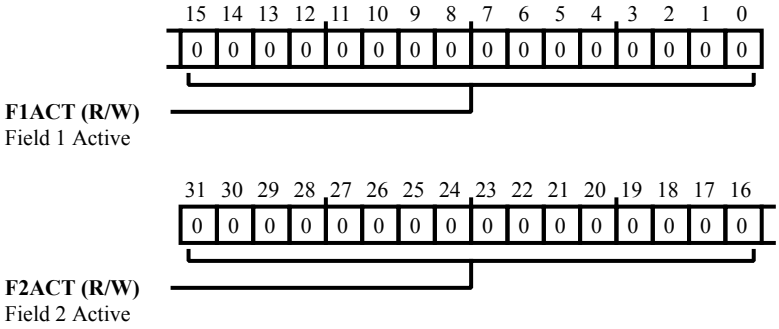


Figure 25-20: EPPI_FS2_PALPF Register Diagram

Table 25-57: EPPI_FS2_PALPF Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	F2ACT	Field 2 Active. The EPPI_FS2_PALPF.F2ACT bit field contains the number of lines of active data in field 2.
15:0 (R/W)	F1ACT	Field 1 Active. The EPPI_FS2_PALPF.F1ACT bit field contains the number of lines of active data in field 1.

FS2 Width Register / EPPI Lines Of Vertical Blanking Register

The EPPI_FS2_WLVB register content varies depending on whether the EPPI is in GP2/3 FS modes or in GP transmit mode.

In GP 2 or 3 FS modes, the EPPI_FS2_WLVB register is used for the generation of frame sync 2. The register contains the width required for EPPI_FS2 based on the EPPI_CLK clock.

In GP transmit mode with the EPPI_CTL.BLANKGEN bit set, this register contains the number or lines of vertical blanking.

Note that for progressive video, the EPPI_FS2_WLVB.F2VBBD and EPPI_FS2_WLVB.F2VBAD bits are ignored.

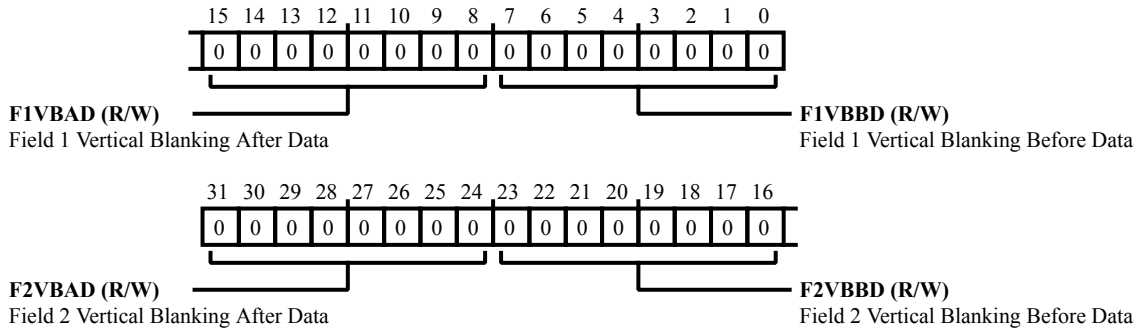


Figure 25-21: EPPI_FS2_WLVB Register Diagram

Table 25-58: EPPI_FS2_WLVB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	F2VBAD	Field 2 Vertical Blanking After Data. The EPPI_FS2_WLVB.F2VBAD bit field contains the number of lines of vertical blanking after field 2.
23:16 (R/W)	F2VBBD	Field 2 Vertical Blanking Before Data. The EPPI_FS2_WLVB.F2VBBD bit field contains the number of lines of vertical blanking before field 2.
15:8 (R/W)	F1VBAD	Field 1 Vertical Blanking After Data. The EPPI_FS2_WLVB.F1VBAD bit field contains the number of lines of vertical blanking after field 1.
7:0 (R/W)	F1VBBD	Field 1 Vertical Blanking Before Data. The EPPI_FS2_WLVB.F1VBBD bit field contains the number of lines of vertical blanking before field 1.

Horizontal Transfer Count Register

The EPPI_HCNT register holds the number of samples to read in or write out per line, after EPPI_HDLY number of cycles have expired since the assertion of EPPI_FS1. Any write to the EPPI_LINE register modifies the EPPI_HCNT register. But, any write to the EPPI_HCNT register does not affect the EPPI_LINE register value. So the EPPI_HCNT register should be programmed after the EPPI_LINE register.

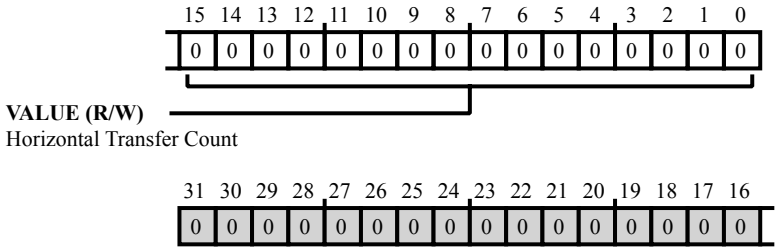


Figure 25-22: EPPI_HCNT Register Diagram

Table 25-59: EPPI_HCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Horizontal Transfer Count. The EPPI_HCNT.VALUE holds the number of samples to read in or write out per line, after EPPI_HDLY number of cycles have expired since the last assertion of EPPI_FS1.

Horizontal Delay Count Register

The EPPI_HDLY register contains the number of clock cycles to delay after the assertion of EPPI_FS1 is detected before starting to read or write data.

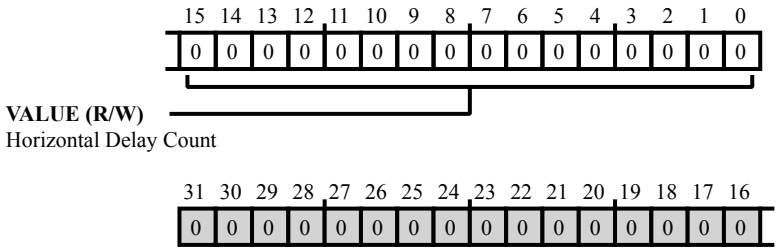


Figure 25-23: EPPI_HDLY Register Diagram

Table 25-60: EPPI_HDLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Horizontal Delay Count. The EPPI_HDLY.VALUE holds the number of EPPI_CLK cycles to delay after assertion of EPPI_FS1 before starting to read or write data.

Interrupt Mask Register

The EPPI_IMSK register permits the masking (if associated bit is set) of EPPI error interrupts for YFIFO underflow or overflow, CFIFO underflow or overflow, line track overflow error, line track underflow error, frame track overflow error, frame track underflow error, and ERR_NCOR (ITU preamble error not corrected. These conditions are flagged in the EPPI_STAT register and cleared by write-1-to-clear.

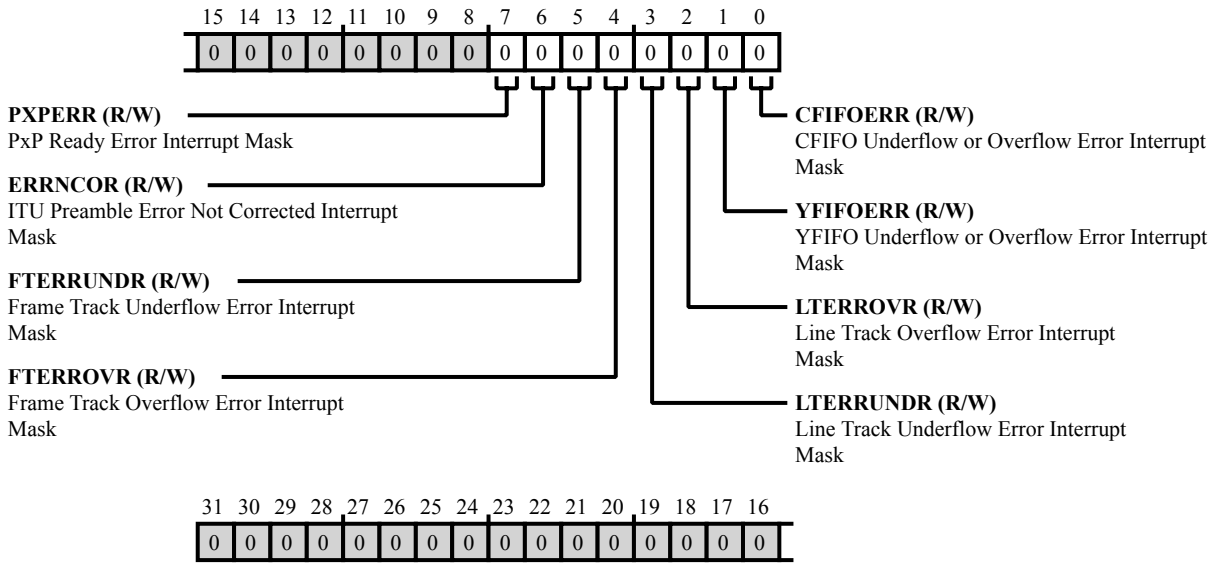


Figure 25-24: EPPI_IMSK Register Diagram

Table 25-61: EPPI_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	PXPERR	PxP Ready Error Interrupt Mask.	
		0	Unmask Interrupt
		1	Mask Interrupt
6 (R/W)	ERRNCOR	ITU Preamble Error Not Corrected Interrupt Mask.	
		0	Unmask Interrupt
		1	Mask Interrupt
5 (R/W)	FTERRUNDR	Frame Track Underflow Error Interrupt Mask.	
		0	Unmask Interrupt
		1	Mask Interrupt
4 (R/W)	FTERROVR	Frame Track Overflow Error Interrupt Mask.	
		0	Unmask Interrupt
		1	Mask Interrupt

Table 25-61: EPPI_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	LTERRUNDR	Line Track Underflow Error Interrupt Mask.	
		0	Unmask Interrupt
		1	Mask Interrupt
2 (R/W)	LTERROVR	Line Track Overflow Error Interrupt Mask.	
		0	Unmask Interrupt
		1	Mask Interrupt
1 (R/W)	YFIFOERR	YFIFO Underflow or Overflow Error Interrupt Mask.	
		0	Unmask Interrupt
		1	Mask Interrupt
0 (R/W)	CFIFOERR	CFIFO Underflow or Overflow Error Interrupt Mask.	
		0	Unmask Interrupt
		1	Mask Interrupt

Samples Per Line Register

The EPPI_LINE register tracks the line track overflow and underflow errors. This register should be programmed with the number of samples expected per line. Any write to the EPPI_LINE register will also write the same value to the EPPI_HCNT register. However, any write to the EPPI_HCNT register does not affect the EPPI_LINE register value. So the EPPI_LINE register should be programmed before the EPPI_HCNT register.

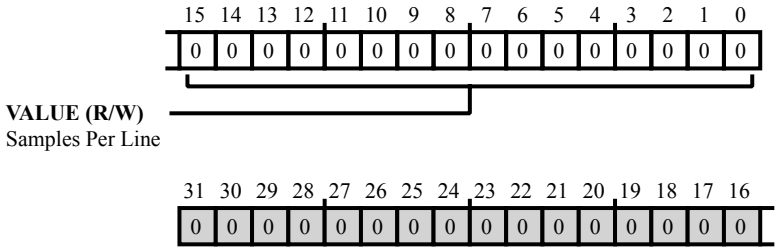


Figure 25-25: EPPI_LINE Register Diagram

Table 25-62: EPPI_LINE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Samples Per Line. The EPPI_LINE.VALUE holds the number of samples expected per line.

Clipping Register for ODD (Chroma) Data Register

The EPPI_ODDCLIP register selects the clipping threshold for chroma data, which provides clipping of individual video components.

The high odd and low odd spaces in EPPI_ODDCLIP are 16-bits wide and (depending on the EPPI_CTL.DLEN bit selection) only the corresponding video component bits are considered for clipping.

For example, if the EPPI is programmed in 10-bit mode, bits [9:0] and bits [25:16] constitute the clipping thresholds. The higher bits are (in this case) ignored.

Using the this method, 8-, 10-, 12- and 16-bit clipping thresholds can be set.

Note that when the EPPI is programmed in 16-, 20-, or 24-bit mode with the EPPI_CTL.SPLTWRD bit set, the luma data gets the clipping threshold levels of the EPPI_EVENCLIP register, and the chroma data gets the clipping threshold levels of the EPPI_ODDCLIP register.

Also, note that the EPPI_EVENCLIP and EPPI_ODDCLIP registers are ignored when the EPPI_CTL.RGBFMTEN bit is set.

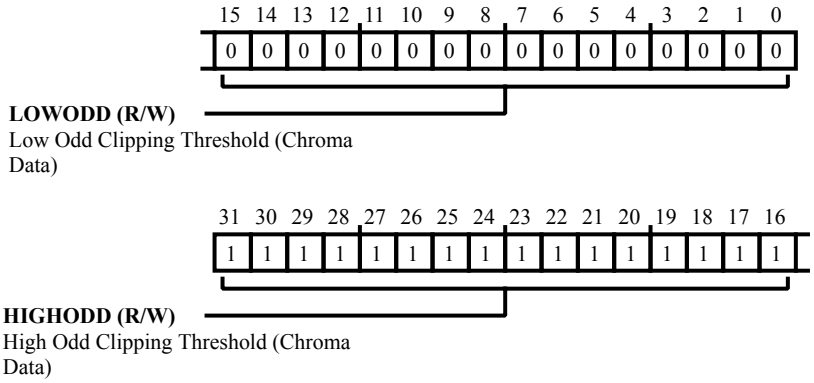


Figure 25-26: EPPI_ODDCLIP Register Diagram

Table 25-63: EPPI_ODDCLIP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	HIGHODD	High Odd Clipping Threshold (Chroma Data). The EPPI_ODDCLIP.HIGHODD bit field selects the clipping threshold for luma data. The high odd spaces are 16-bits wide and (depending on the EPPI_CTL.DLEN selection) only the corresponding video component bits are considered for clipping.
15:0 (R/W)	LOWODD	Low Odd Clipping Threshold (Chroma Data). The EPPI_ODDCLIP.LOWODD bit field selects the clipping threshold for luma data. The low add spaces are 16-bits wide and (depending on the EPPI_CTL.DLEN selection) only the corresponding video component bits are considered for clipping.

Status Register

The EPPI_STAT register contains bits that provide information about the current operating state of the EPPI.

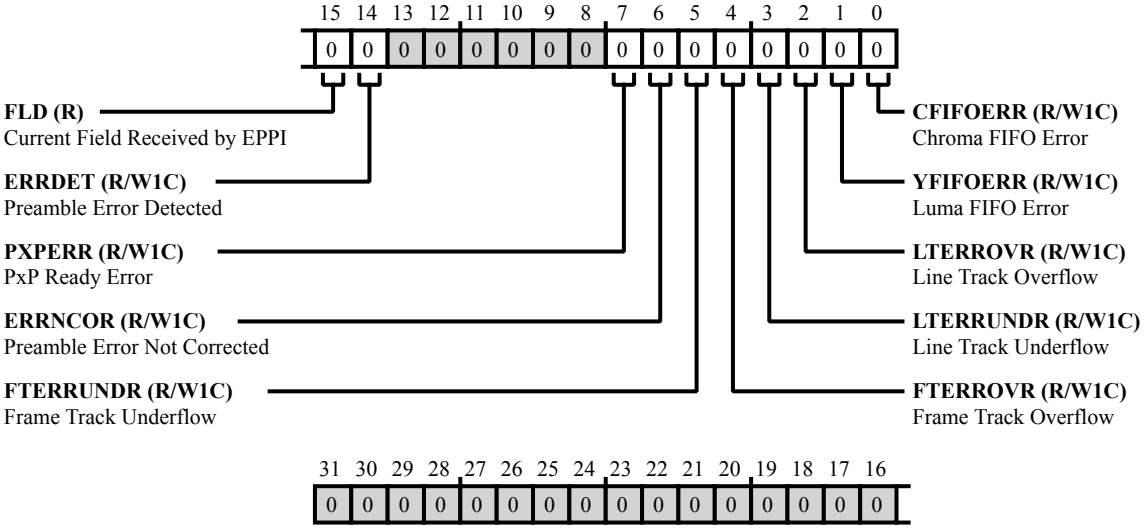


Figure 25-27: EPPI_STAT Register Diagram

Table 25-64: EPPI_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	FLD	Current Field Received by EPPI. The EPPI_STAT.FLD bit indicates whether the current field being received by the PPI is field 1 (if clear) or field 2 (if set).	
		0	Field 1
		1	Field 2
14 (R/W1C)	ERRDET	Preamble Error Detected. The EPPI_STAT.ERRDET bit is useful only in ITU receive modes and indicates if an error has been detected in the status word of EAV or SAV sequences (if set) or not (if clear).	
		0	No Preamble Error Detected
		1	Preamble Error Detected
7 (R/W1C)	PXPERR	PxP Ready Error. The EPPI_STAT.PXPERR bit is valid only in the RX mode. This bit indicates whether the incoming PPI data overflows the PxP interface (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.	

Table 25-64: EPPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1C)	ERRNCOR	Preamble Error Not Corrected. The `EPPI_STAT.ERRNCOR` bit is useful only in the ITU receive modes and indicates if an error in the status word of EAV or SAV sequences can not be cleared (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.	
		0	No uncorrected preamble error has occurred
		1	Preamble error detected but not corrected
5 (R/W1C)	FTERRUNDR	Frame Track Underflow. The `EPPI_STAT.FTERRUNDR` bit indicates whether a frame track underflow error has occurred (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.	
		0	No Error Detected
		1	Error Occurred
4 (R/W1C)	FTERROVR	Frame Track Overflow. The `EPPI_STAT.FTERROVR` bit indicates whether a frame track overflow error has occurred (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.	
		0	No Error Detected
		1	Error Occurred
3 (R/W1C)	LTERRUNDR	Line Track Underflow. The `EPPI_STAT.LTERRUNDR` bit indicates whether a line track underflow error has occurred (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.	
		0	No Error Detected
		1	Error Occurred
2 (R/W1C)	LTERROVR	Line Track Overflow. The `EPPI_STAT.LTERROVR` bit indicates whether a line track overflow error has occurred (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.	
		0	No Error Detected
		1	Error Occurred

Table 25-64: EPPI_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W1C)	YFIFOERR	Luma FIFO Error. For RX modes, the `EPPI_STAT.YFIFOERR` bit indicates whether the Luma FIFO has overflowed (if set) or not (if clear). For TX modes, this bit indicates whether the Luma FIFO has underflowed (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.	
		0	No Error Detected
		1	Error Occurred
0 (R/W1C)	CFIFOERR	Chroma FIFO Error. For RX modes, the `EPPI_STAT.CFIFOERR` bit indicates whether the chroma FIFO has overflowed (if set) or not (if clear). For TX modes, this bit indicates whether the chroma FIFO has underflowed (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.	
		0	No Error Detected
		1	Error Occurred

Vertical Transfer Count Register

The EPPI_VCNT register holds the number of lines to read in or write out, after EPPI_VDLY number of lines from the start of frame. Any write to the EPPI_FRAME register modifies the EPPI_VCNT register. However, any write to the EPPI_VCNT register does not affect the EPPI_FRAME register value. So the EPPI_VCNT register should be programmed after the EPPI_FRAME register.

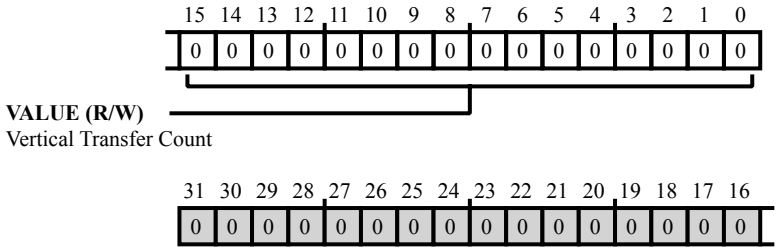


Figure 25-28: EPPI_VCNT Register Diagram

Table 25-65: EPPI_VCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Vertical Transfer Count. The EPPI_VCNT.VALUE holds the number of lines to read in or write out, after EPPI_VDLY number of lines from the start of frame.

Vertical Delay Count Register

The EPPI_VDLY register contains the number of lines to wait after the start of a new frame before starting to read/transmit data.

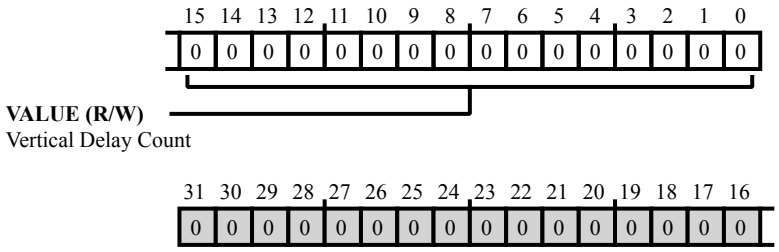


Figure 25-29: EPPI_VDLY Register Diagram

Table 25-66: EPPI_VDLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Vertical Delay Count. The EPPI_VDLY.VALUE holds the number of lines to wait after the start of a new frame before starting to read/transmit data.

26 General-Purpose Timer (TIMER)

The general-purpose timer (GP Timer) module serves as a collection of system timers that support various system-level functions. These functions include:

- Synchronized PWM waveform output capability
- External signal capture
- External event count
- General time-base functionality

Additionally, interrupt requests can be generated upon completion of timer events. Moreover, GP timers can act both as trigger requesters and trigger completers.

GP Timer Features

Each timer can be individually configured in any of these modes:

- Pin interrupt capture mode
- Windowed watchdog mode
- Pulse-width count and capture (WDTH_CAP) mode
- External Event (EXT_CLK) mode
- Pulse-width modulation (PWM_OUT) mode

Other features include:

- Synchronous operation
- Consistent management of period and pulse width values
- Autobaud detection for UART module (where available)
- Graceful bit pattern termination when stopping
- Support for center-aligned PWM patterns
- Error detection on implausible pattern values

- All read and write accesses to 32-bit registers are atomic
- Every timer has its dedicated interrupt request output

ADSP-SC59x TIMER Register List

The General-Purpose Timer block (TIMER) provides timers that may be used for external event capture and measurement, system timing, and PWM waveform generation. A set of registers governs TIMER operations. For more information on TIMER functionality, see the TIMER register descriptions.

Table 26-1: ADSP-SC59x TIMER Register List

Name	Description
TIMER_BCAST_DLY	Broadcast Delay Register
TIMER_BCAST_PER	Broadcast Period Register
TIMER_BCAST_WID	Broadcast Width Register
TIMER_DATA_ILAT	Data Interrupt Latch Register
TIMER_DATA_IMSK	Data Interrupt Mask Register
TIMER_ERR_TYPE	Error Type Status Register
TIMER_RUN	Run Register
TIMER_RUN_CLR	Run Clear Register
TIMER_RUN_SET	Run Set Register
TIMER_STAT_ILAT	Status Interrupt Latch Register
TIMER_STAT_IMSK	Status Interrupt Mask Register
TIMER_STOP_CFG	Stop Configuration Register
TIMER_STOP_CFG_CLR	Stop Configuration Clear Register
TIMER_STOP_CFG_SET	Stop Configuration Set Register
TIMER_TMR[n]_CFG	Timer n Configuration Register
TIMER_TMR[n]_CNT	Timer n Counter Register
TIMER_TMR[n]_DLY	Timer n Delay Register
TIMER_TMR[n]_PER	Timer n Period Register
TIMER_TMR[n]_WID	Timer n Width Register
TIMER_TRG_IE	Trigger Receiver Enable Register
TIMER_TRG_MSK	Trigger Generator Mask Register

ADSP-SC59x TIMER Interrupt List

Table 26-2: ADSP-SC59x TIMER Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
48	TIMER0_TMR00	TIMER0 Timer 0	Level	
49	TIMER0_TMR01	TIMER0 Timer 1	Level	
50	TIMER0_TMR02	TIMER0 Timer 2	Level	
51	TIMER0_TMR03	TIMER0 Timer 3	Level	
52	TIMER0_TMR04	TIMER0 Timer 4	Level	
53	TIMER0_TMR05	TIMER0 Timer 5	Level	
54	TIMER0_TMR06	TIMER0 Timer 6	Level	
55	TIMER0_TMR07	TIMER0 Timer 7	Level	
56	TIMER0_TMR08	TIMER0 Timer 8	Level	
57	TIMER0_TMR09	TIMER0 Timer 9	Level	
58	TIMER0_TMR10	TIMER0 Timer 10	Level	
59	TIMER0_TMR11	TIMER0 Timer 11	Level	
60	TIMER0_TMR12	TIMER0 Timer 12	Level	
61	TIMER0_TMR13	TIMER0 Timer 13	Level	
62	TIMER0_TMR14	TIMER0 Timer 14	Level	
63	TIMER0_TMR15	TIMER0 Timer 15	Level	
64	TIMER0_STAT	TIMER0 Status	Level	

ADSP-SC59x TIMER Trigger List

Table 26-3: ADSP-SC59x TIMER Trigger List Generators

Trigger ID	Name	Description	Sensitivity
137	TIMER0_TMR00_MST	TIMER0 Timer 0	Edge
138	TIMER0_TMR01_MST	TIMER0 Timer 1	Edge
139	TIMER0_TMR02_MST	TIMER0 Timer 2	Edge
140	TIMER0_TMR03_MST	TIMER0 Timer 3	Edge
141	TIMER0_TMR04_MST	TIMER0 Timer 4	Edge
142	TIMER0_TMR05_MST	TIMER0 Timer 5	Edge
143	TIMER0_TMR06_MST	TIMER0 Timer 6	Edge
144	TIMER0_TMR07_MST	TIMER0 Timer 7	Edge
145	TIMER0_TMR08_MST	TIMER0 Timer 8	Edge

Table 26-3: ADSP-SC59x TIMER Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
146	TIMER0_TMR09_MST	TIMER0 Timer 9	Edge
147	TIMER0_TMR10_MST	TIMER0 Timer 10	Edge
148	TIMER0_TMR11_MST	TIMER0 Timer 11	Edge
149	TIMER0_TMR12_MST	TIMER0 Timer 12	Edge
150	TIMER0_TMR13_MST	TIMER0 Timer 13	Edge
151	TIMER0_TMR14_MST	TIMER0 Timer 14	Edge
152	TIMER0_TMR15_MST	TIMER0 Timer 15	Edge

Table 26-4: ADSP-SC59x TIMER Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
125	TIMER0_TMR00_SLV0	TIMER0 Timer 0	Pulse
126	TIMER0_TMR00_SLV1	TIMER0 Timer 0	Pulse
127	TIMER0_TMR01_SLV0	TIMER0 Timer 1	Pulse
128	TIMER0_TMR01_SLV1	TIMER0 Timer 1	Pulse
129	TIMER0_TMR02_SLV0	TIMER0 Timer 2	Pulse
130	TIMER0_TMR02_SLV1	TIMER0 Timer 2	Pulse
131	TIMER0_TMR03_SLV0	TIMER0 Timer 3	Pulse
132	TIMER0_TMR03_SLV1	TIMER0 Timer 3	Pulse
133	TIMER0_TMR04_SLV0	TIMER0 Timer 4	Pulse
134	TIMER0_TMR04_SLV1	TIMER0 Timer 4	Pulse
135	TIMER0_TMR05_SLV0	TIMER0 Timer 5	Pulse
136	TIMER0_TMR05_SLV1	TIMER0 Timer 5	Pulse
137	TIMER0_TMR06_SLV0	TIMER0 Timer 6	Pulse
138	TIMER0_TMR06_SLV1	TIMER0 Timer 6	Pulse
139	TIMER0_TMR07_SLV0	TIMER0 Timer 7	Pulse
140	TIMER0_TMR07_SLV1	TIMER0 Timer 7	Pulse
141	TIMER0_TMR08_SLV0	TIMER0 Timer 8	Pulse
142	TIMER0_TMR08_SLV1	TIMER0 Timer 8	Pulse
143	TIMER0_TMR09_SLV0	TIMER0 Timer 9	Pulse
144	TIMER0_TMR09_SLV1	TIMER0 Timer 9	Pulse
145	TIMER0_TMR10_SLV0	TIMER0 Timer 10	Pulse

Table 26-4: ADSP-SC59x TIMER Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
146	TIMER0_TMR10_SLV1	TIMER0 Timer 10	Pulse
147	TIMER0_TMR11_SLV0	TIMER0 Timer 11	Pulse
148	TIMER0_TMR11_SLV1	TIMER0 Timer 11	Pulse
149	TIMER0_TMR12_SLV0	TIMER0 Timer 12	Pulse
150	TIMER0_TMR12_SLV1	TIMER0 Timer 12	Pulse
151	TIMER0_TMR13_SLV0	TIMER0 Timer 13	Pulse
152	TIMER0_TMR13_SLV1	TIMER0 Timer 13	Pulse
153	TIMER0_TMR14_SLV0	TIMER0 Timer 14	Pulse
154	TIMER0_TMR14_SLV1	TIMER0 Timer 14	Pulse
155	TIMER0_TMR15_SLV0	TIMER0 Timer 15	Pulse
156	TIMER0_TMR15_SLV1	TIMER0 Timer 15	Pulse

Timer Block Diagram

The ***Timer Block Diagram*** shows all of the possible clock sources.

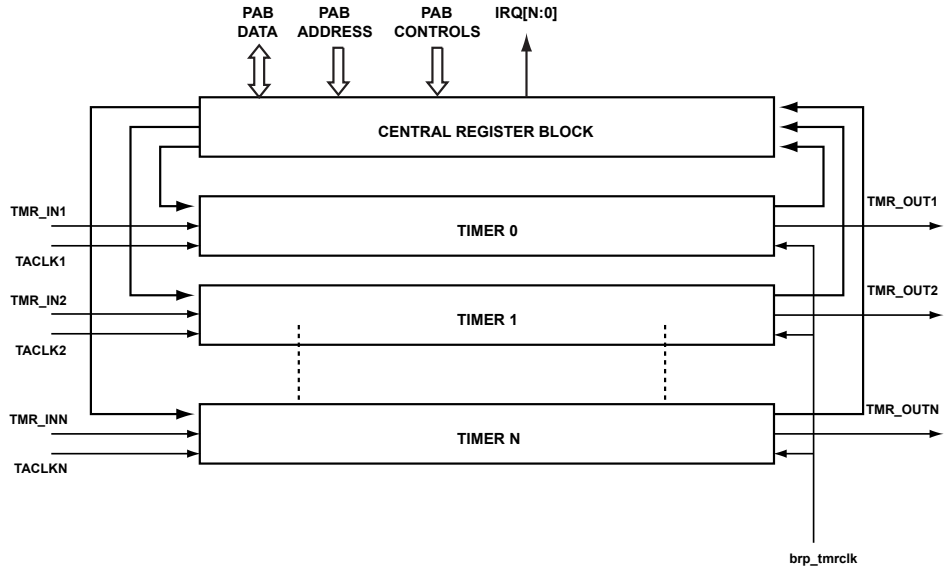


Figure 26-1: Timer Block Diagram

Internal Interface

The processor core always accesses the timer registers through the MMR access bus. Hardware ensures that all read and write operations from and to 32-bit timer registers are atomic. Every timer has a dedicated data interrupt request. There is also one common timer status and error interrupt request output that connects to the system event controller. Whenever a data interrupt request is generated, a data trigger generator pulse is also driven out, if

enabled. Each timer has an individual trigger input line, and each timer can be either started or stopped as a trigger receiver.

In total, the GP timer module can have up to (N + 1) interrupt request output lines and N data trigger lines.

Internal Timer Connections

The Timers support alternate inputs for the clock/capture (see External Interface). Some signals have internal default alternate connections according to the ***Timer Signal Routing*** table.

Table 26-5: Timer Signal Routing

Timer Signal	Connection
TM0_ACLK0	SYS_CLKIN0
TM0_ACI5	DAI0_PB04
TM0_ACLK5	DAI0_PB03
TM0_ACI6	DAI1_PB04
TM0_ACLK6	DAI1_PB03
TM0_ACI07	CNT0_TO
TM0_ACLK7	SYS_CLKIN1
TM0_ACI8	DAI0_PB06
TM0_ACLK8	DAI0_PB05
TM0_ACI9	DAI1_PB06
TM0_ACLK9	DAI1_PB05
TM0_ACI14	DAI0 Group C
TM0_ACI15	DAI1 Group C

External Interface

Each GP timer module can support up to 16 individual timers. However, most processors have less than this number. The exact number of timers available on a given processor is available in the data sheet for the processor.

Every timer has one main input/output signal (`TIMER_TMR[nn]`) and, usually, one auxiliary input pin, used as an alternate capture input (`TIMER_ACI[nn]`). Each timer can either run with a time base of SCLK0 or can reference an external clock on one of two (`TIMER_ACLK[nn]`) pins. The TMR_ALT_CLK0 signal maps to individual alternate clock ((`TIMER_ACLK[nn]`) pins for one or more timers. For instance, a TM_ACLK3 pin would provide an alternate site to supply an external signal that would serve as reference clock for TMR3. Likewise, the TMR_ALT_CLK1 signal from each timer unit connects internally to provide a single global timer clock pin (`TIMER_CLK`) for the GP timer module. It is used as an additional time base.

GP Timer Operating Modes

The following sections provide information on the various operating modes of the GP timer.

General Operation

The core of every timer is a 32-bit counter that can be interrogated through the read-only TIMER_TMR[n]_CNT register. Once the module enables a timer, it loads the timer TIMER_TMR[n]_CNT register with a starting value.

A timer can operate in one of several different modes, configured through the TIMER_TMR[n]_CFG register for that timer. These modes are: PWMOUT, EXTCLK, WIDCAP, WATCHDOG, PININT, and IDLE. The ***Timer Mode Descriptions*** table summarizes the modes.

Table 26-6: Timer Mode Descriptions

Timer Mode	Description
PWMOUT	Generates single or continuous PWM waveforms with programmable pulse width, period, and delay
EXTCLK	Counts edges of an externally applied waveform
WIDCAP	Captures pulse width or period of an externally applied waveform
WATCHDOG	Monitors pulse width or period of an external signal and compares against a window of acceptable values, optionally generating an interrupt when it falls inside or outside of that window
PININT	Can generate an interrupt request on an active edge applied to a timer pin
IDLE	Idle; no activity

Period, Width and Delay Register Interaction

When the timer is started, writes to the buffer registers are immediately copied through to the double-buffered period, pulse width, and delay registers. These values are then ready for use in the first timer period. When a timer is already running, software can write new values to the TIMER_TMR[n]_PER, TIMER_TMR[n]_WID, and TIMER_TMR[n]_DLY registers. The written values are buffered and do not update into the registers until the end of the current period. (The update occurs when the value in the TIMER_TMR[n]_CNT register equals the value in the TIMER_TMR[n]_PER register.)

If new values are not written to these registers, the value from the previous period is reused. Writes to these registers are atomic; it is not possible for the high word to be written without the low word also being written. Values written to the period, pulse width, and delay registers are always stored in the buffer registers. Reads from the same register always return the current, active value of period, pulse width, or delay value. Written values are not readback until they become active.

The usage of the TIMER_TMR[n]_PER, TIMER_TMR[n]_WID, and TIMER_TMR[n]_DLY registers varies, depending on the mode of the timer specified by the TIMER_TMR[n]_CFG.TMODE bits. See the ***Usage of the Period, Width, and Delay Registers in Different Timer Modes*** table for more information.

Table 26-7: Usage of the Period, Width, and Delay Registers in Different Timer Modes

Timer Mode	**TIMER_TMR[n]_PER**	**TIMER_TMR[n]_WID**	**TIMER_TMR[n]_DLY**
IDLE	Not writable	Not writable	Not writable

Table 26-7: Usage of the Period, Width, and Delay Registers in Different Timer Modes (Continued)

Timer Mode	TIMER_TMR[n]_PER	TIMER_TMR[n]_WID	TIMER_TMR[n]_DLY
WATCHDOG	Update on-the-fly. New value takes effect either at timer start or when an asserting edge on the input signal is sensed.	Read-only. Retains value of last measured width or period of the input signal.	Update on-the-fly. New value takes effect either at timer start or when an asserting edge on the input signal is sensed.
WIDCAP	Read-only. Period value captured at the appropriate time and updated from its buffer register simultaneously with the Width register.	Read-only. Width value captured at the appropriate time and updated from its buffer register simultaneously with the Period register.	Not used
PWMOUT	Update on-the-fly. New value takes effect either at timer start or at the end of the current period. A write followed by immediate read returns the current operating values.	Update on-the-fly. New value takes effect either at timer start or at the end of the current period. A write followed by immediate read returns the current operating values.	Update on-the-fly. New value takes effect either at timer start or at the end of the current period. A write followed by immediate read returns the current operating values.
EXTCLK	Can be updated on-the-fly.	Not used	Not used
PININT	Not used	Not used	Not used

If any of the period, pulse width, and delay registers are not used, then programs cannot write into that register. For example, in WIDCAP mode, the delay registers are not used. So, the program is not allowed to write any value to the TIMER_TMR[n]_DLY register. To prevent undesired operation, program the TIMER_TMR[n]_CFG.TMODE bits before programming the period, width, or delay registers.

If a program changes the TIMER_TMR[n]_CFG.TMODE bits from a status register to writable register (for example in PWMOUT mode), hardware does not clear these registers. These values are automatically overwritten by new values specified by software.

In PWMOUT mode with small periods, there may not be enough time between updates from the buffer registers to write these registers. The next period can use one old value and one new value. To prevent (width + pulse delay) > period errors, write the width and delay registers before the period register when decreasing the values. Write the period register before the width and delay registers when increasing the value.

Single-Pulse PWMOUT Mode

In single-pulse PWMOUT mode, the timer generates a single pulse on the TIMER_TMR[nn] pin. This mode is frequently used to implement a precise delay, often with generating an output trigger. The timer module uses the value in the TIMER_TMR[n]_DLY register to control the assertion of a pulse. The value in the TIMER_TMR[n]_WID register defines the pulse width. The TIMER_TMR[n]_PER is not used and cannot be written in this mode. After completion of the pulse, the timer is automatically stopped, and optionally generates an interrupt. The timer uses the TIMER_TMR[n]_CFG.PULSEHI bit to control pulse polarity.

The timer can be configured to generate a data interrupt request after satisfying various conditions specified by the TIMER_TMR[n]_CFG.IRQMODE bits.

It is not necessary to clear the relevant TIMER_RUN bit to stop the timer cleanly. At the end of the pulse, the timer stops automatically and the corresponding TIMER_RUN bit is cleared. To generate multiple discrete pulses (as opposed to a continuous PWM waveform), write a 1 to the appropriate TIMER_RUN bit, and wait for the timer to stop. Then, write another 1 to the same TIMER_RUN bit.

Continuous PWMOUT Mode

In continuous PWMOUT mode, the timer generates a repetitive pulse with a well-defined period, duty cycle, and pulse position. The TIMER_TMR[n]_DLY, TIMER_TMR[n]_PER, and TIMER_TMR[n]_WID registers are programmed with the values of the required PWM pulse. After the timer is started, the counter counts towards the value programmed in the TIMER_TMR[n]_PER register. Initially, the TIMER_TMR[nn] pin remains in a deasserted state. The pin toggles to an asserted state when the value in the TIMER_TMR[n]_CNT register equals the value in the TIMER_TMR[n]_DLY register.

The timer can control the assertion sense of the TIMER_TMR[nn] pin with the TIMER_TMR[n]_CFG.PULSEHI bit. The TIMER_TMR[nn] pin holds this value for the number of clock cycles specified in the TIMER_TMR[n]_WID register. Then, the pin deasserts and holds this value until the completion of the programmed period. The same waveform is generated repeatedly until the timer is disabled.

The timer can be configured to generate a data interrupt request after satisfying any of various conditions specified by the TIMER_TMR[n]_CFG.IRQMODE bits.

It is important to guarantee that the programmed period is greater than or equal to the sum of width and delay. Similarly, delay must be less than period. Violating either of these criteria results in an unpredictable waveform on the TIMER_TMR[nn] pin until the situation is rectified by writing proper values to these registers.

The maximum frequency possible to generate on the TIMER_TMR[nn] pin is achieved by setting TIMER_TMR[n]_PER to 2 and TIMER_TMR[n]_WID to 1. This operation makes the TIMER_TMR[nn] pin toggle each SCLK0 clock cycle (assuming the timer is configured to clock internally), producing a duty cycle of 50%.

When the TIMER_STOP_CFG.TMR[nn] bit of a timer is 0, the timer treats a stop operation as a stop-is-pending condition. When terminated with this setting, the timer automatically completes the current waveform and then stops cleanly, remaining in a deasserted state. This funtionality prevents truncation of the current pulse and unwanted PWM patterns at the TIMER_TMR[nn] pin. The processor can determine when the timer stops running by polling the corresponding TIMER_RUN.TMR[nn] bit until it reads 0 or by waiting for the last interrupt (if enabled).

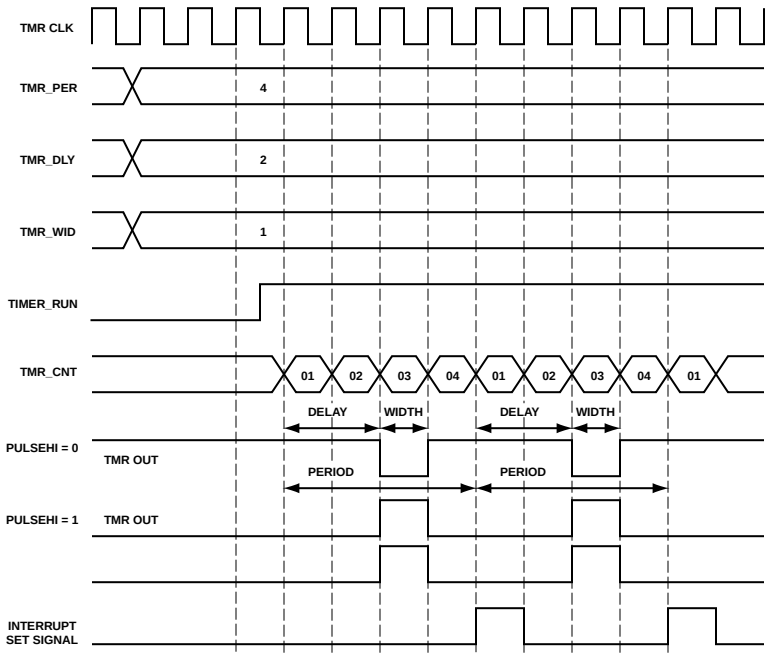


Figure 26-2: Signal Generation in Continuous PWMOUT Mode

The TIMER_TMR[n]_CFG register cannot be reconfigured until after the timer stops and the TIMER_RUN register reads 0.

Programs can force a timer to stop immediately in PWMOUT mode by writing a 1 to the TIMER_STOP_CFG register followed by writing a 1 to the TIMER_RUN_CLR register. (Or a program can stop a timer by writing a 0 to the appropriate TIMER_RUN.TMR[nn] bit.) This operation stops the timer whether the pending stop is waiting for the end of the current period or the end of the current pulse width. The timer can use this feature to regain immediate control of a timer during an error recovery sequence.

Use this feature carefully, as it can corrupt the PWM pattern generated at the TIMER_TMR[nn] pin, though after such a stop the pin deasserts automatically. Each timer samples its TIMER_RUN.TMR[nn] bit at the end of each period. It stops cleanly at the end of the first period after the TIMER_RUN.TMR[nn] bit is low. A timer that is disabled and then restarted (before the end of the current period), continues to run as if nothing happened. Typically, the program disables a PWMOUT timer and then waits for it to stop itself.

Width Capture (WIDCAP) Mode

The timer uses WIDCAP mode, often called capture mode, to measure pulse widths on the TIMER_TMR[nn] or (TIMER_ACI[nn]) inputs. The polarity (active high or low) of the input signal can be selected with the TIMER_TMR[n]_CFG.PULSEHI bit. The ***Timer Signal Flow in Width Capture Mode*** figure shows the control signal flow for WIDCAP_CAP mode.

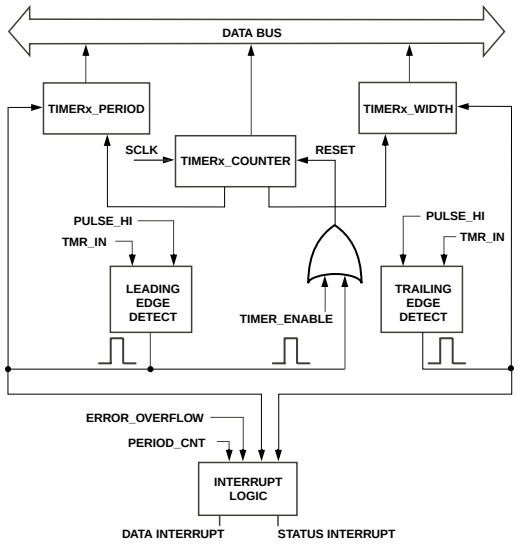


Figure 26-3: Timer Signal Flow in Width Capture Mode

NOTE: SCLK in the ***Timer Signal Flow in Width Capture Mode*** figure is SCLK0.

In this mode, the timer uses the `TIMER_TMR[n]_CFG.TINSEL` bit to select between the `TIMER_TMR[nn]` or(`TIMER_ACI[nn]`) input. The internally clocked timer is used to determine the period and pulse width of the externally applied rectangular waveforms.

When a timer is enabled in this mode, the timer resets the count in its `TIMER_TMR[n]_CNT` register to 0x0000 0001. It does not start counting until it detects a leading edge on the selected input pin.

When the timer detects the first leading edge, it starts incrementing. When it detects a trailing edge of a waveform, it captures the current 32-bit value of its `TIMER_TMR[n]_CNT` register into its width buffer register. At the next leading edge, the timer transfers the current 32-bit value of its `TIMER_TMR[n]_CNT` register into its period buffer register. The `TIMER_TMR[n]_CNT` register is reset to 0x0000 0001 again, and the timer continues counting and capturing until it is disabled.

In this mode, programs can measure both the pulse width and the pulse period of a waveform. The timer does not use the `TIMER_TMR[n]_DLY` register in this mode. The timer uses the `TIMER_TMR[n]_CFG.PULSEHI` bit to control the definition of leading edge and trailing edge of the `TIMER_TMR[nn]` /(`TIMER_ACI[nn]`) pin.

In WIDCAP mode, the following events always occur at the same time as one unit:

1. The `TIMER_TMR[n]_PER` register is updated from the period buffer register.
2. The `TIMER_TMR[n]_WID` register is updated from the width buffer register.
3. The `TIMER_DATA_ILAT.TMR[nn]` bit is set (if enabled).
4. A timer data trigger pulse is generated (if enabled).

The `TIMER_TMR[n]_CFG.TMODE` bit 0 controls the point in time at which this set of events is executed. Taken together, these four events are called a measurement report. The `TIMER_STAT_ILAT` register is not set at a

measurement report. A measurement report occurs, at most, once per input signal period. The current TIMER_TMR[n]_CNT value is always copied to the width buffer and period buffer registers at the trailing and leading edges of the input signal, respectively. But these values are not visible to software. A measurement report event samples the captured values into visible registers and sets the timer interrupt request to signal that the TIMER_TMR[n]_PER and the TIMER_TMR[n]_WID registers are ready to be read.

When the TIMER_TMR[n]_CFG.TMODE bit =b#1011, the measurement report occurs just after the width buffer register captures its value at a falling edge. Then, the TIMER_TMR[n]_WID register reports the pulse width measured in the pulse that has ended, but the TIMER_TMR[n]_PER register reports the pulse period measured at the end of the previous period. If only the first trailing edge has occurred, then the first period value has not yet been measured at the first measurement report. So, the period value is not valid. A read of the TIMER_TMR[n]_PER value in this case returns 0. See the ***Example of Width Capture Deasserted Mode (TMODE=b#1011)*** figure for more information.

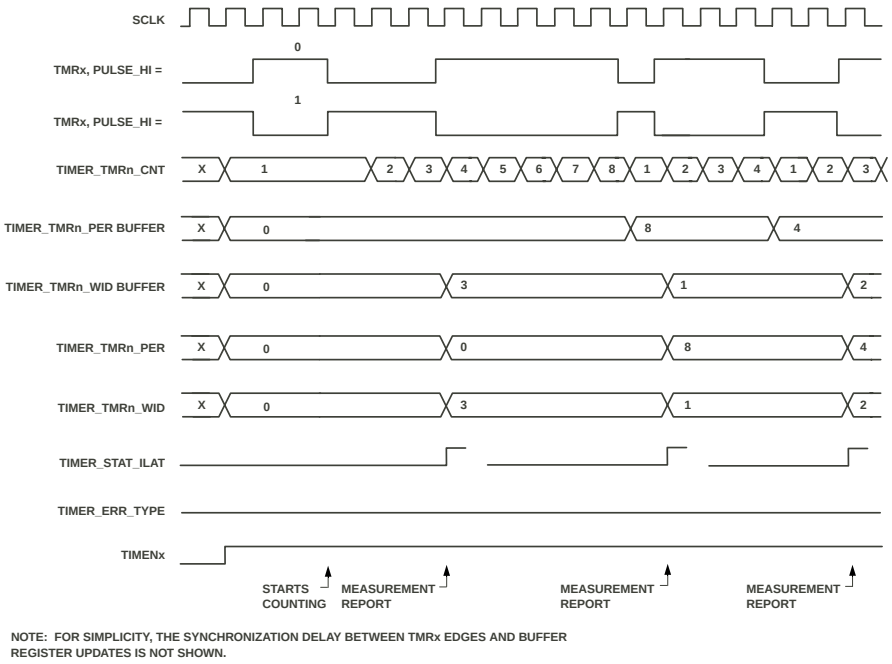


Figure 26-4: Example of Width Capture Deasserted Mode (TMODE=b#1011)

NOTE: SCLK in the ***Example of Width Capture Deasserted Mode (TMODE=b#1011)*** figure is SCLK0.

When the TIMER_TMR[n]_CFG.TMODE bit =b#1010, the measurement report occurs just after the period buffer register captures its value at a leading edge. Then, the TIMER_TMR[n]_PER and TIMER_TMR[n]_WID registers report the pulse period and pulse width measured in the period that has ended. Refer to the ***Example of Width Capture Asserted Mode (TMODE=b#1010)*** figure for more information.

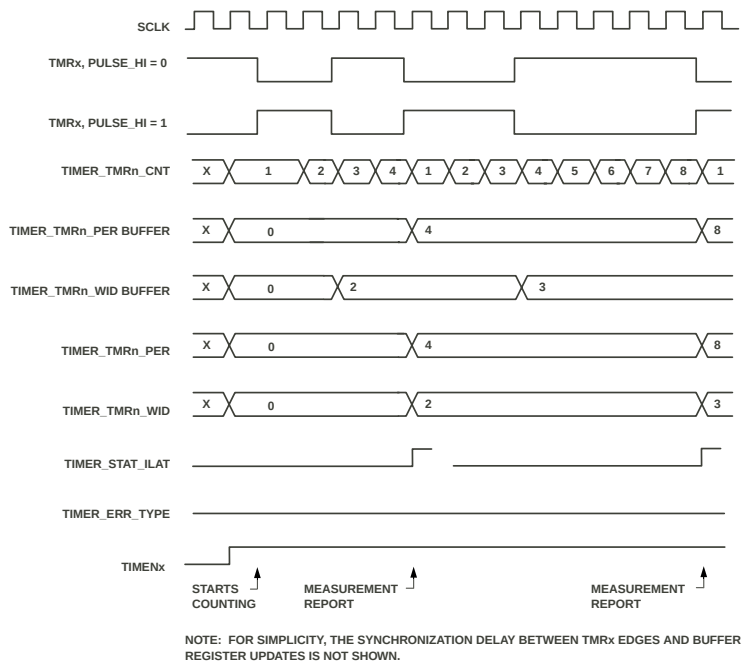


Figure 26-5: Example of Width Capture Asserted Mode (TMODE=b#1010)

NOTE: SCLK in the ***Example of Width Capture Asserted Mode (TMODE=b#1010)*** figure is SCLK0.

To measure the pulse width of a waveform that has only one leading edge and one trailing edge, set `TIMER_TMR[n]_CFG.TMODE` = b#1011. When `TIMER_TMR[n]_CFG.TMODE` = b#1010, no period value is captured in the period buffer register. Instead, the timer generates an error report interrupt request (if enabled) when the `TIMER_TMR[n]_CNT` range is exceeded and the counter wraps around. In this case, both the `TIMER_TMR[n]_PER` and `TIMER_TMR[n]_WID` registers read 0 (because no measurement report occurred to copy the value captured in the width buffer register to the `TIMER_TMR[n]_WID` register).

When using the `TIMER_TMR[n]_CFG.TMODE` bit =b#1010 mode to measure the width of a single pulse, programs can disable the timer after taking the interrupt that ends the measurement interval. When desired, restart the timer as appropriate in preparation for another measurement. This procedure prevents the timer from freely running after the width measurement and logging errors generated by the timer count overflowing.

Width Capture Mode Overflow

A timer status interrupt request (when enabled) is generated when the `TIMER_TMR[n]_CNT` register wraps around from 0xFFFF FFFF to 0 in the absence of a leading edge. At that point, the `TIMER_STAT_ILAT` bit is set and the `TIMER_ERR_TYPE` bits change to indicate a count overflow due to a period greater than the range of the counter. This indication is referred to as an error report. A data interrupt request in WIDCAP mode indicates that a new measurement is ready to be read (a measurement report). Similarly, an interrupt request on the timer status interrupt line (shared interrupt request for all timers) indicates an overflow error when generated in this mode.

The `TIMER_TMR[n]_PER` and `TIMER_TMR[n]_WID` registers are never updated at the time an overflow error is signaled. If the timer overflows and the `TIMER_TMR[n]_CFG.TMODE` bit =b#1010, the `TIMER_TMR[n]_PER` and `TIMER_TMR[n]_WID` registers are not updated. If the timer overflows and the

TIMER_TMR[n]_CFG.TMODE bit =b#1011, the TIMER_TMR[n]_PER and TIMER_TMR[n]_WID registers are updated only if a trailing edge is detected at a previous measurement report.

Software can count the number of error reports between measurement report interrupt requests to measure input signal periods longer than 0xFFFF FFFF. Each error report interrupt request adds a full 2^{32}SCLK0 counts to the total for the period, but the width is ambiguous. Ensure that if software monitors only the status interrupt request, then status interrupt requests from all other timers are masked.

Refer to the ***Example Timing for Width Capture Followed by Period Overflow (TMR_CFG.TMODE=b#1010)*** figure. The period is 0x1 0000 0004, but the pulse width could be either 0x0 0000 0002 or 0x1 0000 0002.

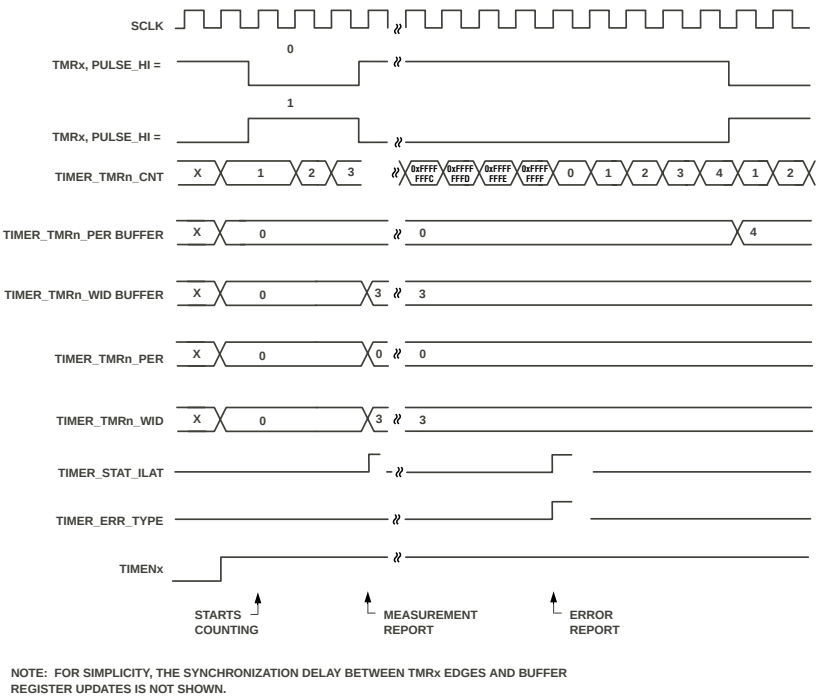


Figure 26-6: Example Timing for Width Capture Followed by Period Overflow (TMR_CFG.TMODE=b#1010)

NOTE: SCLK in the ***Example Timing for Width Capture Followed by Period Overflow*** figure is SCLK0.

The waveform applied to the TIMER_TMR[nn] (or (TIMER_ACI[nn])) pin is not required to have a 50% duty cycle. The minimum input low time is little more than one SCLK0 period. The minimum input high time is a little more than one SCLK0 period. (Refer to the product data sheet for details.) The maximum TIMER_TMR[nn] input frequency is less than SCLK0/2, with a 50% duty cycle. Under these conditions, the WIDCAP mode timer measures: period =2 and pulse width =1.

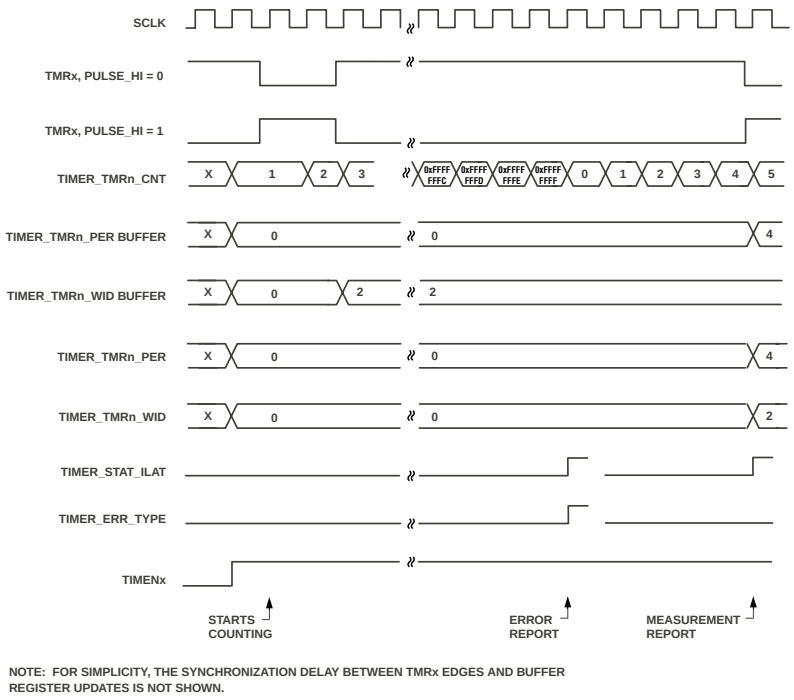


Figure 26-7: Example Timing for Width Capture Followed by Period Overflow (TMR_CFG.TMODE=b#1011)

NOTE: SCLK in the ***Example Timing for Width Capture Followed by Period Overflow*** figure is SCLK0.

Windowed Watchdog (WATCHDOG) Modes

In windowed watchdog (WATCHDOG) modes, a timer can take inputs from one of the `TIMER_TMR[nn]` pins or the `TIMER_ACI[nn]` pins. With this mode, the timer can monitor pulse width (width watchdog mode) or pulse period (period watchdog mode) on the input line. It also compares the measured value against a minimum required value and maximum allowed value and generates an interrupt request appropriately. The timer uses the `TIMER_TMR[n]_CFG.PULSEHI` bit to select polarity of the input signal.

The waveform applied to the input pin in watchdog mode is not required to have a 50% duty cycle. The minimum input pulse low time, high time, and total period specifications are available in the product data sheet.

Windowed Watchdog Width Mode

In windowed watchdog width mode, the timer counter monitors the pulse width of an input signal on one of the `TIMER_TMR[nn]` pins or one of the alternate clock pins (`TIMER_ACLK[nn]`). Program the minimum pulse width (p_{MIN}) in the `TIMER_TMR[n]_DLY` register and the maximum pulse width (p_{MAX}) in the `TIMER_TMR[n]_PER` register. Both values are programmed in terms of number of clock cycles (SCLK0 or alternate clock). The timer can generate an interrupt if the deasserting pulse edge occurs:

- Inside the window (p_{MIN} < pulse width ≤ p_{MAX}), or
- Outside the window (pulse width ≤ p_{MIN} or pulse width > p_{MAX})

After enabling the timer in this mode, it always starts counting at the asserting edge of the input signal. Any pulse that is already active when the timer is enabled is ignored.

With the TIMER_TMR[n]_CFG.IRQMODE bit =b#11, the timer generates an interrupt if the timed pulse width exceeds p_{MAX}, or if the pulse width is less than p_{MIN}. After attaining p_{MAX}, the pulse stays at an active level, and the counter keeps on counting until it sees a deasserting edge. When the input pulse is not active, the counter holds its current value until it again sees an asserting edge, or it restarts. An interrupt can also be generated for when the pulse occurs within the specified window condition, by setting TIMER_TMR[n]_CFG.IRQMODE =b#10.

In this mode, a trailing edge on the input pin triggers capturing of pulse width into the TIMER_TMR[n]_WID register. During the inactive portion of the input signal, the internal counter does not increment. The ***Watchdog Width Mode Timing*** figure shows the signal flow in this mode.

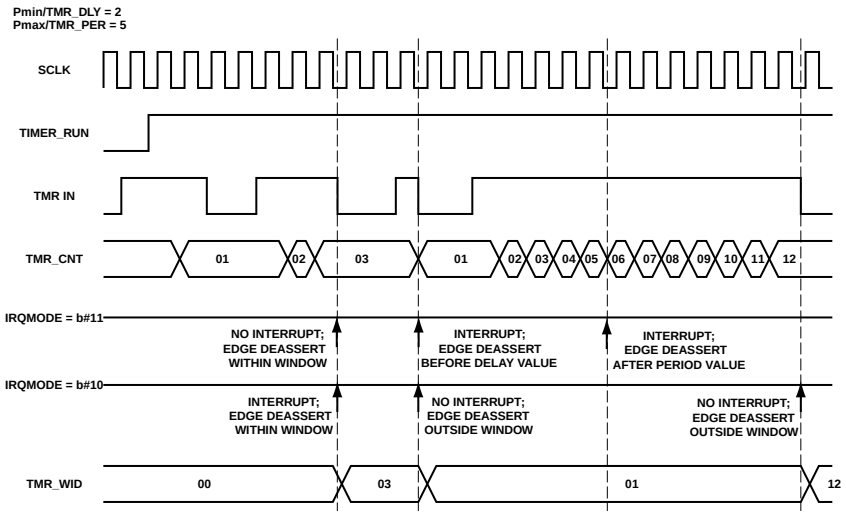


Figure 26-8: Watchdog Width Mode Timing

NOTE: SCLK in the ***Watchdog Width Mode Timing*** figure is SCLK0.

To check only the upper limit on pulse width (p_{MAX} but not p_{MIN}), then program p_{MIN} as 0 or 1. In such a case, it is better to use TIMER_TMR[n]_CFG.IRQMODE =b#11. With TIMER_TMR[n]_CFG.IRQMODE = b#10, a pulse width of 1 clock cycle results in an interrupt. For details, see the ***Windowed Watchdog Width Mode Interpretation*** table.

Table 26-8: Windowed Watchdog Width Mode Interpretation

Timer Delay	Timer Period	Incoming Pulse Width	IRQMODE= b#10	IRQMODE= b#11	Error Interrupt?
0 or 1	Anything ≥ 1	PW = 1	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		PW ≤ TMR_PER	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		PW > TMR_PER	No Interrupt	Interrupt when pulse width exceeds Pmax (Period Register) Value	No Error Interrupt
> 1 but ≤ (Period -1)	Anything > 1	PW ≤ TMR_DLY	No Interrupt	Interrupt at deasserting edge of input signal	No Error Interrupt
		TMR_DLY < PW≤ TMR_PER	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		PW > TMR_PER	No Interrupt	Interrupt when pulse width exceeds Pmax (Period Register) Value	No Error Interrupt
≥ Period	-	PW ≤ TMR_PER	Undefined	Undefined	No Error Interrupt
	-	PW > TMR_PER	Undefined	Undefined	b#11 Error Type
-	0	-	Undefined	Undefined	b#10 Error Type

Windowed Watchdog Period Mode

In this mode, the timer monitors the number of clock cycles between two consecutive rising or falling edges of an input signal on one of the `TIMER_TMR[nn]` or `TIMER_ACI[nn]` pins. Program the required minimum number of clock cycles (t_{MIN}) in the `TIMER_TMR[n]_DLY` register and the required maximum allowed number of clock cycles (t_{MAX}) in the `TIMER_TMR[n]_PER` register. Both values are programmed in terms of number of clock cycles (SCLK0) or alternate time clock (`TIMER_ACLK[nn]`). The timer can generate an interrupt when two consecutive occurrences of an active edge are:

- Within a specified window (t_{MIN} < Pulse Period ≤ t_{MAX}), or
- Outside a specified window (pulse width ≤ (t_{MIN} or t_{MAX} < pulse width)

When the `TIMER_TMR[n]_CFG.IRQMODE` bit =b#11 and the pulse period > t_{MAX} or is ≤ t_{MIN}, the timer generates an interrupt (if unmasked). After attaining the t_{MAX} value, the counter keeps on counting until it sees an active edge on the input line. An interrupt can also be generated for when the pulse occurs within the specified

window condition, by setting `TIMER_TMR[n]_CFG.IRQMODE` =b#10. Refer to the ***Watchdog Period Mode Timing*** figure for timer functionality in period watchdog mode.

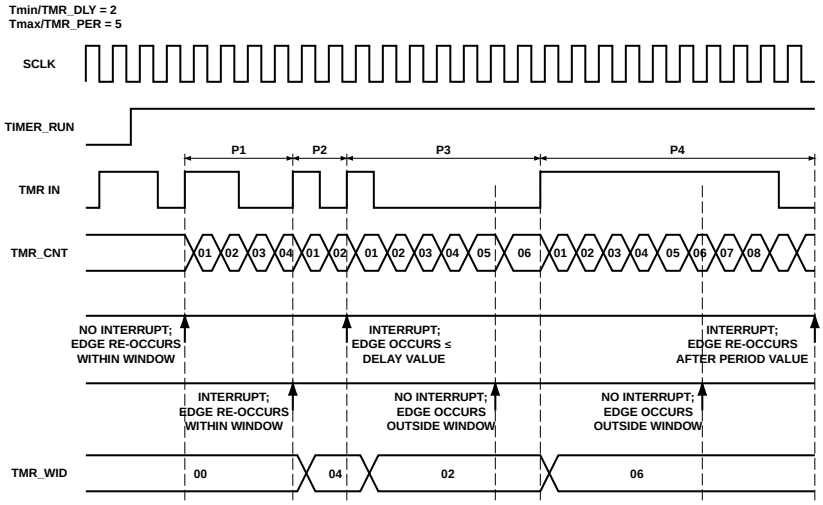


Figure 26-9: Watchdog Period Mode Timing

NOTE: SCLK in the ***Watchdog Period Mode Timing*** figure is SCLK0.

To check only the upper limit on period (the t_{MAX} value, not the t_{MIN} value), program t_{MIN} as 0 or 1. For details, refer to the ***Windowed Watchdog Period Mode Interpretation*** table.

Table 26-9: Windowed Watchdog Period Mode Interpretation

Timer Delay	Timer Period	Incoming Pulse Width	IRQMODE=b#10	IRQMODE =b#11	Error Interrupt?
0 or 1	Anything ≥ 2	Pulse Period ≤ TMR_PER	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		Pulse Period > TMR_PER	No Interrupt	Interrupt when pulse period crosses Pmax (Period Register) value	No Error Interrupt
> 1 but ≤ Period -1	Anything ≥ 2	Pulse Period ≤ TMR_DLY	No Interrupt	Interrupt at deasserting edge of input signal	No Error Interrupt
		TMR_DLY < Pulse Period ≤TMR_PER	Interrupt at deasserting edge of input signal	No Interrupt	No Error Interrupt
		Pulse Period > TMR_PER	No Interrupt	Interrupt when pulse width exceeds Pmax (Period Register) value	No Error Interrupt

Table 26-9: Windowed Watchdog Period Mode Interpretation (Continued)

Timer Delay	Timer Period	Incoming Pulse Width	IRQMODE=b#10	IRQMODE =b#11	Error Interrupt?
≥ Period	-	Pulse Period < TMR_PER	Undefined	Undefined	No Error Interrupt
		Pulse Period ≥ TMR_PER	Undefined	Undefined	b#11 Error Type
-	0 or 1	-	Undefined	Undefined	b#10 Error Type

Pin Interrupt (PININT) Mode

In PININT mode, any active edges on either the `TIMER_TMR[nn]` pin or the (`TIMER_ACI[nn]`) pin can cause an edge-based interrupt, if enabled. (The timer uses the `TIMER_TMR[n]_CFG.TINSEL` register to select the pin). The event on the input pin can set the `TIMER_DATA_ILAT.TMR[nn]` bit and issue a system interrupt request. Program the `TIMER_TMR[n]_CFG.PULSEHI` bit to change active edge polarity.

Since the interrupt request is generated in the SCLK0 clock domain, the width of the input signal must be more than one SCLK0 period. Along with generating the interrupt request, the timer also generates a trigger pulse (configured using the `TIMER_TRG_MSK` register). Due to the configuration of polarity, glitches can cause the generation of an undesired interrupt request at the input. To avoid this problem, programs must ensure that interrupt requests are unmasked only after configuring the desired polarity.

External Clock (EXTCLK) Mode

The timer uses EXTCLK mode, sometimes referred to as the counter mode, to count external events (signal edges), on either the `TIMER_TMR[nn]` or (`TIMER_ACI[nn]`) input pin. The timer works as a counter clocked by an external source (the signal at the pin), which can be asynchronous to SCLK0. The current count in the `TIMER_TMR[n]_CNT` register represents the number of leading-edge events detected. The `TIMER_TMR[n]_PER` register is programmed with the value of the maximum timer external count before stopping or issuing an interrupt request or trigger.

The `TIMER_TMR[n]_CFG.PULSEHI` bit determines the polarity of the leading edge on the input pin. The timer uses the `TIMER_TMR[n]_CFG.TINSEL` bit to select whether the event is counted on the `TIMER_TMR[nn]` or on the (`TIMER_ACI[nn]`) pin. The `TIMER_STAT_ILAT.TMR[nn]` and `TIMER_ERR_TYPE` bits are set if *one* of these conditions is met:

- `TIMER_TMR[n]_CNT` wraps around from 0xFFFF FFFF to 0
- The period = 0 at startup
- `TIMER_TMR[n]_CNT` register rolls over (from count = period to count = 0x1)

The `TIMER_TMR[n]_WID` and `TIMER_TMR[n]_DLY` registers are unused in this mode and must not be written.

The ***EXTCLK Mode Control Flow*** figure shows a flow diagram for EXTCLK mode.

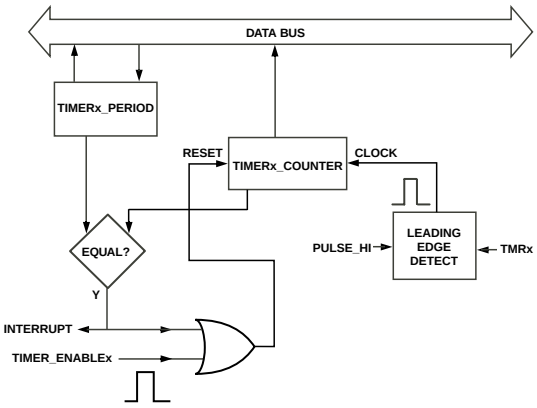


Figure 26-10: EXTCLK Mode Control Flow

The waveform applied to the input pin is not required to have a 50% duty cycle. The minimum input pulse low time, high time, and total period specifications are available in the product data sheet. Program the period to any value from 1 to (2^{32} – 1), inclusive.

After the timer has started, it resets the `TIMER_TMR[n]_CNT` register to 0x0 and then waits for the first leading edge on the input pin. This edge causes `TIMER_TMR[n]_CNT` to be incremented to the value 0x1, and every subsequent leading edge increments it by one. After the `TIMER_TMR[n]_CNT` register reaches the value programmed in the `TIMER_TMR[n]_PER` register, the corresponding `TIMER_DATA_ILAT` bit is set, and an interrupt and trigger are both generated (if enabled). The next leading-edge reloads the `TIMER_TMR[n]_CNT` register with 0x1, and the timer continues counting until it is disabled.

GP Timer Programming Concepts

Using the features, operating modes, and event control for the GP timer to their greatest potential requires an understanding of some GP timer-related concepts.

Setting Up Constantly Changing Timer Conditions

This task shows how to use different period, pulse width, and delay settings for each of the first three timer periods after the timer starts.

1. Program the first set of `TIMER_TMR[n]_PER`, `TIMER_TMR[n]_WID`, and `TIMER_TMR[n]_DLY` register values.
2. Enable the timer using the `TIMER_RUN` register.
3. Immediately program the second set of `TIMER_TMR[n]_PER`, `TIMER_TMR[n]_WID`, and `TIMER_TMR[n]_DLY` register values, as needed.
4. Wait for the first timer interrupt request.
5. Program the third set of `TIMER_TMR[n]_PER`, `TIMER_TMR[n]_WID`, and `TIMER_TMR[n]_DLY` register values.

Each new setting is then programmed when the preceding timer interrupt request is received.

Configuring, Enabling, and Disabling One or More Timers

1. Configure the relevant timers for the operating mode and other properties using the TIMER_TMR[n]_CFG register.
2. Write a 1 to the representative TIMER_RUN.TMR[nn] bit. Or use the TIMER_RUN_SET register to avoid disturbing the settings of other timers that are not going through configuration.

   The timer is enabled and operating.
3. To stop one or more timers, first program the TIMER_STOP_CFG register to determine whether to stop immediately or gracefully upon receiving a stop command.

   ADDITIONAL INFORMATION: PWMOUT modes are the only modes where a timer can be configured for graceful termination.
4. Write a 0 to the representative TIMER_RUN.TMR[nn] bits to stop the timer according to their TIMER_STOP_CFG settings. Alternately, write a 1 to the appropriate TIMER_RUN_CLR.TMR[nn] bits to avoid disturbing the settings of other timers that are not terminating.

   The timers stop.

Configuring Timer Data and Status Interrupts

1. Configure the TIMER_TMR[n]_CFG.IRQMODE bit field with the desired interrupt properties.
2. Unmask the interrupt source.
3. Set the TIMER_TMR[n]_CFG.IRQMODE field but leave the interrupt masked at the system level to poll the TIMER_DATA_ILAT.TMR[nn] bit of the timer without generating an interrupt.
4. Use the TIMER_STAT_IMSK register to generate interrupt requests by overflow or error conditions (incorrect programming values). The timer uses the TIMER_STAT_ILAT.TMR[nn] bits to report interrupt errors, when the timer status interrupt source is unmasked.
5. To poll the TIMER_STAT_ILAT.TMR[nn] bit of the timer without generating an interrupt, unmask the corresponding bit in the TIMER_STAT_IMSK register, but leave the interrupt masked at the system level.

Configuring the Timer as a Trigger Receiver

The timer can be configured to either start or stop or toggle between these two states on the input trigger pulse depending on the configuration of the TIMER_TMR[n]_CFG.TGLTRIG and TIMER_TMR[n]_CFG.SLAVETRIG bits.

- If TIMER_TMR[n]_CFG.TGLTRIG bit =0 and TIMER_TMR[n]_CFG.SLAVETRIG bit =1 then the trigger pulse starts timer, if it is stopped.

- If TIMER_TMR[n]_CFG.TGLTRIG bit =0 and TIMER_TMR[n]_CFG.SLAVETRIG bit =0 then the trigger pulse stops timer, if it is running.
- When TIMER_TMR[n]_CFG.TGLTRIG bit =0, the trigger pulse has no effect when the timer is already in the requested state.

If TIMER_TMR[n]_CFG.TGLTRIG bit is 1, the trigger pulse starts the timer if it is stopped, or stops the timer if it is running. The TIMER_TMR[n]_CFG.SLAVETRIG bit has no effect on trigger mechanism. In continuous PWMOUT mode, the timer stops gracefully or abruptly depending on the stop mechanism programmed in the TIMER_STOP_CFG_CLR register. In other modes, the timer stops immediately.

Ordered Trigger Toggle Mode

The timer can be configured to either start or stop the timer on the input trigger pulse depending on the configuration of the TIMER_TMR[n]_CFG.TGLTRIG and TIMER_TMR[n]_CFG.ORDTGLTRIG bits.

- If TIMER_TMR[n]_CFG.TGLTRIG bit is 1 and TIMER_TMR[n]_CFG.ORDTGLTRIG bit is 0, then the input trigger pulse on trigger receiver 0 or trigger receiver 1 starts the timer if it is stopped, or stops the timer if it is running.
- If TIMER_TMR[n]_CFG.TGLTRIG bit is 1 and TIMER_TMR[n]_CFG.ORDTGLTRIG bit is 1, then the input trigger pulse on trigger receiver 0 starts timer if it is stopped. If the timer is running, it stops when the input trigger pulse on trigger receiver 1 is detected.

  Trigger receiver 0 cannot toggle the timer from the run to halt state. Trigger receiver 1 cannot toggle the timer from the halt to run state.

NOTE: The timer state is not toggled by writes to the TIMER_RUN, TIMER_RUN_SET, and TIMER_RUN_CLR registers in this mode.

Using the Timer Broadcast Feature

The broadcast feature provides a means to update period, width, and delay registers simultaneously across more than one timer.

1. Enable the appropriate broadcast bits (TIMER_TMR[n]_CFG.BPEREN, TIMER_TMR[n]_CFG.BWIDEN are TIMER_TMR[n]_CFG.BDLYEN) for the timers involved in the broadcast. The use of these bits depends on which broadcast registers the timer uses (TIMER_BCAST_PER, TIMER_BCAST_WID, or TIMER_BCAST_DLY).
2. Program the TIMER_BCAST_PER register (for example), to broadcast the period setting across the multiple timers enabled.

   The enabled timers load their TIMER_TMR[n]_PER registers with the value specified in the TIMER_BCAST_PER register.
3. Repeat Step 2 as needed for the TIMER_BCAST_WID and TIMER_BCAST_DLY register settings.

Timer Illegal States

The following sections use these definitions:

- Startup. The first clock period during which the timer counter is running after the timer is started by writing the TIMER_RUN register.
- Rollover. The time when the current count in TIMER_TMR[n]_CNT matches the value in TIMER_TMR[n]_PER and the counter is reloaded with the value 1.
- Overflow. The timer counter was incremented instead of doing a rollover when it was holding the maximum count value of 0xFFFF FFFF. The counter does not have a large enough range to express the next greater value and so it erroneously loads a new value of 0x0000 0000.
- Unchanged. No new error.

When the TIMER_ERR_TYPE register is designated unchanged, it displays the previously reported error code orb# 00 when there has been no error since this timer was enabled.

When the TIMER_STAT_ILAT register is unchanged, it reads 0 when there has been no error or overflow since this timer was enabled. Or, it reads 0 if software has performed a W1C to clear any previous error. If software has not acknowledged a previous error, the TIMER_STAT_ILAT register reads 1. Software can read the TIMER_STAT_ILAT register to check for errors. If a particular bit of a timer is set in this register, software can then read the TIMER_ERR_TYPE register for more information. Once detected, software can W1C the appropriate TIMER_STAT_ILAT bit to acknowledge the error.

Read the following tables as:

- In mode __ at event __,
- if TIMER_TMR[n]_PER is __ and TIMER_TMR[n]_WID is __ and TIMER_TMR[n]_DLY is __,
- then TIMER_ERR_TYPE is __ and TIMER_STAT_ILAT is __.

Startup error conditions do not prevent the timer from starting. Similarly, overflow and rollover error conditions do not stop the timer. Illegal cases can cause unwanted behavior of the TIMER_TMR[nn] pins.

NOTE: For PININT mode, the timer does not use error functionality.

Continuous PWMOUT Mode

Table 26-10: Startup Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≤ 1	Anything other than period [*1]	Anything	Anything	b#10	Set

Table 26-10: Startup Event (Continued)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≥ 2	Anything including 0, excluding TMR_PER value	Anything including 0	≤ PERIOD	Unchanged	Unchanged
	Anything including 0	Anything including 0	> PERIOD	Unchanged *2(Detected at rollover)	Unchanged (Detected at rollover)
	Anything	Anything	> 2^{32} – 1	b#11	Set
	=Period	=0	=Period	No error	Unchanged (Detected at rollover)

*1 If delay = period, the #10 error is rolled to a #11 error.

*2 This case is not detected at startup, but when the timer counts to the complete period.

Table 26-11: Rollover Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≥ 1	Anything	Anything	Anything	b#10[timer running at SCLK0] b#11 [timer running at ALT_CLKx]	Set
≥ 2	Anything including 0, excluding TMR_PER value	Anything including 0	≤PERIOD	Unchanged	Unchanged
	Anything including 0, excluding TMR_PER value	Anything >0	>PERIOD	b#11	Set
	Anything	Anything	> 2^{32}– 1	b#11	Set
	= Period*1	=0	=Period	b#11	Set
	>Period	=0	>Period	Unchanged	Unchanged

*1 In case of graceful termination, this error is not generated.

Table 26-12: Overflow Event (On TMR_PER Register Programming Error Only)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
Anything	Anything	Anything	Anything	b#01	Set

Single Pulse PWMOUT Mode

For single pulse PWMOUT mode, there are no rollover events.

Table 26-13: Startup Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_STAT_ILAT (if enabled)	TIMER_STAT_ILAT (if enabled)
N/A	Anything	== 0	Anything	b#11*1	Set
N/A	Anything including 0	≥1	≤ 2^{32}- 1	Unchanged	Unchanged
N/A	Anything including 0	≥1	> 2^{32}- 1	b#11	Set

*1 The timer does not run in this case.

Table 26-14: Overflow Event (On another error, such as DELAY + WIDTH ≥ 2^{32}- 1)

	TIMER_TMR[n]_DLY		TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY		TIMER_STAT_ILAT (if enabled)
Anything	Anything	Anything	Anything	b#01	Set

WIDCAP Mode

For WIDCAP mode, the `TIMER_TMR[n]_PER` and `TIMER_TMR[n]_WID` registers are read-only and the `TIMER_TMR[n]_DLY` register is not used. Therefore, no startup or rollover errors are possible.

Table 26-15: Overflow Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
Anything	N/A	Anything	N/A	b#01	Set

EXTCLK Mode

Table 26-16: Startup Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
=0	N/A	N/A	N/A	b#01	Set

Table 26-16: Startup Event (Continued)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≥1	N/A	N/A	N/A	Unchanged	Unchanged

Table 26-17: Rollover Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
=0	N/A	N/A	N/A	b#01	Set
≥1	N/A	N/A	N/A	Unchanged	Unchanged

Table 26-18: Overflow Event (On TMR_PER Register = 0 Only)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
Anything	N/A	N/A	N/A	b#01	Set

WATCHDOG Events

Table 26-19: Startup Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≤ Allowed MIN*1	Anything < PERIOD	N/A	N/A	b#01	Set
> Allowed MIN	Anything < PERIOD	N/A	N/A	Unchanged	Unchanged
> Allowed MIN	Anything ≥ PERIOD	Refer to WATCHDOG Mode tables			

*1 Allowed MIN=0 for width watchdog mode. Allowed MIN=1 for period watchdog mode.

Table 26-20: Rollover Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
≤ Allowed MIN*1	Anything < PERIOD	N/A	N/A	b#01	Set

Table 26-20: Rollover Event (Continued)

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
> Allowed MIN	Anything	N/A	N/A	Unchanged	Unchanged
> Allowed MIN	Anything ≥ PERIOD	Refer to WATCHDOG Mode tables			

*1 In case of graceful termination, this error is not generated.

Table 26-21: Overflow Event

TIMER_TMR[n]_PER	TIMER_TMR[n]_DLY	TIMER_TMR[n]_WID	TIMER_TMR[n]_WID + TIMER_TMR[n]_DLY	TIMER_ERR_TYPE	TIMER_STAT_ILAT (if enabled)
Anything	Anything	N/A	N/A	b#01	Set

ADSP-SC59x TIMER Register Descriptions

General-Purpose Timer Block (TIMER) contains the following registers.

Table 26-22: ADSP-SC59x TIMER Register List

Name	Description
TIMER_BCAST_DLY	Broadcast Delay Register
TIMER_BCAST_PER	Broadcast Period Register
TIMER_BCAST_WID	Broadcast Width Register
TIMER_DATA_ILAT	Data Interrupt Latch Register
TIMER_DATA_IMSK	Data Interrupt Mask Register
TIMER_ERR_TYPE	Error Type Status Register
TIMER_RUN	Run Register
TIMER_RUN_CLR	Run Clear Register
TIMER_RUN_SET	Run Set Register
TIMER_STAT_ILAT	Status Interrupt Latch Register
TIMER_STAT_IMSK	Status Interrupt Mask Register
TIMER_STOP_CFG	Stop Configuration Register
TIMER_STOP_CFG_CLR	Stop Configuration Clear Register
TIMER_STOP_CFG_SET	Stop Configuration Set Register
TIMER_TMR[n]_CFG	Timer n Configuration Register

Table 26-22: ADSP-SC59x TIMER Register List (Continued)

Name	Description
TIMER_TMR[n]_CNT	Timer n Counter Register
TIMER_TMR[n]_DLY	Timer n Delay Register
TIMER_TMR[n]_PER	Timer n Period Register
TIMER_TMR[n]_WID	Timer n Width Register
TIMER_TRG_IE	Trigger Receiver Enable Register
TIMER_TRG_MSK	Trigger Generator Mask Register

Broadcast Delay Register

For timers with TIMER_TMR[n]_CFG.BDLYEN enabled, a write to the TIMER_BCAST_DLY register concurrently updates the delay (TIMER_TMR[n]_DLY) registers of only those timers. A read of the TIMER_BCAST_DLY register returns 0x00000000, and no bus error is generated. To read back a written value, read that TMR's TIMER_TMR[n]_DLY register.

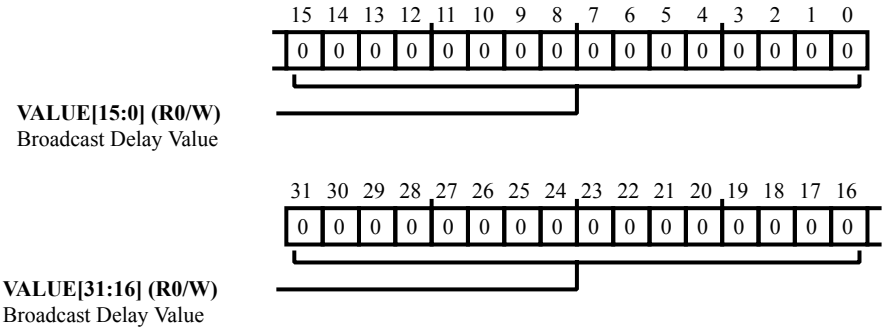


Figure 26-11: TIMER_BCAST_DLY Register Diagram

Table 26-23: TIMER_BCAST_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W)	VALUE	Broadcast Delay Value. A write to the TIMER_BCAST_DLY.VALUE bit field concurrently updates the delay (TIMER_TMR[n]_DLY) registers of only those timers. A read of the TIMER_BCAST_DLY.VALUE bit field returns 0x0000 0000, and no bus error is generated.

Broadcast Period Register

For timers with TIMER_TMR[n]_CFG.BPEREN enabled, a write to the TIMER_BCAST_PER register concurrently updates the period (TIMER_TMR[n]_PER) registers of only those timers. A read of TIMER_BCAST_PER returns 0x00000000, and no bus error is generated. To read back a written value, read that TMR's TIMER_TMR[n]_PER register.

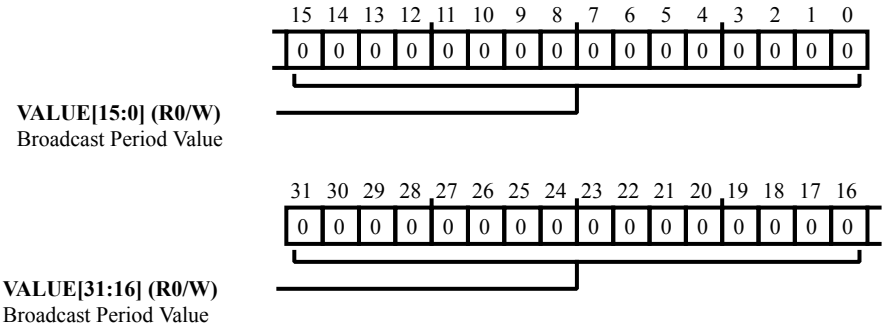


Figure 26-12: TIMER_BCAST_PER Register Diagram

Table 26-24: TIMER_BCAST_PER Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W)	VALUE	Broadcast Period Value. A write to the TIMER_BCAST_PER.VALUE bit field concurrently updates the period (TIMER_TMR[n]_PER) registers of only those timers. A read of the TIMER_BCAST_PER.VALUE bit fields returns 0x0000 0000, and no bus error is generated.

Broadcast Width Register

For timers with TIMER_TMR[n]_CFG.BWIDEN enabled, a write to the TIMER_BCAST_WID register concurrently updates the width (TIMER_TMR[n]_WID) registers of only those timers. A read of the TIMER_BCAST_WID register returns 0x00000000, and no bus error is generated. To read back a written value, read that TMR's TIMER_TMR[n]_WID register.

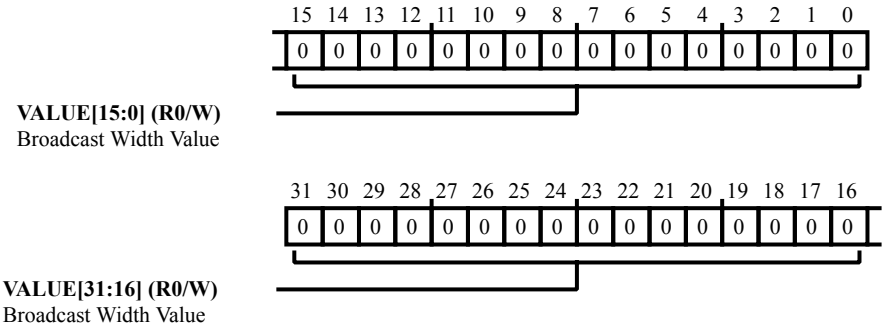


Figure 26-13: TIMER_BCAST_WID Register Diagram

Table 26-25: TIMER_BCAST_WID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W)	VALUE	Broadcast Width Value. A write to the TIMER_BCAST_WID.VALUE bit field concurrently updates the width (TIMER_TMR[n]_WID) registers of only those timers. A read of the TIMER_BCAST_WID.VALUE bit field returns 0x0000 0000, and no bus error is generated.

Data Interrupt Latch Register

The TIMER_DATA_ILAT holds the latched interrupt status for interrupt requests that have been unmasked (enabled) by the TIMER_DATA_IMSK register and generated according to the conditions selected by the TIMER_TMR[n]_CFG.IRQMODE bits. If a bit in TIMER_DATA_ILAT is already set and the corresponding interrupt is masked in TIMER_DATA_IMSK, the latch holds its old value, leaving the interrupt request asserted until it is reset by software with a W1C operation.

Note that interrupt service routines (ISRs) should clear the appropriate bits in TIMER_DATA_ILAT before returning from the ISR, to ensure that the interrupt is not re-issued. To make sure that no timer event is missed, the latch should be reset at the very beginning of the ISR when in EXTCLK mode.

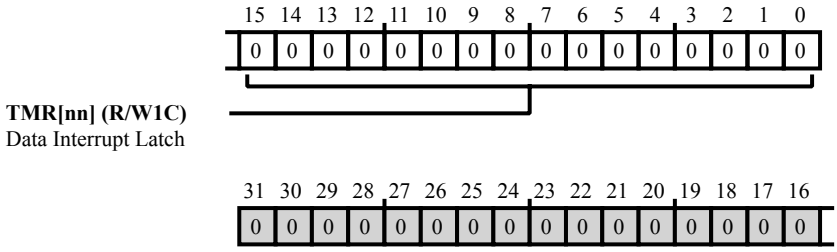


Figure 26-14: TIMER_DATA_ILAT Register Diagram

Table 26-26: TIMER_DATA_ILAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W1C)	TMR[nn]	Data Interrupt Latch. For all TIMER_DATA_ILAT.TMR[nn] bits, status of =0 indicates no interrupt is latched, and status of =1 indicates a latched interrupt (indicating an unmasked interrupt request from a timer with a condition matching the one selected with corresponding TIMER_TMR[n]_CFG.IRQMODE bit has occurred).

Data Interrupt Mask Register

Each timer may generate a unique processor data interrupt request signal. The TIMER_DATA_IMSK register contains an interrupt mask for these requests, masking (disabling) or unmasking (enabling) the interrupts as programmed. The reset value of the TIMER_DATA_IMSK register is 0xFFFF, masking these interrupts after reset.

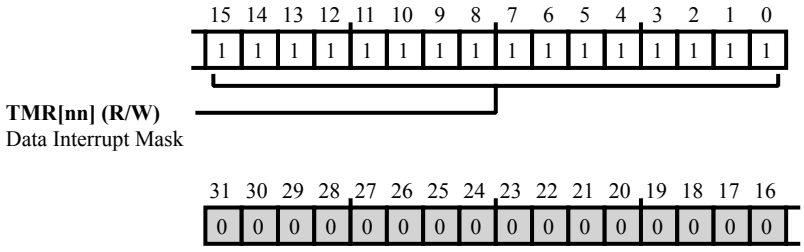


Figure 26-15: TIMER_DATA_IMSK Register Diagram

Table 26-27: TIMER_DATA_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Data Interrupt Mask. For all TIMER_DATA_IMSK.TMR[nn] bits, write =0 unmasks (enables) the corresponding data interrupt request, and write =1 masks (disables) the corresponding data interrupt request.

Error Type Status Register

The TIMER_ERR_TYPE register contains error type status bits for each timer. These bits indicate the type of error (if any) in a running timer. This register is read-only. These status bits are cleared at reset and when a particular timer is enabled.

Each time an error request interrupt is latched in the TIMER_STAT_ILAT register, the corresponding TERRx bits in the TIMER_ERR_TYPE register are loaded with a code that identifies the type of error that was detected. This status value is held until the next error or until a particular timer is restarted. No bus error is generated if a write is performed on the TIMER_ERR_TYPE register.

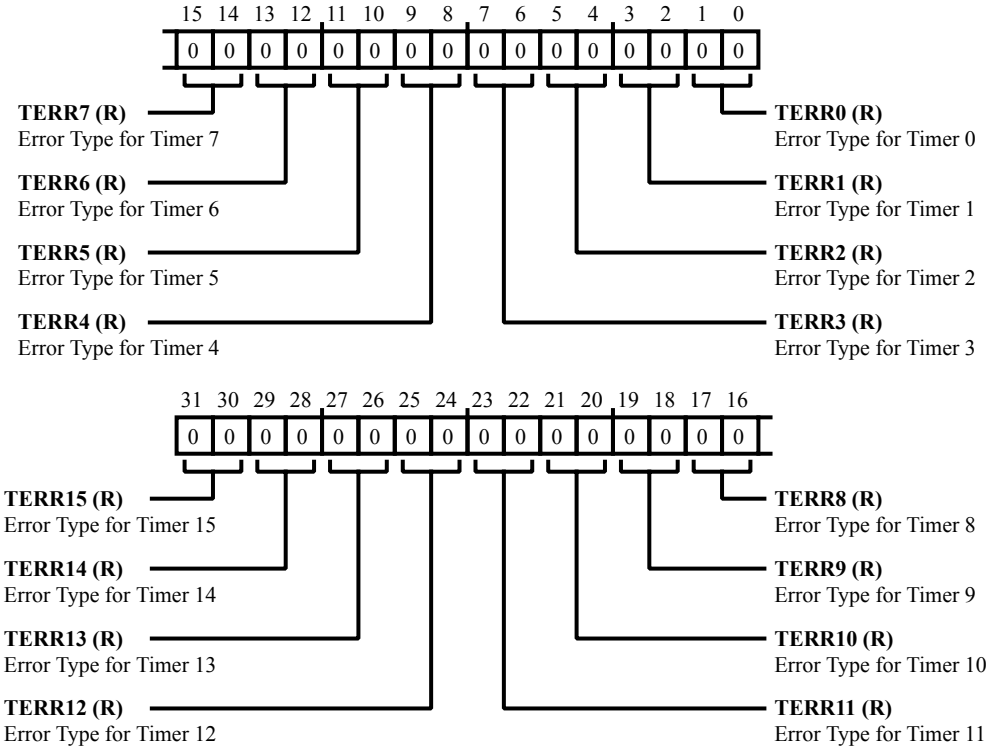


Figure 26-16: TIMER_ERR_TYPE Register Diagram

Table 26-28: TIMER_ERR_TYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	TERR15	Error Type for Timer 15.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error

Table 26-28: TIMER_ERR_TYPE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29:28 (R/NW)	TERR14	Error Type for Timer 14.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
27:26 (R/NW)	TERR13	Error Type for Timer 13.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
25:24 (R/NW)	TERR12	Error Type for Timer 12.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
23:22 (R/NW)	TERR11	Error Type for Timer 11.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
21:20 (R/NW)	TERR10	Error Type for Timer 10.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
19:18 (R/NW)	TERR9	Error Type for Timer 9.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error

Table 26-28: TIMER_ERR_TYPE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/NW)	TERR8	Error Type for Timer 8.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
15:14 (R/NW)	TERR7	Error Type for Timer 7.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
13:12 (R/NW)	TERR6	Error Type for Timer 6.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
11:10 (R/NW)	TERR5	Error Type for Timer 5.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
9:8 (R/NW)	TERR4	Error Type for Timer 4.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
7:6 (R/NW)	TERR3	Error Type for Timer 3.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error

Table 26-28: TIMER_ERR_TYPE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/NW)	TERR2	Error Type for Timer 2.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
3:2 (R/NW)	TERR1	Error Type for Timer 1.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error
1:0 (R/NW)	TERR0	Error Type for Timer 0.	
		0	No Error
		1	Counter Overflow Error
		2	PER Register Programming Error
		3	WID or DLY Register Programming Error

Run Register

The TIMER_RUN allows all timers to be enabled simultaneously, permitting them to run synchronously. For each timer, there is a single start/stop control bit. Writing a 1 to this bit starts the corresponding timer; writing a 0 stops the timer with mechanism specified in the timer stop configuration TIMER_STOP_CFG register.

The start/stop control bits can be set/reset individually or in any combination. While starting or stopping one particular timer directly with this register, software must perform a read-modify write, so the bits corresponding to other timers remain unchanged. To avoid this need, software can use the TIMER_RUN_CLR register.

Reading the TIMER_RUN register shows the start status for the corresponding timer. A 1 indicates that the timer is running.

If a timer is in run state (corresponding run bit is =1), a software write of 1 in this bit does not have any effect on the timer state. The write does not result in restarting the timer.

Note that the TIMER_RUN register is not used in PININT mode. PININT mode starts as soon as the TIMER_TMR[n]_CFG.TMODE bits are set to 111.

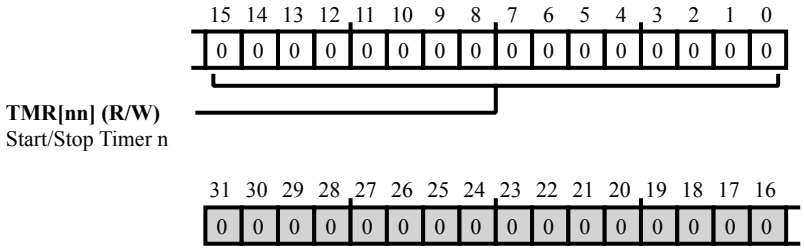


Figure 26-17: TIMER_RUN Register Diagram

Table 26-29: TIMER_RUN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Start/Stop Timer n. For all TIMER_RUN.TMR[nn] bits, write =0 for stop, and write =1 for start. Read =1 when timer is running.

Run Clear Register

The TIMER_RUN_CLR register is an alias register, providing a mechanism to clear a specific start/stop bit in the TIMER_RUN register without affecting other bits in TIMER_RUN. To stop a particular timer, software must write a 1 into the corresponding TIMER_RUN_CLR bit. Writing a 0 has no effect. Because TIMER_RUN_CLR is a write-only register, the result of any write to this register must be checked by reading the TIMER_RUN register. A read of the TIMER_RUN_CLR returns 0x0000.

Note that the stopping mechanism of a timer may be abrupt or graceful (after completion of current waveform period) depending on the selection in the TIMER_STOP_CFG register.

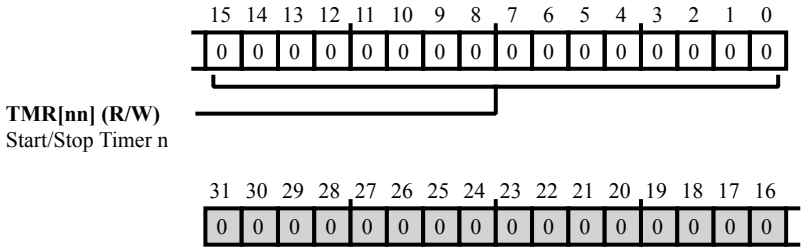


Figure 26-18: TIMER_RUN_CLR Register Diagram

Table 26-30: TIMER_RUN_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W1C)	TMR[nn]	RUN Clear Alias. For all TIMER_RUN_CLR.TMR[nn] bits, write =0 has no effect, and write =1 for stop (clearing the corresponding in start/stop bit in the TIMER_RUN register). Using TIMER_RUN_CLR to clear start/stop bits permits stopping specific timers without influencing run status of other timers.

Run Set Register

The TIMER_RUN_SET register is an alias register, providing a mechanism to set a specific start/stop bit in the TIMER_RUN register without affecting other bits in TIMER_RUN. To start a particular timer, software must write a 1 into the corresponding TIMER_RUN_SET bit. Writing a zero has no effect. For an example, to start timer 3 without affecting any other timer, write 0x0008 into TIMER_RUN_SET. Because TIMER_RUN_SET is a write-only register, the result of any write to this register must be checked by reading the TIMER_RUN register. A read of the TIMER_RUN_SET returns 0x0000.

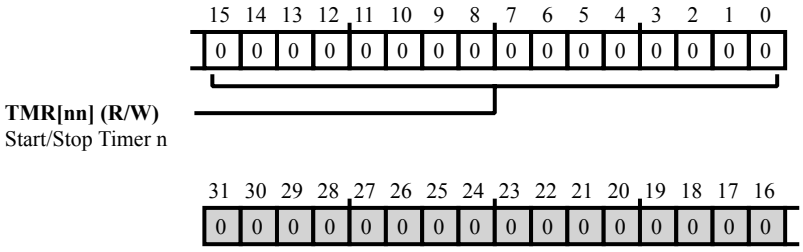


Figure 26-19: TIMER_RUN_SET Register Diagram

Table 26-31: TIMER_RUN_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W1S)	TMR[nn]	RUN Set Alias. For all TIMER_RUN_SET.TMR[nn] bits, write =0 has no effect, and write =1 for start (setting the corresponding start/stop bit in the TIMER_RUN register). Using TIMER_RUN_SET to set start/stop bits permits starting specific timers without influencing the run status of other timers.

Status Interrupt Latch Register

The TIMER_STAT_ILAT holds the latched interrupt status for error interrupt requests, indicating a timer overflow condition or indicating that prohibited programming has occurred for a timer. These interrupt status bits are sticky and are W1C. The bits in the TIMER_STAT_ILAT register provide information regarding each timer interrupt request source.

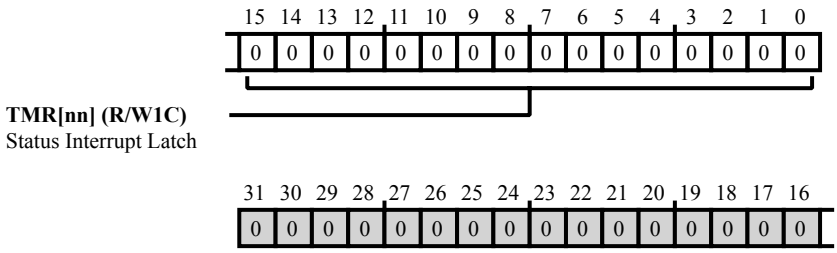


Figure 26-20: TIMER_STAT_ILAT Register Diagram

Table 26-32: TIMER_STAT_ILAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W1C)	TMR[nn]	Status Interrupt Latch. For all TIMER_STAT_ILAT.TMR[nn] bits, status of 0 indicates no error interrupt request is latched, and status of 1 indicates a timer counter overflow or programming error interrupt request is latched.

Status Interrupt Mask Register

While each timer may generate a status interrupt request, these requests are OR'ed to generate a single status interrupt signal to the system event controller. The TIMER_STAT_IMSK register contains an interrupt mask for these requests, masking (disabling) or unmasking (enabling) the interrupts as programmed. The reset value of the TIMER_STAT_IMSK register is 0xFFFF, masking these interrupts after reset.

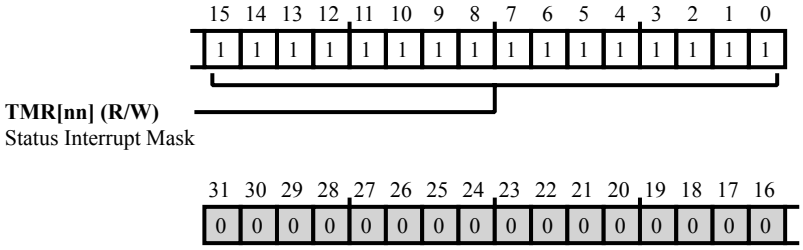


Figure 26-21: TIMER_STAT_IMSK Register Diagram

Table 26-33: TIMER_STAT_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Status Interrupt Mask. For all TIMER_STAT_IMSK.TMR[nn] bits, write =0 unmasks (enables) the corresponding status interrupt request, and write =1 masks (disables) the corresponding status interrupt request.

Stop Configuration Register

The TIMER_STOP_CFG register selects the stop mode for each timer. Timers may be stopped abruptly (immediate halt - all modes) or gracefully in PWMOUT modes (single pulse and continuous). The halt is achieved through either a write =0 to the corresponding bit in TIMER_RUN or a write =1 to the corresponding bit in TIMER_RUN_CLR. A read of TIMER_STOP_CFG returns the last value written.

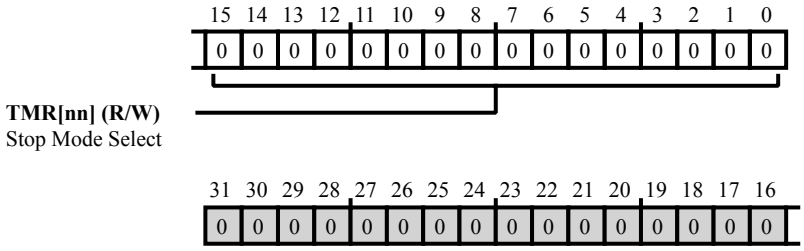


Figure 26-22: TIMER_STOP_CFG Register Diagram

Table 26-34: TIMER_STOP_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Stop Mode Select. For all TIMER_STOP_CFG.TMR[nn] bits, write =0 for graceful termination (PWMOUT modes only), and write =1 for abrupt (immediate halt) on stop.

Stop Configuration Clear Register

This is an alias register, providing a mechanism to clear a specific bit in the TIMER_STOP_CFG register without affecting other bits in TIMER_STOP_CFG. To clear a bit in TIMER_STOP_CFG, software must write a 1 to the corresponding bit of TIMER_STOP_CFG_CLR register. Writing a zero has no effect. Because the TIMER_STOP_CFG_CLR register is a write-only register, the result of any write to this register must be checked by reading the TIMER_STOP_CFG register. A read of the TIMER_STOP_CFG_CLR register returns 0x0000.

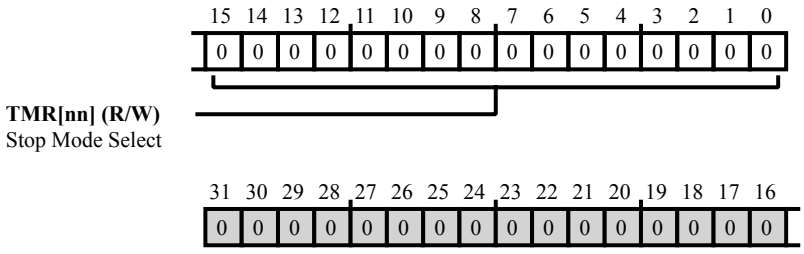


Figure 26-23: TIMER_STOP_CFG_CLR Register Diagram

Table 26-35: TIMER_STOP_CFG_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W1C)	TMR[nn]	STOP_CFG Clear Alias. For all TIMER_STOP_CFG_CLR.TMR[nn] bits, write =0 has no effect, and write =1 for graceful stop in PWMOUT modes (clearing the corresponding stop mode select bit in the TIMER_STOP_CFG register). Using TIMER_STOP_CFG_CLR to clear stop mode bits permits configuring specific timers without influencing the stop mode configuration of other timers.

Stop Configuration Set Register

This is an alias register, providing a mechanism to set a specific bit in the TIMER_STOP_CFG register without affecting other bits in TIMER_STOP_CFG. To set a bit in the TIMER_STOP_CFG register, software must write a 1 to the corresponding bit of the TIMER_STOP_CFG_SET register. Writing a zero has no effect. Because the TIMER_STOP_CFG_SET register is a write-only register, the result of any write to this register must be checked by reading the TIMER_STOP_CFG register. A read of the TIMER_STOP_CFG_SET register returns 0x0000.

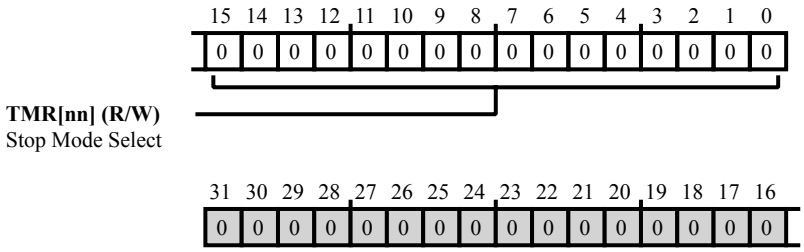


Figure 26-24: TIMER_STOP_CFG_SET Register Diagram

Table 26-36: TIMER_STOP_CFG_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W1S)	TMR[nn]	STOP_CFG Set Alias. For all TIMER_STOP_CFG_SET.TMR[nn] bits, write =0 has no effect, and write =1 for abrupt stop (setting the corresponding stop mode select bit in the TIMER_STOP_CFG register). Using TIMER_STOP_CFG_SET to set stop mode bits permits configuring specific timers without influencing the stop mode configuration of other timers.

Timer n Configuration Register

Each timer has a TIMER_TMR[n]_CFG register that specifies its operating mode. Only write to a TIMER_TMR[n]_CFG register when the corresponding timer is not running.

After disabling a timer operating in PWMOUT mode, verify that the timer has stopped running by checking the start/stop status of the timer in the TIMER_RUN register before writing to the timer's TIMER_TMR[n]_CFG register.

Note that a timer's TIMER_TMR[n]_CFG register may be read at any time.

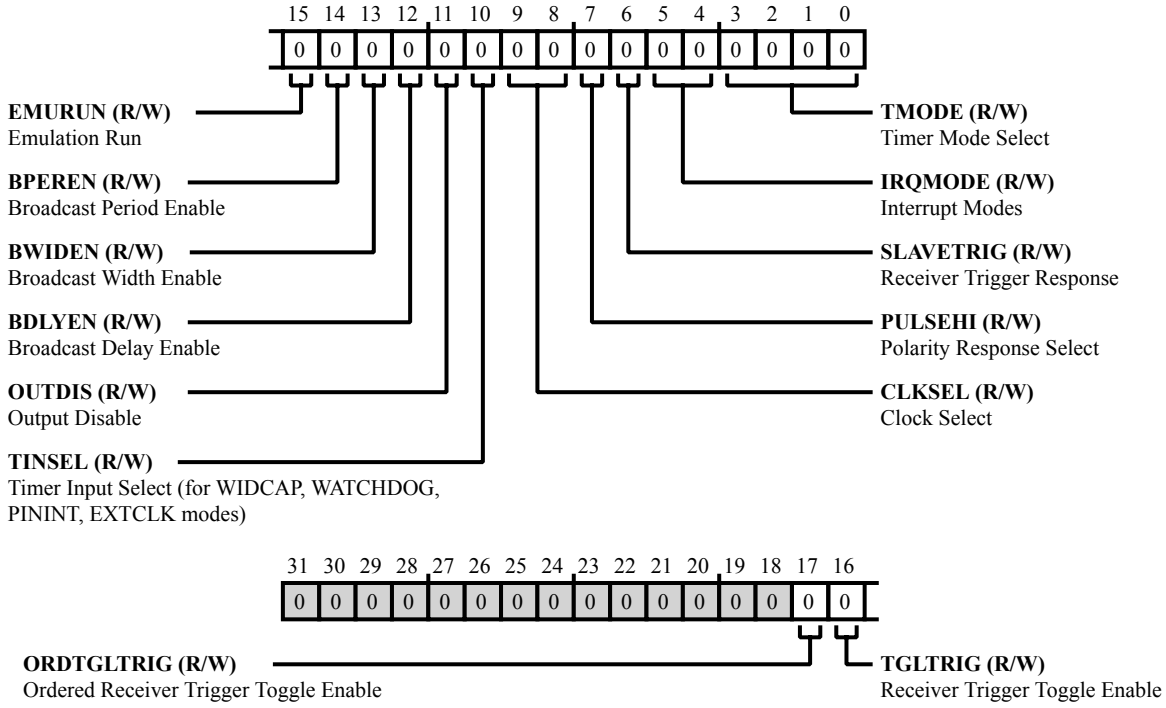


Figure 26-25: TIMER_TMR[n]_CFG Register Diagram

Table 26-37: TIMER_TMR[n]_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	ORDTGLTRIG	Ordered Receiver Trigger Toggle Enable. The TIMER_TMR[n]_CFG.ORDTGLTRIG bit controls ordered receive trigger toggle mode. The timer(receiver) state is not toggled by writes to the TIMER_RUN, TIMER_RUN_SET and TIMER_RUN_CLR registers when this bit is enabled.	
		0	No effect
		1	Trigger pulse on trigger receiver 0 starts the timer if it is in the halt state. Trigger pulse in trigger receiver 1 stops the timer if it is running.
16 (R/W)	TGLTRIG	Receiver Trigger Toggle Enable. The TIMER_TMR[n]_CFG.TGLTRIG bit stops the timer if it is running and starts the timer if it is halted (in the stop state). If the TIMER_TMR[n]_CFG.TGLTRIG bit is set, then the setting of the TIMER_TMR[n]_CFG.SLAVETRIG bit is ignored.	
		0	Receiver trigger response depends on SLAVETRIG bit
		1	Receiver trigger toggles timer state
15 (R/W)	EMURUN	Emulation Run. The TIMER_TMR[n]_CFG.EMURUN bit causes the timer to run (count) during emulation.	
		0	Stop Timer During Emulation
		1	Run Timer During Emulation
14 (R/W)	BPEREN	Broadcast Period Enable. The TIMER_TMR[n]_CFG.BPEREN bit enables updates to the TIMER_TMR[n]_PER register simultaneously across more than one timer.	
		0	Disable Broadcast to PER Register
		1	Enable Broadcast to PER Register
13 (R/W)	BWIDEN	Broadcast Width Enable. The TIMER_TMR[n]_CFG.BWIDEN bit enables updates to the TIMER_TMR[n]_WID register simultaneously across more than one timer.	
		0	Disable Broadcast to WID Register
		1	Enable Broadcast to WID Register

Table 26-37: TIMER_TMR[n]_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	BDLYEN	Broadcast Delay Enable. The `TIMER_TMR[n]_CFG.BDLYEN` bit enables updates to the `TIMER_TMR[n]_DLY` register simultaneously across more than one timer.	
		0	Disable Broadcast to DLY Register
		1	Enable Broadcast to DLY Register
11 (R/W)	OUTDIS	Output Disable. The `TIMER_TMR[n]_CFG.OUTDIS` bit enables or disables the timer pin output buffer.	
		0	Enable TMR Pin Output Buffer
		1	Disable TMR Pin Output Buffer
10 (R/W)	TINSEL	Timer Input Select (for WIDCAP, WATCHDOG, PININT, EXTCLK modes).	
		0	Use TMR Pin Input
		1	Use TMR Alternate Capture Input
9:8 (R/W)	CLKSEL	Clock Select. The `TIMER_TMR[n]_CFG.CLKSEL` bit field selects the TIMER clock to use.	
		0	Use SCLK0
		1	Use TMR_ALT_CLK0 as TMR Clock
		3	Use TMR_ALT_CLK1 as TMR Clock
7 (R/W)	PULSEHI	Polarity Response Select. The `TIMER_TMR[n]_CFG.PULSEHI` bit defines specific behaviors of the timer based on the operating mode. For more information, see the specific operating mode in the Programming Concepts section.	
		0	Negative Response or Pulse. A Negative Edge Response or Negative Action Pulse on the TMR pin.
		1	Positive Response or Pulse. A Positive Edge Response or Positive Action Pulse on the TMR pin.
6 (R/W)	SLAVETRIG	Receiver Trigger Response. The `TIMER_TMR[n]_CFG.SLAVETRIG` bit controls the trigger response. The trigger pulse has no effect (to stop or start the timer) if the timer is already in the requested state.	
		0	Pulse Stops Timer if it is Running
		1	Pulse Starts Timer if it is Stopped

Table 26-37: TIMER_TMR[n]_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:4 (R/W)	IRQMODE	Interrupt Modes. The TIMER_TMR[n]_CFG.IRQMODE bit field selects the interrupt request mode. Note that any mismatched combination of the TIMER_TMR[n]_CFG.IRQMODE and the TIMER_TMR[n]_CFG.TMODE bits results in no interrupt being generated. In WIDCAP modes, the position of the interrupt is controlled with the TIMER_TMR[n]_CFG.TMODE bit, and the TIMER_TMR[n]_CFG.IRQMODE bit is ignored. Whenever an interrupt is generated, a trigger generator pulse is also occurs, when enabled in the TIMER_TRG_MSK register.	
		0	Active Edge Mode. The timer generates an interrupt at every active edge. The active edge polarity depends on the state of the TIMER_TMR[n]_CFG.PULSEHI bit. Valid for PININT mode only.
		1	Delay Expired Mode. The timer generates an interrupt when the TIMER_TMR[n]_CNT value reaches the value in the TIMER_TMR[n]_DLY register. This mode is valid for all PWMOUT modes.
		2	Width Plus Delay Expired Mode. The timer generates an interrupt when the TIMER_TMR[n]_CNT value reaches the value in the TIMER_TMR[n]_WID register plus the value in the TIMER_TMR[n]_DLY register. (PWMOUT modes only) The timer generates an interrupt if the deasserting edge is within the specified window. (WATCHDOG modes only)
		3	Period Expired Mode. The timer generates an interrupt when the TIMER_TMR[n]_CNT value reaches the value in the TIMER_TMR[n]_PER register. (Continuous PWMOUT and EXTCLK modes only) The timer generates an interrupt if the deasserting edge is outside the specified window.(WATCHDOG modes only)
3:0 (R/W)	TMODE	Timer Mode Select. The TIMER_TMR[n]_CFG.TMODE bit field selects the operating mode of each timer.	
		0-7	Idle Mode
		8	Period Watchdog Mode
		9	Width Watchdog Mode

Table 26-37: TIMER_TMR[n]_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		10	Measurement Report at Asserting Edge of Waveform
		11	Measurement Report at Deasserting Edge of Waveform
		12	Continuous PWMOUT Mode
		13	Single Pulse PWMOUT Mode
		14	EXTCLK Mode
		15	PININT (pin interrupt) Mode

Timer n Counter Register

The TIMER_TMR[n]_CNT register holds the current timer count. After enabling, the count is re-initialized to either 0x0 or 0x1, depending on the configuration and mode. The TIMER_TMR[n]_CNT register is read-only and may be read at any time (whether the timer is running or stopped). Reading the TIMER_TMR[n]_CNT register returns an atomic 32-bit value.

Depending on the timer operation mode, the counter increment can be clocked by a number of sources, including SCLK0, the TMR or alternate capture input pins, ACLK[n]. The counter retains its value after the timer is disabled.

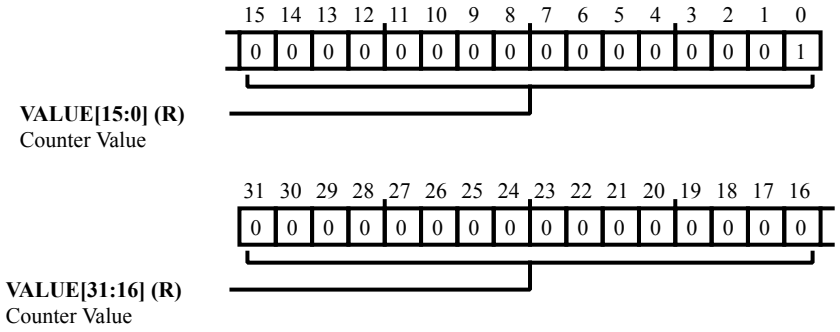


Figure 26-26: TIMER_TMR[n]_CNT Register Diagram

Table 26-38: TIMER_TMR[n]_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Counter Value. The TIMER_TMR[n]_CNT.VALUE bit field holds the current timer count.

Timer n Delay Register

The TIMER_TMR[n]_DLY register holds the delay value for the corresponding timer. This register's use is based on the selected timer mode.

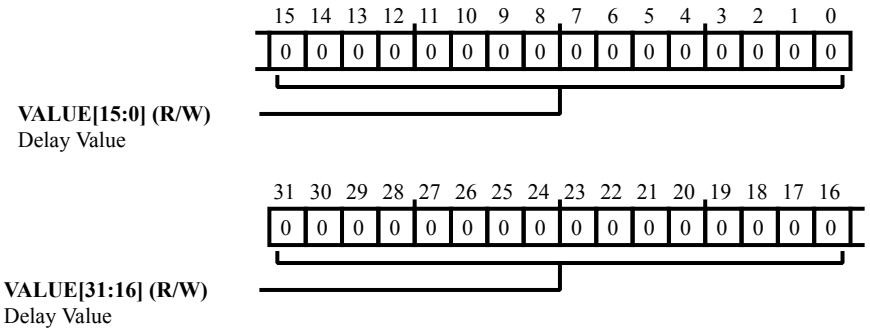


Figure 26-27: TIMER_TMR[n]_DLY Register Diagram

Table 26-39: TIMER_TMR[n]_DLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Delay Value. The TIMER_TMR[n]_DLY.VALUE bit field holds the delay value for the corresponding timer.

Timer n Period Register

The TIMER_TMR[n]_PER register holds the period value for the corresponding timer. This register's use is based on the selected timer mode.

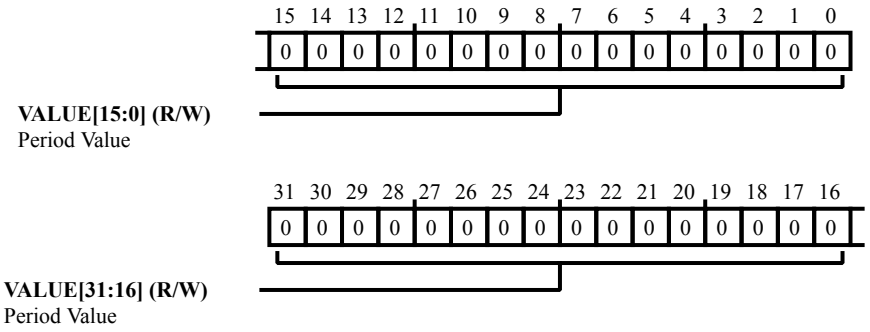


Figure 26-28: TIMER_TMR[n]_PER Register Diagram

Table 26-40: TIMER_TMR[n]_PER Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Period Value. The TIMER_TMR[n]_PER.VALUE bit field holds the period value for the corresponding timer.

Timer n Width Register

The TIMER_TMR[n]_WID register holds the width value for the corresponding timer. This register's use is based on the selected timer mode.

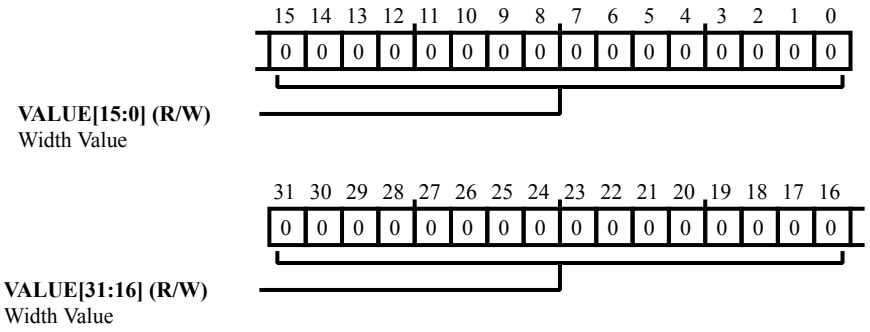


Figure 26-29: TIMER_TMR[n]_WID Register Diagram

Table 26-41: TIMER_TMR[n]_WID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Width Value. The TIMER_TMR[n]_WID.VALUE bit field holds the width value for the corresponding timer.

Trigger Receiver Enable Register

As a trigger receiver, each timer can generate a unique data trigger pulse signal. The `TIMER_TRG_IE` contains trigger input enable bits for these signals, disabling or enabling the triggers as programmed. The reset value of the `TIMER_TRG_IE` register is 0xFFFF, masking these triggers after reset.

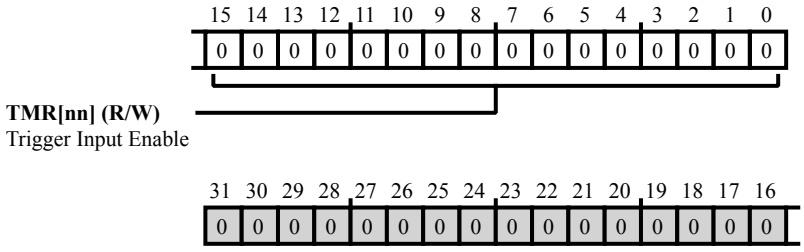


Figure 26-30: TIMER_TRG_IE Register Diagram

Table 26-42: TIMER_TRG_IE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Trigger Input Enable. For all `TIMER_TRG_IE.TMR[nn]` bits, write =0 disables the corresponding trigger input, and write =1 enables the corresponding trigger input.

Trigger Generator Mask Register

As a trigger generator, each timer can generate a unique data trigger pulse signal. The TIMER_TRG_MSK register contains a trigger mask for these outputs, masking (disabling) or unmasking (enabling) the triggers as programmed. The reset value of the TIMER_TRG_MSK register is 0xFFFF, masking these triggers after reset.

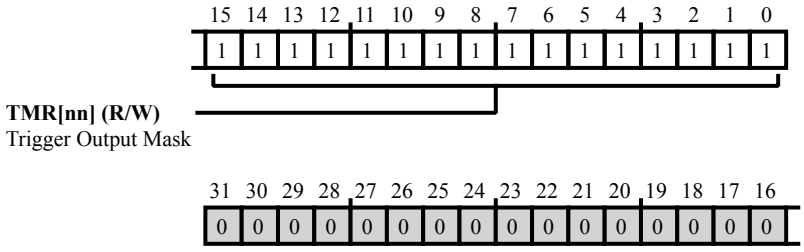


Figure 26-31: TIMER_TRG_MSK Register Diagram

Table 26-43: TIMER_TRG_MSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TMR[nn]	Trigger Output Mask. For all TIMER_TRG_MSK.TMR[nn] bits, write =0 unmasks (enables) the corresponding data trigger output, and write =1 masks (disables) the corresponding data trigger output.

27 General-Purpose Counter (CNT)

The GP counter converts pulses from incremental position encoders into data that is representative of the actual position of the pulse. This conversion is done by integrating (counting) pulses on one or two inputs. Since integration provides relative position, some devices also feature a zero-position input (zero marker). The GP counter can use the zero position input feature to establish a reference point for verifying that the acquired position does not drift over time. In addition, the GP counter can use the incremental position information to determine speed, if the time intervals are measured.

The GP counter provides flexible ways to establish position information. When used with the GP timer block, the GP counter can allow for the acquisition of coherent position or time stamp information that enables speed calculation.

GP Counter Features

The GP counter includes the following features:

- 32-bit up or down counter
- Quadrature encode mode (Gray code)
- Binary encoder mode
- Alternative frequency-direction mode
- Timed direction and up or down counting modes
- Zero marker or push-button support
- Capture event timing in association with GP Timer
- Boundary comparison and boundary setting features

GP Counter Functional Description

The ***GP Counter Block Diagram*** shows a block diagram of the GP counter. The `CNT_UD` and `CNT_DG` pins accept various forms of incremental inputs. The 32-bit counter processes the inputs. The GP counter uses the `CNT_ZM` pin to sense the pressing of a push button.

NOTE: When enabled, the GP counter requires 3 SCLK0 cycles of initialization before recognizing valid toggles on its input pins.

The three input pins can be filtered (debounced) before the GP counter evaluates them.

The GP counter features a flexible boundary comparison. In all of the operating modes, the counter can be compared to an upper and lower limit. It takes various actions when reaching these limits.

The GP counter has a flexible input selection. Apart from accepting inputs from the CNT0_UD and CNT0_DG and PORT pins, the counter can be configured to accept trigger inputs by setting PADS_PCFG0.CNT0UDSEL and PADS_PCFG0.CNT0DGSEL bits. Refer to the PADS_PCFG0 register description in General-Purpose Ports (PORT) chapter for details.

The module can optionally generate an interrupt request to the system through its IRQ line. On many processors, a GP timer module can use an output to generate time stamps on certain events.

ADSP-SC59x CNT Register List

The GP Counter (CNT) provides support for manually controlled rotary controllers, such as the volume wheel on a radio device. This unit also supports industrial encoders.

The CNT converts pulses from incremental position encoders into data that is representative of the actual position. To complete this task, the CNT integrates (counts) pulses on one or two inputs. Because integration provides relative position, a zero position input (zero marker) is usually provided that establishes a reference point, verifying that the acquired position does not drift over time. The incremental position information may also be used to determine speed, if the relevant time intervals are measured. The CNT provides flexible ways to establish position information. When used in conjunction with the General-purpose Timer (TIMER), the CNT allows acquisition of coherent position/time stamp information, enabling speed calculation.

A set of registers govern CNT operations. For more information on CNT functionality, see the CNT register descriptions.

Table 27-1: ADSP-SC59x CNT Register List

Name	Description
CNT_CFG	Configuration Register
CNT_CMD	Command Register
CNT_CNTR	Counter Register
CNT_DEBNCE	Debounce Register
CNT_IMSK	Interrupt Mask Register
CNT_MAX	Maximum Count Register
CNT_MIN	Minimum Count Register
CNT_STAT	Status Register

ADSP-SC59x CNT Interrupt List

Table 27-2: ADSP-SC59x CNT Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
156	CNT0_STAT	CNT0 Status	Level	

ADSP-SC59x CNT Trigger List

Table 27-3: ADSP-SC59x CNT Trigger List Generators

Trigger ID	Name	Description	Sensitivity
3	CNT0_STAT	CNT0 Status	Level
4	CNT0_UD	CNT0 CNT0 Count Up and Direction	Level
5	CNT0_DG	CNT0 CNT0 Count Down and Gate	Level
6	CNT0_TO	CNT0 CNT0 Output to Timer Block	Level

Table 27-4: ADSP-SC59x CNT Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
0	CNT0_UD	CNT0 CNT0 Count Up and Direction	Pulse
1	CNT0_DG	CNT0 CNT0 Count Down and Gate	Pulse

GP Counter Operating Modes

The GP counter has the following modes of operation.

1. Quadrature Encoder
2. Binary Encoder
3. Up/Down Counter
4. Direction Counter
5. Timed Direction

With the exception of the timed direction mode, the GP counter can operate with the GP timer block to capture additional timing information (time stamps) associated with events detected by this block.

Quadrature Encoder Mode

In this mode, the CNT_UD and CNT_DG inputs expect a quadrature-encoded signal that is interpreted as a two-bit Gray code. The order of transitions of the CNT_UD and CNT_DG inputs determines whether the counter increments or decrements. The CNT_CNTR register contains the number of transitions that have occurred as shown in the ***Quadrature Events and Counting Mechanism*** table. Optionally, an interrupt is generated when both inputs

change within one SCLK0 cycle. Gray coding prohibits such transitions. Therefore, the CNT_CNTR register remains unchanged, and an error condition is signaled.

Table 27-5: Quadrature Events and Counting Mechanism

CNT_COUNTER Register Value	-4	-3	-2	-1	0	+1	+2	+3	+4
CDG, CUD Inputs	00	01	11	10	00	01	11	10	00

It is possible to reverse the count direction of the Gray-coded signal by enabling the polarity inverter of either the CNT_UD pin or the CNT_DG pin. Inverting both pins does not alter the behavior. The GP counter can enable this feature with the CNT_CFG.CDGINV and CNT_CFG.CUDINV bits.

As an example, the CNT_DG and CNT_UD inputs are 00 and the next transition is to 01. These inputs normally change the counter in increments as shown in the table. If the CNT_UD polarity is inverted, this condition generates a received input of 01 followed by 00. The results is a decrement of the counter, altering the behavior of the connected hardware.

Binary Encoder Mode

This mode is almost identical to quadrature encoder mode, with the exception that the CNT_UD: CNT_DG inputs expect a binary-encoded signal. The order of transitions of the CNT_UD and CNT_DG inputs determines whether the counter increments or decrements. The CNT_CNTR register contains the number of transitions that have occurred as shown in the ***Binary Events and Counting Mechanism*** table. Optionally, an interrupt is generated when the detected code steps by more than 1 (in binary arithmetic) within one SCLK0 cycle. Such transitions are erroneous. Therefore, the CNT_CNTR register remains unchanged, and an error condition is signaled.

Table 27-6: Binary Events and Counting Mechanism

CNT_COUNTER Register Value	-4	-3	-2	-1	0	+1	+2	+3	+4
CDG:CUD Inputs	00	01	10	11	00	01	10	11	00

Reversing the CNT_UD and CNT_DG pin polarity has a different effect in binary encoder mode than for the quadrature encoder mode. Inverting the polarity of the CNT_UD pin only or inverting both the CNT_UD and CNT_DG pins, results in reversing the count direction.

Up/Down Counter Mode

In this mode, the counter increments or decrements at every active edge of the input pins. The GP counter uses the CNT_CFG.CUDINV bit to select an active edge and has the following results.

- If the GP counter module detects an active edge at the CNT_UD input, the counter increments.
- If the GP counter module detects an active edge at the CNT_DG input, the counter decrements.
- If simultaneous edges occur on the CNT_DG and CNT_UD pins, the counter remains unchanged, and both up-count and down-count events are signaled in the CNT_STAT register.

Direction Counter Mode

In this mode, the counter is incremented or decremented at every active edge of the CNT_DG input pin. The state of the CNT_UD input determines whether the counter increments or decrements. The GP counter uses the CNT_CFG.CUDINV bit to select the polarity.

If the GP counter detects an active edge at the CNT_DG input, the counter value changes by one in the selected direction.

Timed Direction Mode

In this mode, the counter is incremented or decremented at each SCLK0 cycle. The state of the CNT_UD input determines whether the counter increments or decrements. The GP counter uses the CNT_CFG.CUDINV bit to select the polarity. The CNT_DG pin can be used to gate the clock. The GP counter uses the CNT_CFG.CDGINV bit to select the polarity.

GP Counter Programming Model

The following sections provide information for programming the interface.

GP Counter General Programming Flow

The following are general guidelines for configuring and enabling the GP counter.

1. Initialize (but do not enable) the GP counter for the desired mode and settings through the CNT_CFG register.
2. Usually, events of interest are processed using interrupts rather than by polling status bits. In this case, clear all status bits and activate the interrupt generation requests with the CNT_IMSK register.
3. Configure interrupts at the system level to insure desired interrupt signaling to the system.
4. If timing information is required, set up the relevant GP Timer in width capture mode.
5. Finally, enable interrupt requests and the GP Counter itself using the CNT_IMSK and CNT_CFG registers, respectively.

GP Counter Mode Configuration

The GP counter can use the CNT_ZM input pin to sense the zero marker output of a rotary device or to detect the pressing of a push button. There are four programming schemes, which are functional in all counter modes:

- Push-button mode
- Zero-marker-zeros-counter mode
- Zero-marker-error mode
- Zero-once mode

Configuring GP Counter Push-Button Operation

Use the following procedure to configure push-button operation:

1. Set CNT_IMSK.CZM to enable (unmask) the zero marker interrupt.
2. Select the active edge polarity through the CNT_CFG.CZMINV bit.
3. Proceed with any other desired configuration steps and enable the peripheral.

An active edge at the CNT_ZM input sets the CNT_IMSK.CZM bit.

Configuring Zero-Marker-Zeros-Counter Mode

The following provides information on configuring zero-marker-zeros-counter mode for the GP counter.

1. Set CNT_IMSK.CZMZ to enable CNT_CNTR. The zero marker interrupt zeroes the counter.
2. Set CNT_CFG.ZMZC to enable ZMZC mode.
3. Select the active edge polarity through the CNT_CFG.CZMINV bit.
4. Proceed with any other desired configuration steps and enable the peripheral.

This configuration causes an active level at the CNT_ZM pin to clear the CNT_CNTR register and keep it cleared until the CNT_ZM pin is deactivated. In addition, the CNT_STAT.CZMZ bit is set.

Configuring Zero-Marker-Error Mode

The GP counter uses this mode to detect discrepancies between counter-value and the zero marker output of certain rotary encoder devices.

1. Set the CNT_STAT.CZME bit to enable this mode.
2. Select the active edge of the CNT_ZM pin through the CNT_CFG.CZMINV bit.
3. Proceed with any other desired configuration steps and enable the peripheral.

When the GP counter detects an active edge at the CNT_ZM input pin, it compares the four LSBs of the CNT_CNTR register to zero. If they are not zero, the GP counter uses CNT_STAT.CZME bit to signal a mismatch.

Configuring Zero-Once Mode

The GP counter uses this mode to perform an initial reset of the counter-value when it detects an active zero marker. After that, the zero marker is ignored (the counter is no longer reset).

1. Set the CNT_CMD.W1ZMONCE bit to enable this mode.
2. Select the active edge of the CNT_ZM pin through the CNT_CFG.CZMINV bit.
3. Ensure that at least one of the following bits is enabled: CNT_IMSK.CZM, CNT_IMSK.CZME, CNT_IMSK.CZMZ.

4. Proceed with any other desired configuration steps and enable the peripheral.

The CNT_CNTR register and the CNT_CMD.W1ZMONCE bit are cleared on the next active edge of the CNT_ZM pin. Now the CNT_CMD.W1ZMONCE bit can be read to check whether the event has already occurred.

Configuring Boundary Auto-Extend Mode

In this mode, hardware modifies the boundary registers (CNT_MIN and CNT_MAX) whenever the CNT_CNTR value reaches either of them. The GP counter uses this mode to monitor the widest angle a thumb wheel even if the software did not generate interrupts.

1. Initialize CNT_CNTR with the desired value.
2. Set both CNT_MIN and CNT_MAX to this same value.
3. Configure the CNT_CFG.BNDMODE field for auto extend mode.
4. Proceed with any other desired configuration steps and enable the peripheral.

The CNT_MAX register is loaded with the current CNT_CNTR value when the latter increments beyond the CNT_MAX value. Similarly, the CNT_MIN register is loaded with the CNT_CNTR value when the latter decrements below the CNT_MIN value. The CNT_STAT.MAXC and CNT_STAT.MINC status bits are set when the CNT_CNTR value matches the respective boundary register value.

Configuring Boundary Capture Mode

In this mode, the CNT_CNTR value is latched into the CNT_MIN register at one detected edge of the CNT_ZM input pin and latched into the CNT_MAX boundary register at the opposite edge.

1. To capture the CNT_ZM pin rising edge into CNT_MIN and the falling edge into CNT_MAX, program CNT_CFG.CZMINV for active high polarity. Conversely, to capture the CNT_ZM pin falling edge into CNT_MIN and the rising edge into CNT_MAX, program CNT_CFG.CZMINV for active low polarity.
2. Program the CNT_IMSK.MAXC and CNT_IMSK.MINC interrupt mask bits according to interrupt generation requirements.
3. Configure the CNT_CFG.BNDMODE field for boundary capture mode.
4. Proceed with any other desired configuration steps and enable the peripheral.

The CNT_STAT.MAXC and CNT_STAT.MINC status bits report the capture event, depending on how interrupt masks are configured.

Configuring Boundary Compare and Boundary Zero Modes

In these modes, the two boundary registers (CNT_MAX and CNT_MIN) are compared to the value in the CNT_CNTR register.

1. Program CNT_MAX and CNT_MIN registers with appropriate upper and lower range values.

2. Program the CNT_IMSK.MAXC and CNT_IMSK.MINC interrupt mask bits according to interrupt generation requirements.
3. Configure the CNT_CFG.BNDMODE field for boundary compare mode.
4. Proceed with any other desired configuration steps and enable the peripheral.

If after incrementing, CNT_CNTR = CNT_MAX, then the CNT_STAT.MAXC bit is set. Similarly if after decrementing, CNT_CNTR = CNT_MIN, then the CNT_STAT.MINC bit is set.

Additionally, for boundary zero mode, the counter-value in CNT_CNTR is set to zero. The CNT_STAT.MAXC and CNT_STAT.MINC bits are not set when software updates the CNT_MAX or CNT_MIN registers.

Configuring GP Counter Push-Button Operation

Use the following procedure to configure push-button operation:

1. Set CNT_IMSK.CZM to enable (unmask) the zero marker interrupt.
2. Select the active edge polarity through the CNT_CFG.CZMINV bit.
3. Proceed with any other desired configuration steps and enable the peripheral.

An active edge at the CNT_ZM input sets the CNT_IMSK.CZM bit.

GP Counter Programming Concepts

Using the features, operating modes, and event control for the GP counter to their greatest potential requires an understanding of some GP counter-related concepts. Some key aspects to consider are input noise filtering and capturing timing information.

CNT Input Noise Filtering

In all modes, the three input pins can be filtered to present clean signals to the GP counter logic. The GP counter uses the CNT_CFG.DEBEN bit to enable or disable this filtering. The ***Programmable Noise Filtering*** figure shows the filtering operation for the CNT_UD pin.

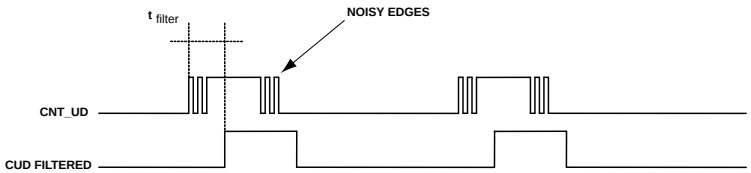


Figure 27-1: Programmable Noise Filtering

The CNT module implements the filtering mechanism using counters for each GP counter pin, where each counter is initialized from the CNT_DEBNCE.DPRESCALE field. When a transition is detected on a pin, the corresponding counter starts counting up to the programmed number of SCLK0 cycles. The state of the pin is latched after time t_{filter} and passed on to the GP counter logic.

The following formula determines the time t_{filter}, given SCLK0 and the CNT_DEBNCE.DPRESCALE value, where lower values of CNT_DEBNCE.DPRESCALE result in shorter debounce delays:

$t_{filter} = 128 \times (2^{DPRESCALE} \times SCLK0)$

Capturing Counter Interval and CNT_CNTR Read Timing

When the count speed is low, it is often useful to capture the time elapsed since the last count event. Program the TIMER_TMR[n]_CFG register of the associated GP timer in a width capture mode with the following bit settings.

- TIMER_TMR[n]_CFG.PULSEHI = 0
- TIMER_TMR[n]_CFG.TMODE = b#1011
- TIMER_TMR[n]_CFG.TINSEL = 1

The ***Capture Intervals*** figure shows and the following list describes the operation of the GP counter and the GP timer in this mode.

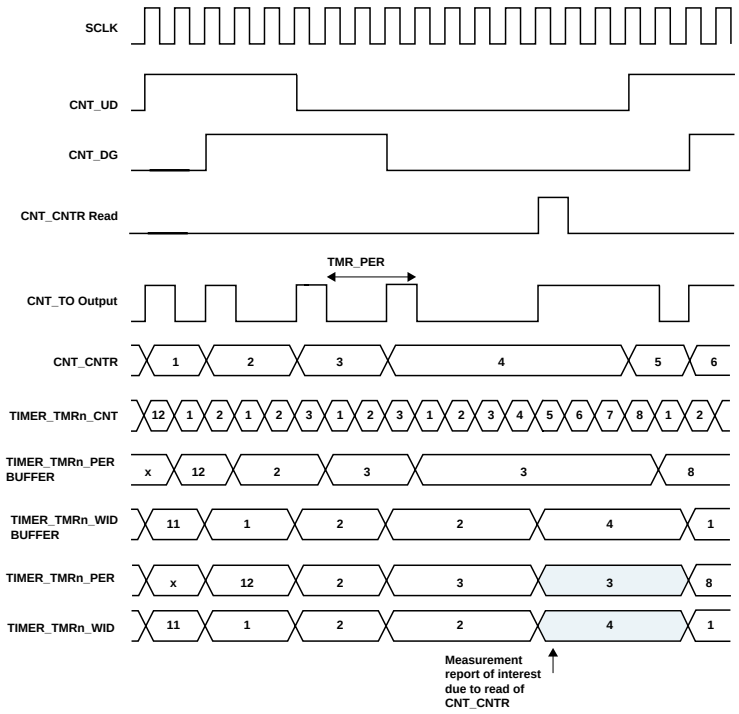


Figure 27-2: Capture Intervals

NOTE: SCLK in the ***Capture Intervals*** figure is SCLK0.

1. The CNT_TO signal generates a pulse every time a count event occurs. In addition, when the processor reads the CNT_CNTR register, the CNT_TO signal presents a pulse which is extended (high) until the next count event.
2. The GP timer updates its TIMER_TMR[n]_PER register with the period (measured from falling edge to falling edge, because TIMER_TMR[n]_CFG.PULSEHI = 0) of the CNT_TO signal.

3. The TIMER_TMR[n]_WID register is updated with the pulse width (the portion where CNT_TO is low, again because TIMER_TMR[n]_CFG.PULSEHI = 0).
4. Both registers are updated at every rising edge of the CNT_TO signal (because TIMER_TMR[n]_CFG.TMODE= b#011).

The TIMER_TMR[n]_PER register contains the period between the last two count events. The TIMER_TMR[n]_WID register contains the time since the last count event and the read of the CNT_CNTR register, both measured in SCLK0 cycles.

Read the CNT_CNTR register to latch the two time measurements, providing a coherent triplet of information to calculate speed and position.

NOTE: Speed restrictions apply to the use of the CNT_TO signal. Therefore, programs must not operate at count event rates that are high. For instance, if CNT_CNTR is incremented or decremented every SCLK0 cycle (timed direction mode), the CNT_TO signal is not valid.

Capturing Time Interval Between Successive Counter Events

When the required timing information is the interval between successive count events, program the associated timer in a width capture mode. Set the TIMER_TMR[n]_CFG bit of TIMER_TMR[n]_CFG.PULSEHI = 1, TIMER_TMR[n]_CFG.TMODE = b#1010 and TIMER_TMR[n]_CFG.TINSEL = 1. Typically, this information is sufficient if the speed of GP counter events does not to reach low values.

The ***Period Register Timing*** figure shows the operation of the GP counter and the GP timer in this mode.

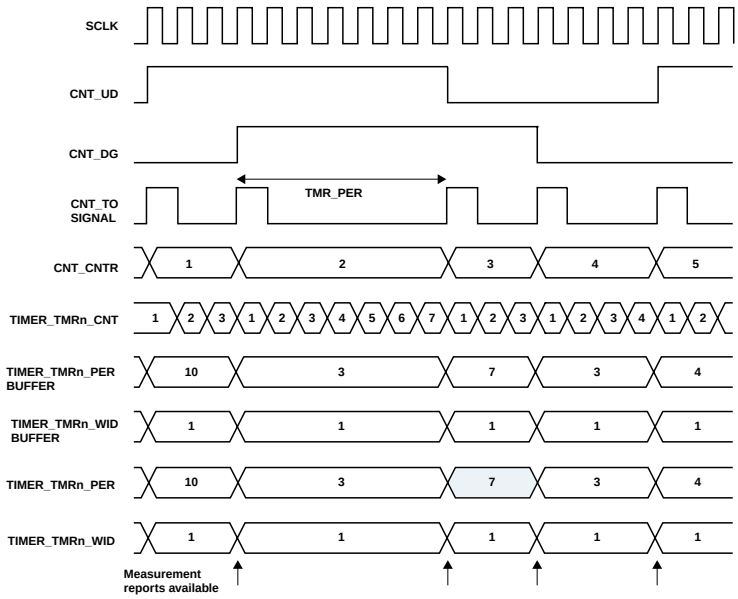


Figure 27-3: Period Register Timing

NOTE: SCLK in the ***Period Register Timing*** figure is SCLK0.

The CNT_TO signal generates a pulse every time a count event occurs. The GP timer updates its TIMER_TMR[n]_PER register with the period (measured from rising edge to rising edge) of the CNT_TO signal. The TIMER_TMR[n]_PER register is updated at every rising edge of the CNT_TO signal and contains the number of SCLK0 cycles that have elapsed since the previous rising edge.

Incidentally, the TIMER_TMR[n]_WID register is also updated at the same time, but is of no interest in this mode of operation. If no reads of the CNT_CNTR register occur between counter events, the TIMER_TMR[n]_WID register only contains the width of the CNT_TO pulse. If a read of CNT_CNTR has occurred between events, the TIMER_TMR[n]_WID register contains the time between the read of CNT_CNTR and the next event.

This mode can also be used with TIMER_TMR[n]_CFG.PULSEHI = 0. In this case, the period of CNT_TO is measured between falling edges. It results in the same values as in the previous case, only the latching occurs one SCLK0 cycle later.

GP Counter Event Control

Eleven events can be signaled to the processor using status information and optional interrupt requests. The GP counter uses the respective bits in the CNT_IMSK register to enable the interrupt requests. It uses dedicated bits in the CNT_STAT register to report events. When an interrupt request from the GP counter is serviced, the application software is responsible for correct interpretation of the events. It is recommended to logically AND the content of the CNT_IMSK and CNT_STAT registers to identify pending interrupt requests.

Perform a write-one-to-clear (W1C) operation to the CNT_STAT register to clear the interrupt requests. Hardware does not clear the status bits automatically unless the counter module is disabled.

The following sections describe the events associated with the GP counter.

Illegal Gray and Binary Code Events

When illegal transitions occur in quadrature encoder or binary encoder modes, the CNT_STAT.IC bit is set. If enabled by the CNT_STAT.IC bit, the counter module generates an interrupt request. Set the CNT_STAT.IC bit only in the quadrature encoder or binary encoder modes.

Up/Down Count Events

The GP counter uses the CNT_STAT.UC bit to indicate whether the counter has been incremented. Similarly, the CNT_STAT.DC bit reports decrements. The two events are independent. For instance, if the counter increments by one and then decrements by two, both bits remain set, even though the resulting counter-value shows a decrement by one.

In up/down counter mode, hardware can detect simultaneous active edges on the CNT_UD and CNT_DG inputs. In that case, the CNT_CNTR remains unchanged, but both the CNT_STAT.UC and CNT_STAT.DC bits are set. Interrupt requests for these events can be enabled through the CNT_IMSK.UC and CNT_IMSK.DC bits. Use this feature carefully when the counter is clocked at high rates. This suggestion is especially critical when the counter operates in DIR_TMR mode, as interrupts are generated every SCLK0 cycle.

These events can also be used for more push buttons, when GP counter features are unnecessary. When up/down counter mode is enabled, the GP counter can use these count events to report interrupts from push buttons that connect to the CNT_UD and CNT_DG inputs.

Zero-Count Events

The CNT_STAT.CZERO status bit indicates that the CNT_CNTR has reached a value equal to 0x0000 0000 after an increment or decrement. This bit is not set when the counter value is set to zero by a write to CNT_CNTR or by setting the CNT_CMD.W1LCNTZERO bit. If enabled by the CNT_IMSK.CZERO bit, the GP counter module generates an interrupt request.

Overflow Events

There are two status bits that indicate whether the signed counter-register has overflowed from a positive to a negative value or conversely. The CNT_STAT.COV31 bit reports that the 32-bit CNT_CNTR register has either incremented from 0x7FFF FFFF to 0x8000 0000, or decremented from 0x8000 0000 to 0x7FFF FFFF.

If enabled by the CNT_IMSK.COV31 bit, an interrupt request is generated. Similarly, in applications where only the lower 16 bits of the counter are of interest, the CNT_STAT.COV15 status bit reports counter transitions from 0xXXXX 7FFF to 0xXXXX 8000, or from 0xXXXX 8000 to 0xXXXX 7FFF. If enabled by the CNT_IMSK.COV15 bit, an interrupt request is generated.

Boundary Match Events

The CNT_STAT.MINC and CNT_STAT.MAXC status bits report boundary events as described in Configuring Boundary Capture Mode. These bits are not set if the software updates the CNT_CNTR, CNT_MAX, or CNT_MIN registers or writes to the CNT_CMD register. The CNT_IMSK.MINC and CNT_IMSK.MAXC bits enable interrupt request generation on boundary events.

Zero Marker Events

The CNT_STAT.CZM, CNT_STAT.CZME, and CNT_STAT.CZMZ bits are associated with zero marker events, as described in Configuring GP Counter Push-Button Operation. Each of these events can optionally generate an interrupt request, when enabled by the corresponding CNT_IMSK.CZM, CNT_IMSK.CZME and CNT_IMSK.CZMZ bits.

ADSP-SC59x CNT Register Descriptions

General-Purpose Counter (CNT) contains the following registers.

Table 27-7: ADSP-SC59x CNT Register List

Name	Description
CNT_CFG	Configuration Register
CNT_CMD	Command Register

Table 27-7: ADSP-SC59x CNT Register List (Continued)

Name	Description
CNT_CNTR	Counter Register
CNT_DEBNCE	Debounce Register
CNT_IMSK	Interrupt Mask Register
CNT_MAX	Maximum Count Register
CNT_MIN	Minimum Count Register
CNT_STAT	Status Register

Configuration Register

The CNT_CFG register configures counter modes, configures input pins, and enables the CNT.

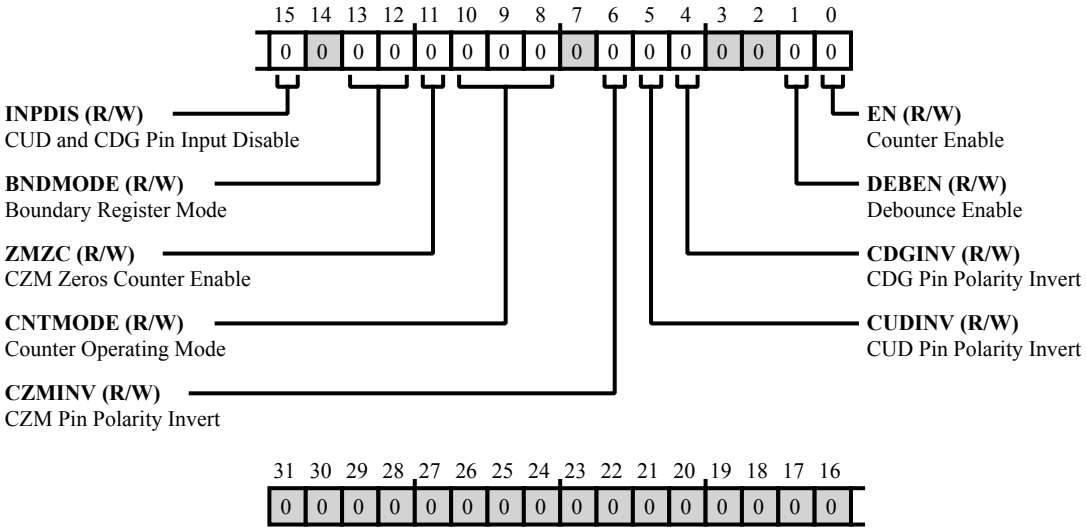


Figure 27-4: CNT_CFG Register Diagram

Table 27-8: CNT_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	INPDIS	CUD and CDG Pin Input Disable. The CNT_CFG.INPDIS disables or enables the CNT_UD input pin and the CNT_DG pin.	
		0	Enable
		1	Pin Input Disable
13:12 (R/W)	BNDMODE	Boundary Register Mode. The CNT_CFG.BNDMODE bit field selects the mode for the CNT_MIN and CNT_MAX boundary registers.	
		0	BND_COMP. Boundary Compare Mode
		1	BND_ZERO. Boundary Zero Mode
		2	BND_CAPT. Boundary Capture Mode
		3	BND_AEXT. Boundary Auto-extend Mode

Table 27-8: CNT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	ZMZC	CZM Zeros Counter Enable. The CNT_CFG.ZMZC bit enables or disables level sensitive - active CNT_ZM pin operation to zero the CNT_CNTR register.	
		0	Disable
		1	Enable
10:8 (R/W)	CNTMODE	Counter Operating Mode. The CNT_CFG.CNTMODE bit field selects the operating mode for the CNT_UD input pin and the CNT_DG pin.	
		0	QUAD_ENC. Quadrature Encoder Mode
		1	BIN_ENC. Binary Encoder Mode
		2	UD_CNT. Rotary Counter Mode
		4	DIR_CNT. Direction Counter Mode
		5	DIR_TMR. Direction Timer Mode
6 (R/W)	CZMINV	CZM Pin Polarity Invert. The CNT_CFG.CZMINV bit selects the polarity for the CNT_ZM pin. This polarity must be configured before the counter is enabled. It must not change on-the-fly while the counter is enabled.	
		0	Active High, Rising Edge
		1	Active Low, Falling Edge
5 (R/W)	CUDINV	CUD Pin Polarity Invert. The CNT_CFG.CUDINV bit selects the polarity for the CNT_UD pin. This polarity must be configured before the counter is enabled. It must not change on-the-fly while the counter is enabled.	
		0	Active High, Rising Edge
		1	Active Low, Falling Edge
4 (R/W)	CDGINV	CDG Pin Polarity Invert. The CNT_CFG.CDGINV bit selects the polarity for the CNT_DG pin. This polarity must be configured before the counter is enabled. It must not change on-the-fly while the counter is enabled.	
		0	Active High, Rising Edge
		1	Active Low, Falling Edge

Table 27-8: CNT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	DEBEN	Debounce Enable. The `CNT_CFG.DEBEN` bit enables or disables CNT input debounce filtering operation selected with the `CNT_DEBNCE` register.	
		0	Disable
		1	Enable
0 (R/W)	EN	Counter Enable. The `CNT_CFG.EN` bit enables or disables CNT operation.	
		0	Counter Disable
		1	Counter Enable

Command Register

The CNT_CMD register configures the CNT, enabling operations such as zeroing a counter register and copying or swapping boundary registers. These actions are taken by setting the appropriate bit.

Read operations from this register do not return meaningful values, with the exception of the CNT_CMD.W1ZMONCE bit, where a set bit indicates that the bit has been set by software before, but a zero marker event has not yet been detected on the CNT_ZM pin yet. For more information, see the CNT functional description.

The CNT_CNTR, CNT_MIN, and CNT_MAX registers can be initialized to zero by setting the CNT_CMD.W1LCNTZERO, CNT_CMD.W1LMINZERO, and CNT_CMD.W1LMAXZERO bits. In addition to clearing registers, the CNT_CMD register permits modifying the CNT_MIN and CNT_MAX boundary registers in a number of ways. The current counter value in the CNT_CNTR register can be captured and loaded into either of the two boundary registers to create new boundary limits. This operation is performed by setting the CNT_CMD.W1LMAXCNT and CNT_CMD.W1LMINCNT bits. Alternatively, the counter can be loaded from CNT_MAX or CNT_MIN using the CNT_CMD.W1LCNTMAX and CNT_CMD.W1LCNTMIN bits. It is also possible to transfer the current CNT_MAX value into CNT_MIN (or conversely) through the CNT_CMD.W1LMINMAX and CNT_CMD.W1LMAXMIN bits.

Another counter operation is the ability to only have the zero marker clear the CNT_CNTR register once. For more information, see the CNT functional description.

It is possible for multiple actions to be performed simultaneously by setting multiple bits in the CNT_CMD register. However, there are restrictions. The bits associated with each command have been grouped together such that all bits that involve a write to the CNT_CNTR, CNT_MAX, or CNT_MIN registers are located within bits 4-bit groups of the CNT_CMD register.

Note that a maximum of three commands can be issued at any one time, excluding the CNT_CMD.W1ZMONCE command. Also, note that CNT_CMD.W1LCNTMIN, CNT_CMD.W1LCNTMAX, and CNT_CMD.W1LCNTZERO bits have to be used exclusively. Never set more than one of them at the same time.

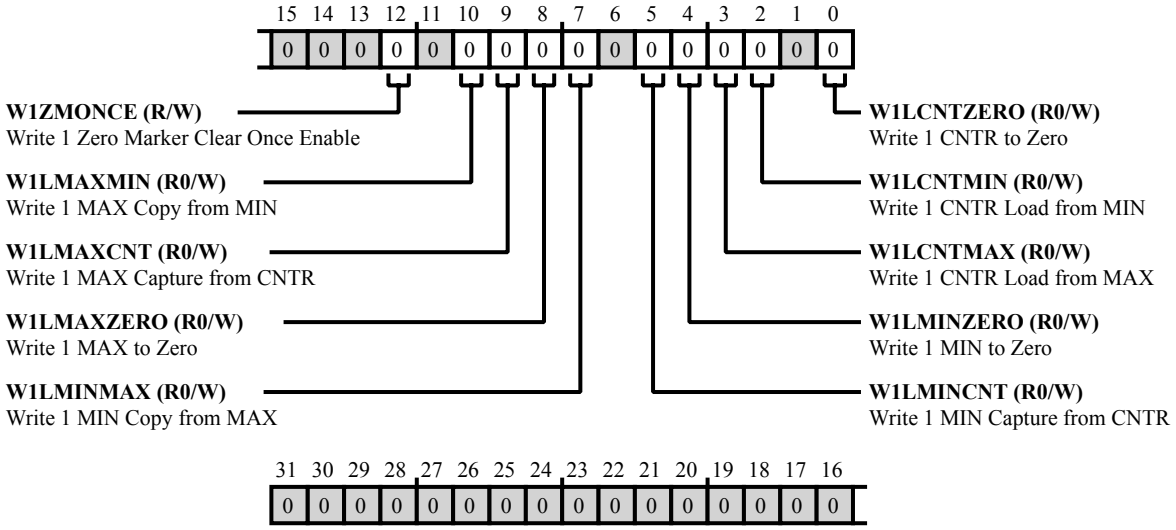


Figure 27-5: CNT_CMD Register Diagram

Table 27-9: CNT_CMD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12 (R/W)	W1ZMONCE	Write 1 Zero Marker Clear Once Enable. The CNT_CMD.W1ZMONCE enables a single zero marker clear of the CNT_CNTR register. Reading a 1 in this bit indicates that the bit has been set by software before, but no zero marker event has been detected on the CNT_ZM pin yet.
10 (R0/W)	W1LMAXMIN	Write 1 MAX Copy from MIN. The CNT_CMD.W1LMAXMIN bit transfers the current CNT_MIN register value into CNT_MAX register.
9 (R0/W)	W1LMAXCNT	Write 1 MAX Capture from CNTR. The CNT_CMD.W1LMAXCNT bit loads the current value in the CNT_CNTR register into the CNT_MAX register to create a new boundary limit.
8 (R0/W)	W1LMAXZERO	Write 1 MAX to Zero. Writing a 1 to the CNT_CMD.W1LMAXZERO bit clears the CNT_MAX register.
7 (R0/W)	W1LMINMAX	Write 1 MIN Copy from MAX. The CNT_CMD.W1LMINMAX bit transfers the current CNT_MAX register value into CNT_MIN register.
5 (R0/W)	W1LMINCNT	Write 1 MIN Capture from CNTR. The CNT_CMD.W1LMINCNT bit loads the current value in the CNT_CNTR register into the CNT_MIN register to create a new boundary limit.
4 (R0/W)	W1LMINZERO	Write 1 MIN to Zero. Writing a 1 to the CNT_CMD.W1LMINZERO bit clears the CNT_MIN register.

Table 27-9: CNT_CMD Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R0/W)	W1LCNTMAX	Write 1 CNTR Load from MAX. The CNT_CMD.W1LCNTMAX bit loads the current value in the CNT_MAX register into the CNT_CNTR register to create a new boundary limit.
2 (R0/W)	W1LCNTMIN	Write 1 CNTR Load from MIN. The CNT_CMD.W1LCNTMIN bit loads the current value in the CNT_MIN register into the CNT_CNTR register to create a new boundary limit.
0 (R0/W)	W1LCNTZERO	Write 1 CNTR to Zero. Writing a 1 to the CNT_CMD.W1LCNTZERO bit clears the CNT_CNTR register.

Counter Register

The CNT_CNTR register holds the 32-bit, two's-complement count value. It can be read and written at any time. Hardware ensures that reads and write are atomic, by providing respective shadow registers. This register can be accessed with either 32-bit or 16-bit operations. This allows use of the CNT as a 16-bit counter if sufficient for the application.

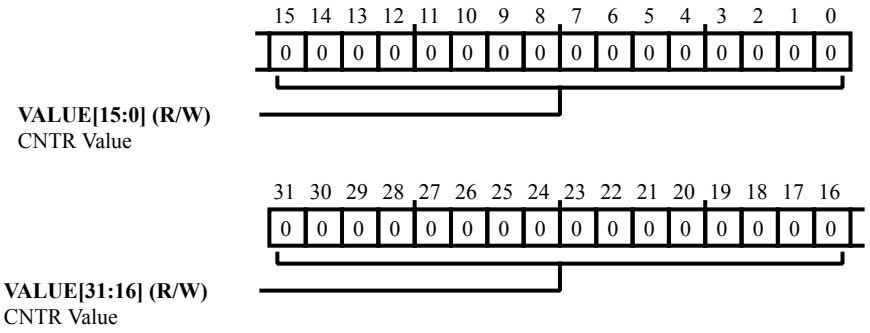


Figure 27-6: CNT_CNTR Register Diagram

Table 27-10: CNT_CNTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	CNTR Value. The CNT_CNTR.VALUE bit field holds the 32-bit, two's-complement count value.

Debounce Register

The CNT_DEBNCE register selects the noise filtering characteristic of the three input pins according to the formula:

t_{filter} = 128 x ($2^{DPRESCALE}$/ SCLK0)

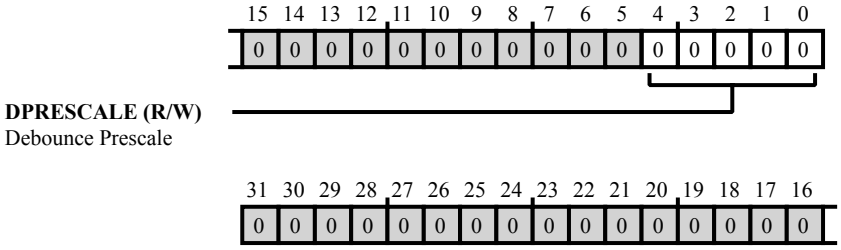


Figure 27-7: CNT_DEBNCE Register Diagram

Table 27-11: CNT_DEBNCE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4:0 (R/W)	DPRESCALE	Debounce Prescale. The CNT_DEBNCE.DPRESCALE selects the desired number of input filtering cycles (and resulting input debounce time) in multiples of SCLK0.	
		0	1x Cycles = 128 SCLK0 Cycles
		1	2x Cycles
		2	4x Cycles
		3	8x Cycles
		4	16x Cycles
		5	32x Cycles
		6	64x Cycles
		7	128x Cycles
		8	256x Cycles
		9	512x Cycles
		10	1024x Cycles
		11	2048x Cycles
		12	4096x Cycles
		13	8192x Cycles
		14	16384x Cycles
		15	32768x Cycles
		16	65536x Cycles

Table 27-11: CNT_DEBNCE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		17	131072x Cycles
		18	Reserved from This Value. The values 10010 - 11111 are reserved.
		31	Reserved Until This Value

Interrupt Mask Register

The CNT_IMSK register supports enabling (unmasking) interrupt request generation from each of the CNT events.

All bits in CNT_IMSK either disable/mask an interrupt request (if bit cleared) or enable/unmask an interrupt request (if bit set).

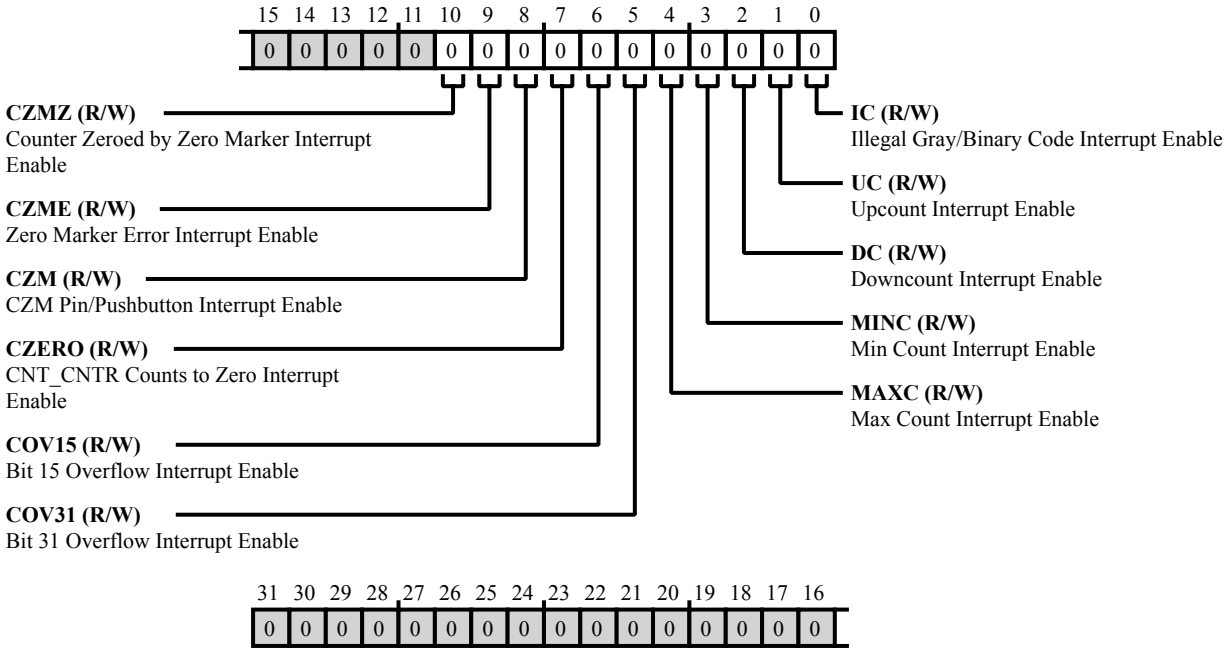


Figure 27-8: CNT_IMSK Register Diagram

Table 27-12: CNT_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	CZMZ	Counter Zeroed by Zero Marker Interrupt Enable. The CNT_IMSK.CZMZ bit enables (unmasks) the counter zeroed by zero marker interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
9 (R/W)	CZME	Zero Marker Error Interrupt Enable. The CNT_IMSK.CZME bit enables (unmasks) the zero marker error interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt

Table 27-12: CNT_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	CZM	CZM Pin/Pushbutton Interrupt Enable. The CNT_IMSK.CZM bit enables (unmasks) the CZM pin/pushbutton interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
7 (R/W)	CZERO	CNT_CNTR Counts to Zero Interrupt Enable. The CNT_IMSK.CZERO bit enables (unmasks) the counts to zero interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
6 (R/W)	COV15	Bit 15 Overflow Interrupt Enable. The CNT_IMSK.COV15 bit enables (unmasks) the bit 15 overflow interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
5 (R/W)	COV31	Bit 31 Overflow Interrupt Enable. The CNT_IMSK.COV31 bit enables (unmasks) the bit 31 overflow interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
4 (R/W)	MAXC	Max Count Interrupt Enable. The CNT_IMSK.MAXC bit enables (unmasks) the max count interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
3 (R/W)	MINC	Min Count Interrupt Enable. The CNT_IMSK.MINC bit enables (unmasks) the min count interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
2 (R/W)	DC	Downcount Interrupt Enable. The CNT_IMSK.DC bit enables (unmasks) the down count interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt

Table 27-12: CNT_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	UC	Upcount Interrupt Enable. The `CNT_IMSK.UC` bit enables (unmasks) the up count interrupt request.	
		0	Mask Interrupt
		1	Unmask Interrupt
0 (R/W)	IC	Illegal Gray/Binary Code Interrupt Enable. The `CNT_IMSK.IC` bit enables (unmasks) the illegal Gray/Binary Code interrupt request and should only be used in these modes.	
		0	Mask Interrupt
		1	Unmask Interrupt

Maximum Count Register

The CNT_MAX register holds the 32-bit, two's-complement, higher boundary value. It can be read and written at any time. Hardware ensures that reads and write are atomic, by providing respective shadow registers. This register can be accessed with either 32-bit or 16-bit operations. This allows for using the CNT as a 16-bit counter if sufficient for the application.

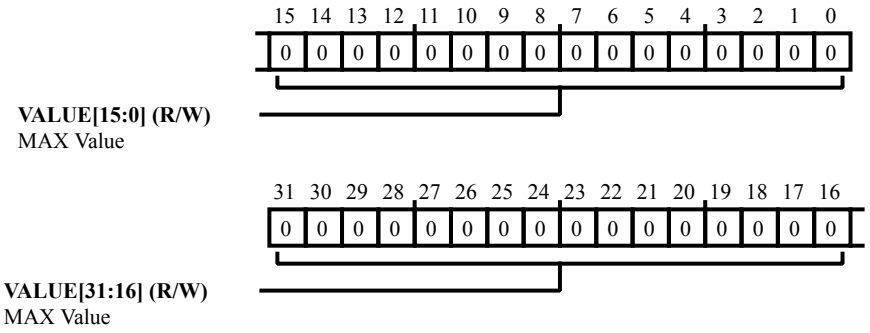


Figure 27-9: CNT_MAX Register Diagram

Table 27-13: CNT_MAX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	MAX Value. The CNT_MAX.VALUE bit field holds the 32-bit, two's-complement, higher boundary value.

Minimum Count Register

The CNT_MIN register holds the 32-bit, two's-complement, lower boundary value. It can be read and written at any time. Hardware ensures that reads and write are atomic, by providing respective shadow registers. This register can be accessed with either 32-bit or 16-bit operations. This allows for using the CNT as a 16-bit counter if sufficient for the application.

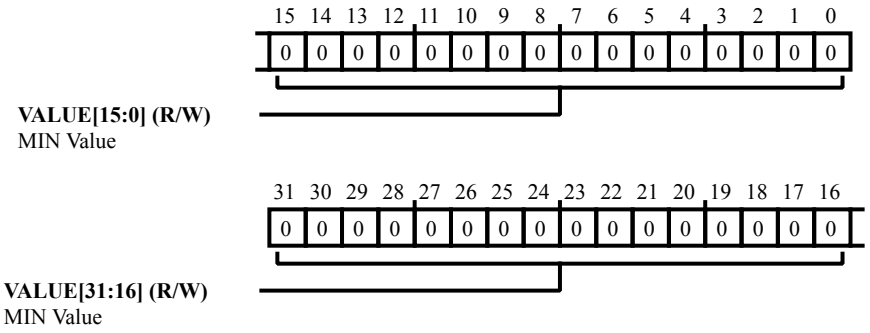


Figure 27-10: CNT_MIN Register Diagram

Table 27-14: CNT_MIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	MIN Value. The CNT_MIN.VALUE bit field holds the 32-bit, two's-complement, lower boundary value.

Status Register

The CNT_STAT register provides status information for each of the CNT events as configured in the CNT_IMSK register. When a CNT event is detected, the corresponding bit in this register is set. It remains set until either software writes a 1 to the bit (write-1-to-clear) or the CNT is disabled.

All bits in the CNT_STAT register indicate either no interrupt request pending (if bit cleared) or an interrupt request pending (if bit set).

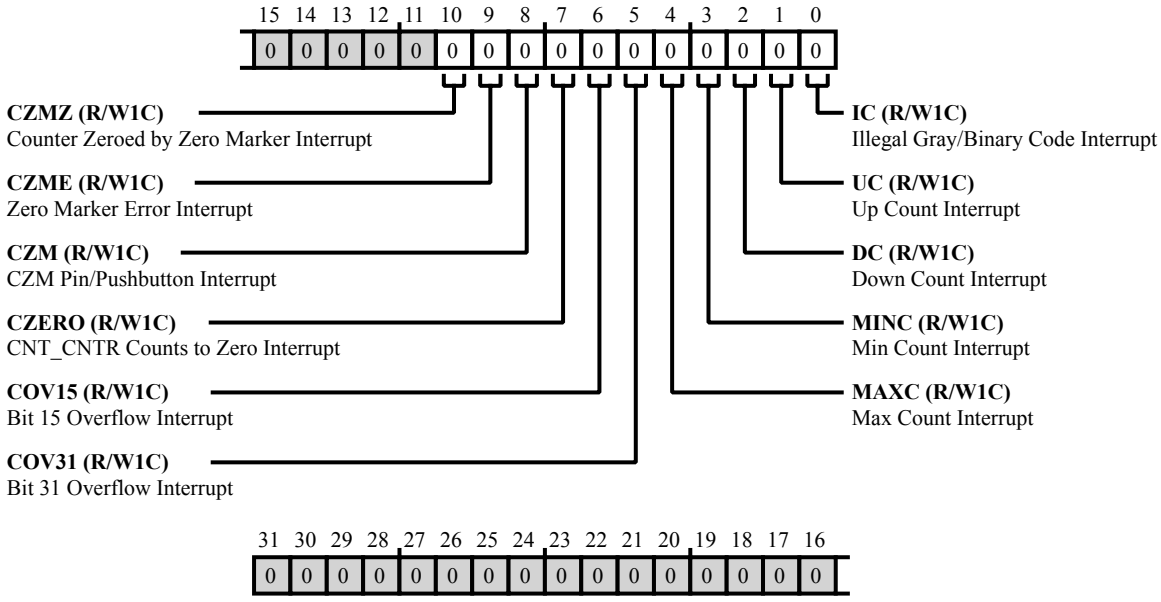


Figure 27-11: CNT_STAT Register Diagram

Table 27-15: CNT_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W1C)	CZMZ	Counter Zeroed by Zero Marker Interrupt. The CNT_STAT.CZMZ bit indicates a zero marker error. If the CNT_CFG.ZMZC bit =1, this interrupt request is generated when the CZMII latch reports a significant edge on the CZM input. Once cleared by software the CNT_STAT.CZM bit is not set again when the CZM input remains active without pulsing.	
		0	No Error
		1	Error Occurred

Table 27-15: CNT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W1C)	CZME	Zero Marker Error Interrupt. The CNT_STAT.CZME bit behaves similarly to the CNT_STAT.CZM bit, with the exception that CNT_STAT.CZME is not set on the CZM edge when the lower four bits of the CNT_CNTR are not zero. In many applications this indicates an error condition, as the zero marker might be out of sync with the counter.	
		0	No Error
		1	Error Occurred
8 (R/W1C)	CZM	CZM Pin/Pushbutton Interrupt. The CNT_STAT.CZM bit indicates a CZM pin/pushbutton error. This interrupt request is generated when a significant edge is seen on the CZM pin, regardless what mode the counter is operating in. This is often used to sense push buttons (especially with the debouncing circuit enabled).	
		0	No Error
		1	Error Occurred
7 (R/W1C)	CZERO	CNT_CNTR Counts to Zero Interrupt. The CNT_STAT.CZERO bit indicates a counts to zero error. This error is generated when the CNT_CNTR register has incremented or decremented toward 0x0000.0000. The latch is not set when software writes to the CNT_CNTR register directly or when the counter is zeroed by writes to the CNT_CMD register.	
		0	No Error
		1	Error Occurred
6 (R/W1C)	COV15	Bit 15 Overflow Interrupt. The CNT_STAT.COV15 bit indicates a bit 15 overflow error. This error is generated when the 16-bit twos-complement CNT_CNTR register has incremented from 0xxxxx.7FFF to 0xxxxx.8000 or decremented from 0xxxxx.8000 to 0xxxxx.7FFF.	
		0	No Error
		1	Error Occurred
5 (R/W1C)	COV31	Bit 31 Overflow Interrupt. The CNT_STAT.COV31 bit indicates a bit 31 overflow error. This error is generated when the 32-bit twos-complement CNT_CNTR register has incremented from 0x7FFF.FFFF to 0x8000.0000 or decremented from 0x8000.0000 to 0x7FFF.FFFF.	
		0	No Error
		1	Error Occurred

Table 27-15: CNT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W1C)	MAXC	Max Count Interrupt. The CNT_STAT.MAXC bit indicates a max count error. This interrupt is used in boundary compare (BND_COMP) mode. If after incrementing the CNT_CNTR register equals CNT_MAX, the CNT_STAT.MAXC bit is set.	
		0	No Error
		1	Error Occurred
3 (R/W1C)	MINC	Min Count Interrupt. The CNT_STAT.MINC bit indicates a minimum count error. This interrupt is used in boundary compare (BND_COMP) mode. If, after decrementing, the CNT_CNTR register equals CNT_MIN, the CNT_STAT.MINC bit is set.	
		0	No Error
		1	Error Occurred
2 (R/W1C)	DC	Down Count Interrupt. The CNT_STAT.DC bit indicates a down count error. This interrupt is generated when the CNT_CNTR register decrements.	
		0	No Error
		1	Error Occurred
1 (R/W1C)	UC	Up Count Interrupt. The CNT_STAT.UC bit indicates an up count interrupt. This interrupt is generated when the CNT_CNTR register increments.	
0 (R/W1C)	IC	Illegal Gray/Binary Code Interrupt. The CNT_STAT.IC bit indicates a illegal Gray/Binary Code interrupt and should only be used in these modes. In normal operation those codes can increment or decrement the CNT_CNTR register by one at a time. If the sensed inputs instruct the counter to increment or decrement by two, the CNT_STAT.IC bit is set. Hardware sets the CNT_STAT.IC bit in QUAD_ENC and BIN_ENC encoder modes only.	
		0	No Error
		1	Error Occurred

28 Media Local Bus (MLB)

Media Local Bus (MediaLB®) is an on-PCB or inter-chip communication bus, which allows an application to access the MOST network data. Media Local Bus supports all the MOST network data transport methods including synchronous stream data, asynchronous packet data, control message data and isochronous data. The MLB topology supports communication among the MLB controller and MLB devices, where the MLB controller is the interface between the MLB devices and the MOST network.

The MLB module serves as an interface between the MediaLB and the processor, implementing the requirements of the physical layer and the link layer outlined in the MediaLB specification. It supports up to 64 logical channels with up to 468 bytes of data per MediaLB frame. Transmit and receive data can be transferred between MediaLB and on-chip memory with DMA block transfers.

The MLB supports the MOST25, MOST50 and MOST150 standards and this document assumes familiarity with these standards. For more information, refer to the Media Local Bus specification version 4.2.

- Copyright 1998-2016 Microchip Technology Inc. All rights reserved. Portions of this chapter are included with permission from Microchip Technology, Inc.

Features

The objective of MLB is to map all the MOST Network data types (transport methods) into a single low-cost, scalable, and standardized hardware interface between a MediaLB controller and at least one other MediaLB Device. The adoption of MediaLB simplifies the hardware interface, reduces the pin count, and facilitates the design of modular reusable hardware. From a software development perspective, the use of MediaLB relieves the system developer from the complexity of the MOST network, which simplifies software development and enables the design of reusable software for different applications. This simplified, standardized interface shortens time to market and makes software maintenance effortless.

- Compliant to Media Local Bus specification version 4.2
- Support MOST25, MOST50, MOST150 standards
- Support 3-pin and 6-pin mode
    - 6-pin mode supports a data rate of 2048 × FS
    - 3-pin mode support various data rates of 256 × FS, 512 × FS, 1024 × FS

- Support 64 logic channels
- Dedicated PLL for clock recovery and phase alignment
- Dedicated pins for 6-pin mode (LVDS)
- Shared pins for 3-pin mode
- Dedicated internal RAM for data buffering and channel table
- Recovered clock (after division) available out on a shared pin for DAI

NOTE: Only one MLB interface (3-pin mode or 6-pin mode) is active at any given time.

MLB Definitions

The following are standard MLB and MOST terms that are used in this chapter.

AGU

Address Generation Units. To access a particular HBI channel the HC must first configure one of two HBI address generation units. AGUs can be configured by writing the HBI command registers, HCMD0 and HCMD1.

CAT

Channel Allocation Table. The Channel Allocation Table (CAT) is comprised of 16 CTR entries. Each 16-bit CAT entry represents a logical connection to or from a transmit/receive device (for example MediaLB or HBI channel).

CDT

Channel Descriptor Table.

FCE

Flow Control Enable bit. The FCE bit is used by MediaLB isochronous Rx channels only.

HBI

Host Bus Interface. The HBI block provides 16-bit parallel completer access to all MOST channels and data types for the external Host Controller (HC). The HBI supports up to 64 independent channels.

HC

Host Controller (external).

HCMDx

HBI Command registers.

HSTSx

HBI status registers.

MFE

Multi-Frame per sub-buffer enable bit. The MFE bit is used by MediaLB synchronous channels only.

PML

Packet Message Length.

Clocking

The MLB controller provides an external clock pin—the media local bus clock. The MLB controller generates the clock. It is synchronized to the MOST network and provides the timing for the entire MLB interface at FS = 48 kHz.

Functional Description

The ***MediaLB Block Diagram*** figure shows the MLB high-level architecture. The MLB core serves as an interface between the MediaLB and the processor, implementing the requirements of the physical layer and the link layer outlined in the MediaLB specification. The MLB core has the following responsibilities.

- Transmit commands and data when functioning as the transmitting device associated with a *Channel Address*
- Receive data and transmit Rx status responses when functioning as the receiving device associated *Channel Address*
- MLB lock detection
- System channel command handling

The MediaLB device can function as either a MediaLB 3-pin interface (single-ended) or MediaLB 6-pin interface (differential) but only one interface can be active at a time. The MediaLB interfaces are capable of exchanging data at speeds up to 1024 × Fs in 3-pin mode or 2048 × Fs in 6-pin mode.

A set of physical channels for exchanging data over the MediaLB bus is supported. These physical channels (4 bytes in length, or a quadlet) can be grouped into logical channels, where each logical channel is referenced using a channel address and represents a uni-directional datapath between a specific MediaLB device transmitting the data and the MediaLB device(s) receiving the data. The MediaLB 6-pin interface provides support for up to 468 bytes of data per frame. The logical channels, configured by system software, can be any combination of channel types (synchronous, asynchronous, isochronous, or control) and direction (transmit or receive).

ADSP-SC59x MLB Register List

The MediaLB Device Interface Macro 2 (MediaLB DIM 2), also referred to as OS62420, implements the required functionality of a Media Local Bus (MediaLB) device. This logic serves as an interface between the inter-chip MediaLB bus and a customer IC, implementing the physical- and link-layer requirements outlined in the MediaLB Specification.

Table 28-1: ADSP-SC59x MLB Register List

Name	Description
MLB_ACMR0	Peripheral Channel Mask 0 Register
MLB_ACMR1	Peripheral Channel Mask 1 Register
MLB_ACSR0	Peripheral Channel Status 0 Register
MLB_ACSR1	Peripheral Channel Status 1 Register
MLB_ACTL	Bus Control Register
MLB_CTL0	MediaLB Control 0 Register
MLB_CTL1	Control 1 Register
MLB_GCTL	MLB Global Control Register
MLB_HCBR0	HBI Channel Busy 0 Register
MLB_HCBR1	HBI Channel Busy 1 Register
MLB_HCER0	HBI Channel Error 0 Register
MLB_HCER1	HBI Channel Error 1 Register
MLB_HCMR0	HBI Channel Mask 0 Register
MLB_HCMR1	HBI Channel Mask 1 Register
MLB_HCTL	HBI Control Register
MLB_MADR	Memory Interface Address Register
MLB_MCTL	Memory Interface Control Register
MLB_MDAT0	Memory Interface Control Data 0 Register
MLB_MDAT1	Memory Interface Control Data 1 Register
MLB_MDAT2	Memory Interface Control Data 2 Register
MLB_MDAT3	Memory Interface Control Data 3 Register
MLB_MDWE0	Memory Interface Control Data Write Enable 0 Register
MLB_MDWE1	Memory Interface Control Data Write Enable 1 Register
MLB_MDWE2	Memory Interface Control Data Write Enable 2 Register
MLB_MDWE3	Memory Interface Control Data Write Enable 3 Register
MLB_MIEN	Interrupt Enable Register

Table 28-1: ADSP-SC59x MLB Register List (Continued)

Name	Description
MLB_MS0	Channel Status 0 Register
MLB_MS1	Channel Status 1 Register
MLB_MSD	System Data Register
MLB_MSS	System Status Register
MLB_PCTL0	MediaLB 6-pin Control 0 Register

ADSP-SC59x MLB Interrupt List

Table 28-2: ADSP-SC59x MLB Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
187	MLB0_INT0	MLB0 Interrupt 0		
188	MLB0_INT1	MLB0 Interrupt 1		
189	MLB0_STAT	MLB0 Status		

MediaLB Protocol

The MediaLB topology supports communication among all MediaLB devices, including the MediaLB controller. The bus interface consists of a uni-directional line for clock (MLBC), a bidirectional line for signal information (MLBS), and a bidirectional line for data transfer (MLBD). The MediaLB topology supports one controller connected to one or more devices, where the controller is the interface between the MediaLB devices and the MOST network.

The MediaLB controller includes MediaLB device functionality, and also generates the MediaLB clock (MLBC) that is synchronized to the MOST Network. This generated clock provides the timing for the entire MediaLB interface. The MLBS line is a multiplexed signal which carries channel addresses generated by the MediaLB controller, as well as command and RxStatus bytes from MediaLB devices. The MLBD line is driven by the transmitting MediaLB device and is received by all other MediaLB devices, including the MediaLB controller. The MLBD line carries the actual data (synchronous, asynchronous, control, or isochronous).

Once per MOST network frame, the MLB controller generates a unique frame sync pattern on the MLB_SIG line. The end of the frame sync pattern defines the byte boundary and the channel boundary for the MLB_SIG and MLB_DAT lines of all MLB devices.

The MLB controller manages the arbitration for all the channels on the MLB and grants bandwidth for all the MLB devices. An MLB *physical channel* is defined as four bytes wide, or a quadlet. Physical channels can be grouped into multiple quadlets (which do not have to be consecutive) to form an MLB *logical channel*, which is defined by a unique channel address.

As shown in ***MLB Data Structure***, the MLB controller initiates communication by sending out a channel address on the MLB_SIG line for each physical channel. The channel address indicates which MLB device is transmitting and which MLB devices are receiving in the following physical channel. Therefore, four bytes after the controller outputs the channel address on the MLB_SIG line, the transmitting device outputs a command byte command on the MLB_SIG line and outputs the respective data on the MLB_DAT line, concurrently. The MLB command byte contains the type of data currently being transmitted (for example synchronous, asynchronous or control).

The MLB device receiving the channel data outputs a status byte, RxStatus, on the MLB_SIG line immediately after the transmitting device outputs the command byte. The status response can indicate that the receiving device is busy and cannot receive the data at present, or the device is ready to receive the data. Since synchronous stream data is sent in a broadcast fashion, receiving devices cannot return a busy status and should not drive RxStatus onto the MLB_SIG line.

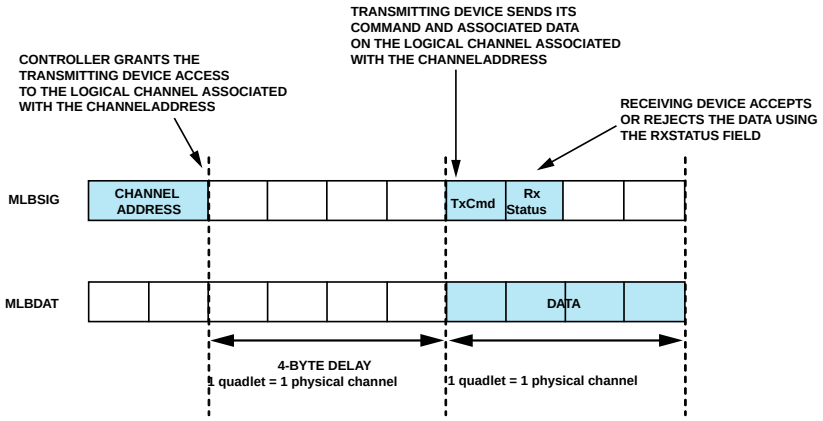


Figure 28-1: MLB Data Structure

MLB Architectural Concepts

The following sections provide information about the MLB architecture.

MediaLB Block Diagram

The ***MediaLB Block Diagram*** shows the various blocks within the interface its connections to the processor.

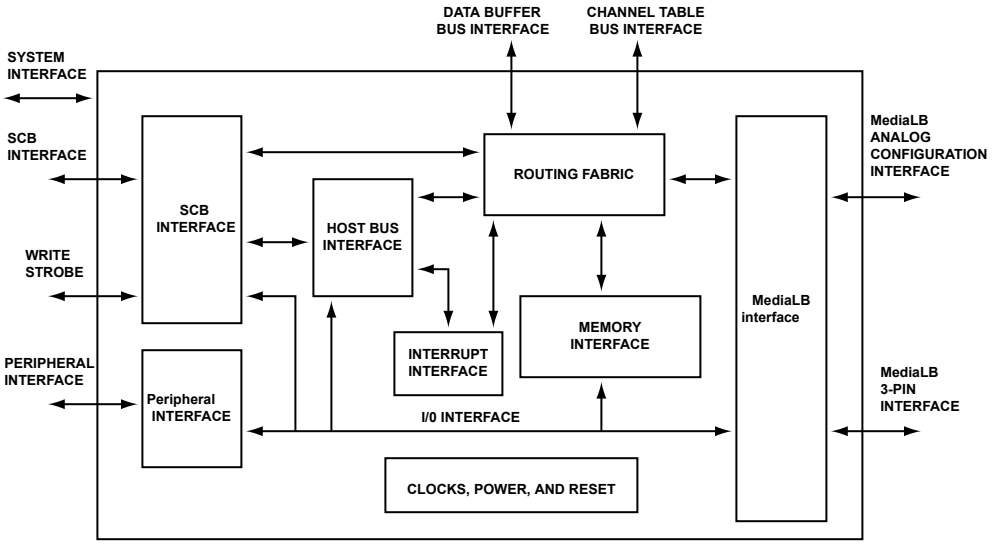


Figure 28-2: MediaLB Block Diagram

MediaLB Interface

The Media Local Bus (MediaLB) block supports both a MediaLB 3-pin interface and MediaLB 6-pin interface; however, only one MediaLB interface can be active at any given time. Both MediaLB interfaces provide real-time access to all network data types - synchronous, asynchronous, control, and isochronous data.

- MediaLB 3-pin interface–supports the MediaLB protocol for single-ended 3-pin mode, with a maximum data rate of 1024 FS (49.152 MHz at FS = 48 kHz).
- MediaLB 6-pin interface–supports the MediaLB protocol for high-speed differential 6-pin mode, with a maximum data rate of 2048 FS (98.304 MHz at FS = 48 kHz).

MediaLB Channel Address to Logical Channel Mapping

The MediaLB channel addresses are mapped to the logical channels as shown in the ***MediaLB Channel Address to Logical Channel Mapping*** table.

Table 28-3: MediaLB Channel Address to Logical Channel Mapping

Channel Address	Logical Address
0x0002	1
0x0004	2
0x0006	3
...	...
0x007C	62
0x007E	63
0x01FE	0 (Logical channel 0 is the system channel and is reserved)

Routing Fabric

The Routing Fabric (RF) block manages the flow of data between the MediaLB port and the HBI port. It manages accessing the Channel Table RAM (CTR) and Data Buffer RAM (DBR), which are explained in the following subsections. The routing fabric uses channel descriptors (stored in the CTR) to manage access to dynamic buffers in the DBR.

Data Buffer RAM

The Data Buffer RAM (DBR) is an 8-bit x 16k entriy buffer, single-port synchronous SRAM, and provides dynamic circular buffering between the transmit and receive devices. The size and location of each data buffer is defined by software in the Channel Descriptor Table (CDT), which is in the Channel Table RAM (CTR), described in following sections.

Channel Table RAM

The Channel Table RAM (CTR) is a 128-bit x 144-entry table that allows system software to dynamically configure channel routing and allocate data buffers in the DBR. The CTR is logically divided into three tables:

- Channel Descriptor Tables
- AHB Descriptor Table (ADT)
- Channel Allocation Table

Address Mapping

The ***CTR Address Mapping*** table shows the address mapping for the CTR.

Table 28-4: CTR Address Mapping

Label	Address	Bits [127:96]	Bits [95:64]	Bits [63:32]	Bits [31:0]
Channel Descriptor Table (CDT)					
CDT	0x00	CDT0[127:0], CL = 0			
	0x01	CDT1[127:0], CL = 1			
	0x02	CDT2[127:0], CL = 2			
	...				
	0x3D	CDT61[127:0], CL = 61			
	0x3E	CDT62[127:0], CL = 62			
	0x3F	CDT63[127:0], CL = 63			
AHB Descriptor Table (ADT)					
ADT	0x40	ADT0[127:0], CL = 0			
	0x41	ADT1[127:0], CL = 1			
	0x42	ADT2[127:0], CL = 2			
	...				

Table 28-4: CTR Address Mapping (Continued)

Label	Address	Bits [127:96]		Bits [95:64]		Bits [63:32]		Bits [31:0]	
	0x7D	ADT61[127:0], CL = 61							
	0x7E	ADT62[127:0], CL = 62							
	0x7F	ADT63[127:0], CL = 63							
Channel Allocation Table (CAT)									
CAT for MediaLB	0x80	CAT7	CAT6	CAT5	CAT4	CAT3	CAT2	CAT1	CAT0
	...	...	...	...	...	...	...	...	...
	0x87	CAT63	CAT62	CAT61	CAT60	CAT59	CAT58	CAT57	CAT56
CAT for HBI*1	0x88	CAT71	CAT70	CAT69	CAT68	CAT67	CAT66	CAT65	CAT64
	...	...	...	...	...	...	...	...	...
	0x8F	CAT127	CAT126	CAT125	CAT124	CAT123	CAT122	CAT121	CAT120

*1 A fixed relationship exists between ADT entries and HBI CAT entries. When using HBI channel 0 (CAT64), program ADT0. When using HBI channel 1 (CAT65), program ADT1, and so on.

Channel Allocation Table

The ***Channel Allocation Table (CAT)*** table is comprised of 16 CTR entries (addresses 0x80 0x8F) as shown in the ***CTR Entry Map*** table. Each 16-bit CAT entry represents a logical connection to or from a transmit or receive device. (for example, MediaLB channel). All entries are indexed according to a fixed physical address assigned to every RX/TX channel as shown in the ***CAT Entry Formats*** table. The value stored in a CAT entry includes a 6-bit connection label, which provides a pointer to the CDT. To complete a logical channel and form a routing connection, system software must assign the same connection label to both the RX and TX channels.

Table 28-5: CAT Entry Map

Peripheral	TX Channels	RX Channels	CAT Start Index	CAT End Index	Entries
MediaLB	0 to 64	64 TX Channels	0	63	64
HBI	0 to 64	64 TX Channels	64	127	64

The format of a full CAT entry is shown in the ***CAT Entry Formats*** table, with field descriptions described in the ***CAT Field Definitions*** table. All reserved bits of a CAT entry field should be written as zero.

Table 28-6: CAT Entry Formats

Channel Type	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Isochronous	rsvd	FCE	rsvd	RNW	CE	CT[2:0] = 011			rsvd		CL[5:0]					
Asynchronous	rsvd		MT	RNW	CE	CT[2:0] = 010			rsvd		CL[5:0]					
Control	rsvd		MT	RNW	CE	CT[2:0] = 001			rsvd		CL[5:0]					

Table 28-6: CAT Entry Formats (Continued)

Channel Type	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Synchronous	rsvd	MFE	MT	RNW	CE	CT[2:0] = 000			rsvd		CL[5:0]					

Table 28-7: CAT Field Definitions

Field	Description
CL[5:0]	Connection Label (offset into CDT)
CT[2:0]	Channel Type (others) 111 = Reserved 110 = Reserved 101 = Reserved 100 = Reserved 011 = Isochronous 010 = Asynchronous 001 = Control 000 = Synchronous
CE	Channel Enable. 0 = Disabled, 1 = Enabled
RNW	Read Not Write. 0 = Write, 1 = Read
MT	Mute Enable. 0 = Disabled When set for synchronous channels, the MT bit forces RX channels to write zeros into the channel data buffer, and TX channels to output zeros on the physical interface. When set for asynchronous and control channels, the MT bit causes DMA to halt at a packet boundary. Not valid for isochronous channels.
FCE	Flow Control Enable. 0 = Disabled, 1 = Enabled The FCE bit is used by MediaLB isochronous RX channels only.
MFE	Multi-Frame per sub-buffer enable. 0 = Disabled, 1 = Enabled The MFE bit is used by MediaLB synchronous channels only.
rsvd	Reserved. Software writes a 0 to all reserved bits when the entry is initialized. These bits are read-only after initialization.

Channel Set Up

Data direction is in reference to the DBR. The data direction of CAT entries corresponding to the same channel is reversed for the HBI CAT and the MediaLB CAT.

- For a Tx channel (from the HC to the MediaLB interface):
  - HBI CAT entry: RNW = 0 (write)
  - MediaLB CAT entry: RNW = 1 (read)
- Conversely, for an Rx channel (data from MediaLB to HC):

- HBI CAT entry: RNW = 1 (read)
- MediaLB CAT entry: RNW = 0 (write)

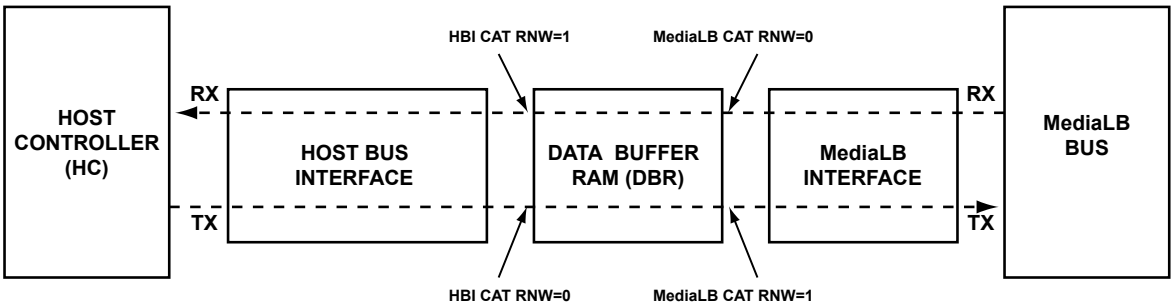


Figure 28-3: DBR Directional Relationship

Channel Descriptor Tables

The Channel Descriptor Table (CDT) is comprised of 64 CTR entries (addresses 0x00 - 0x3F), as shown in the Table 28-4 CTR Address Mapping table. Each 128-bit CDT entry (also referred to as a channel descriptor) is referenced by a connection label and contains information about a data buffer in the DBR (for example buffer size, address pointers). The format of each CDT entry is dependent on the channel type (synchronous, isochronous, asynchronous, or control).

NOTE: All reserved channel descriptor bits must be written to 0 by software when initialized.

Synchronous Channel Descriptors

The format and field definitions for a synchronous CDT entry are shown in the ***Synchronous CDT Entry Format*** and ***Synchronous CDT Entry Field Definitions*** tables.

Table 28-8: Synchronous CDT Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WSBC[1:0]		Reserved													
16	RSBC[1:0]		Reserved													
32	Reserved															
48	Reserved															
64	WSTS[3:0]				WPTR[11:0]											
80	RSTS[3:0]				RPTR[11:0]											
96	Reserved				BD[11:0]											
112	Reserved		BA[13:0]													

Table 28-9: Synchronous CDT Entry Field Definitions

Field	Description	Details	Access
	Buffer Base Address	Can start at any byte in the 16k DBR	RW

Table 28-9: Synchronous CDT Entry Field Definitions (Continued)

Field	Description	Details	Access
BA			
BD	Buffer Depth	BD = size of buffer in bytes 1. Buffer end address = BA + BD. BD = 4 m bpf 1 where: m = frames per sub-buffer (for MFE = 0, m = 1) bpf = bytes per frame.	RW
RPTR	Read Pointer	Software initializes to 0, hardware updates. Counts the read address offset within a buffer. DMA read address = BA + RPTR.	RWU
WPTR	Write Pointer	Software initializes to 0, hardware updates. Counts the write address offset within a buffer. DMA read address = BA + WPTR.	RWU
RSBC	Read Sub-buffer Counter	Software initializes to 0, hardware updates. Counts the read sub-buffer offset. DMA uses for pointer management.	RWU
WSBC	Write Sub-buffer Counter	Software initializes to 0, hardware updates. Counts the write sub-buffer offset. DMA uses for pointer management.	RWU
RSTS	Read Status	Software initializes to 0, hardware updates. RSTS States: xxx0 = normal operation (no mute) xxx1 = normal operation (mute) xx0x = idle	RWU
WSTS	Write Status	Software initializes to 0, hardware updates. WSTS States: xxx0 = normal operation (no mute) xxx1 = normal operation (mute) xx0x = idle 1xxx = command protocol error	RWU
Reserved. Software writes a 0 to all these bits when the entry is initialized. Reserved bits are RO after initialization.			RWU

Isochronous Channel Descriptors

The format and field definitions for a synchronous CDT entry are shown in the ***Isochronous Entry Formats*** and ***Isochronous CDT Entry Field Definitions*** tables.

Table 28-10: Isochronous Entry Formats

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Reserved															
16	Reserved															
32	Reserved								BS[8:0]							
48	Reserved															
64	WSTS[3:0]				WPTR[12:0]											
80	RSTS[3:0]				RPTR[12:0]											
96	Reserved				BD[12:0]											
112	BF	Rsvd		BA[13:0]												

Table 28-11: Isochronous CDT Entry Field Definitions

Field	Description	Details	Access
BA	Buffer Base Address	Can start at any byte in the 16k DBR.	RW
BD	Buffer Depth	BD = size of buffer in bytes 1. Buffer end address = BA + BD. Isochronous buffers must be large enough to hold at least 3 blocks (packets) of data. BD Must be an integer multiple of blocks.	RW
BF	Buffer Full	Software initializes to 0, hardware updates. DMA write hardware sets BF when the buffer is full. DMA read hardware clears BF when the buffer is empty. BF is valid only when buffer is full or empty, otherwise ignore.	RWU
BS	Block Size	BS defines when to begin the DMA to the data buffer. BS = buffer block size in bytes 1. For RX channels, the DMA writes start when the number of empty bytes in the data buffer the block size. For TX channels, the DMA reads start when the number of valid bytes in the data buffer the block size.	RWU
RPTR	Read Pointer	Software initializes to 0, hardware updates. Counts the read address offset within a buffer. DMA read address = BA + RPTR.	RWU
WPTR	Write Pointer	Software initializes to 0, hardware updates. Counts the write address offset within a buffer. DMA write address = BA + WPTR.	RWU

Table 28-11: Isochronous CDT Entry Field Definitions (Continued)

Field	Description	Details	Access
RSTS	Read Status	Software initializes to 0, hardware updates. RSTS States: xx1 = active xx0 = idle	RWU
WSTS	Write Status	Software initializes to 0, hardware updates. WSTS States: xxx0 =active xxx1 = idle xx0x = command protocol error 1xxx = buffer overflow (FCE = 0 only)	RWU
Reserved. Software writes a 0 to all these bits when the entry is initialized. Reserved bits are RO after initialization.			RWU

Asynchronous/Control Channel Descriptors

The format and field definitions for an Asynchronous/Control CDT entry are shown in the ***Asynchronous/Control CDT Entry Format*** and ***Asynchronous/Control CDT Entry Field Definitions*** tables.

Table 28-12: Asynchronous/Control CDT Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WPC[4:0]					Reserved										
16	RPC[4:0]					Reserved										
32	Rsvd	WPC[7:5]				Reserved										
48	Rsvd	RPC[7:5]				Reserved										
64	WSTS[3:0]					WPTR[11:0]										
80	RSTS[3:0]					RPTR[11:0]										
96	RSTS[4]	WSTS[4]	Rsvd			BD[11:0]										
112	Rsvd		BA[13:0]													

Table 28-13: Asynchronous/Control CDT Entry Field Definitions

Field	Description	Details	Access
BA	Buffer Base Address	Can start at any byte in the 16k DBR.	RW
BD	Buffer Depth	BD = size of buffer in bytes 1. Buffer end address = BA + BD. BD = max packet length 1.	RW

Table 28-13: Asynchronous/Control CDT Entry Field Definitions (Continued)

Field	Description	Details	Access
RPC	Read Packet Count	Software initializes to 0, hardware updates. Used with RPC, RPTR and WPTR to determine if the buffer is empty or full.	RWU
WPC	Write Packet Count	Software initializes to 0, hardware updates. Used with RPC, RPTR and WPTR to determine if the buffer is empty or full.	RWU
RPTR	Read Pointer	Software initializes to 0, hardware updates. Counts the read address offset within a buffer. DMA read address = BA + RPTR.	RWU
WPTR	Write Pointer	Software initializes to 0, hardware updates. Counts the write address offset within a buffer. DMA read address = BA + WPTR.	RWU
RSTS	Read Status	Software initializes to 0, hardware updates. RSTS States: x0x00 = idle xx1xx = *ReceiverProtocolError* response received from RX device 1xxxx = *ReceiverBreak* command received from RX device	RWU
WSTS	Write Status	Software initializes to 0, hardware updates. Status States (only valid for DMA pointers associated with the MLB block, not HBI block pointers): xxx0 = idle xxx1 = command protocol error detected xx0x = *AsyncBreak*/*ControlBreak* command received from TX device	RWU
Reserved. Software writes a 0 to all these bits when the entry is initialized. Reserved bits are RO after initialization.			RWU

AHB Descriptor Table (ADT)

The AHB block manages data exchange between local channel data buffers within MLB module and the system memory buffer. To support system memory buffering, a ping-pong memory structure is implemented on a per channel basis using 128-bit descriptors for AHB Descriptor Table (ADT) entries. The Table 28-4 CTR Address Mapping table provides a complete address map of the CTR, including the location of the ADT.

Each logical channel is assigned a separate 128-bit descriptor, defining the data buffers in the system memory used by the DMA interface for that channel. The descriptors are stored at fixed addresses in the CTR as described in previous section. The ***ADT Field Definitions*** table provides an overview of field definitions for ADT entries.

Table 28-14: ADT Field Definitions

Field	No. of Bits	Description	Access
CE	1	Channel Enable. 0 = Disabled	RW, U
LE	1	Endianness Select. 0 = Big Endian, 1 = Little Endian	RW
PG	1	Page pointer. Software initializes to 0, hardware writes thereafter. 0 = Ping buffer, 1 = Pong buffer	RW, U
RDY1	1	Buffer Ready bit for ping buffer page. 0 = Not ready, 1 = Ready	RW
RDY2	1	Buffer Ready bit for pong buffer page. 0 = Not ready, 1 = Ready	RW
DNE1	1	Buffer Done bit for ping buffer page. 0 = Not done, 1 = Done	R, U, c0
DNE2	1	Buffer Done bit for pong buffer page. 0 = Not done, 1 = Done	R, U, c0
ERR1	1	Error Response detected for ping buffer page. 0 = No error, 1 = Error	R, U, c0
ERR2	1	Error Response detected for pong buffer page. 0 = No error, 1 = Error	R, U, c0
PS1	1	Packet Start bit for ping buffer page. 0 = No packet start, 1 = Packet start Reserved for synchronous and isochronous channels.	RW, U both TX and RX
PS2	1	Packet Start bit for pong buffer page. 0 = No packet start, 1 = Packet start Reserved for synchronous and isochronous channels.	RW, U both TX and RX
MEP1	1	Most Ethernet Packet indicator for ping buffer page. 0 = Not MEP, 1 = MEP. MEP1 only valid for the first page of a segmented buffer. Reserved for control synchronous and isochronous channels.	Rsvd for TX, R, U, c0 for RX
MEP2	1	Most Ethernet Packet indicator for pong buffer page. 0 = Not MEP, 1 = MEP. MEP2 only valid for the first page of a segmented buffer. Reserved for control synchronous and isochronous channels.	Rsvd for TX, R, U, c0 for RX
BD1	11 to 13	Buffer Depth for ping buffer page. 11 or 12 bits for asynchronous and control channels. 13 bits for synchronous and isochronous channels.	RW
BD2	11 to 13	Buffer Depth for pong buffer page. 11 or 12 bits for asynchronous and control channels. 13 bits for synchronous and isochronous channels.	RW
BA1	32	Buffer Base Address for ping buffer page	RW
BA2	32	Buffer Base Address for pong buffer page.	RW
Reserved	varies	Reserved. Software writes a 0 to all these bits when the entry is initialized. Reserved bits are RO after initialization.	RW, U

The ***Ping-Pong System Memory Structure*** figure shows that this system memory structure is similar for all channel types and shows the relationship between the BAn, BDn, and PG descriptor fields.

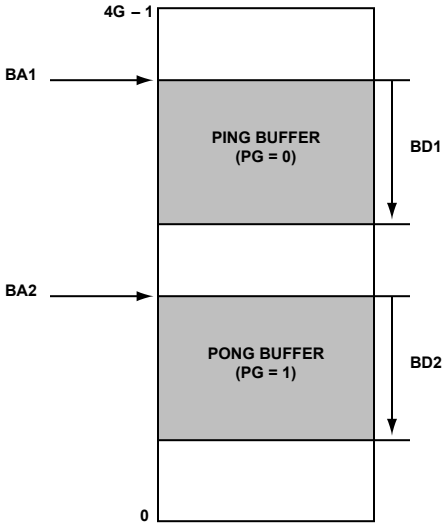


Figure 28-4: Ping-Pong System Memory Structure

Each ADT entry (also referred to as a *Channel Descriptor)* holds a 32-bit BAn field which defines the start of each ping or pong buffer within system memory. The BDn field is used to indicate the size for the respective ping or pong page. The maximum size is 2k entries for asynchronous and control channels and 8k entries for isochronous and synchronous channels.

Synchronous Channel Descriptors

The synchronous buffering scheme allows each ping or pong buffer to contain a single frame or a multiple number of frames. For this reason, the synchronous buffer depth (BDn) must be defined in terms of an integer number (n), frames per sub-buffer (m) and bytes per frame (bpf) of data (for example BDn = n m bpf 1). The ***Synchronous ADT Entry Format*** table shows the format for a synchronous ADT entry. The field definitions are defined in the ***ADT Field Definitions*** table. Each synchronous channel buffer can be up to 8k-bytes deep.

Table 28-15: Synchronous ADT Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CE	LE	PG	Reserved												
16	Reserved															
32	RDY1	DNE1	ERR1	BD1[12:0]												
48	RDY2	DNE2	ERR2	BD2[12:0]												
64	BA1[15:0]															
80	BA1[31:16]															
96	BA2[15:0]															
112	BA2[31:16]															

Isochronous Channel Descriptors

The isochronous buffering scheme allows each ping or pong buffer to contain a single block or a multiple number of blocks. For this reason, the isochronous buffer depth (BDn) must be defined in terms of an integer number (n) and block size (BS) (for example BDn = n (BS + 1) 1).

The ***Isochronous ADT Entry Format*** table shows the format for an isochronous ADT entry. The field definitions are defined in the ***ADT Field Definitions*** table. Each isochronous channel buffer can be up to 8k-bytes deep.

Table 28-16: Isochronous ADT Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CE	LE	PG	Reserved												
16	Reserved															
32	RDY1	DNE1	ERR1	BD1[12:0]												
48	RDY2	DNE2	ERR2	BD2[12:0]												
64	BA1[15:0]															
80	BA1[31:16]															
96	BA2[15:0]															
112	BA2[31:16]															

Asynchronous and Control Channel Descriptors

Every asynchronous and control packet adheres to the Port Message Protocol (PMP), which designates the first two bytes of each packet as the packet length (PML). Each packet must be no more than 2048-bytes (PML 2048).

Software must set the buffer ready bit (RDYn) for each buffer as it programs the DMA. As hardware processes each buffer, it sets the done bit (DNEn) and generates an interrupt to inform HC. When hardware finishes processing a buffer, it can begin processing another buffer if RDYn is set. The application is responsible for setting up and configuring the channel buffer descriptor prior to every DMA access on the channel.

Two packet buffering modes are supported by hardware for programming the DMA, single-packet mode (`MLB_ACTL.MPB` =0) and multiple-packet mode (`MLB_ACTL.MPB` =1). The MPB is written prior to enabling the channel DMA.

Single Packet Mode. The single-packet mode asynchronous and control buffering scheme supports a maximum of one packet per buffer (for example, ping or pong). Both non-segmented and segmented data packets are allowed while using single-packet mode. Non-segmented packets are exchanged when only one buffer (for example, ping or pong) is needed for packet transfer. Segmented packets are exchanged when a single packet is too long for one buffer and the packet must span multiple buffers.

The ***Single-Packet Asynchronous and Control Entry Format*** table shows the format for single-packet mode asynchronous and control ADT entries. The field definitions are defined in the ***ADT Field Definitions*** table.

Table 28-17: Single-Packet Asynchronous and Control Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CE	LE	PG	Reserved												
16	Reserved															
32	RDY1	DNE1	ERR1	PS1	MEP1	BD1[10:0]										
48	RDY2	DNE2	ERR2	PS2	MEP2	BD2[10:0]										
64	BA1[15:0]															
80	BA1[31:16]															
96	BA2[15:0]															
112	BA2[31:16]															

Multiple Packet Mode. The multiple-packet mode asynchronous and control buffering scheme supports more than one packet per system memory buffer, as shown in the ***Asynchronous/Control CDT Entry Format*** table. Multiple-packet mode reduces the interrupt rate for packet channels at the cost of increasing buffering and latency.

For TX packet channels in multiple-packet mode, software sets the packet start bit (PSn) for every buffer. Setting PSn informs hardware that the first two bytes of the buffer contains the port message length (PML) of the first packet. After the first packet, hardware keeps track of where packets start and end within the current buffer. Software should not write to PSn while the buffer is active (RDYn = 1 and DNEn = 0). For TX packet channels, the buffer is done (DNEn = 1) when the last byte of the last packet in the buffer is read from system memory. Software should set the buffer depth to contain the exact number of complete packets for that buffer. Segmented buffers are not supported for TX packet channels in multiple-packet mode.

NOTE: The PS1 and PS2 bits are only valid for TX channels. Set PS1 and PS2 = 1 at the start of the buffer.

Table 28-18: Multiple-Packet Asynchronous and Control Entry Format

Bit Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CE	LE	PG	Reserved												
16	Reserved															
32	RDY1	DNE1	ERR1	PS1	BD1[11:0]											
48	RDY2	DNE2	ERR2	PS2	BD2[11:0]											
64	BA1[15:0]															
80	BA1[31:16]															
96	BA2[15:0]															
112	BA2[31:16]															

Interrupt Interface Block

The interrupt interface raises an interrupt when specific changes to HBI channel descriptors occur, including:

- For asynchronous and control read/write channels:
  - a packet is available to read in the channel buffer, or
  - sufficient empty space is available in the channel buffer to accept a requested packet write
- For isochronous read/write channels:
  - the number of valid bytes in the channel buffer exceeds the block size, or
  - the number of empty bytes in the channel buffer exceeds the block size

Operating Modes

The following sections describe the operating modes of the MLB interface. The channel type selection enables the logical channels to operate in synchronous, asynchronous, isochronous, or control channels.

NOTE: The logical channels can be any combination of channel type (for example synchronous, asynchronous, or control) and direction (transmit or receive).

Isochronous Data Exchange

An isochronous HBI channel is initially opened and synchronized with HCMD0.CMD[2:0]= 010. For isochronous channels, no further synchronization is required from the HBI perspective; however, an optional resynchronization command is available for HC flexibility. Setting HCMD0.CMD[2:0]= 011 reinitializes the address pointer within the data buffer, ensuring that subsequent data exchange with the channel is aligned at an isochronous packet boundary. When the HC must close an isochronous channel before it has read or written an entire data packet, setting HCMD0.CMD[2:0]= 000 reopens the channel without synchronizing the address pointer in the buffer. This action allows reading and writing to continue where the HC previously stopped.

For isochronous data transmission, the ***Exchanging Isochronous Data on an HBI Channel*** figure shows the flow control.

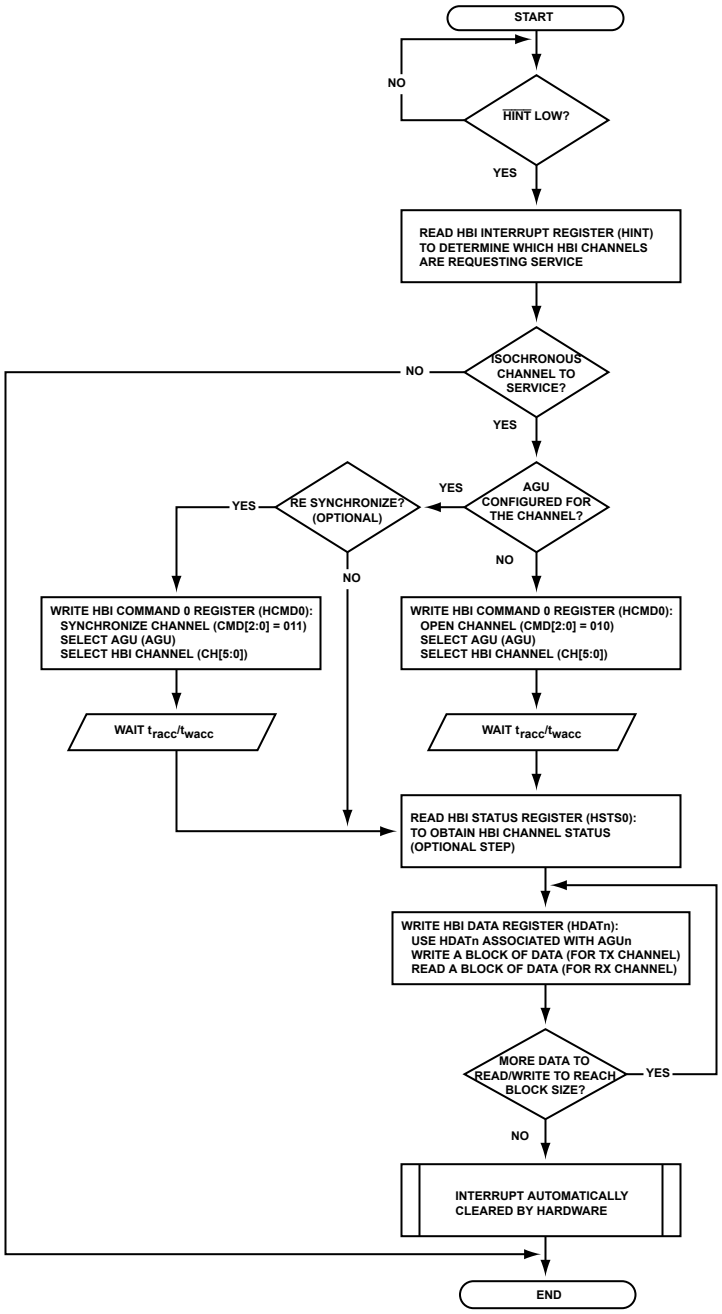


Figure 28-5: Exchanging Isochronous Data on an HBI Channel

Asynchronous and Control Data Exchange

An asynchronous or control HBI channel is initially opened using HCMD0.CMD[2:0]= 010. Occasionally, the HC may need to close a packet channel before it has completed writing or reading the current packet of data (for example, the HC needs to use this AGU to service another HBI channel). In this case, the HC can reopen the previous channel with HCMD0.CMD[2:0]= 000. This situation allows the HC to continue writing or reading a packet from

the point it left off. In this situation, the PML is already known, and the packet length is not reread or rewritten by the HC.

During the reading of a packet, if the HC sets HCMD0.CMD[2:0]= 010 for the channel before the last byte of the packet is read, an internal hardware pointer is reset to the beginning of the packet buffer. This situation requires that the HC reread the PML (from HSTS1) and reread the packet data (from HDATn) from the beginning. In the same manner, if the HC resets HCMD0.CMD[2:0]= 010 for the channel before the last byte of a packet is written, an internal hardware pointer is reset to the beginning of the packet buffer. This situation requires that the HC rewrite the PML (to HCMD1) and rewrite the packet data (to HDATn) from the beginning. Any previous packet data in the buffer is overwritten.

Frame synchronization is not supported for asynchronous channels.

Synchronous Data Exchange

The MLB core provides two modes of operation; standard and multi-frame per sub-buffer which provide flexibility for implementing synchronous channels. Channels configured for standard mode require less buffer space but have higher interrupt rates and more stringent latency requirements. Channels setup for each multi-frame per sub-buffer mode requires more buffer space, but has lower interrupt rates and less stringent latency requirements.

To set up a channel in multi-frame per sub-buffer mode:

1. Program the `MLB_CTL0.FCNT` bit field to select the number of frames per sub-buffer.
2. Program the CAT to enable multi-frame sub-buffering (MFE= 1) for each channel.
3. Set the buffer depth in the CDT: BD= 4 × m × bpf 1 where: m = frames per sub-buffer, bpf = bytes per frame.
4. Repeat for additional synchronous channels

A sample synchronous data buffer is shown in the ***Synchronous Data Buffer Structure*** figure. Each data buffer contains four sub-buffers and each sub-buffer contains space for 1 to 64 frames of data, determined by the `MLB_CTL0.FCNT` bits.

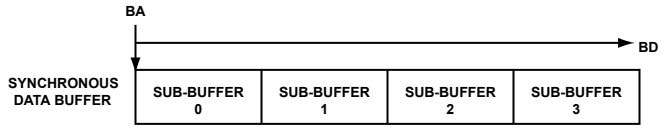


Figure 28-6: Synchronous Data Buffer Structure

Data Transfer

Two modes of operation are supported for transferring channel data between the MLB and internal memory. DMA allows the multi-channel DMA engine to manage data transfers without core intervention. Core driven mode (I/O mode) allows software to manage the transfer of data between MLB and internal memory.

NOTE: All hardware channels must use the same data transfer method. Mixed mode operation where hardware channels operate in both I/O mode and DMA mode is not supported.

DMA

The processor supports DMA mode which uses only INCR8, INCR4 and SINGLE beat bursts. Program the `MLB_ACTL.DMAMODE` = 1 to use DMA mode.

Programming Model

This section provides general guidelines for programming the MediaLB interface.

Channel Initialization

The software flow required to initialize a channel must be performed to ensure proper operation.

Configure the Hardware

1. Initialize the CTR and registers.
    a. Set all the CTR (CAT, CDT, and ADT) bits to 0.
    b. Set all bits of all registers to 0.
2. Configure the MediaLB interface.
    a. Select 3-pin or 6-pin MediaLB operation: `MLB_CTL0.PEN` =0 (3-pin), `MLB_CTL0.PEN` =1 (6-pin).
    b. Select MediaLB clock speed via `MLB_CTL0.CLK`.
    c. Set MediaLB enable via `MLB_CTL0.EN`.
3. Configure the HBI interface.
    a. Set `MLB_HCMR0` and `MLB_HCMR1` = 0xFFFFFFFF to activate all channels.
    b. Set the HBI enable bit: `MLB_HCTL.EN` =1.

Program the CAT and the CDT

1. Initialize all bits of the CAT to 0.
2. Select a logical channel: N = 1 – 63.
3. Program the CDT for channel N.
    a. Set the 14-bit base address (BA)
    b. Set the 12-bit or 13-bit buffer depth (BD): BD = buffer depth in bytes 1

- For synchronous channels: (BD + 1) = 4 frames per sub-buffer (m) bytes-per-frame (bpf)
- For isochronous channels: (BD + 1) mod (BS + 1) = 0

- For asynchronous channels: (BD + 1) max packet length (1024 for a MOST Data Packet (MDP)
- 1536 for a MOST Ethernet Packet (MEP)
- For control channels: (BD + 1) max packet length (64)

4. Program the CAT for the inbound DMA.
    a. For TX channels (to MediaLB) HBI is the inbound DMA
    b. For RX channels (from MediaLB) MediaLB is the inbound DMA
    c. Set the channel direction: RNW = 0
    d. Set the channel type: CT[2:0] = 010 (asynchronous), 001 (control), 011 (isochronous), or 000 (synchronous)
    e. Set the connection label: CL[5:0] = N
    f. If CT[2:0] = 000 (synchronous), set the mute bit (MT = 1)
    g. Set the channel enable: CE = 1
    h. Set all other bits of the CAT to 0
5. Program the CAT for the outbound DMA.
    a. For TX channels (to MediaLB) HBI is the outbound DMA
    b. For RX channels (from MediaLB) MediaLB is the outbound DMA
    c. Set the channel direction: RNW = 1
    d. Set the channel type: CT[2:0] = 010 (asynchronous), 001 (control), 011 (isochronous), or 000 (synchronous)
    e. Set the channel label: CL[5:0] = N
    f. If CT[2:0] = 000 (synchronous), set the mute bit (MT = 1)
    g. Set the channel enable: CE = 1
    h. Set all other bits of the CAT to 0
6. Repeat steps 2 through 5 to initialize all logical channels.

Program the ADT

1. Initialize all bits of the ADT to 0
2. Select a logical channel: N = 1 – 63
3. Program the AMBA AHB block ping page for channel N
    a. Set the 32-bit base address (BA1)

b. Set the 11-bit buffer depth (BD1): BD1 = buffer depth in bytes – 1
   - For synchronous channels: (BD1 + 1) = n frames per sub-buffer (m) bytes-per-frame (bpf)
   - For isochronous channels: (BD1 + 1) mod (BS + 1) = 0
   - For asynchronous channels: 5 (BD1 + 1) 4096 (max packet length)
   - For control channels: 5 (BD1 + 1) 4096 (max packet length)

c. For asynchronous and control Tx channels set the packet start bit (PS1) when the page contains the start of the packet

d. Clear the page done bit (DNE1)

e. Clear the error bit (ERR1)

f. Set the page ready bit (RDY1)

4. Program the AMBA AHB block pong page for channel N
   a. Set the 32-bit base address (BA2)
   b. Set the 11-bit buffer depth (BD2): BD2 = buffer depth in bytes – 1
      - For synchronous channels: (BD2 + 1) = n frames per sub-buffer (m) bytes-per-frame (bpf)
      - For isochronous channels: (BD2 + 1) mod (BS + 1) = 0
      - For asynchronous channels: 5 (BD2 + 1) 4096 (max packet length)
      - For control channels: 5 (BD2 + 1) 4096 (max packet length)
   c. For asynchronous and control TX channels set the packet start bit (PS2) when the page contains the start of the packet
   d. Clear the page done bit (DNE2)
   e. Clear the error bit (ERR2)
   f. Set the page ready bit (RDY2)
5. Select Big Endian (LE = 0) or Little Endian (LE = 1)
6. Select the active page: PG = 0 (ping), PG = 1 (pong)
7. Set the channel enable (CE) bit for all active logical channels
8. Repeat steps 2 through 7 for all active logical channels.

Service

After initialization, each channel will require periodic servicing. Use the procedures in the following sections to service DMA and MLB interrupts and to poll for MLB system commands.

Servicing the DMA Channel Interrupts

1. Program the MLB_ACMR0/MLB_ACMR1 registers to enable interrupts from all active DMA channels.
2. Select the status clear method. MLB_ACTL.SCE =0 (hardware clears on read), MLB_ACTL.SCE = 1 (software writes a 1 to clear).
3. Select 1 or 2 interrupt signals. Configure the MLB_ACTL.SMX bit =0 (one interrupt for channels 0 through 31 and another interrupt for channels 32 through 63 on). Configure the MLB_ACTL.SMX bit =1 (single interrupt for all channels).
4. Wait for an interrupt.
5. Read the MLB_ACSR0/MLB_ACSR1 registers to determine which channel or channels are causing the interrupt.
6. If the MLB_ACTL.SCE bit =1, write the results of step 5 back to the MLB_ACSR0 and MLB_ACSR1 registers to clear the interrupt.
7. Select a logical channel (N = 0 – 63) with an interrupt to service.
8. Read the ADT entry for channel N to:
    a. Determine the active page (ping or pong) via the PG bit
    b. Determine which page(s) are done via the DNEn bits
    c. Determine which channels encountered an AHB error via the ERRn bit
    d. Determine which asynchronous and control Rx channel pages contain a packet start via the PSn bit (extract the PML)
9. Repeat steps 6 through 8 for all channels with pending interrupts
10. Repeat steps 4 through 9 while there are active channels.

Servicing the MediaLB Status Interrupts

1. Select the MediaLB channel status register (MLB_MS0/MLB_MS1) to be cleared by software, writing a 0 to the appropriate bits.
2. Program the MLB_MIEN register to enable protocol error interrupts for all active MediaLB channels. (MLB_MIEN.CTXPE =1, MLB_MIEN.CRXPE =1, MLB_MIEN.ATXPE =1, MLB_MIEN.ARXPE =1, MLB_MIEN.SYNCPE =1, and MLB_MIEN.ISOCPE =1).
3. Wait for an interrupt on the MLB_INT0/MLB_INT1 signal.
4. Read the MLB_MS0/MLB_MS1 registers to determine which channel(s) are causing the interrupt.
5. Read the RSTS/WSTS of the appropriate CDT(s) to determine the interrupt type.

6. Clear the RSTS/WSTS errors to ensure that the status of channel operations is reflected in the register:

Option	Description
For synchronous RX channels	WSTS[3] = 0
For synchronous TX channels	RSTS[3] = 0
For isochronous RX channels	WSTS[2:1] = 00
For isochronous TX channels	RSTS[2:1] = 00
For asynchronous and control RX channels	WSTS[4] = 0 and WSTS[2] = 0
For asynchronous and control TX channels	RSTS[4] = 0 and RSTS[2] = 0

Polling for MediaLB System Commands

The MediaLB System status (`MLB_MSS`) register is used to detect a system command received from the MediaLB controller. The processor's peripheral automatically sends the appropriate system response to the MediaLB controller. The procedure for the application is:

1. The application periodically polls the `MLB_MSS` register.
2. Clear by writing a 0 to the appropriate bit in the `MLB_MSS` register after the application finishes the service.
3. If `MLB_MSS.SWSYSCMD` = 1, read the `MLB_MSD` register to receive the system data sent from MediaLB controller.

ADSP-SC59x MLB Register Descriptions

MediaLB Device Interface Macro 2 (MLB) contains the following registers.

Table 28-19: ADSP-SC59x MLB Register List

Name	Description
`MLB_ACMR0`	Peripheral Channel Mask 0 Register
`MLB_ACMR1`	Peripheral Channel Mask 1 Register
`MLB_ACSR0`	Peripheral Channel Status 0 Register
`MLB_ACSR1`	Peripheral Channel Status 1 Register
`MLB_ACTL`	Bus Control Register
`MLB_CTL0`	MediaLB Control 0 Register
`MLB_CTL1`	Control 1 Register
`MLB_GCTL`	MLB Global Control Register
`MLB_HCBR0`	HBI Channel Busy 0 Register
`MLB_HCBR1`	HBI Channel Busy 1 Register

Table 28-19: ADSP-SC59x MLB Register List (Continued)

Name	Description
MLB_HCER0	HBI Channel Error 0 Register
MLB_HCER1	HBI Channel Error 1 Register
MLB_HCMR0	HBI Channel Mask 0 Register
MLB_HCMR1	HBI Channel Mask 1 Register
MLB_HCTL	HBI Control Register
MLB_MADR	Memory Interface Address Register
MLB_MCTL	Memory Interface Control Register
MLB_MDAT0	Memory Interface Control Data 0 Register
MLB_MDAT1	Memory Interface Control Data 1 Register
MLB_MDAT2	Memory Interface Control Data 2 Register
MLB_MDAT3	Memory Interface Control Data 3 Register
MLB_MDWE0	Memory Interface Control Data Write Enable 0 Register
MLB_MDWE1	Memory Interface Control Data Write Enable 1 Register
MLB_MDWE2	Memory Interface Control Data Write Enable 2 Register
MLB_MDWE3	Memory Interface Control Data Write Enable 3 Register
MLB_MIEN	Interrupt Enable Register
MLB_MS0	Channel Status 0 Register
MLB_MS1	Channel Status 1 Register
MLB_MSD	System Data Register
MLB_MSS	System Status Register
MLB_PCTL0	MediaLB 6-pin Control 0 Register

Peripheral Channel Mask 0 Register

The MLB_ACMR0 register allows control over which channel(s) generate interrupts on MLB_INT[1:0]. All of the bits in this register default to 0 (masked) so the HC must initially write to the MLB_ACMR0 register to enable interrupts. Each bit of this register is read/write accessible.

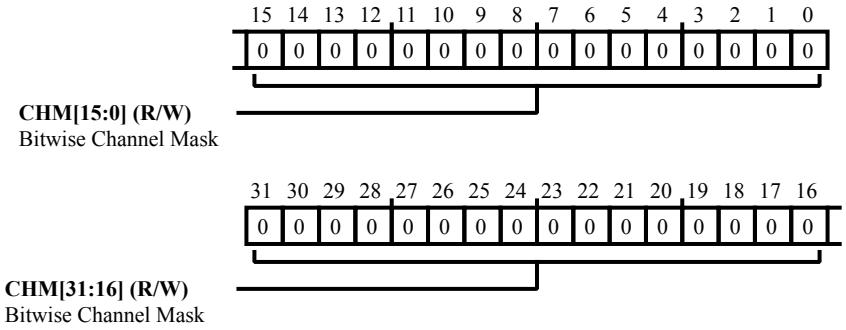


Figure 28-7: MLB_ACMR0 Register Diagram

Table 28-20: MLB_ACMR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	CHM	Bitwise Channel Mask. The MLB_ACMR0.CHM bit field masks or unmasks channels 31-0.	
		0	Mask interrupt for Channel 31:0 (bitwise; all channels shown)
		4294967295	Unmask interrupt for Channel 31:0 (bitwise; all channels shown)

Peripheral Channel Mask 1 Register

The MLB_ACMR1 register allows control over which channel(s) generate interrupts on MLB_INT[1:0]. All of the bits in this register default to 0 (masked) so the HC must initially write to the MLB_ACMR1 register to enable interrupts. Each bit of this register is read/write accessible.

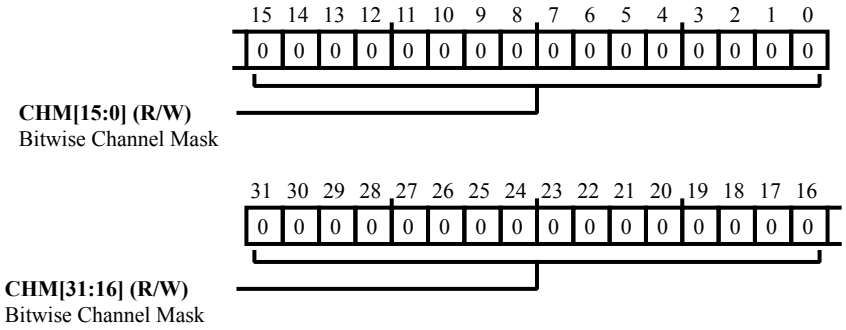


Figure 28-8: MLB_ACMR1 Register Diagram

Table 28-21: MLB_ACMR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W)	CHM	Bitwise Channel Mask. The MLB_ACMR1.CHM bit field masks or unmasks channels 63-32.	
		0	Mask interrupt for Channel 63:32 (bitwise; all channels shown)
		4294967295	Unmask interrupt for Channel 63:32 (bitwise; all channels shown)

Peripheral Channel Status 0 Register

The MLB_ACSR0 register contains interrupt bits for each of the 64 physical channels. When a bit in this register is set, it indicates that the corresponding physical channel has an interrupt pending.

A peripheral interrupt is triggered when either DNEn or ERRn is set within the Bus Channel Descriptor. The HC is notified of the channel interrupt via MLB_INT[1:0]. When an interrupt occurs in ACCUSER (for channels 31 to 0) MLB_INT[0] is set. When an interrupt occurs in MLB_ACSR1 (for channels 63 to 32) MLB_INT[1] is set.

Interrupts in the MLB_ACSR0 and MLB_ACSR1 registers can be optionally multiplexed onto a single interrupt signal, MLB_INT[0], if the MLB_ACTL.SMX bit = 1.

If the MLB_ACTL.SCE bit =0, hardware automatically clears the interrupt bit(s) after the HC reads the peripheral channel status registers. Alternatively, if the MLB_ACTL.SCE bit =1, software must write a 1 to the appropriate bit(s) of the peripheral channel status registers to clear the interrupt(s).

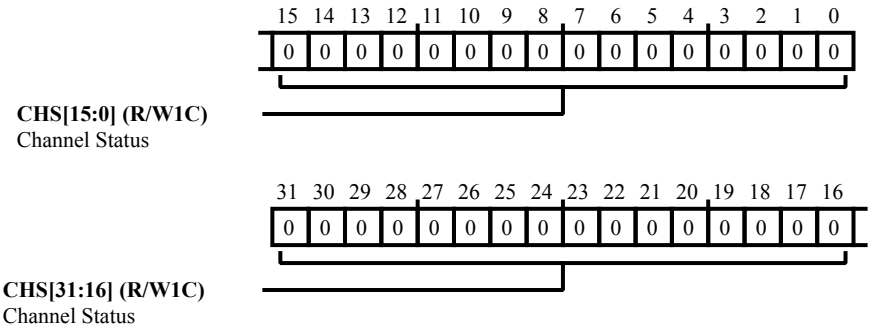


Figure 28-9: MLB_ACSR0 Register Diagram

Table 28-22: MLB_ACSR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W1C)	CHS	Channel Status. The MLB_ACSR0.CHS bit field indicates channel status for channels 31-0.	
		0	No interrupt on Channel 31:0 (bitwise; all channels shown)
		4294967295	Interrupt on Channel 31:0 (bitwise; all channels shown)

Peripheral Channel Status 1 Register

The MLB_ACSR1 register contains interrupt bits for each of the 64 physical channels. When a bit in this register is set, it indicates that the corresponding physical channel has an interrupt pending.

A peripheral interrupt is triggered when either DNEn or ERRn is set within the Bus Channel Descriptor. The HC is notified of the channel interrupt via MLB_INT[1:0]. When an interrupt occurs in ACCUSER (for channels 31 to 0) MLB_INT[0] is set. When an interrupt occurs in MLB_ACSR1 (for channels 63 to 32) MLB_INT[1] is set.

Interrupts in the MLB_ACSR1 and MLB_ACSR0 registers can be optionally multiplexed onto a single interrupt signal, MLB_INT[0], if the MLB_ACTL.SMX bit = 1.

If the MLB_ACTL.SCE bit =0, hardware automatically clears the interrupt bit(s) after the HC reads the peripheral channel status registers. Alternatively, if the MLB_ACTL.SCE bit =1, software must write a 1 to the appropriate bit(s) of peripheral channel status registers to clear the interrupt(s).

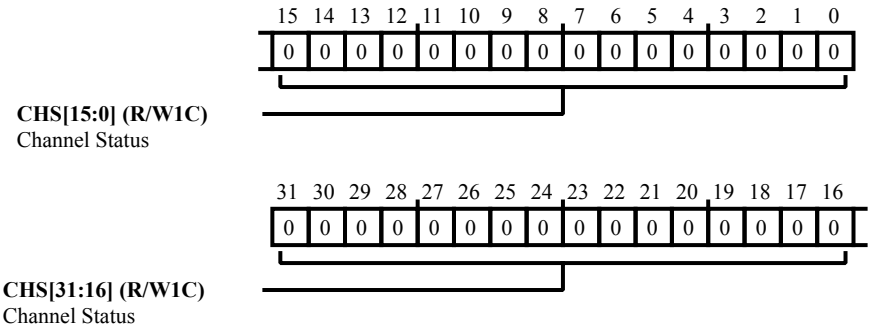


Figure 28-10: MLB_ACSR1 Register Diagram

Table 28-23: MLB_ACSR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:0 (R/W1C)	CHS	Channel Status. The MLB_ACSR1.CHS bit field indicates channel status for channels 63-32.	
		0	No interrupt on Channel 63:32 (bitwise; all channels shown)
		4294967295	Interrupt on Channel 63:32 (bitwise; all channels shown)

Bus Control Register

The MLB_ACTL register is written by the HC to configure the block for channel interrupts. This register contains bits to select the packet buffering mode and the DMA mode. This register also contains bits that are used to multiplex channel interrupts onto a single interrupt signal and to select the method of clearing channel interrupts.

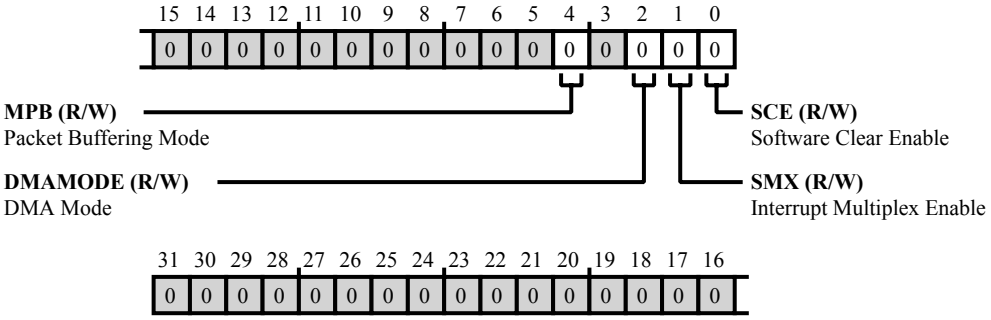


Figure 28-11: MLB_ACTL Register Diagram

Table 28-24: MLB_ACTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	MPB	Packet Buffering Mode. The MLB_ACTL.MPB bit selects whether the buffering mode is single-packet or multiple-packet.	
		0	single-packet mode
		1	multiple-packet mode
2 (R/W)	DMAMODE	DMA Mode. The MLB_ACTL.DMAMODE bit selects between DMA mode 1 and 0. DMA Mode 0 uses incrementing bursts of an unspecified length. This allows the block to perform single beat transfers as well as an incrementing (INCR) burst of unspecified length up to the maximum specified burst length (8 beats). DMA Mode 1 uses only INCR8, INCR4, and SINGLE beat bursts. The hburst[2:0] signal encodes the burst transfer used for a DMA operation.	
		0	DMA Mode 0
		1	DMA Mode 1

Table 28-24: MLB_ACTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	SMX	Interrupt Multiplex Enable. The `MLB_ACTL.SMX` bit selects whether ACSR0 generates an interrupt on MLB_INT[0] and ACSR1 generates an interrupt on MLB_INT[1] or ACSR0 and ACSR1 generate an interrupt on MLB_INT[0] only.	
		0	ACSR0 generates an interrupt on MLB_INT[0]; ACSR1 generates an interrupt on MLB_INT[1]
		1	ACSR0 and ACSR1 generate an interrupts on MLB_INT[0] only
0 (R/W)	SCE	Software Clear Enable. The `MLB_ACTL.SCE` bit selects whether hardware clears the interrupt after a ACSRn register read or software clears the interrupt.	
		0	Hardware clears interrupt after a ACSRn register read
		1	Software clears interrupt

MediaLB Control 0 Register

The MLB_CTL0 register contains bit that enable the module and provide module status. Note that the maximum speed for ML6-pin mode is 2048 x FS.

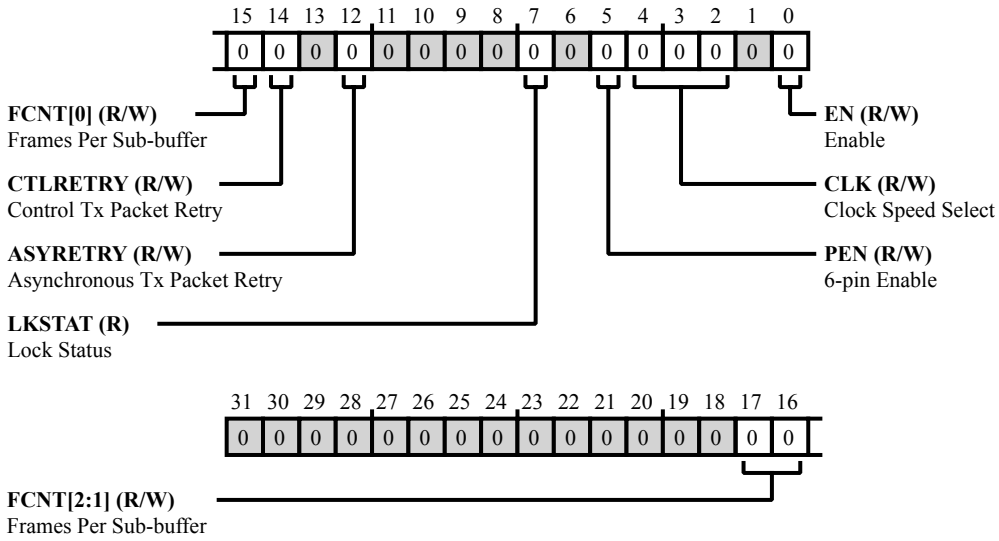


Figure 28-12: MLB_CTL0 Register Diagram

Table 28-25: MLB_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17:15 (R/W)	FCNT	Frames Per Sub-buffer. The MLB_CTL0.FCNT bit field configures the frames per sub-buffer on synchronous channels.	
		0	1 Frame per sub-buffer (Operation is the same as Standard mode)
		1	2 frames per sub-buffer
		2	4 frames per sub-buffer
		3	8 frames per sub-buffer
		4	16 frames per sub-buffer
		5	32 frames per sub-buffer
		6	64 frames per sub-buffer
14 (R/W)	CTLRETRY	Control Tx Packet Retry. When the MLB_CTL0.CTLRETRY bit is set, a control packet that is flagged with a Break or Protocol error by the receiver is retransmitted. When cleared, a control packet that is flagged with a Break or Protocol error by the receiver is skipped.	

Table 28-25: MLB_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	ASYRETRY	Asynchronous Tx Packet Retry. When the MLB_CTL0.ASYRETRY bit is set, an asynchronous packet that is flagged with a Break or Protocol error by the receiver is retransmitted. When cleared, an asynchronous packet that is flagged with a Break or ProtocolError by the receiver is skipped.	
7 (R/NW)	LKSTAT	Lock Status. When the MLB_CTL0.LKSTAT bit is set (=0), the MediaLB block is synchronized to the incoming MediaLB frame with the following conditions. • If MLB_CTL1.LOCK =0 (unlocked), MLB_CTL1.LOCK is set after a FRAMESYNC is detected at the same position for three consecutive frames. • If MLB_CTL1.LOCK =1 (locked), MLB_CTL1.LOCK is cleared after not receiving a FRAMESYNC at the expected time for two consecutive frames. In this case FRAMESYNC patterns occurring at locations other than the expected one are ignored.	
5 (R/W)	PEN	6-pin Enable. The MLB_CTL0.PEN bit configures the MLB for 6-pin or 3-pin mode.	
		0	3-pin interface enabled
		1	6-pin interface enabled
4:2 (R/W)	CLK	Clock Speed Select. The MLB_CTL0.CLK bit field sets the clock speed.	
		0	256 x Fs (MLB_CTL.PEN = 0)
		1	512 x Fs (MLB_CTL.PEN = 0)
		2	1024 x Fs (MLB_CTL.PEN = 0)
		3	2048 x Fs (MLB_CTL.PEN = 1)
		4	Reserved
		5	Reserved
		6	Reserved
		7	Reserved
0 (R/W)	EN	Enable. When the MLB_CTL0.EN bit is set (=1), MediaLB clock, signal, and data are received and transmitted on the appropriate MediaLB pins.	

Control 1 Register

The MLB_CTL1 register contains bits that provide lock status and control system commands.

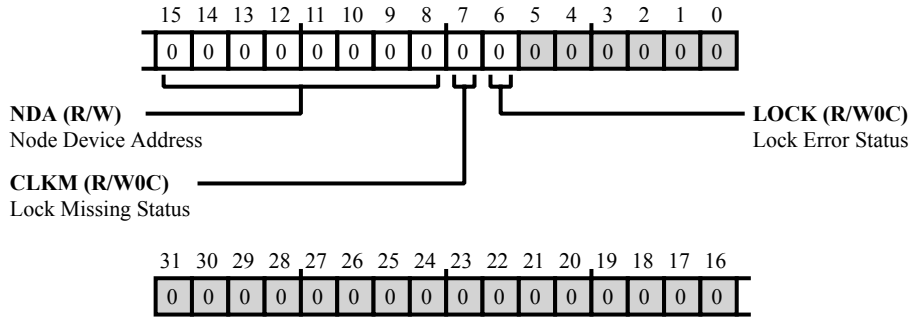


Figure 28-13: MLB_CTL1 Register Diagram

Table 28-26: MLB_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	NDA	Node Device Address. The MLB_CTL1.NDA bit field is used for system commands directed to individual MediaLB nodes.
7 (R/W0C)	CLKM	Lock Missing Status. The MLB_CTL1.CLKM bit is set when the MediaLB clock is not toggling at the pin. This bit is cleared by software.
6 (R/W0C)	LOCK	Lock Error Status. The MLB_CTL1.LOCK bit is set when the MediaLB is unlocked. This bit is cleared by software.

MLB Global Control Register

The MLB_GCTL register contains bits that manage the MLB clock.

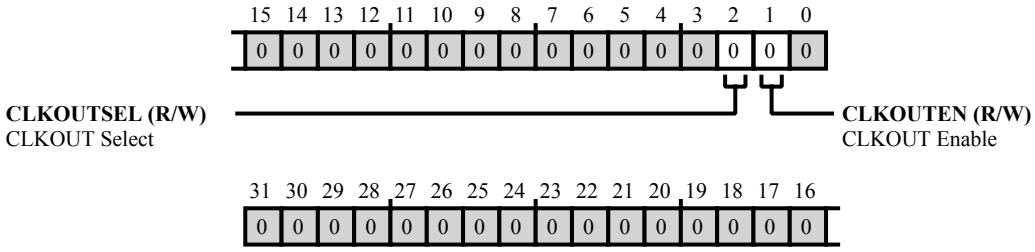


Figure 28-14: MLB_GCTL Register Diagram

Table 28-27: MLB_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	CLKOUTSEL	CLKOUT Select. The MLB_GCTL.CLKOUTSEL bit selects either the MLB 3 pin clock or the 6 pin clock as MLBCLKOUT.	
		0	MLB 3 pin clock is selected for MLBCLKOUT
		1	MLB 6 pin clock is selected for MLBCLKOUT
1 (R/W)	CLKOUTEN	CLKOUT Enable.	

HBI Channel Busy 0 Register

The HC can determine which channel(s) are busy by reading the MLB_HCBR0 register. An HBI channel is busy if:

- it is currently loaded into one of the two AGUs
- the channel is enabled, CE = 1 from the Channel Allocation Table
- the DMA is active

When an HBI channel is busy, hardware may write back its local copy of the channel descriptor at any time. System software should not write a CDT descriptor for a channel that is busy. Only two HBI channels can be busy at any given time. Each bit of this register is read-only.

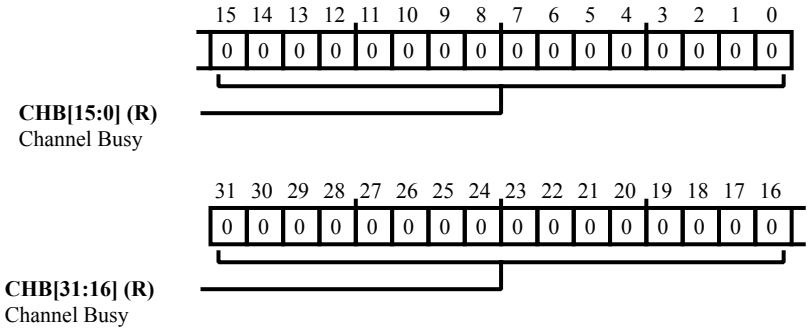


Figure 28-15: MLB_HCBR0 Register Diagram

Table 28-28: MLB_HCBR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CHB	Channel Busy. The MLB_HCBR0.CHB bit field contains the bitwise channel busy bit for channels 31:0. When a bit is cleared (=0) the channel is idle. When a bit is set (=1) the channel is busy.

HBI Channel Busy 1 Register

The HC can determine which channel(s) are busy by reading the MLB_HCBR1 register. An HBI channel is busy if:

- it is currently loaded into one of the two AGUs
- the channel is enabled, CE = 1 from the Channel Allocation Table
- the DMA is active

When an HBI channel is busy, hardware may write back its local copy of the channel descriptor at any time. System software should not write a CDT descriptor for a channel that is busy. Only two HBI channels can be busy at any given time. Each bit of this register is read-only.

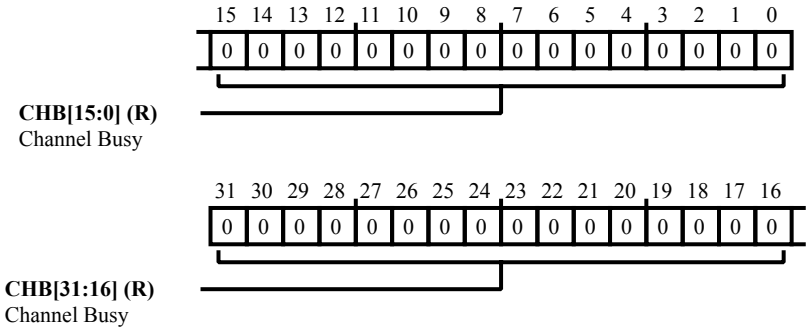


Figure 28-16: MLB_HCBR1 Register Diagram

Table 28-29: MLB_HCBR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CHB	Channel Busy. The MLB_HCBR1.CHB bit field contains the bitwise channel busy bit for channels 63:32. When a bit is cleared (=0) the channel is idle. When a bit is set (=1) the channel is busy.

HBI Channel Error 0 Register

The MLB_HCER0 register indicates which channels (channels 31:0) have encountered fatal errors.

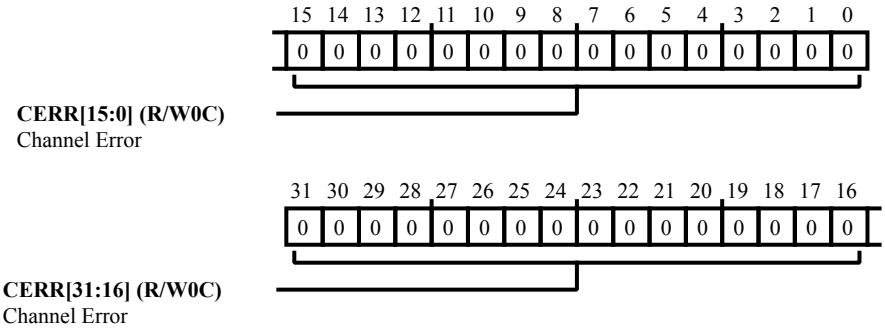


Figure 28-17: MLB_HCER0 Register Diagram

Table 28-30: MLB_HCER0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W0C)	CERR	Channel Error. The MLB_HCER0.CERR bit field reports bitwise errors for channels 31:0.

HBI Channel Error 1 Register

The MLB_HCER1 register indicates which channel(s) have encountered fatal errors.

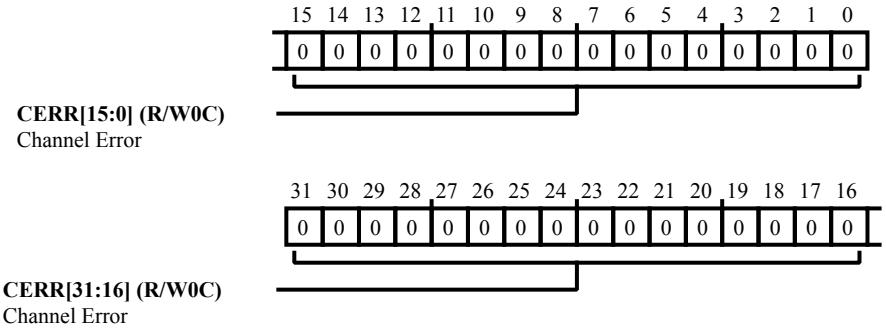


Figure 28-18: MLB_HCER1 Register Diagram

Table 28-31: MLB_HCER1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W0C)	CERR	Channel Error. The MLB_HCER1.CERR bite field reports bitwise errors for channels 63:32.

HBI Channel Mask 0 Register

The MLB_HCMR0 register controls which channels (for channels 31:0) are able to generate an HBI interrupt.

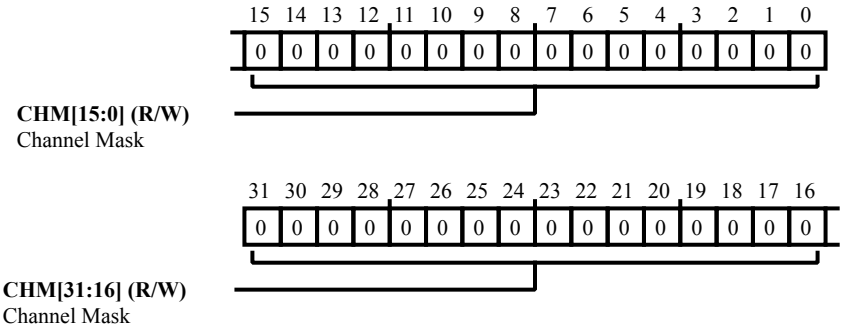


Figure 28-19: MLB_HCMR0 Register Diagram

Table 28-32: MLB_HCMR0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CHM	Channel Mask. The MLB_HCMR0.CHM bit field contains the bitwise channel mask bit for channels 31:0. When a bit is cleared (=0) the channel is masked. When set (=1) the channel is unmasked.

HBI Channel Mask 1 Register

The MLB_HCMR1 register controls which channels (for channels 63:32) are able to generate an HBI interrupt.

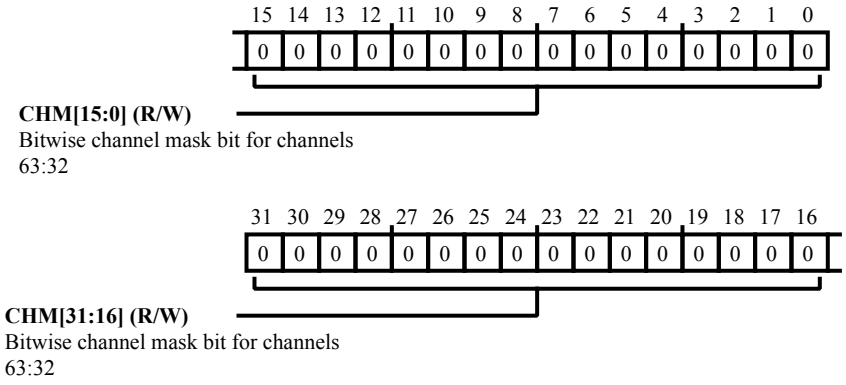


Figure 28-20: MLB_HCMR1 Register Diagram

Table 28-33: MLB_HCMR1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	CHM	Bitwise channel mask bit for channels 63:32. The MLB_HCMR1.CHM bit field contains the bitwise channel mask bit for channels 63:32. When a bit is cleared (=0) the channel is masked. When set (=1) the channel is unmasked.

HBI Control Register

The MLB_HCTL register controls and monitors general operation of the HBI block through the Address Generation Units) by reading and writing the register through the I/O interface. Each bit of this register is read/write.

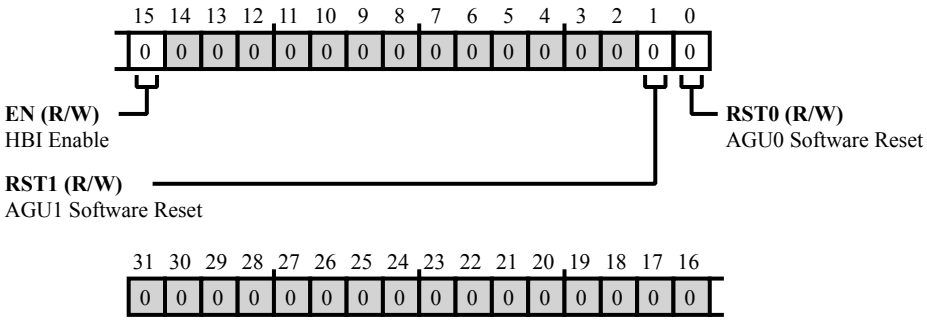


Figure 28-21: MLB_HCTL Register Diagram

Table 28-34: MLB_HCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	EN	HBI Enable. Setting the MLB_HCTL.EN bit enables the HBI.
1 (R/W)	RST1	AGU1 Software Reset. Setting the MLB_HCTL.RST1 bit resets AGU1.
0 (R/W)	RST0	AGU0 Software Reset. Setting the MLB_HCTL.RST0 bit resets AGU0.

Memory Interface Address Register

The MLB_MADR register contains bit fields that contain the Channel Table RAM (CTR) or Data Buffer RAM (DBR) addresses. It also contains bits that set target location and read-not-write parameters.

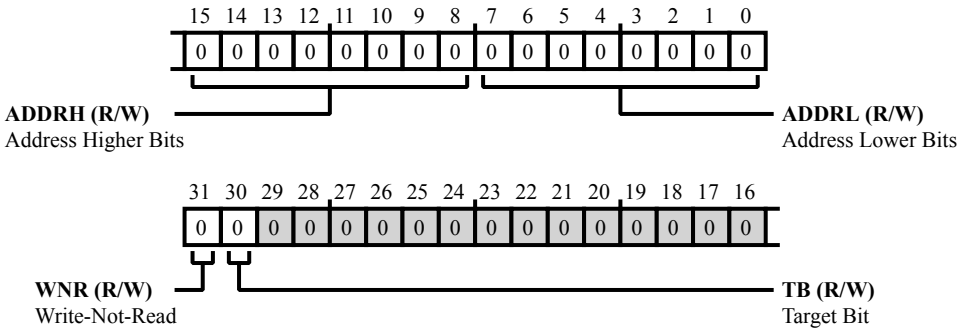


Figure 28-22: MLB_MADR Register Diagram

Table 28-35: MLB_MADR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	WNR	Write-Not-Read. The MLB_MADR.WNR bit selects Write-Not-Read.	
		0	Read
		1	Write
30 (R/W)	TB	Target Bit. The MLB_MADR.TB bit sets the target location.	
		0	Channel Table RAM (CTR)
		1	Data Buffer RAM (DBR)
15:8 (R/W)	ADDRH	Address Higher Bits. The MLB_MADR.ADDRH bit field contains the DBR address of the 8-bit entry (bits [13:8]).	
7:0 (R/W)	ADDRL	Address Lower Bits. The MLB_MADR.ADDRL bit field contains the CTR address of the 128-bit entry or the DBR address of the 8-bit entry (bits [7:0]).	

Memory Interface Control Register

The MLB_MCTL register contains a bit that indicates that the data transfer is complete.

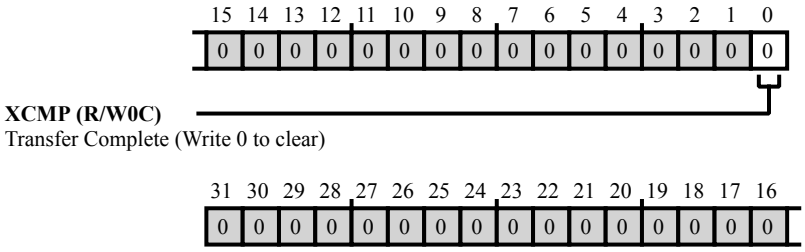


Figure 28-23: MLB_MCTL Register Diagram

Table 28-36: MLB_MCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W0C)	XCMP	Transfer Complete (Write 0 to clear). The MLB_MCTL.XCMP bit indicates that the data transfer is complete.

Memory Interface Control Data 0 Register

The MLB_MDAT0 register contains Channel Table RAM (CTR) data (bits [31:0] of 128-bit entry) or Data Buffer RAM (DBR) data (bits [7:0] of 8-bit entry).

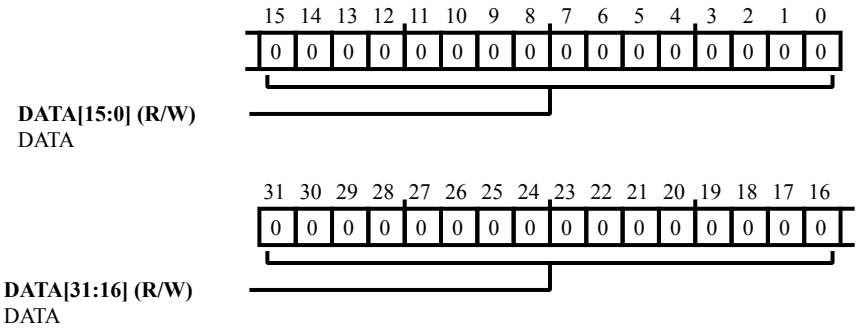


Figure 28-24: MLB_MDAT0 Register Diagram

Table 28-37: MLB_MDAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	DATA. The MLB_MDAT0.DATA bit field contains CTR data or DBR data.

Memory Interface Control Data 1 Register

The MLB_MDAT1 register contains Channel Table RAM (CTR) data (bits [63:32] of 128-bit entry).

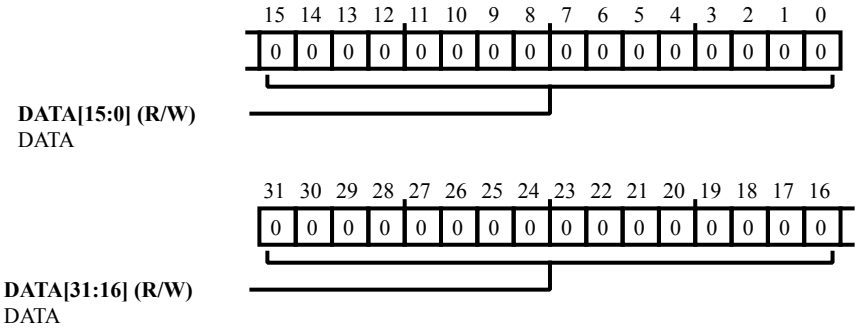


Figure 28-25: MLB_MDAT1 Register Diagram

Table 28-38: MLB_MDAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	DATA. The MLB_MDAT1.DATA bit field contains CTR data.

Memory Interface Control Data 2 Register

The MLB_MDAT2 register contains Channel Table RAM (CTR) data (bits [95:64] of 128-bit entry).

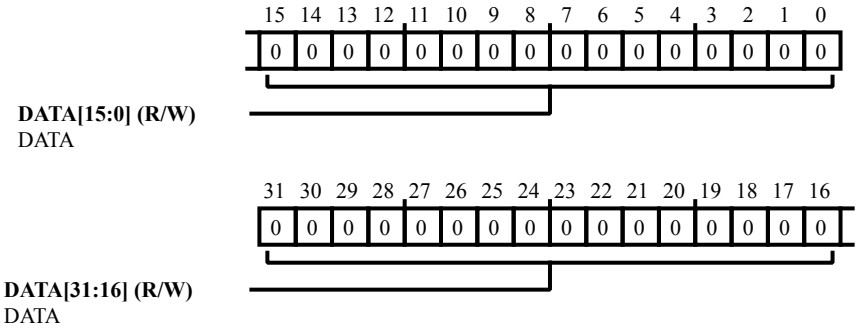


Figure 28-26: MLB_MDAT2 Register Diagram

Table 28-39: MLB_MDAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	DATA. The MLB_MDAT2.DATA bit field contains CTR data.

Memory Interface Control Data 3 Register

The MLB_MDAT3 register contains Channel Table RAM (CTR) data (bits [127:96] of 128-bit entry).

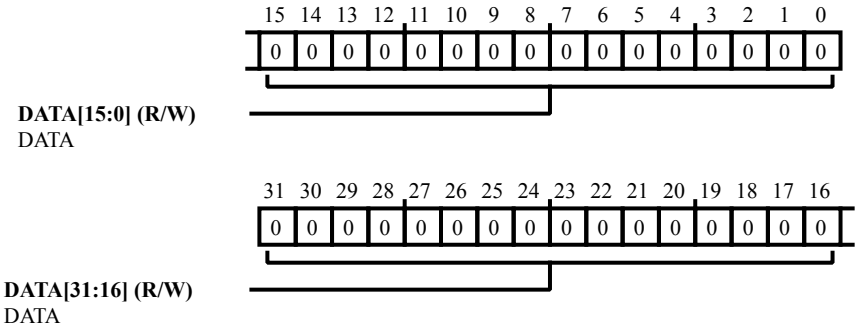


Figure 28-27: MLB_MDAT3 Register Diagram

Table 28-40: MLB_MDAT3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	DATA. The MLB_MDAT3.DATA bit field contains CTR data.

Memory Interface Control Data Write Enable 0 Register

The MLB_MDWE0 register contains the bitwise write enable for Channel Table RAM (CTR) data bits [31:0]. When cleared (=0) the bit is disabled, when set (=1) the bit is enabled.

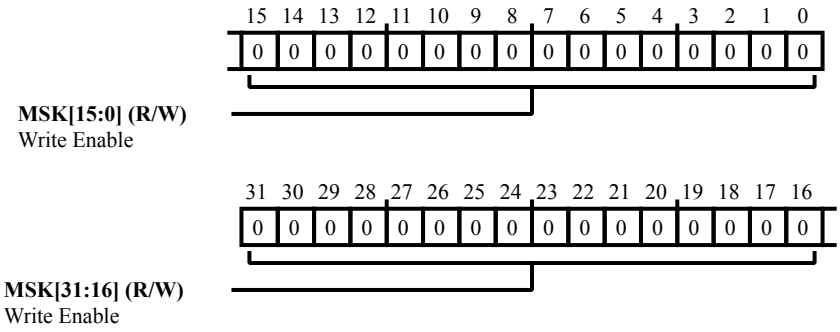


Figure 28-28: MLB_MDWE0 Register Diagram

Table 28-41: MLB_MDWE0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSK	Write Enable. The MLB_MDWE0.MSK bit field contains the bitwise write enable for CTR data.

Memory Interface Control Data Write Enable 1 Register

The MLB_MDWE1 register contains the bitwise write enable for Channel Table RAM (CTR) data bits [63:32]. When cleared (=0), the bit is disabled. When set (=1), the bit is enabled.

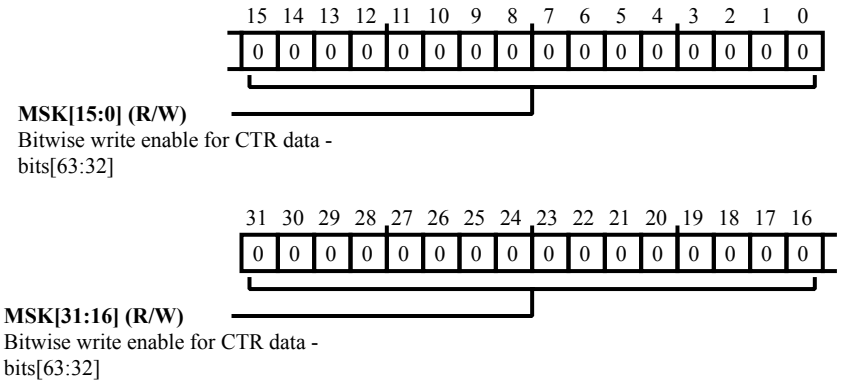


Figure 28-29: MLB_MDWE1 Register Diagram

Table 28-42: MLB_MDWE1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSK	Bitwise write enable for CTR data - bits[63:32]. The MLB_MDWE1.MSK bit field contains the bitwise write enable for CTR data.

Memory Interface Control Data Write Enable 2 Register

The MLB_MDWE2 register contains the bitwise write enable for Channel Table RAM (CTR) data bits [95:64]. When cleared (=0), the bit is disabled. When set (=1), the bit is enabled.

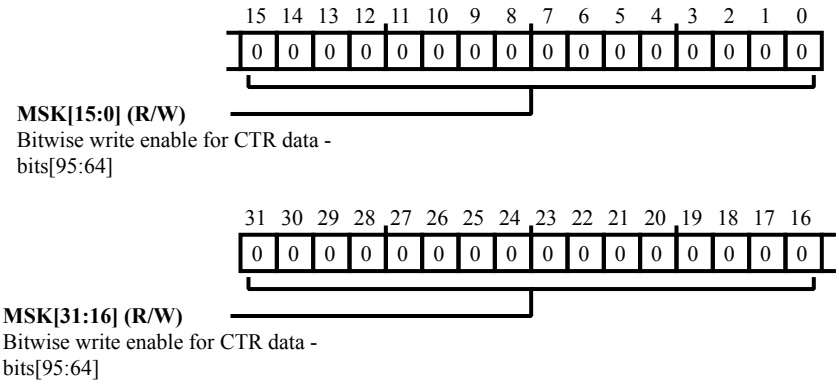


Figure 28-30: MLB_MDWE2 Register Diagram

Table 28-43: MLB_MDWE2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSK	Bitwise write enable for CTR data - bits[95:64]. The MLB_MDWE2.MSK bit field contains the bitwise write enable for CTR data.

Memory Interface Control Data Write Enable 3 Register

The MLB_MDWE3 register contains the bitwise write enable for Channel Table RAM (CTR) data bits [127:96]. When cleared (=0), the bit is disabled. When set (=1), the bit is enabled.

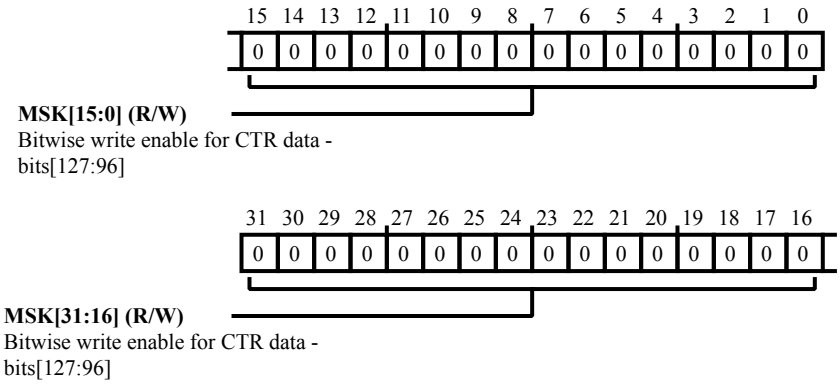


Figure 28-31: MLB_MDWE3 Register Diagram

Table 28-44: MLB_MDWE3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MSK	Bitwise write enable for CTR data - bits[127:96]. The MLB_MDWE3.MSK bit field contains the bitwise write enable for CTR data.

Interrupt Enable Register

The MLB_MIEN register is used to enable various interrupt conditions.

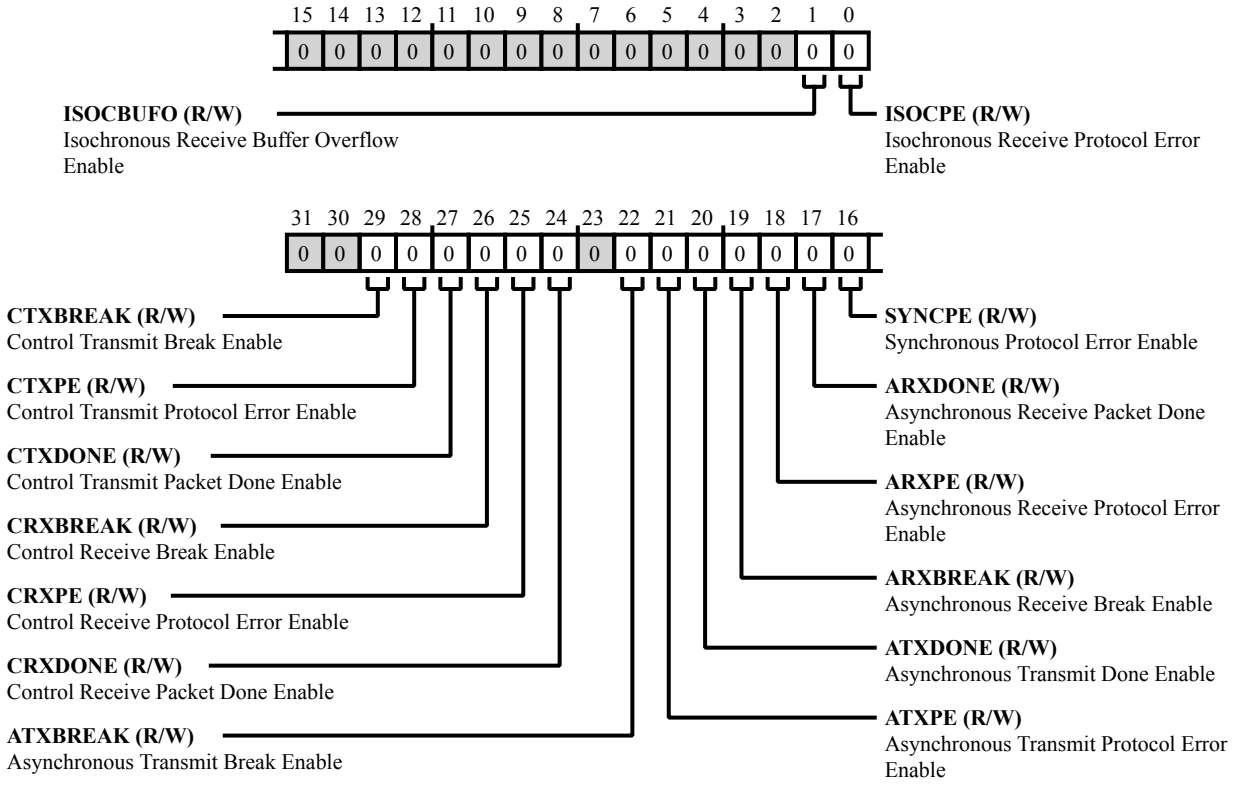


Figure 28-32: MLB_MIEN Register Diagram

Table 28-45: MLB_MIEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	CTXBREAK	Control Transmit Break Enable. When the MLB_MIEN.CTXBREAK bit is set, a ReceiverBreak response received from the receiver on a control Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
28 (R/W)	CTXPE	Control Transmit Protocol Error Enable. When the MLB_MIEN.CTXPE bit is set, a ProtocolError generated by the receiver on a control Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
27 (R/W)	CTXDONE	Control Transmit Packet Done Enable. When the MLB_MIEN.CTXDONE bit is set, a packet transmitted with no errors on a control Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.

Table 28-45: MLB_MIEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
26 (R/W)	CRXBREAK	Control Receive Break Enable. When the MLB_MIEN.CRXBREAK bit is set, a ControlBreak command received from the transmitter on a control Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
25 (R/W)	CRXPE	Control Receive Protocol Error Enable. When the MLB_MIEN.CRXPE bit is set, a ProtocolError detected on a control Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
24 (R/W)	CRXDONE	Control Receive Packet Done Enable. When the MLB_MIEN.CRXDONE bit is set, a packet received with no errors on a control Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
22 (R/W)	ATXBREAK	Asynchronous Transmit Break Enable. When the MLB_MIEN.ATXBREAK bit is set, a ReceiverBreak response received from the receiver on an asynchronous Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
21 (R/W)	ATXPE	Asynchronous Transmit Protocol Error Enable. When the MLB_MIEN.ATXPE bit is set, a ProtocolError generated by the receiver on an asynchronous Tx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
20 (R/W)	ATXDONE	Asynchronous Transmit Done Enable. When the MLB_MIEN.ATXDONE bit is set, a packet transmitted with no errors on an asynchronous Tx channel channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
19 (R/W)	ARXBREAK	Asynchronous Receive Break Enable. When the MLB_MIEN.ARXBREAK bit is set, a AsyncBreak command received from the transmitter on an asynchronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
18 (R/W)	ARXPE	Asynchronous Receive Protocol Error Enable. When the MLB_MIEN.ARXPE bit is set, a protocol error detected on an asynchronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.
17 (R/W)	ARXDONE	Asynchronous Receive Packet Done Enable. When the MLB_MIEN.ARXDONE bit is set, a packet received with no errors on an asynchronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.

Table 28-45: MLB_MIEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	SYNCPE	Synchronous Protocol Error Enable. When the MLB_MIEN.SYNCPE bit is set, a protocol error detected on a synchronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set. This occurs only when isochronous flow control is disabled.
1 (R/W)	ISOCBUFO	Isochronous Receive Buffer Overflow Enable. When the MLB_MIEN.ISOCBUFO bit is set, a buffer overflow on an isochronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set. This occurs only when isochronous flow control is disabled.
0 (R/W)	ISOCPE	Isochronous Receive Protocol Error Enable. When the MLB_MIEN.ISOCPE bit is set, a protocol error detected on an isochronous Rx channel causes the appropriate channel bit in the MLB_MS0 or MLB_MS1 registers to be set.

Channel Status 0 Register

The MLB_MS0 register indicates the channel status for MediaLB channels 31 to 0. Channel status bits are set by hardware and cleared by software. Status is only set if the appropriate bits in the MLB_MIEN register are set.

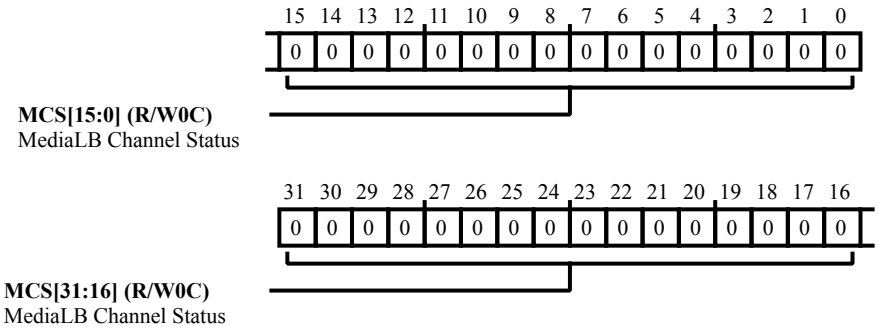


Figure 28-33: MLB_MS0 Register Diagram

Table 28-46: MLB_MS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W0C)	MCS	MediaLB Channel Status. The MLB_MS0.MCS bit field indicates the MediaLB channel status for channels 31 to 0.

Channel Status 1 Register

The MLB_MS1 register indicates the channel status for MediaLB channels 32 to 63. Channel status bits are set by hardware and cleared by software. Status is only set if the appropriate bits in the MLB_MIEN register are set.

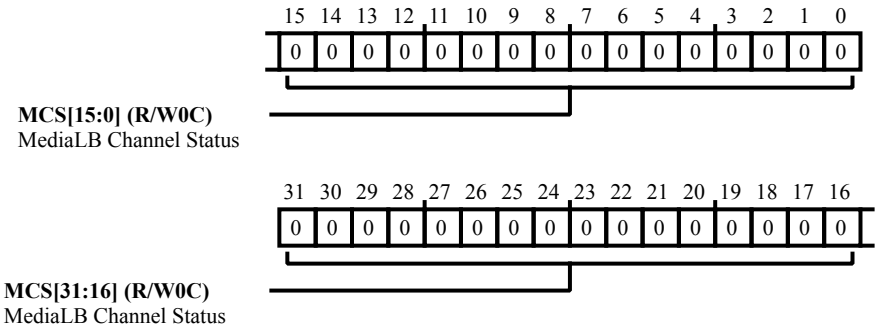


Figure 28-34: MLB_MS1 Register Diagram

Table 28-47: MLB_MS1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W0C)	MCS	MediaLB Channel Status. The MLB_MS1.MCS bit field indicates the MediaLB channel status for channels 63 to 32.

System Data Register

The MLB_MSD register allows system software to receive control information from the MLB controller. The MLB_MSD register is updated once per frame by the hardware during the MLB system channel.

The MLB_MSD register is loaded with the data from the MLBDAT_IN signal during the system channel quadlet. System software must read this register before the start of the next MLB frame to prevent the current data from being lost.

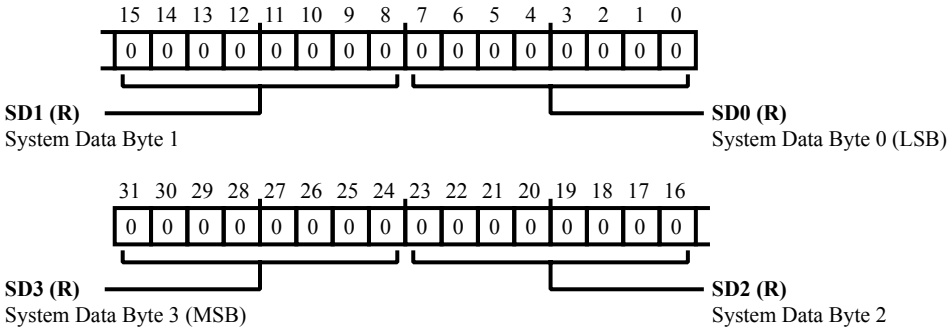


Figure 28-35: MLB_MSD Register Diagram

Table 28-48: MLB_MSD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	SD3	System Data Byte 3 (MSB). The MLB_MSD.SD3 bits are updated with MediaLB Data [31:24] when a MediaLB software system command is received in the system quadlet. If the MLB_MSS.SWSYSCMD bit is already set, then SD3 is not updated.
23:16 (R/NW)	SD2	System Data Byte 2. The MLB_MSD.SD2 bits are updated with MediaLB Data [23:16] when a MediaLB software system command is received in the system quadlet. If the MLB_MSS.SWSYSCMD bit is already set, then SD2 is not updated.
15:8 (R/NW)	SD1	System Data Byte 1. The MLB_MSD.SD1 bits are updated with MediaLB Data [15:8] when a MediaLB software system command is received in the system quadlet. If the MLB_MSS.SWSYSCMD bit is already set, then SD1 is not updated.
7:0 (R/NW)	SD0	System Data Byte 0 (LSB). The MLB_MSD.SD0 bits are updated with MediaLB Data [7:0] when a MediaLB software system command is received in the system quadlet. If the MLB_MSS.SWSYSCMD bit is already set, then SD0 is not updated.

System Status Register

The MLB_MSS register allows system software to monitor and control the status of the MLB network. The register is updated once per frame by hardware during the MLB system channel. The bits of this register are not valid until the processor is locked to the MLB interface (except for the bits associated with MLB lock and unlock, SDMU and SDML). System software must service events before the start of the next MLB frame to prevent the current frame status from being lost.

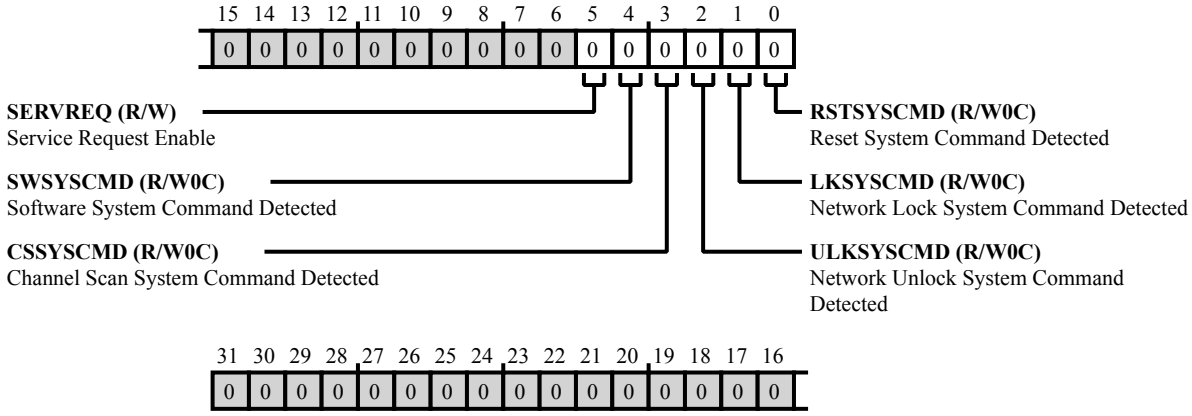


Figure 28-36: MLB_MSS Register Diagram

Table 28-49: MLB_MSS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W)	SERVREQ	Service Request Enable. When the MLB_MSS.SERVREQ bit set, the MediaLB block responds with a device present, request service system response if a matching channel scan system command is detected. When cleared, the MediaLB block responds with a device present system response.
4 (R/W0C)	SWSYSCMD	Software System Command Detected. The MLB_MSS.SWSYSCMD bit indicates that a software system command was detected (in the system quadlet). The MLB_MSS.SWSYSCMD bit is set by hardware, cleared by software. Data is stored in the MLB_MSD register for this command.
3 (R/W0C)	CSSYSCMD	Channel Scan System Command Detected. The MLB_MSS.CSSYSCMD bit indicates that a channel scan system command was detected (in the system quadlet). The MLB_MSS.CSSYSCMD bit is set by hardware, cleared by software. If the node address specified in the data quadlet matches the value in the MLB_CTL1.NDA bit field, the device responds either device present or device present, request service system response in the next system quadlet.

Table 28-49: MLB_MSS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/W0C)	ULKSYSCMD	Network Unlock System Command Detected. The `MLB_MSS.ULKSYSCMD` bit indicates a network unlock system command was detected (in the system quadlet). The `MLB_MSS.ULKSYSCMD` bit is by hardware, cleared by software.
1 (R/W0C)	LKSYSCMD	Network Lock System Command Detected. The `MLB_MSS.LKSYSCMD` bit indicates a network lock system command was detected (in the system quadlet). The `MLB_MSS.LKSYSCMD` bit is set by hardware and cleared by software.
0 (R/W0C)	RSTSYSCMD	Reset System Command Detected. The `MLB_MSS.RSTSYSCMD` bit indicates a reset system command was detected (in the system quadlet). The `MLB_MSS.RSTSYSCMD` bit is set by hardware, cleared by software.

MediaLB 6-pin Control 0 Register

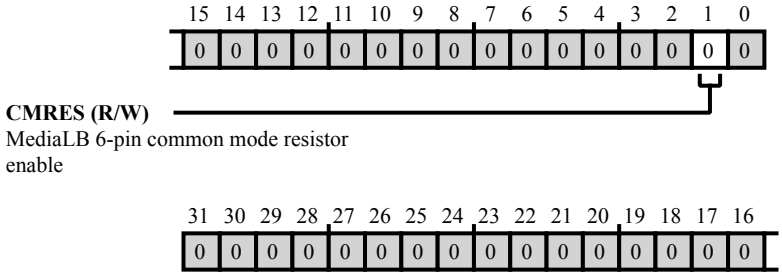


Figure 28-37: MLB_PCTL0 Register Diagram

Table 28-50: MLB_PCTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/W)	CMRES	MediaLB 6-pin common mode resistor enable. This signal is no longer used.

29 Two-Wire Interface (TWI)

The processor has a two-wire interface (TWI), that provides a simple exchange method of control data between multiple devices. The TWI module is compatible with the widely used I^2C bus standard. Additionally, the TWI module is fully compatible with serial camera control bus (SCCB) functionality for easier control of various CMOS camera sensor devices.

The TWI module offers the capabilities of simultaneous requester and completer operation and support for both 7-bit addressing and multimedia data arbitration. The TWI interface uses two pins for transferring clock (`TWI_SCL`) and data (`TWI_SDA`) and supports the protocol at speeds up to 400K bits/s. The TWI interface pins are compatible with 3.3V logic levels.

To preserve processor bandwidth, the TWI module can be set up with transfer-initiated interrupts to only service FIFO buffer data reads and writes. Protocol-related interrupts are optional. The TWI externally moves 8-bit data while maintaining compliance with the I^2C bus protocol.

NOTE: All TWI instances operate in the SCLK0 domain.

TWI Features

The TWI is fully compatible with the widely used I^2C bus standard.

The TWI controller includes the following features.

- Simultaneous requester and completer operation on multiple device systems
- Support for multi-controller bus arbitration
- 7-bit addressing
- 100K bits/second and 400K bits/second data rates
- General call address support
- Controller clock synchronization and support for clock low extension
- Separate multiple-byte receive and transmit FIFOs
- Low interrupt rate

- Individual override control of data and clock lines in the event of bus lock-up
- Input filter for spike suppression
- Serial camera control bus support as specified in the *OmniVision Serial Camera Control Bus (SCCB) Functional Specification*
- Programmable drive/tolerance of TWI pins. Refer to the PADS_PCFG0 register description in General-Purpose Ports (PORT) chapter for details.

TWI Functional Description

The TWI interface is a shift register that serially transmits and receives data bits. It moves data 1 bit at a time at the SCL rate, to and from other TWI devices. The SCL signal synchronizes the shifting and sampling of the data on the serial data pin.

ADSP-SC59x TWI Register List

The Two-Wire Interface controller TWI allows a device to interface to an inter-IC bus as specified by the Philips I^2C Bus Specification version 2.1, dated January 2000. A set of registers governs TWI operations. For more information on TWI functionality, see the TWI register descriptions.

Table 29-1: ADSP-SC59x TWI Register List

Name	Description
TWI_CLKDIV	SCL Clock Divider Register
TWI_CTL	Control Register
TWI_FIFOCTL	FIFO Control Register
TWI_FIFOSTAT	FIFO Status Register
TWI_IMSK	Interrupt Mask Register
TWI_ISTAT	Interrupt Status Register
TWI_MSTRADDR	Controller Mode Address Register
TWI_MSTRCTL	Controller Mode Registers
TWI_MSTRSTAT	Controller Mode Status Register
TWI_RXDATA16	Rx Data Double-Byte Register
TWI_RXDATA8	Rx Data Single-Byte Register
TWI_SLVADDR	Target Mode Address Register
TWI_SLVCTL	Target Mode Control Register
TWI_SLVSTAT	Target Mode Status Register
TWI_TXDATA16	Tx Data Double-Byte Register
TWI_TXDATA8	Tx Data Single-Byte Register

ADSP-SC59x TWI Interrupt List

Table 29-2: ADSP-SC59x TWI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
150	TWI0_DATA	TWI0 Data Interrupt	Level	
151	TWI1_DATA	TWI1 Data Interrupt	Level	
152	TWI2_DATA	TWI2 Data Interrupt	Level	
153	TWI3_DATA	TWI3 Data Interrupt	Level	
154	TWI4_DATA	TWI4 Data Interrupt	Level	
155	TWI5_DATA	TWI5 Data Interrupt	Level	

TWI Block Diagram

The *TWI Block Diagram* figure shows the basic blocks of the TWI interface.

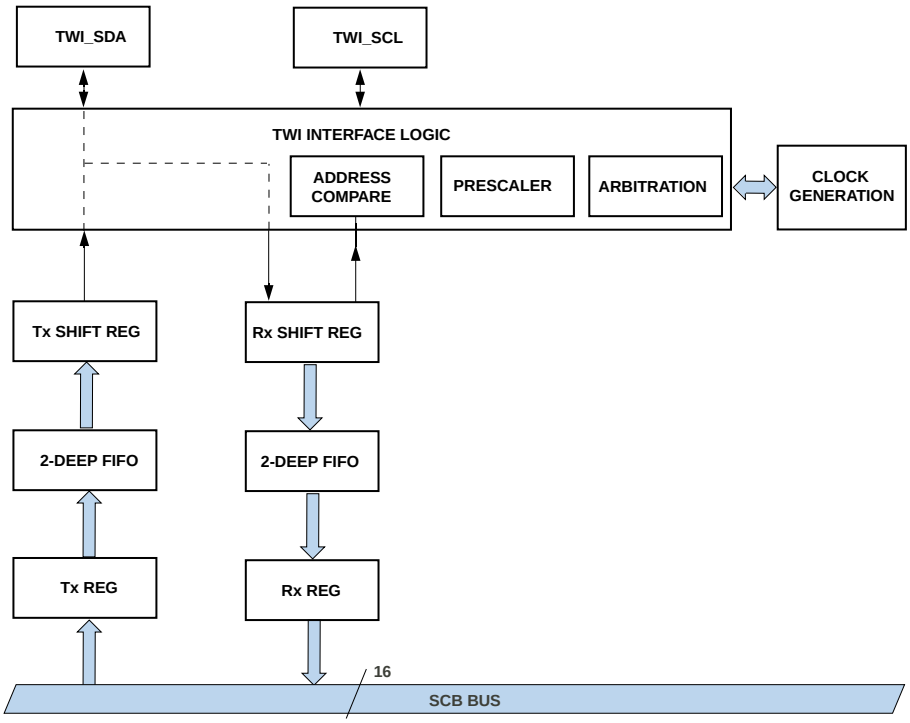


Figure 29-1: TWI Block Diagram

External Interface

The `TWI_SDA` (serial data) and `TWI_SCL` (serial clock) signals are open drain and require pull-up resistors. These bidirectional signals externally interface the TWI controller to the I^2C bus, and no other external connections or logic are necessary.

Serial Clock Signal (SCL)

The serial clock signal (TWI_SCL) is an input in completer mode. In requester mode, the TWI controller must set this signal to the desired frequency.

The TWI controller supports the standard mode of operation (up to 100 kHz) or fast mode (up to 400 kHz). The TWI control register (TWI_CTL) sets the TWI_CTL.PRESCALE value which sets the relationship between the system clock (SCLK0) and the internally timed events of the TWI controller. The internal time reference is derived from SCLK0 using a prescaled value. The prescale value is the number of SCLK0 periods used in the generation of one internal time reference. Set the value of prescale to create an internal time reference with a period of 10 MHz. It is represented as a 7-bit binary value as follows.

PRESCALE = f_{SCLK0}/10MHz

NOTE: It is not always possible to achieve 10-MHz accuracy. In such cases, it is safe to round up the PRESCALE value to the next highest integer. For example, if SCLK0 is 100 MHz, the PRESCALE value is calculated as 100 MHz/10 MHz = 10. A prescale value of 14 in this case ensures that all timing requirements are met.

During requester mode operation, the TWI module uses the TWI_CLKDIV register values to create the minimum TWI_CLKDIV.CLKHI and TWI_CLKDIV.CLKLO durations of the TWI_SCL signal. The TWI_CLKDIV.CLKHI field specifies the minimum number of 10-MHz time reference periods the TWI_SCL waits before a new clock low period begins, assuming a single requester. (The 10-MHz time reference periods are represented as an 8-bit binary value). The TWI uses the TWI_CLKDIV.CLKLO field to specify the minimum number of internal time reference periods (represented as an 8-bit binary value); the TWI_SCL signal is held low.

Serial clock frequencies can vary from 400 kHz to less than 20 kHz. The resolution of the clock generated is 1/10 MHz or 100 ns. The following equation describes the frequency.

CLKDIV = TWI_SCL period/10 MHz time reference.

For example, for an TWI_SCL of 400 kHz (period = 1/400 kHz = 2500 ns) and an internal time reference of 10 MHz (period = 100 ns), the following equation applies:

CLKDIV =2500 ns/100 ns = 25

Therefore, a TWI_SCL signal with a 30% duty cycle has TWI_CLKDIV.CLKLO=17 and TWI_CLKDIV.CLKHI=8. Adding TWI_CLKDIV.CLKLO and TWI_CLKDIV.CLKHI equals CLKDIV.

NOTE: The TWI_CLKDIV.CLKHI and TWI_CLKDIV.CLKLO fields are not intended to guarantee a certain frequency. Rather, they guarantee a certain minimum high and low duration for the TWI_SCL signal. Slew rate controls falling edges. The *RC* time constant governs the rising edges. The pull-up resistor and the TWI_SCL capacitance form the time constant. See the “Register Descriptions” section for more details.

Serial Data Signal (SDA)

The TWI transmits and receives serial data, depending on the direction of the transfer, on the bidirectional serial data signal (SDA).

Internal Interface

The peripheral bus interface supports the transfer of 16-bit wide data. The processor uses the interface in the support of register and FIFO buffer reads and writes. The TWI internal interface is comprised of the blocks described as follows.

Register block. Contains all control and status bits and reflects what can be written or read as outlined by the programming model. Each function block updates their corresponding status bits.

FIFO buffer. Configured as a 1-byte-wide, 2-deep transmit FIFO buffer and a 1-byte-wide, 2-deep receive FIFO buffer.

Transmit shift register. Serially shifts its data out externally off chip. The output can be controlled for generation of acknowledgments, or it can be manually overwritten.

Receive shift register. Receives its data serially from off chip. The receive shift register is 1 byte wide and data received can either be transferred to the FIFO buffer or used in an address comparison.

Address compare block. Supports address comparison in the event the TWI controller module is accessed as a completer.

Prescaler block. Must be programmed to generate a 10-MHz time reference relative to the system clock. The block uses this time base for filtering of data and timing events specified by the electrical data sheet (See the Philips specification). The block uses the time base to generate the `TWI_SCL` clock as well.

Clock generation module. Generates an external `TWI_SCL` clock when in requester mode. It includes the logic necessary for synchronization in a multi-requester clock configuration and clock stretching when configured in completer mode.

NOTE: The TWI does not support DMA based operation.

TWI Architectural Concepts

The TWI controller follows the transfer protocol of the Philips I^2C Bus specification version 2.1 dated January 2000.

NOTE: The TWI unit does not support DMA-based operation.

TWI Protocol

The ***Data Transfer*** figure shows a simple complete transfer.

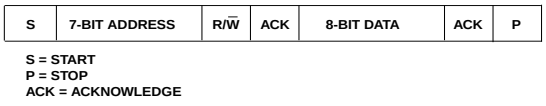


Figure 29-2: Data Transfer

The TWI controller register contents maps to a basic transfer. The ***Data Transfer with Bit Illustration*** figure details the same transfer from the ***Data Transfer*** figure noting the corresponding TWI controller bit names. In this

illustration, the TWI controller successfully transmits 1 byte of data. The completer has acknowledged both address and data.

Figure 29-3: Data Transfer with Bit Illustration

Clock Generation and Synchronization

The TWI controller implementation only issues a clock during requester mode operation and only at the time a transfer initiates. If arbitration for the bus is lost, the serial clock output immediately three-states. If multiple clocks attempt to drive the serial clock line, the TWI controller synchronizes its clock with the other remaining clocks. The ***Clock Synchronization*** figure shows this functionality.

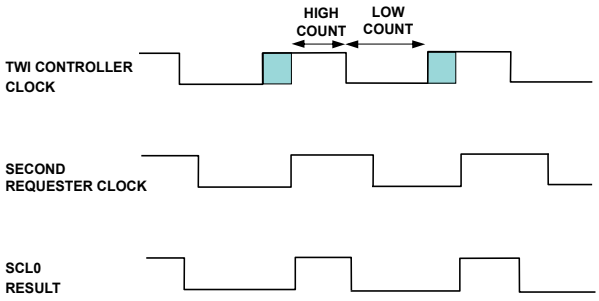


Figure 29-4: Clock Synchronization

The TWI controller serial clock (`TWI_SCL`) output follows these rules:

- Once the clock high (`TWI_CLKDIV.CLKHI`) count is complete, the serial clock output is driven low and the clock low (`TWI_CLKDIV.CLKLO`) count begins.
- Once the clock low count is complete, the serial clock line is three-stated. This state allows the external pull-up resistor to pull the `TWI_SCL` signal high. The clock synchronization logic enters a delay mode (shaded area) until the `TWI_SCL` signal is detected at logic 1 level. Now, the clock high count begins.

Bus Arbitration

The TWI controller initiates a requester mode transmission only when the bus is idle. Whenthe bus is idle and two requesters initiate a transfer, arbitration for the bus begins. The ***Bus Arbitration*** figure shows the arbitration.

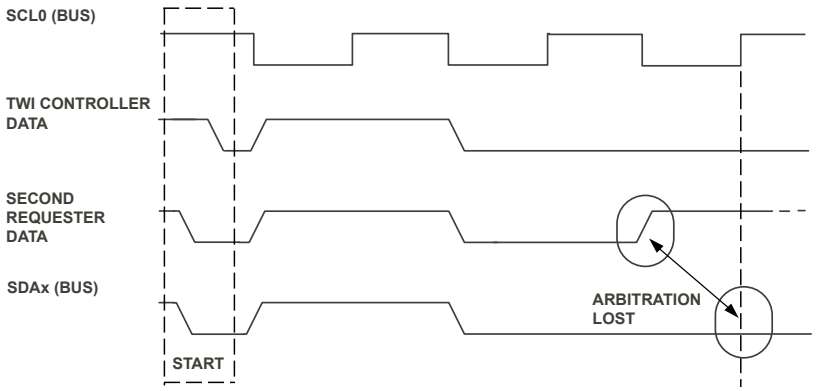


Figure 29-5: Bus Arbitration

The TWI controller monitors the serial data bus (SDA) while the `TWI_SCL` signal is high. If the `TWI_SDA` signal is determined to be an active logic 0 level while the data of the TWI controller is a logic 1 level, the TWI controller has lost arbitration. It stops generating the clock and data signals. Arbitration is not only performed at the serial clock edges, but also during the entire time the `TWI_SCL` signal is high.

Start and Stop Conditions

Start and stop conditions involve serial data transitions while the serial clock is a logic 1 level. The TWI controller generates and recognizes these transitions. Typically, start and stop conditions occur at the beginning and at the conclusion of a transmission, except repeated start combined transfers. The ***Start and Stop Conditions*** figure shows the transitions.

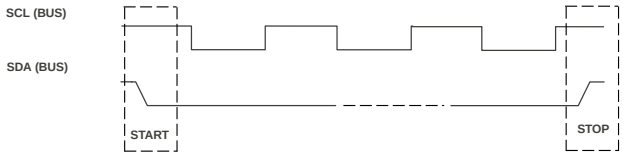


Figure 29-6: Start and Stop Conditions

The TWI special case start and stop conditions of the TWI controller include the following.

- Controller addressed as a completer-receiver. If the requester asserts a stop condition during the data phase of a transfer, the TWI controller concludes the transfer (`TWI_ISTAT.SCOMP`).
- Controller addressed as a completer-transmitter. If the requester asserts a stop condition during the data phase of a transfer, the TWI controller concludes the transfer (`TWI_ISTAT.SCOMP`) and indicates a completer transfer error (`TWI_ISTAT.SERR`).
- Controller as a requester-transmitter or requester-receiver. If the stop bit (`TWI_MSTRCTL.STOP`) is set during an active requester transfer, the TWI controller issues a stop condition as soon as possible avoiding any error conditions. The TWI controller operates as if data transfer count had been reached.

General Call Support

The TWI controller always decodes and acknowledges a general call address if:

- The TWI controller is enabled as a target
- General call is enabled

The TWI_SLVCTL.GEN bit configures general call addressing (0x00) only when the TWI controller is a target-receiver.

If the data associated with the transfer is (NAK) not acknowledged, the TWI_SLVCTL.NAK bit can be set. If the TWI controller issues a general call as a requester-transmitter, set the appropriate address (TWI_MSTRADDR register) and transfer direction (TWI_MSTRCTL.DIR bit) and load the transmit FIFO data.

NOTE: The byte following the general call address usually defines the completer's response to the call. The interpretation of the command in the second byte is based on the value of its LSB. For a TWI completer device, the bytes received after the general call address are considered data.

Fast Mode

Fast mode uses the same mechanics as the standard mode of operation. Fast mode affects electrical specifications and timing. When fast mode is enabled, (FAST) timing is modified to meet the following electrical requirements.

- Serial data rise times before arbitration evaluation (t_r)
- Stop condition set-up time from serial clock to serial data (t_{SUSTO})
- Bus free time between a stop and start condition (t_{BUF})

TWI Operating Modes

The TWI has two modes of operation: repeated start and clock stretching. The following sections describe the operating modes.

Repeated Start

A repeated start condition is the absence of a stop condition between two transfers. The two transfers can be of any direction type. Examples include a transmit followed by a receive, or a receive followed by a transmit. The following sections guide the programmer in developing a service routine.

Transmit Receive Repeated Start

The ***Repeated Start Followed by Data Receive*** figure shows a repeated start followed by a data receive sequence. The shading in the figure indicates that the completer has control of the bus.

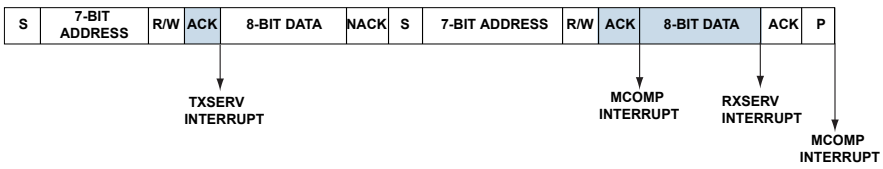


Figure 29-7: Repeated Start Followed by Data Receive

The tasks performed at each interrupt are:

- Transmit FIFO service (TWI_ISTAT.TXSERV) interrupt request. This interrupt is generated due to a FIFO access. Since this byte is the last of this transfer, the TWI uses the TWI_FIFOSTAT register to indicate that the transmit FIFO is empty. When read, TWI_MSTRCTL.DCNT bit field=0. Set the TWI_MSTRCTL.RSTART bit to indicate a repeated start and set the TWI_MSTRCTL.DIR bit if the following transfer is a data receive.
- Requester transfer complete (TWI_ISTAT.MCOMP) interrupt. This interrupt request is generated when all data transfers (TWI_MSTRCTL.DCNT bit field=0). If no errors occur, a start condition initiates. Clear the TWI_MSTRCTL.RSTART bit and program the TWI_MSTRCTL.DCNT bits with the desired number of bytes to receive.
- Receive FIFO service (TWI_ISTAT.RXSERV) interrupt. This interrupt request is generated due to the arrival of a byte in the receive FIFO. Simple data handling is the only requirement.
- Requester transfer complete (TWI_ISTAT.MCOMP) interrupt. This interrupt request is generated when the transfer completes.

Receive Transmit Repeated Start

The ***Repeated Start Data Receive Followed by Data Transmit*** figure illustrates a repeated start data receive followed by a data transmit sequence. The shading in the figure indicates that the completer has control of the bus.

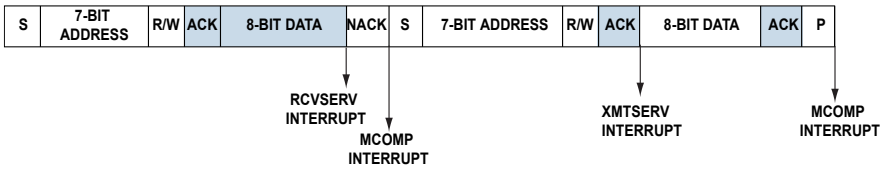


Figure 29-8: Repeated Start Data Receive Followed by Data Transmit

The tasks performed at each interrupt are:

- Receive FIFO service (TWI_ISTAT.RXSERV) interrupt. This interrupt request is generated due to the arrival of a data byte in the receive FIFO. Set the TWI_MSTRCTL.RSTART bit to indicate a repeated start and clear the TWI_MSTRCTL.DIR bit if the following transfer is a data transmit.
- Requester transfer complete (TWI_ISTAT.MCOMP) interrupt. This interrupt request has occurred due to the completion of the data receive transfer. If no errors occur, a start condition initiates. Clear the TWI_MSTRCTL.RSTART bit and program the TWI_MSTRCTL.DCNT bits with the desired number of bytes to transmit.
- Transmit FIFO service (TWI_ISTAT.TXSERV) interrupt. This interrupt request is generated due to a FIFO access. Simple data handling is the only requirement.
- Requester transfer complete (TWI_ISTAT.MCOMP) interrupt. This interrupt request is generated when the transfer completes.

NOTE: There is no timing constraint to meet the conditions—program the bits as required. Refer to Clock Stretching During Repeated Start section for more on how the controller stretches the clock during repeated start transfers.

Clock Stretching

Clock stretching is an added function of the TWI controller in requester mode operation. This behavior uses self-induced stretching of the I ^{2}C clock while waiting to service interrupts. Hardware initiates stretching automatically. No programming is necessary. The TWI controller as a requester supports three modes of clock stretching:

- Clock Stretching During FIFO Underflow
- Clock Stretching During FIFO Overflow
- Clock Stretching During Repeated Start

Clock Stretching During FIFO Underflow

During a requester mode transmit, an interrupt request occurs the instant the transmit FIFO becomes empty. The most recent byte begins transmission. If the `TWI_ISTAT.TXSERV` interrupt request is not serviced, the concluding acknowledge phase of the transfer stretches.

Stretching of the clock continues until new data bytes are written to the transmit FIFO (`TWI_TXDATA8` or `TWI_TXDATA16` registers). No other action is required to release the clock and continue the transmission. This behavior continues until the transmission completes (`TWI_MSTRCTL.DCNT`=0). The transmission concludes (`TWI_ISTAT.MCOMP`). The ***Clock Stretching during FIFO Underflow*** figure and table show the stretching.

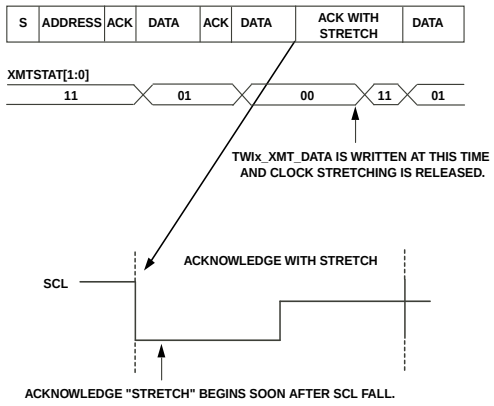


Figure 29-9: Clock Stretching during FIFO Underflow

TWI Controller	Processor
Interrupt: XMTSERV – Transmit FIFO buffer is empty	Acknowledge: Clear the interrupt request source bits. Write to the transmit FIFO buffer
...	...
Interrupt: MCOMP – requester transmit complete (DCNT= 0x00)	Acknowledge: Clear the interrupt request source bits

Clock Stretching During FIFO Overflow

During a requester mode receive operation, an interrupt occurs at the instant the receive FIFO becomes full. It is during the acknowledge phase of this received byte that clock stretching begins. The TWI module makes no attempt to initiate the reception of another byte. Stretching of the clock continues until the data bytes previously received are read from the receive FIFO buffer (`TWI_RXDATA8` or `TWI_RXDATA16` registers). No other action is required to release the clock and continue the reception of data. This behavior continues until the reception is complete (`TWI_MSTRCTL.DCNT=0`). Reception concludes (`TWI_ISTAT.MCOMP`). The ***Clock Stretching During FIFO Overflow*** figure and table show the clock stretching.

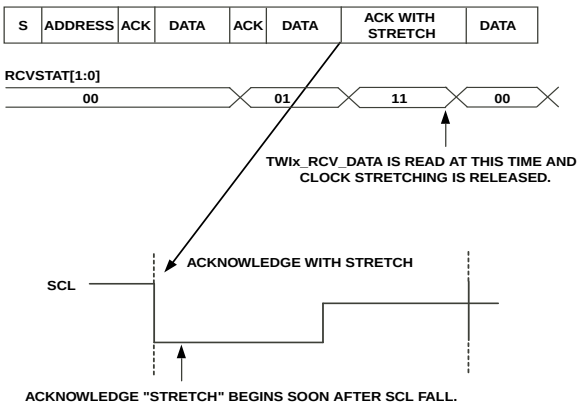


Figure 29-10: Clock Stretching During FIFO Overflow

TWI Controller	Processor
Interrupt: RCVSERV – Receive FIFO buffer is full	Acknowledge: Clear the interrupt request source bits. Read the receive FIFO buffer
...	...
Acknowledge: Clear the interrupt source bits	Interrupt: MCOMP – requester receive complete

Clock Stretching During Repeated Start

The repeated start feature in I^2C protocol requires a transition between two subsequent transfers. With the use of clock stretching, the task of managing transitions becomes simpler and common to all transfer types.

Once an initial TWI requester transfer completes (transmit or receive), the clock initiates a stretch during the repeated start phase between transfers. Concurrent with this event, the initial transfer generates a `TWI_ISTAT.MCOMP` interrupt to signify the initial transfer has completed (`TWI_MSTRCTL.DCNT=0`). This initial transfer is handled without any special bit setting sequences or timing.

The clock stretching logic described applies here. With no system-related timing constraints, the subsequent transfer (receive or transmit) is set up and activated. This sequence can repeat as many times as required to string a series of repeated start transfers together. The ***Clock Stretching during Repeated Start Condition*** figure and table show the clock stretching.

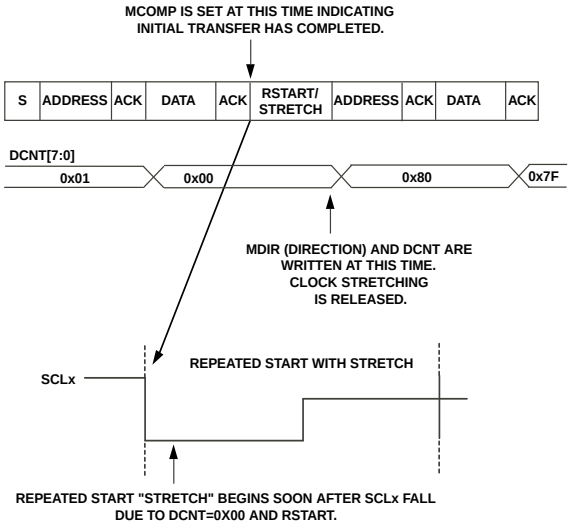


Figure 29-11: Clock Stretching during Repeated Start Condition

TWI Controller	Processor
Interrupt: MCOMP – Initial transmit has completed and DCNT = 0x00. Note: transfer in progress, RSTART previously set.	Acknowledge: Clear the interrupt request source bits. Write to TWIx_MASTER_CTL, setting MDIR (receive), clearing RSTART, and setting new DCNT value (nonzero).
Interrupt: RCVSERV – Receive FIFO is full	Acknowledge: Clear the interrupt request source bits. Read the receive FIFO buffer
...	...
Interrupt: MCOMP – requester receive complete	Acknowledge: Clear the interrupt request source bits

TWI Programming Model

The topics in this section provide information on the basic programming steps required to set up and run the two wire interface.

General Setup

General setup refers to register writes that are required for both completer mode and requester mode operations.

Perform general setup before setting either the requester or completer enable bits.

1. Program the `TWI_CTL.EN` bit to enable the TWI controller and set the prescale value (`TWI_CTL.PRESCALE` bit).
2. Program the prescale value to the binary representation of f_{SCLK0}/10 MHz. Round up all values to the next whole number.
3. Set the `TWI_CTL.EN` bit to enable the controller.

Once the TWI controller is enabled, a bus busy condition can be detected. This condition clears after t_{BUF} has expired, assuming no additional bus activity has been detected.

Completer Mode

When enabled, completer mode operation supports both receive and transmit data transfers.

It is not possible to enable only one data transfer direction and not acknowledge (NAK) the other. The following setup reflects this functionality.

1. Program the TWI_SLVADDR register. The TWI uses the appropriate 7 bits in determining a match during the address phase of the transfer.
2. Program the TWI_TXDATA8.VALUE or TWI_TXDATA16 registers. These values are the initial data values for transmission when the completer is addressed, and transmission is needed. This step is optional. If no data is written when the completer is addressed and transmission is needed, the serial clock (TWI_SCL) stretches. An interrupt is generated until data is written to the transmit FIFO.
3. Program the TWI_IMSK register. There are enable-bits associated with the desired interrupt sources. For example, programming the value 0x000F results in an interrupt request output to the processor, when the TWI module detects a valid address match. An interrupt request also occurs when a valid completer transfer completes or has an error, or a subsequent transfer has begun, and the previous transfer has not been serviced.
4. Program the TWI_SLVCTL register. This step prepares and enables completer mode operation. For example, programming the value 0x0005 enables completer mode operation and requires 7-bit addressing. It indicates that data in the transmit FIFO buffer is for completer mode transmission.

The ***Completer Mode Interaction*** table and ***TWI Completer Mode Program Flow*** diagram represent the interaction between the TWI controller and the processor using this example.

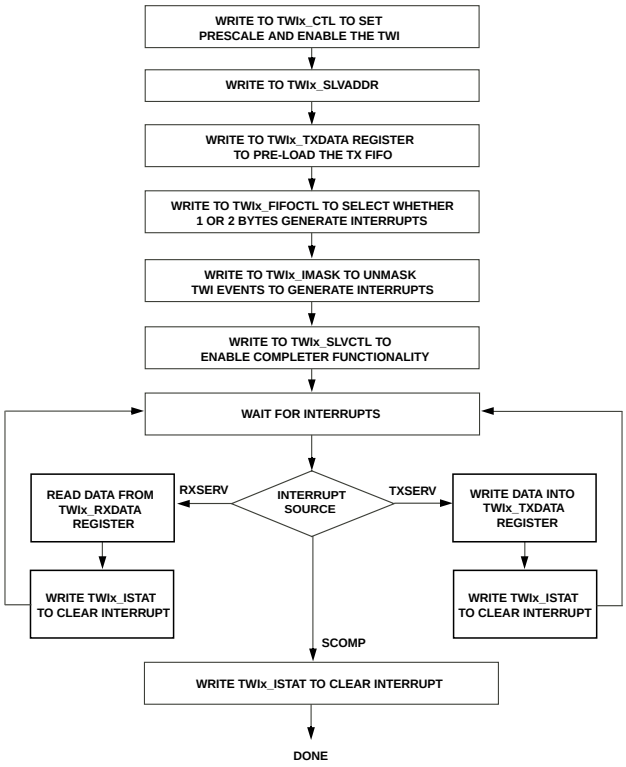


Figure 29-12: TWI Completer Mode Program Flow

Table 29-3: Completer Mode Interaction

TWI Controller	Processor
Interrupt: SINIT – completer transfer in progress.	Acknowledge: Clear the interrupt source bits.
Interrupt: RCVSERV – Receive buffer is full.	Acknowledge: Clear the interrupt source bits. Read TWIx_FIFO_STAT. Read the receive FIFO buffer.
...	...
Interrupt: SCOMP – completer transfer complete.	Acknowledge: Clear the interrupt source bits. Read the receive FIFO buffer.

Requester Mode Program Flow

The ***Requester Mode Program Flow*** figure shows the program for the TWI in requester mode.

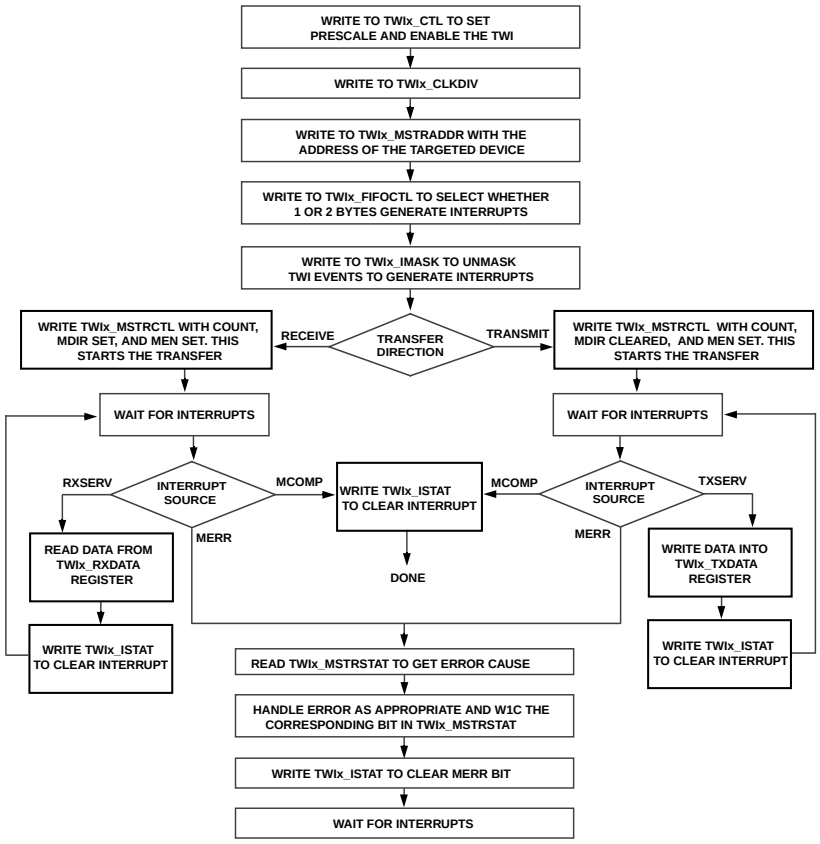


Figure 29-13: Requester Mode Program Flow

Requester Mode Clock Setup

Requester mode operation is set up and executed on a per-transfer basis.

An example of programming steps for a receive and for a transmit is given separately in following sections. The programming step for clock setup listed here is common to both transfer types.

1. Program the `TWI_CLKDIV` register to define the minimum high and minimum low duration for the clock.

The `TWI_CLKDIV.CLKHI` and `TWI_CLKDIV.CLKLO` fields do not guarantee a certain frequency. Rather, they guarantee a certain minimum high and low duration for `TWI_SCL`. The slew rate controls falling edges. The RC time constant formed by the pull-up resistor and the SCL capacitance govern rising edges. See the “Register Descriptions” section for more details.

Requester Mode Transmit

Follow these programming steps for a single requester mode transmission:

1. Program the `TWI_MSTRADDR` register. This step defines the address transmitted during the address phase of the transfer.

2. Program the TWI_TXDATA8 or TWI_TXDATA16 register. This step configures the initial data transmitted. It is an error to complete the address phase of the transfer and not have data available in the transmit FIFO buffer.
3. Program the TWI_FIFOCTL register. The programming indicates if the transmit FIFO buffer interrupt requests occur with each byte transmitted (8-bits) or with every 2 bytes transmitted (16-bits).
4. Program the TWI_IMSK register. This step enables the bits associated with the desired interrupt request sources. For example, programming the value 0x0030 results in an interrupt output to the processor when the requester transfer completes, and the requester transfer has an error.
5. Program the TWI_MSTRCTL register. This step prepares and enables requester mode operation. For example, programming the value 0x0201: enables requester mode operation, generates a 7-bit address, sets the direction to requester-transmit, uses standard mode timing, and transmits 8 data bytes before generating a stop condition.

The ***Requester Mode Transmit Setup Interaction*** table represents the interaction between the TWI controller and the processor using this example.

Table 29-4: Requester Mode Transmit Setup Interaction

TWI Controller	Processor
Interrupt: XMTSERV – Transmit buffer is empty	Acknowledge: Clear the interrupt request source bits and write to the transmit FIFO buffer
...	...
Interrupt: MCOMP – requester transfer complete	Acknowledge: Clear the interrupt request source bits

Requester Mode Receive

Follow these programming steps for a single requester mode receive.

1. Program the TWI_MSTRADDR register. This step defines the address transmitted during the address phase of the transfer.
2. Program the TWI_FIFOCTL register. This step indicates if the receive FIFO buffer interrupt requests occur with each byte received (8-bits) or with every 2 bytes received (16-bits).
3. Program the TWI_IMSK register. This step configures the enable bits associated with the desired interrupt sources. For example, programming the value 0x0030 results in an interrupt request output to the processor when the requester transfer completes, and the requester transfer has an error.
4. Program the TWI_MSTRCTL register. This step prepares and enables requester mode operation. For example, programming the value 0x0205: enables requester mode operation, generates a 7-bit address, sets the direction to requester-receive, uses standard mode timing, and receives 8 data bytes before generating a stop condition.

The ***Requester Mode Receive Setup Interaction*** table shows the interaction between the TWI controller and the processor using this example.

Table 29-5: Requester Mode Receive Setup Interaction

TWI Requester	Processor
Interrupt: RCVSERV – receive buffer is full	Acknowledge: Clear the interrupt request source bits. Read the receive FIFO buffer.
...	...
Interrupt: MCOMP – requester transfer complete	Acknowledge: Clear the interrupt request source bits and read the receive FIFO buffer

NOTE: After the `TWI_MSTRCTL.DCNT` bit decrements to zero, the TWI controller device sends a NAK to indicate to the completer transmitter to release the bus. This operation allows the requester to send the stop signal to terminate the transfer.

ADSP-SC59x TWI Register Descriptions

Two-Wire Interface (TWI) contains the following registers.

Table 29-6: ADSP-SC59x TWI Register List

Name	Description
`TWI_CLKDIV`	SCL Clock Divider Register
`TWI_CTL`	Control Register
`TWI_FIFOCTL`	FIFO Control Register
`TWI_FIFOSTAT`	FIFO Status Register
`TWI_IMSK`	Interrupt Mask Register
`TWI_ISTAT`	Interrupt Status Register
`TWI_MSTRADDR`	Controller Mode Address Register
`TWI_MSTRCTL`	Controller Mode Registers
`TWI_MSTRSTAT`	Controller Mode Status Register
`TWI_RXDATA16`	Rx Data Double-Byte Register
`TWI_RXDATA8`	Rx Data Single-Byte Register
`TWI_SLVADDR`	Target Mode Address Register
`TWI_SLVCTL`	Target Mode Control Register
`TWI_SLVSTAT`	Target Mode Status Register
`TWI_TXDATA16`	Tx Data Double-Byte Register
`TWI_TXDATA8`	Tx Data Single-Byte Register

SCL Clock Divider Register

During controller mode operation, the TWI_CLKDIV holds values, which the TWI uses to create the high and low durations of the serial clock (SCL). The clock signal SCL is an output in controller mode and an input in target mode. The values in the TWI_CLKDIV.CLKLO and TWI_CLKDIV.CLKHI fields add up to the CLKDIV value the following equation.

CLKDIV = TWI SCL period / 10 MHz time reference

Serial clock frequencies can vary from 400 KHz to less than 20 KHz. The resolution of the clock generated is 1/10 MHz or 100 ns. For example, for an SCL of 400 KHz (period = 1/400 KHz = 2500 ns) and an internal time reference of 10 MHz (period = 100 ns):

CLKDIV = 2500 ns / 100 ns = 25

For an SCL with a 30% duty cycle, use TWI_CLKDIV.CLKLO = 17 and TWI_CLKDIV.CLKHI = 8.

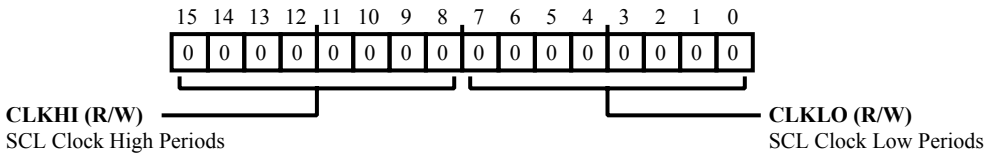


Figure 29-14: TWI_CLKDIV Register Diagram

Table 29-7: TWI_CLKDIV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	CLKHI	SCL Clock High Periods. The TWI_CLKDIV.CLKHI specifies the number of 10 MHz time reference periods the serial clock (SCL) waits before a new clock low period begins, assuming a single requester.
7:0 (R/W)	CLKLO	SCL Clock Low Periods. The TWI_CLKDIV.CLKLO specifies the number of internal time reference periods the serial clock (SCL) is held low.

Control Register

The TWI_CTL enables the TWI, establishes a relationship between the system clock (SCLK0) and the TWI controller's internally timed events, and enables SCCB compatibility.

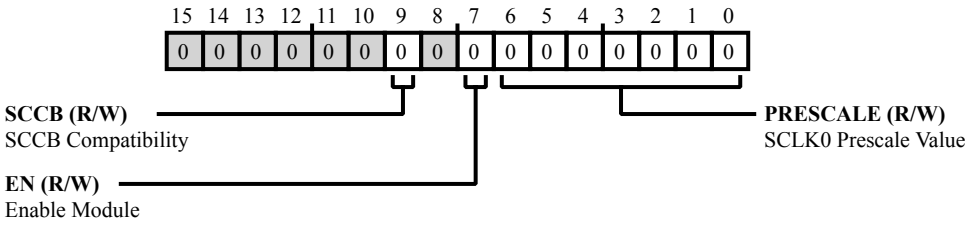


Figure 29-15: TWI_CTL Register Diagram

Table 29-8: TWI_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	SCCB	SCCB Compatibility. The TWI_CTL.SCCB enables SCCB compatible operation for the TWI. SCCB compatibility is an optional feature and should not be used in an I^2C bus system. When this feature is enabled, all completer asserted acknowledgement bits are ignored by this requester. This feature is valid only during transfers where the TWI is requesting an SCCB bus. Completer mode transfers should be avoided when this feature is enabled because the TWI controller always generates an acknowledge in completer mode.	
		0	Disable SCCB compatibility. When disabled, requester transfers are not SCCB compatible.
		1	Enable SCCB compatibility. When enabled, requester transfers are SCCB compatible. All completer-asserted acknowledgment bits are ignored by this requester.
7 (R/W)	EN	Enable Module. The TWI_CTL.EN enables TWI controller operation for either requester and/or completer mode of operation. It is recommended that this bit be set at the time TWI_CTL.PRESCALE is initialized and remain set. This method guarantees accurate operation of bus busy detection logic.	
		0	Disable
		1	Enable

Table 29-8: TWI_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	PRESCALE	SCLK0 Prescale Value. The `TWI_CTL.PRESCALE` holds the pre-scaled value for the TWI internal time reference. This reference is derived from SCLK0 according to the formula: `TWI_CTL.PRESCALE` = f_{SCLK0}/10MHz The `TWI_CTL.PRESCALE` specifies the number of system clock (SCLK0) periods used in the generation of one internal time reference. The value of `TWI_CTL.PRESCALE` must be set to create an internal time reference with a period of 10 MHz. It is represented as a 7-bit binary value.

FIFO Control Register

The TWI_FIFOCTL control bits affect only the FIFO and are not tied in any way with controller or target mode operation.

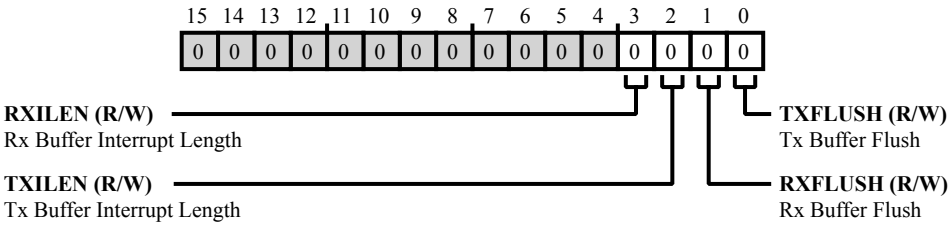


Figure 29-16: TWI_FIFOCTL Register Diagram

Table 29-9: TWI_FIFOCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXILEN	Rx Buffer Interrupt Length. The TWI_FIFOCTL.RXILEN determines the rate at which receive buffer interrupts are to be generated. Interrupts may be generated with each byte received or after two bytes are received. Interrupt status is available in TWI_FIFOSTAT.RXSTAT.	
		0	RXSERVI on 1 or 2 Bytes in FIFO
		1	RXSERVI on 2 Bytes in FIFO
2 (R/W)	TXILEN	Tx Buffer Interrupt Length. The TWI_FIFOCTL.TXILEN determines the rate at which transmit buffer interrupts are to be generated. Interrupts may be generated with each byte transmitted or after two bytes are transmitted. Interrupt status is available in TWI_FIFOSTAT.TXSTAT.	
		0	TXSERVI on 1 Byte of FIFO Empty
		1	TXSERVI on 2 Bytes of FIFO Empty
1 (R/W)	RXFLUSH	Rx Buffer Flush. The TWI_FIFOCTL.RXFLUSH directs the TWI to flush the contents of the receive buffer and update TWI_FIFOSTAT.RXSTAT to indicate the buffer is empty. This state is held until this bit is cleared. During an active receive, the receive buffer in this state responds to the receive logic as if it is full.	
		0	Normal Operation of Rx Buffer
		1	Flush Rx Buffer

Table 29-9: TWI_FIFOCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	TXFLUSH	Tx Buffer Flush. The `TWI_FIFOCTL.TXFLUSH` directs the TWI to flush the contents of the transmit buffer and update `TWI_FIFOSTAT.TXSTAT` to indicate the buffer is empty. This state is held until this bit is cleared. During an active transmit, the transmit buffer in this state responds to the transmit logic as if it is empty.	
		0	Normal Operation of Tx Buffer
		1	Flush Tx Buffer

FIFO Status Register

The TWI_FIFOSTAT fields indicate the state of the FIFO buffers' receive and transmit contents. The FIFO buffers do not discriminate between controller data and target data. By using the status and control bits provided, the FIFO can be managed to allow simultaneous controller and target operation.

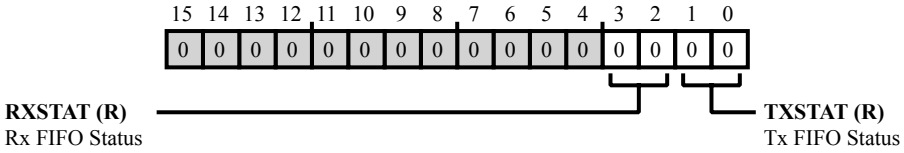


Figure 29-17: TWI_FIFOSTAT Register Diagram

Table 29-10: TWI_FIFOSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/NW)	RXSTAT	Rx FIFO Status. The read-only TWI_FIFOSTAT.RXSTAT indicates the number of valid data bytes in the receive FIFO buffer. The status is updated with each FIFO buffer read using the peripheral data bus or write access by the receive shift register. Simultaneous accesses are allowed.	
		0	Empty. The FIFO is empty.
		1	Contains 1 Byte. The FIFO contains one byte of data. A single byte peripheral read of the FIFO is allowed.
		2	Reserved
		3	Full. The FIFO is full and contains two bytes of data. Either a single or double byte peripheral read of the FIFO is allowed.
1:0 (R/NW)	TXSTAT	Tx FIFO Status. The read-only TWI_FIFOSTAT.TXSTAT field indicates the number of valid data bytes in the FIFO buffer. The status is updated with each FIFO buffer write using the peripheral data bus or read access by the transmit shift register. Simultaneous accesses are allowed.	
		0	Empty. The FIFO is empty. Either a single or double byte peripheral write of the FIFO is allowed.
		1	Contains 1 Byte. The FIFO contains one byte of data. A single byte peripheral write of the FIFO is allowed.
		2	Reserved
		3	Full. The FIFO is full and contains two bytes of data.

Interrupt Mask Register

The TWI_IMSK enables interrupt sources to assert the interrupt output. Each mask bit corresponds with one interrupt request source bit in TWI_ISTAT. Reading and writing TWI_IMSK does not affect the contents of the TWI_ISTAT.

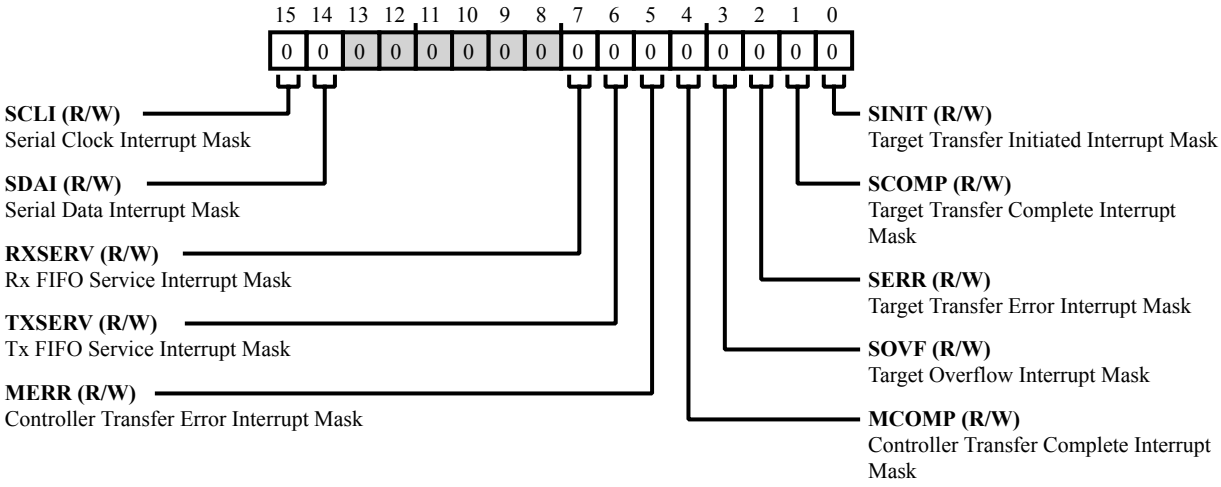


Figure 29-18: TWI_IMSK Register Diagram

Table 29-11: TWI_IMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	SCLI	Serial Clock Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
14 (R/W)	SDAI	Serial Data Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
7 (R/W)	RXSERV	Rx FIFO Service Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
6 (R/W)	TXSERV	Tx FIFO Service Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt

Table 29-11: TWI_IMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	MERR	Controller Transfer Error Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
4 (R/W)	MCOMP	Controller Transfer Complete Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
3 (R/W)	SOVF	Target Overflow Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
2 (R/W)	SERR	Target Transfer Error Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
1 (R/W)	SCOMP	Target Transfer Complete Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt
0 (R/W)	SINIT	Target Transfer Initiated Interrupt Mask.	
		0	Mask (Disable) Interrupt
		1	Unmask (Enable) Interrupt

Interrupt Status Register

The TWI_ISTAT contains information about functional areas requiring servicing. Many of the bits serve as an indicator to further read and service various status registers. After servicing the interrupt source associated with a bit, the user must clear that interrupt source bit by writing a 1 to it.

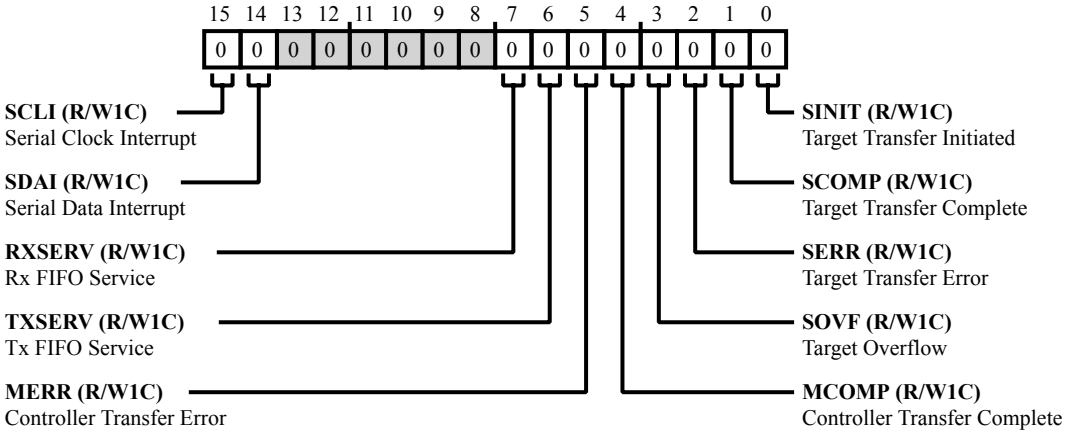


Figure 29-19: TWI_ISTAT Register Diagram

Table 29-12: TWI_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	SCLI	Serial Clock Interrupt. If the TWI is enabled (TWI_CTL.EN), SCLI is set on a high-to-low transition of the serial clock pin (SCLx). Normally, this bit is not required for I^2C bus transfers. It will be initially set on an I^2C transfer and does not require clearing.	
		0	No Interrupt. No transition was detected on the SCLx pin.
		1	Interrupt Detected. A high-to-low transition was detected on the SCLx pin. This bit is W1C.
14 (R/W1C)	SDAI	Serial Data Interrupt. If the TWI is enabled (TWI_CTL.EN), SDAI is set on a high-to-low transition of the serial data pin (SDAx). Normally, this bit is not required for I^2C bus transfers. It will be initially set on an I^2C transfer and does not require clearing.	
		0	No Interrupt. No transition was detected on the SDAx pin.
		1	Interrupt Detected. A high-to-low transition was detected on the SDAx pin. This bit is W1C.

Table 29-12: TWI_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W1C)	RXSERV	Rx FIFO Service. If TWI_FIFOCTL.RXILEN =0, the TWI_ISTAT.RXSERV is set each time the TWI_FIFOSTAT.RXSTAT field is updated to either 01 or 11. If TWI_FIFOCTL.RXILEN =1, the TWI_ISTAT.RXSERV is set each time TWI_FIFOSTAT.RXSTAT is updated to 11.	
		0	No Interrupt. The FIFO does not require servicing, or the TWI_FIFOSTAT.RXSTAT field has not changed since this bit was last cleared.
		1	Interrupt Detected. The receive FIFO buffer has one or two 8-bit words of data available to be read.
6 (R/W1C)	TXSERV	Tx FIFO Service. If TWI_FIFOCTL.TXILEN =0, the TWI_ISTAT.TXSERV is set each time the TWI_FIFOSTAT.TXSTAT field is updated to either 01 or 00. If TWI_FIFOCTL.TXILEN =1, the TWI_ISTAT.TXSERV is set each time TWI_FIFOSTAT.TXSTAT is updated to 00.	
		0	No Interrupt. FIFO does not require servicing, or the TWI_FIFOSTAT.TXSTAT field has not changed since this bit was last cleared.
		1	Interrupt Detected. The transmit FIFO buffer has one or two 8-bit locations available to be written.
5 (R/W1C)	MERR	Controller Transfer Error. The TWI_ISTAT.MERR indicates that a requester error has occurred. The conditions surrounding the error are indicated by the requester status register (TWI_MSTRSTAT).	
		0	No Interrupt
		1	Interrupt Detected
4 (R/W1C)	MCOMP	Controller Transfer Complete. The TWI_ISTAT.MCOMP indicates that the initiated requester transfer has completed. In the absence of a repeat start, the bus has been released.	
		0	No Interrupt
		1	Interrupt Detected

Table 29-12: TWI_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W1C)	SOVF	Target Overflow. The `TWI_ISTAT.SOVF` indicates that the `TWI_ISTAT.SCOMP` bit was set at the time a subsequent transfer has acknowledged an address phase. The transfer continues, however, it may be difficult to delineate data of one transfer from another.	
		0	No Interrupt
		1	Interrupt Detected
2 (R/W1C)	SERR	Target Transfer Error. The `TWI_ISTAT.SERR` indicates that a completer error has occurred. A restart or stop condition has occurred during the data receive phase of a transfer.	
		0	No Interrupt
		1	Interrupt Detected
1 (R/W1C)	SCOMP	Target Transfer Complete. The `TWI_ISTAT.SCOMP` indicates that the transfer is complete and either a stop, or a restart was detected.	
		0	No Interrupt
		1	Interrupt Detected
0 (R/W1C)	SINIT	Target Transfer Initiated. The `TWI_ISTAT.SINIT` indicates whether or not a completer transfer is in progress.	
		0	No Interrupt. A transfer is not in progress, or an address match has not occurred since the last time this bit was cleared.
		1	Interrupt Detected. The completer has detected an address match, and a transfer has been initiated.

Controller Mode Address Register

During the addressing phase of a transfer, the TWI controller, with its controller enabled, transmits the contents of TWI_MSTRADDR. When programming this register, omit the read/write bit. That is, only the upper 7 bits that make up the target address should be written to this register. For example, if the target address is b#1010000X, where X is the read/write bit, the TWI_MSTRADDR is programmed with b#1010000, which corresponds to 0x50. When sending out the address on the bus, the TWI controller appends the read/write bit as appropriate based on the state of the TWI_MSTRCTL.DIR bit.

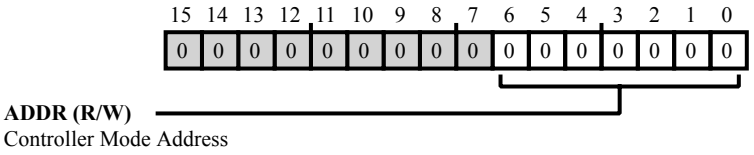


Figure 29-20: TWI_MSTRADDR Register Diagram

Table 29-13: TWI_MSTRADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	ADDR	Controller Mode Address.

Controller Mode Registers

The TWI_MSTRCTL controls the logic associated with controller mode operation. Bits in this register do not affect target mode operation and should not be modified to control target mode functionality.

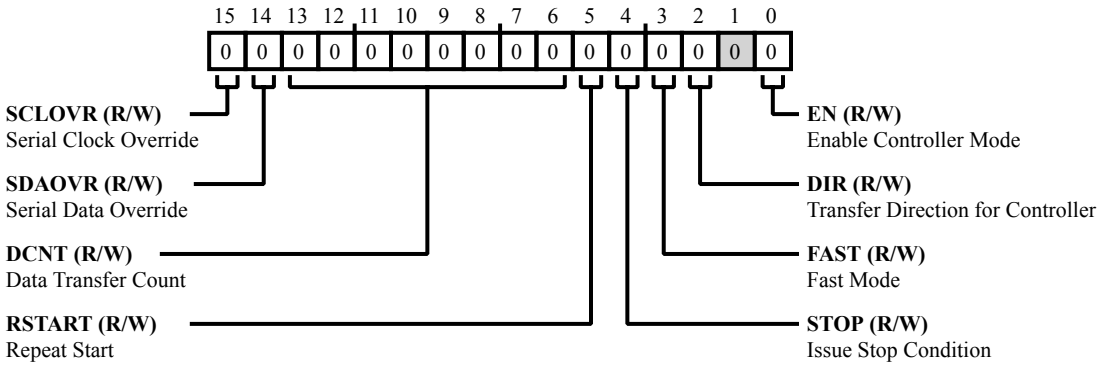


Figure 29-21: TWI_MSTRCTL Register Diagram

Table 29-14: TWI_MSTRCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	SCLOVR	Serial Clock Override. The TWI_MSTRCTL.SCLOVR provides direct control of the serial clock line when required. Normal requester and completer mode operation should not require override operation. When TWI_MSTRCTL.SCLOVR is set, the TWI overrides normal serial clock output, driving it to an active 0 level and overriding all other logic. This state is held until this bit is cleared. When TWI_MSTRCTL.SCLOVR is cleared, the TWI permits normal serial clock operation under the control of requester mode clock generation and completer mode clock stretching logic.	
		0	Permit Normal SCL Operation
		1	Override Normal SCL Operation
14 (R/W)	SDAOVR	Serial Data Override. The TWI_MSTRCTL.SDAOVR provides direct control of the serial data line when required. Normal requester and completer mode operation should not require override operation. When TWI_MSTRCTL.SDAOVR is set, the TWI overrides normal serial data operation under the control of the transmit shift register and acknowledge logic, driving serial data output to an active 0 level and overriding all other logic. This state is held until this bit is cleared. When TWI_MSTRCTL.SDAOVR is cleared, the TWI permits normal serial data operation.	
		0	Permit Normal SDA Operation
		1	Override Normal SDA Operation

Table 29-14: TWI_MSTRCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13:6 (R/W)	DCNT	Data Transfer Count. The TWI_MSTRCTL.DCNT indicates the number of data bytes to transfer. As each data word is transferred, the TWI decrements this counter. When TWI_MSTRCTL.DCNT decrements to 0, a stop condition is generated. Setting TWI_MSTRCTL.DCNT to 0xFF disables the counter. In this transfer mode, data continues to be transferred until it is concluded by setting the TWI_MSTRCTL.STOP bit. In the event a requester transmit is aborted due to a completer data NAK, the value of TWI_MSTRCTL.DCNT equals the number of bytes not sent. The byte which was NAK'ed by the completer is counted as a sent byte.	
5 (R/W)	RSTART	Repeat Start. The TWI_MSTRCTL.RSTART enables the TWI to issue a repeat start condition at the conclusion of the current transfer (TWI_MSTRCTL.DCNT =0) and begin the next transfer. The current transfer concludes with updates to the appropriate status and interrupt bits. If errors occurred during the previous transfer, a repeat start does not occur. In the absence of any errors, requester enable (TWI_MSTRCTL.EN) does not self clear on a repeat start.	
		0	Disable Repeat Start
		1	Enable Repeat Start
4 (R/W)	STOP	Issue Stop Condition. The TWI_MSTRCTL.STOP directs the TWI to issue a stop condition. The transfer concludes as soon as possible avoiding any error conditions (as if data transfer count had been reached). At that time, the TWI_IMSK is updated along with any associated status bits.	
		0	Permit Normal Operation
		1	Issue Stop
3 (R/W)	FAST	Fast Mode. The TWI_MSTRCTL.FAST selects whether the TWI operates in fast mode or standard mode. In fast mode, the TWI uses timing specifications for transfers at up to 400K bits/s. In standard mode, the TWI uses timing specifications for transfers at up to 100K bits/s.	
		0	Select Standard Mode
		1	Select Fast Mode
2 (R/W)	DIR	Transfer Direction for Controller. The TWI_MSTRCTL.DIR selects the transfer direction for the TWI as requester initiated receive or transmit.	
		0	Requester Transmit
		1	Requester Receive

Table 29-14: TWI_MSTRCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Enable Controller Mode. The `TWI_MSTRCTL.EN` enables requester mode functionality. A start condition is generated if the bus is idle. This bit self clears at the completion of a transfer (after `TWI_MSTRCTL.DCNT` decrements to zero), including transfers terminated due to errors. If disabled (=0) during operation, the transfer is aborted, and all logic associated with requester mode transfers are reset. Serial data and serial clock (SDA, SCL) are no longer driven. Write-1-to-clear status bits are not affected.	
		0	Disable
		1	Enable

Controller Mode Status Register

The TWI_MSTRSTAT holds information during controller mode transfers and at their conclusion. Generally, controller mode status bits are not directly associated with the generation of interrupt requests, but these bits offer information on the current transfer. Target mode operation does not affect controller mode status bits.

Note that while TWI_MSTRSTAT.SCLSEN is set (this condition could be due to having no pull-up resistor on TWI_SCL or another agent is driving TWI_SCL low), the acknowledge bits (TWI_MSTRSTAT.ANAK and TWI_MSTRSTAT.DNAK) do not update. This result occurs because the acknowledge conditions are sampled during the high phase of TWI_SCL.

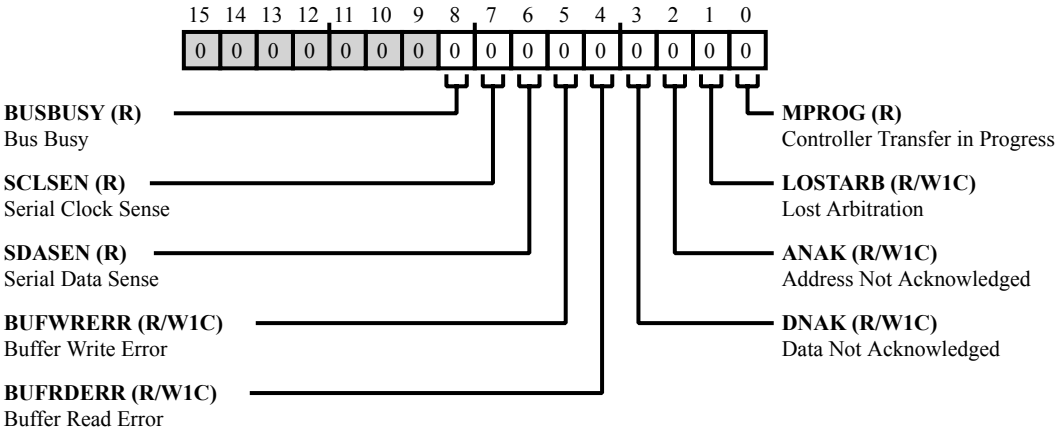


Figure 29-22: TWI_MSTRSTAT Register Diagram

Table 29-15: TWI_MSTRSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	BUSBUSY	Bus Busy. The TWI_MSTRSTAT.BUSBUSY indicates whether the bus is currently busy or free. This indication is not limited to only this device but is for all devices. On a start condition, the setting of the register value is delayed due to the input filtering. On a stop condition the clearing of the register value occurs after t_{BUF}.	
		0	Bus Free. The bus is free. The clock and data bus signals have been inactive for the appropriate bus free time.
		1	Bus Busy. The bus is busy. Clock or data activity has been detected.

Table 29-15: TWI_MSTRSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	SCLSEN	Serial Clock Sense. The `TWI_MSTRSTAT.SCLSEN` indicates the active or inactive state of the serial clock. Use this status bit when direct sensing of the serial clock line is required. The register value is delayed due to the input filter (nominally 50 ns). Normal requester and completer mode operation should not require this feature.	
		0	SCL Inactive "One". An inactive "one" is being sensed on the serial clock.
		1	SCL Active "Zero". An active "zero" is being sensed on the serial clock. The source of the active driver is not known and can be internal or external.
6 (R/NW)	SDASEN	Serial Data Sense. The `TWI_MSTRSTAT.SDASEN` indicates the active or inactive status of the serial data. Use this status bit when direct sensing of the serial data line is required. The register value is delayed due to the input filter (nominally 50 ns). Normal requester and completer mode operation should not require this feature.	
		0	SDA Inactive "One". An inactive "one" is currently being sensed on the serial data line.
		1	SDA Active "Zero". An active "zero" is currently being sensed on the serial data line. The source of the active driver is not known and can be internal or external.
5 (R/W1C)	BUFWRERR	Buffer Write Error. The `TWI_MSTRSTAT.BUFWRERR` indicates whether the current requester transfer was aborted due to a receive buffer write error. The receive buffer and receive shift register were both full at the same time. This bit is W1C.	
		0	No Status
		1	Buffer Write Error
4 (R/W1C)	BUFRDERR	Buffer Read Error. The `TWI_MSTRSTAT.BUFRDERR` indicates whether the current requester transfer was aborted due to a transmit buffer read error. The error occurs if the buffer is empty when data is required by the transmit shift register. This bit is W1C.	
		0	No Status
		1	Buffer Read Error

Table 29-15: TWI_MSTRSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W1C)	DNAK	Data Not Acknowledged. The `TWI_MSTRSTAT.DNAK` indicates whether the current requester transfer was aborted due to the detection of a NAK during data transmission. This bit is W1C.	
		0	No Status
		1	Data NAK
2 (R/W1C)	ANAK	Address Not Acknowledged. The `TWI_MSTRSTAT.ANAK` indicates whether the current requester transfer was aborted due to the detection of a NAK during the address phase of the transfer. This bit is W1C.	
		0	No Status
		1	Address NAK
1 (R/W1C)	LOSTARB	Lost Arbitration. The `TWI_MSTRSTAT.LOSTARB` indicates whether the current transfer was aborted due to the loss of arbitration with another requester. This bit is W1C.	
		0	No Status
		1	Lost Arbitration
0 (R/NW)	MPROG	Controller Transfer in Progress. The `TWI_MSTRSTAT.MPROG` indicates whether or not a requester transfer is in progress. If clear (`TWI_MSTRSTAT.MPROG` =0), currently no transfer is taking place. This can occur after a transfer is complete or while an enabled requester is waiting for an idle bus.	
		0	No Status
		1	Requester Transfer in Progress

Rx Data Double-Byte Register

The TWI_RXDATA16 holds a 16-bit data value read from the FIFO buffer. To reduce interrupt output rates and peripheral bus access times, a double byte receive data access can be performed. Two data bytes can be read, effectively emptying the receive FIFO buffer with a single access.

The data is read in little endian byte order, where byte 0 is the first byte received and byte 1 is the second byte received. With each access, the receive status (TWI_FIFOSTAT.RXSTAT) field is updated to indicate it is empty. If an access is performed while the FIFO buffer is not full, the read data is unknown and the existing FIFO buffer data and its status remains unchanged.

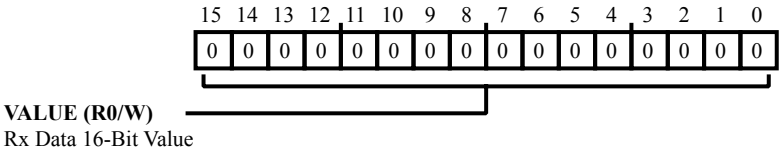


Figure 29-23: TWI_RXDATA16 Register Diagram

Table 29-16: TWI_RXDATA16 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W)	VALUE	Rx Data 16-Bit Value.

Rx Data Single-Byte Register

The TWI_RXDATA8 holds an 8-bit data value read from the FIFO buffer. Receive data is read from the corresponding receive buffer in a first-in first-out order. Although peripheral bus reads are 16 bits, a read access to TWI_RXDATA8 accesses only one transmit data byte from the FIFO buffer. With each access, the receive status (TWI_FIFOSTAT.RXSTAT) field is updated. If an access is performed while the FIFO buffer is empty, the data is unknown and the FIFO buffer status remains indicating it is empty.

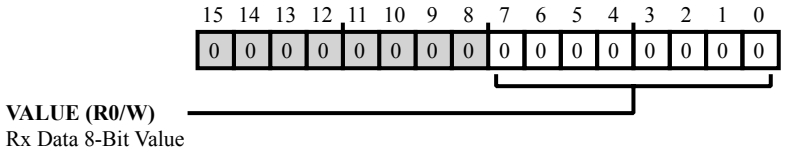


Figure 29-24: TWI_RXDATA8 Register Diagram

Table 29-17: TWI_RXDATA8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R0/W)	VALUE	Rx Data 8-Bit Value.

Target Mode Address Register

The TWI_SLVADDR holds the target mode address, which is the valid address to which the target-enabled TWI controller responds. The TWI controller compares this value with the received address during the addressing phase of a transfer.

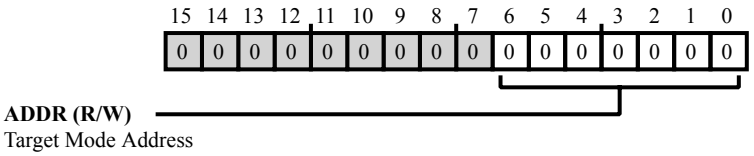


Figure 29-25: TWI_SLVADDR Register Diagram

Table 29-18: TWI_SLVADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	ADDR	Target Mode Address.

Target Mode Control Register

The TWI_SLVCTL controls the logic associated with target mode operation. Settings in this register do not affect controller mode operation and should not be modified to manage controller mode functionality.

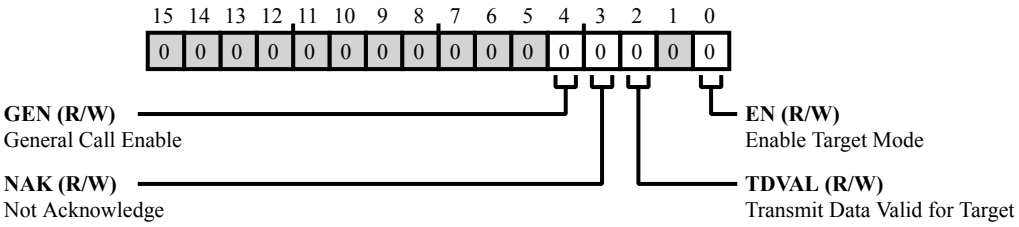


Figure 29-26: TWI_SLVCTL Register Diagram

Table 29-19: TWI_SLVCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	GEN	General Call Enable. The TWI_SLVCTL.GEN enables general call address matching. When enabled, a general call completer receive transfer is accepted. All status and interrupt source bits associated with transfers are updated. Note that general call address detection is available only when completer mode is enabled.	
		0	Disable General Call Matching
		1	Enable General Call Matching
3 (R/W)	NAK	Not Acknowledge. The TWI_SLVCTL.NAK directs the TWI to generate a NAK (if set) or an ACK (if cleared) at the conclusion of data transfer for completer receive. For NAK, the completer is still considered to be addressed at the conclusion of transfer.	
		0	Generate ACK
		1	Generate NAK
2 (R/W)	TDVAL	Transmit Data Valid for Target. The TWI_SLVCTL.TDVAL selects whether the data in the transmit FIFO is available (valid) for completer transmission (TWI_SLVCTL.TDVAL set). If the FIFO data is not available (invalid) for completer transmission (TWI_SLVCTL.TDVAL cleared), the data in the transmit FIFO is for requester mode transmits, and the data is not allowed to be used during a completer transmit; the transmit FIFO is treated as if it is empty.	
		0	Data Invalid for Completer Tx
		1	Data Valid for Completer Tx

Table 29-19: TWI_SLVCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	Enable Target Mode. The `TWI_SLVCTL.EN` enables completer operation. Enabling completer and requester modes of operation concurrently is allowed. If disabled, no attempt is made to identify a valid address. If `TWI_SLVCTL.EN` is cleared during a valid transfer, clock stretching ceases, the serial data line is released, and the current byte is not acknowledged.	
		0	Disable
		1	Enable

Target Mode Status Register

During and at the conclusion of register target mode transfers, the TWI_SLVSTAT holds information on the current transfer. Generally target mode status bits are not associated with the generation of interrupt requests. Controller mode operation does not affect target mode status bits.

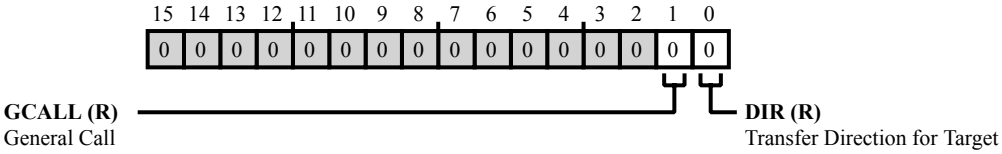


Figure 29-27: TWI_SLVSTAT Register Diagram

Table 29-20: TWI_SLVSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	GCALL	General Call. The TWI_SLVSTAT.GCALL indicates whether or not--at the time of addressing--the address was determined to be a general call. This bit self clears if completer mode is disabled (TWI_SLVCTL.EN =0).	
		0	Not a General Call Address
		1	General Call Address
0 (R/NW)	DIR	Transfer Direction for Target. The TWI_SLVSTAT.DIR indicates whether--at the time of addressing--the transfer direction was determined to be completer transmit or receive. This bit self clears if completer mode is disabled (TWI_SLVCTL.EN =0).	
		0	Completer Receive
		1	Completer Transmit

Tx Data Double-Byte Register

The TWI_TXDATA16 register holds a 16-bit data value written into the FIFO buffer. To reduce interrupt latency output rates and peripheral bus access times, a double byte transfer data access can be done. Two data bytes can be written, effectively filling the transmit FIFO buffer with a single access.

The data is written in little endian byte order, where byte 0 is the first byte to be transferred and byte 1 is the second byte to be transferred. With each access, the transmit status (TWI_FIFOSTAT.TXSTAT) field is updated. If an access is performed while the FIFO buffer is not empty, the write is ignored and the existing FIFO buffer data and its status remains unchanged. This register when read back returns zero.

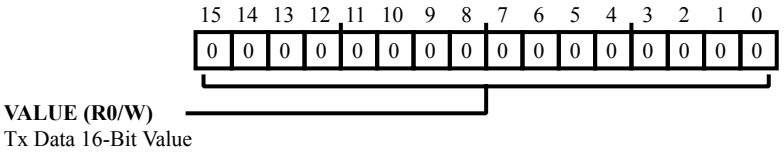


Figure 29-28: TWI_TXDATA16 Register Diagram

Table 29-21: TWI_TXDATA16 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R0/W)	VALUE	Tx Data 16-Bit Value.

Tx Data Single-Byte Register

The TWI_TXDATA8 register holds an 8-bit data value written into the FIFO buffer. Transmit data is entered into the corresponding transmit buffer in a first-in first-out order. For 16-bit peripheral bus writes, a write access to this register adds only one transmit data byte to the FIFO buffer. With each access, the transmit status (TWI_FIFOSTAT.TXSTAT) field is updated. If an access is performed while the FIFO buffer is full, the write is ignored and the existing FIFO buffer data and its status remains unchanged. This register returns zero when read back.

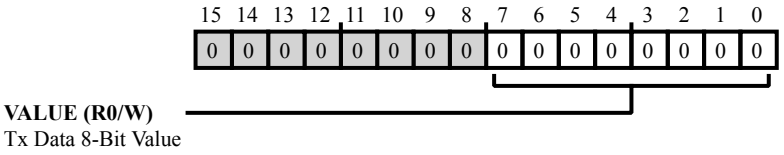


Figure 29-29: TWI_TXDATA8 Register Diagram

Table 29-22: TWI_TXDATA8 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R0/W)	VALUE	Tx Data 8-Bit Value.

30 Ethernet Media Access Controller (EMAC)

The EMAC peripheral in the processor enables network connectivity to applications through an Ethernet interface.

The module is fully compliant to the following standards:

- IEEE 802.3-2015 for Physical Layer Specifications.
- IEEE 1588-2008 for Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.
- IEEE 802.1AS-2011 and 802.1-Qav-2009 for Forwarding and Queuing Enhancements for Time-Sensitive Streams or Audio Video (AV) traffic.
- IEEE 802.1AS-Rev/D4.0 for Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks.
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE).
- IEEE 802.1Qbv-2015, 802.1Qbu-2016, and 802.1AS-Rev D5.0 for Time-Sensitive Networking (TSN).
- IEEE 802.1Qaz-2011 and 802.1Qbb-2011 for Data Center Bridging (DCB).
- Reduced Gigabit Media Independent Interface (RGMII), Revision 2.6, HP/Marvell.
- Reduced Media Independent Interface Specification, Revision 1.2, RMII Consortium.

NOTE: Copyright © 2020 Synopsys, Inc.; portions of this chapter are included with permission from Synopsys, Inc.

NOTE: References to MII in this document are applicable to both MII and RMII; references to GMII are applicable to RGMII.

EMAC Features

The EMAC features include the following:

MAC Tx and Rx Features

- 10/100/1000 Mbps data transfer rates with external PHY interfaced through RGMII/RMII/MII

- Support for both full-duplex and half-duplex operations
- CSMA/CD protocol for half-duplex operation
- IEEE 802.3x flow control for full-duplex and using backpressure support for half-duplex

MAC Transmit Features

- Automatic CRC and pad generation controllable on a per-frame basis
- Programmable packet length to support Standard or Jumbo Ethernet packets with up to 16 KB of size
- Programmable Inter-Frame Gap (IFG)
- Source address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted frames with per-frame control
- Insertion, replacement, or deletion of up to two VLAN tags
- Option to transmit packets with reduced preamble size in full-duplex mode
- Insert, replace, or delete queue/channel based VLAN tags
- Frame preemption for MAC Tx

MAC Receive Features

- Automatic pad and CRC stripping options
- Preamble and SFD deletion
- Programmable watchdog timeout limit
- Flexible address filtering modes
- Additional packet filtering (VLAN tag based, layer 3, layer 4)
- IEEE 802.1Q VLAN tag detection
- Frame preemption for MAC Rx

MAC Transaction Layer (MTL) Features

- Eight queues each on the transmit and receive path
- Programmable threshold capability for each queue
- Programmable burst length

MTL Transmit Features

- Store-and-forward mechanism or threshold mode (cut-through) for transmission to the MAC

- Programmable queue size in configurations with multiple queues. Each queue size can be programmed in terms of 256 bytes
- Automatic retransmission of collision packets in half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and underrun conditions with appropriate status
- Option to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum
- Programmable interrupt options for different operational conditions
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Statistics related to bandwidth consumption by each queue of up to 16 blocks over a 125 us period
- Optional packet-level control for:
    - VLAN tag insertion or replacement
    - Ethernet source address insertion
    - Layer 3/layer 4 checksum insertion control
    - One-step time stamp
    - Time stamp control
    - CRC and pad control
- Following scheduling algorithms in configurations with multiple queues:
    - Weighted Round Robin (WRR)
    - (When Data Center Bridging is enabled) Deficit Weighted Round Robin (DWRR)
    - (When Data Center Bridging is enabled) Weighted Fair Queuing (WFQ)
    - Strict Priority (SP)
    - (When Audio-Video Bridging is enabled) Credit-based Shaper (CBS)
    - (When TSN is enabled), Enhancement to Scheduled Traffic (EST)
    - (When TSN is enabled), Time Based Scheduling (TBS)Option to support dropping of Tx status to improve the transmit throughput

MTL Receive Features

- Programmable Rx queue threshold in threshold (or cut through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets

- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of pause packet control or backpressure signal to the MAC based on the Rx queue fill level
- Arbitration among queues when multiple queues are present. The following arbitration schemes are supported:
  - Weighted Round Robin (WRR)
  - Weighted Strict priority (WSP)
  - Strict Priority (SP)
- Option to replicate received multicast packets for transfer by multiple Rx DMA channel
- Option to have a programmable lookup table based flexible parser for filtering and steering the Rx packets
- Address Recognition Protocol (ARP) offload for IPv4 packets

DMA Block Features

- Multichannel transmit and receive engines (eight transmit and eight receive channels)
- Separate DMA channel in the transmit path for each queue in MFL
- Single or multiple DMA channels for any number of queues in MFL receive path
- Dual-buffer (ring) descriptor support
- Comprehensive status reporting for normal operation and transfers with errors
- Individual programmable burst length for Tx DMA and Rx DMA engines
- Programmable interrupt options for different operational conditions
- Per-packet transmit or receive complete interrupt control
- Round-robin or fixed-priority arbitration between the receive and transmit engines
- Fixed-priority, weighted-strict-priority or weighted-round-robin arbitration for data read requests from multiple transmit DMA engines
- Start and stop modes
- Support for TCP Segmentation Offload (TSO) and UDP Fragmentation Offload (UFO) with selectable number of Tx DMA channels
- Routing of received packets to the DMA channels based on the DA or VLAN priority in multichannel DMA configurations
- Option to split the packet header (layer 3 and layer 4) and payload in different buffers
- Time-sensitive conditional packet fetching from system memory by comparing the slot time or IEEE 1588 time information provided in the descriptor (useful for AV applications)

- Programmable control for transmit descriptor posted writes to improve the throughput
- Separate channels or queues for AV data transfer in 100 Mbps and 1000 Mbps modes
- Up to eight queues on the receive paths for AV traffic and seven queues on the transmit path for AV traffic
- IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm for transmit channels
- Single Tx FIFO and Rx FIFO (MFL) for all selected queues
- Programmable slot interval with range from 1µs to 4096 µs and granularity of 1µs

Data Center Bridging Features

- Separate channels or queues for DCB data transfer in 100 Mbps and 1000 Mbps modes
- Up to eight queues on the transmit and receive paths for DCB traffic
- The following Tx queues scheduling mechanisms meet the compliance specification in IEEE 802.1-Qaz Enhanced Transmission Selection (ETS) algorithm:
  - Weighted Round Robin (WRR)
  - Deficit Weighted Round Robin (DWRR)
  - Weighted Fair Queuing (WFQ)
  - Strict Priority (SP)
- Common memory for all selected Tx or Rx queues
- Priority-based Flow Control (PFC) on individual queues configured for DCB
- Programmable control to route received VLAN tagged packets to channels or queues

Time Sensitive Networking Features

- IEEE 802.1Qbv-2015, Enhancements to Scheduling Traffic
- IEEE802.1Qbu/802.3br, Frame preemption and Interspersing Express Traffic

Precision Time Protocol (PTP) Features

- Ethernet frame time stamping as described in IEEE 1588–2002 and IEEE 1588–2008. The transmit or receive status of each frame include 64-bit time stamps
- Hardware assisted time stamping capable of up to 8 ns resolution
- Automatic detection of PTP messages through Ethernet, IPv4, and IPv6 packets
- Four programmable PPS outputs that physically represent PTP system time
- Auxiliary snapshot to time stamp external events - four auxiliary input pins available

- PTP time stamp offload support
- Support for one-step time stamp

Table 30-1: EMAC0/EMAC1 Feature Differences

Feature	EMAC0		EMAC1	
	SC594	SC598	SC594	SC598
Speed	10/100/1000 Mbps		10/100 Mbps	
Line Interface	MII, RMII, RGMII		RMII	
Double VLAN Processing	No	Yes	No	
Queue/Channel-based VLAN Tag Insertion on Tx	No	Yes	No	
Additional MAC Address Registers	1	31	1	
Address Filter Hash Table Size	64 bits	256 bits	64 bits	
Extended RX VLAN Tag-based Filters	0	4	0	4
Layer 3 and Layer 4 Filters	1	8	1	1
Multicast Packet Duplication Support	No	Yes	No	
Number of Tx/Rx Queues and DMA Channels	3	8	1	
Data Center Bridging (DCB)	No	Yes	No	
Audio Video Bridging (AVB)	Yes	Yes	No	
Additional AVB DMA channels	2	7	N/A	
Precision Time Protocol (PTP)	Yes	Yes	No	
PTP - Number of Auxiliary snapshots	4 with 4 deep FIFO	4 with 16 deep FIFO	N/A	
PTP - Timestamp Offloading (Auto Send PTP frames)	No	Yes	N/A	
PTP - One Step Timestamp Feature	No	Yes	N/A	
TSN-EST (Enhancements to Scheduling Traffic) (802.1Qbv)	No	Yes	No	
Frame Preemption Support (802.3br, 802.1Qbu)	No	Yes	No	
Time Based Scheduling (launch time)	No	Yes	No	

Table 30-1: EMAC0/EMAC1 Feature Differences (Continued)

Feature	EMAC0		EMAC1	
	SC594	SC598	SC594	SC598
TCP Segmentation Offloading for TCP/IP Packets	No	Yes	No	
Split Header Feature	No	Yes	No	
IPv4 ARP Offload	No	Yes	No	
Energy Efficient Ethernet (EEE)	Yes	Yes	Yes	No

EMAC Functional Description

This section provides information on the function of Ethernet MAC peripheral.

Hardware for the Media Access Control protocol

This function allows applications to support TCP/IP based network communication. At the system end, the module supports direct connection with the system crossbar bus for memory or MMR transactions. It supports RMII (Reduced Media Independent Interface), RGMII (Reduced Gigabit Media Independent Interface), MII (Media Independent Interface), and SMI (Station Management Interface) for interfacing with the external PHY chip.

Dedicated DMA Controller with independent read/write channels

Performs both data and status transfers between the application and the media independent interfaces. Internal transmit and receive FIFOs are used to buffer and regulate the frames. Dedicated interrupt lines connect the EMAC interrupt sources to the System Event Controller (SEC) for the SHARC core or Generic Interrupt Controller (GIC) for ARM core.

MAC Management Counters (MMC) block

An extended set of registers that collect various statistics compliant with IEEE 802.3 definitions regarding the operation of the interface. The registers are updated for each new transmitted or received frame when the condition to update the counter is met. The EMAC provides a set of such counters, along with extended usage control.

PTP (Precision Time Protocol) engine

Provides hardware assistance for the implementation of the IEEE 1588 version 1 and version 2 standards, which allows time synchronization between systems.

Audio Video (AV) functionality

Enables transmission of time-sensitive traffic over bridged local area networks (LANs). The EMAC provides hardware support for IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm. In addition, slot number function allows scheduling of fetching of data by DMA from system memory.

Data Center Bridging (DCB)

Data center bridging (DCB) is a set of enhancements to the Ethernet local area network communication protocol for use in data center environments, for use with clustering and storage area networks. DCB aims, for selected traffic, to eliminate loss due to queue overflow (sometimes called lossless Ethernet) and to be able to allocate bandwidth on links. DCB enables, to some extent, the treatment of different priorities as if they were different pipes.

Time Sensitive Networking

Time sensitive network (TSN) is a set of standards developed by the IEEE 802.1 work group. The standards define mechanisms for the time-sensitive transmission of data over Ethernet.

ADSP-SC59x EMAC Register List

RGMII, RMII, MII Ethernet Controller (EMAC)

Table 30-2: ADSP-SC59x EMAC Register List

Name	Description
EMAC_DMA[n]_CTL	DMA Channel n Control Register
EMAC_DMA[n]_RXBUF_CUR	DMA Channel n Current Application Receive Buffer Address Register
EMAC_DMA[n]_RXDSC_CUR	DMA Channel n Current Application Receive Descriptor Register
EMAC_DMA[n]_TXBUF_CUR	DMA Channel n Current Application Transmit Buffer Address Register
EMAC_DMA[n]_TXDSC_CUR	DMA Channel n Current Application Transmit Descriptor Register
EMAC_DMA[n]_IEN	DMA Channel n Interrupt Enable Register
EMAC_DMA[n]_MISSFRM_CNT	DMA Channel n Missed Frame Counter Register
EMAC_DMA[n]_RXDSC_ADDR	DMA Channel n Receive Descriptor List Address Register
EMAC_DMA[n]_RXDSC_TLPTR	DMA Channel n Receive Descriptor Tail Pointer Register
EMAC_DMA[n]_RXCTL	DMA Channel n Receive Control Register
EMAC_DMA[n]_RXCTL2	DMA Channel n Extended Receive Control Register
EMAC_DMA[n]_RXINTWDTMR	DMA Channel n Receive Interrupt Watchdog Timer Register
EMAC_DMA[n]_SFN_CTLSTAT	DMA Channel n Slot Function Control and Status register
EMAC_DMA[n]_STAT	DMA Channel n Status Register
EMAC_DMA[n]_TXDSC_ADDR	DMA Channel n Transmit Descriptor List Address Register

Table 30-2: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_DMA[n]_TXDSC_RLEN	DMA Channel n Transmit Descriptor Ring Length Register
EMAC_DMA[n]_TXDSC_TLPTR	DMA Channel n Transmit Descriptor Tail Pointer Register
EMAC_DMA[n]_TXCTL	DMA Channel n Transmit Control Register
EMAC_DMA_DBG_STAT0	DMA Debug Status 0 Register
EMAC_DMA_DBG_STAT1	DMA Debug Status 1 Register
EMAC_DMA_DBG_STAT2	DMA Debug Status 2 Register
EMAC_DMA_ISTAT	DMA Interrupt Status Register
EMAC_DMA_MODE	DMA Mode Register
EMAC_DMA_SYSBMODE	DMA System Bus Mode Register
EMAC_DMA_TBS_CTL0	DMA TBS Control 0 Register
EMAC_DMA_TBS_CTL1	DMA TBS Control 1 Register
EMAC_DMA_TBS_CTL2	DMA TBS Control 2 Register
EMAC_DMA_TBS_CTL3	DMA TBS Control 3 Register
EMAC_ONEUS_TIC_CNT	LPI One Microsecond Tic Counter Register
EMAC_ARP_ADDR	Address Resolution Protocol (ARP) Register
EMAC_ADDR0_HI	MAC Address 0 High Register
EMAC_ADDR0_LO	MAC Address 0 Low Register
EMAC_ADDR[i]_HI	MAC Address n High Register
EMAC_ADDR[i]_LO	MAC Address n Low Register
EMAC_TM_AUX_CTL	Auxiliary Time Stamp Control Register
EMAC_TM_AUXTS_NSEC	Auxiliary Time Stamp Nanoseconds Register
EMAC_TM_AUXTS_SEC	Auxiliary Time Stamp Seconds Register
EMAC_CSR_SW_CTL	CSR Software Control Register
EMAC_MAC_CFG	MAC Configuration Register
EMAC_MAC_DBG	Debug Register
EMAC_SPLM_OFST_CFG	MAC Split Mode and Offset Configuration Register
EMAC_MAC_EXT_CFG	MAC Extended Configuration Register
EMAC_FPE_CTLSTAT	Frame Preemption Control and Status Register
EMAC_HW_FTR_PSNT0	Hardware Feature Presence Register 0
EMAC_HW_FTR_PSNT1	Hardware Feature Presence Register 1
EMAC_HW_FTR_PSNT2	Hardware Feature Presence Register 2

Table 30-2: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_HW_FTR_PSNT3	Hardware Feature Presence Register 3
EMAC_HASHTBL_REG0	Hash Table Register 0
EMAC_HASHTBL_REG1	Hash Table Register 1
EMAC_HASHTBL_REG2	Hash Table Register 2
EMAC_HASHTBL_REG3	Hash Table Register 3
EMAC_HASHTBL_REG4	Hash Table Register 4
EMAC_HASHTBL_REG5	Hash Table Register 5
EMAC_HASHTBL_REG6	Hash Table Register 6
EMAC_HASHTBL_REG7	Hash Table Register 7
EMAC_INDR_ACC_CTL	Indirect Access Control Register
EMAC_INDR_ACC_DAT	Indirect Access Data Register
EMAC_INVLAN_INCL	Inner VLAN Tag Inclusion or Replacement Register
EMAC_MAC_IEN	MAC Interrupt Enable Register
EMAC_MAC_ISTAT	MAC Interrupt Status Register
EMAC_L3L4_CTL7	Layer 3 and Layer 4 Control Register
EMAC_L3L4_CTL[i]	Layer 3 and Layer 4 Control Register
EMAC_LPI_CTLSTAT	LPI Control and Status Register
EMAC_LPI_ENTR_TMR	Transmit LPI Entry Timer Register
EMAC_LPI_TMRSCTL	LPI Timers Control Register
EMAC_L3_ADDR0_REG7	Layer 3 Address 0 Register 7
EMAC_L3_ADDR0_REG[i]	Layer 3 Address 0 Register [i]
EMAC_L3_ADDR1_REG7	Layer 3 Address 1 Register 7
EMAC_L3_ADDR1_REG[i]	Layer 3 Address 1 Register [i]
EMAC_L3_ADDR2_REG7	Layer 3 Address 2 Register 7
EMAC_L3_ADDR2_REG[i]	Layer 3 Address 2 Register [i]
EMAC_L3_ADDR3_REG7	Layer 3 Address 3 Register 7
EMAC_L3_ADDR3_REG[i]	Layer 3 Address 3 Register [i]
EMAC_L4_ADDR7	Layer 4 Address Register [i]
EMAC_L4_ADDR[i]	Layer 4 Address Register [i]
EMAC_LOG_MSG_INTVL	PTP Log Message Interval Register
EMAC_MDIO_ADDR	MDIO Address Register

Table 30-2: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_MDIO_DATA	MDIO Data Register
EMAC_PHYIF_CTLSTAT	PHY Interface Control and Status Register
EMAC_PPS[n]_INTVL	PPS n Interval Register
EMAC_PPS[n]_TGTM_NSEC	PPS n Target Time Nanoseconds Register
EMAC_PPS[n]_TGTM_SEC	PPS n Target Time Seconds Register
EMAC_PPS[n]_WID	PPS n Width Register
EMAC_PPS_CTL	PPS Control Register
EMAC_PTO_CTL	PTP Time Stamp Offload Control Register
EMAC_MACPKT_FILT	MAC Packet Filter Register
EMAC_PRSNTM_UPDT	Presentation Time Update Register
EMAC_PRSNTM_NSEC	Presentation Time Nanosecond Register
EMAC_Q0_TXFLOW_CTL	Queue 0 Transmit Flow Control Register
EMAC_Q[i]_TXFLOW_CTL	Queue [i] Transmit Flow Control Register
EMAC_RXQ_CTL0	Receive Queue Control 0 Register
EMAC_RXQ_CTL1	Receive Queue Control 1 Register
EMAC_RXQ_CTL2	Receive Queue Control 2 Register
EMAC_RXQ_CTL3	Receive Queue Control 3 Register
EMAC_RXQ_CTL4	Receive Queue Control 4 Register
EMAC_RXFLOW_CTL	Receive Flow Control Register
EMAC_RXTX_STAT	Receive Transmit Status Register
EMAC_SRCPRT_IDNTY0	Source Port Identity 0 Register
EMAC_SRCPRT_IDNTY1	Source Port Identity 1 Register
EMAC_SRCPRT_IDNTY2	Source Port Identity 2 Register
EMAC_TM_SUBSEC	Subsecond Increment Register
EMAC_TM_HISEC	System Time Higher Word Seconds Register
EMAC_TM_NSEC	System Time Nanoseconds Register
EMAC_TM_NSECUPDT	System Time Nanoseconds Update Register
EMAC_TM_SEC	System Time Seconds Register
EMAC_TM_SECUPDT	System Time Seconds Update Register
EMAC_TM_ADDEND	Time Stamp Addend Register
EMAC_TM_CTL	Time Stamp Control Register

Table 30-2: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_TM_EGASYM_CORR	Time Stamp Egress Asymmetry Correction Register
EMAC_TM_EGCORR_NSEC	Time Stamp Egress Correction Nanosecond Register
EMAC_TM_EGCORR_SNSEC	Time Stamp Egress Correction Sub Nanosecond Register
EMAC_TM_EG_LTNCY	Time Stamp Egress Latency Register
EMAC_TM_INGASYM_CORR	Time Stamp Ingress Asymmetry Correction Register
EMAC_TM_INGCORR_NSEC	Time Stamp Ingress Correction Nanosecond Register
EMAC_TM_INGCORR_SNSEC	Time Stamp Ingress Correction Sub Nanosecond Register
EMAC_TM_ING_LTNCY	Time Stamp Ingress Latency Register
EMAC_TM_STAT	Time Stamp Status Register
EMAC_TXQ_PRTY_MAP0	Transmit Queue Priority Mapping 0 Register
EMAC_TXQ_PRTY_MAP1	Transmit Queue Priority Mapping 1 Register
EMAC_TXTMSTAT_NSEC	Transmit Time Stamp Status Nanoseconds Register
EMAC_TXTMSTAT_SEC	Transmit Time Stamp Status Seconds Register
EMAC_VLAN_HASHTBL	VLAN Hash Table Register
EMAC_VLAN_INCL	VLAN Tag Inclusion or Replacement Register
EMAC_VLANTAG_CTL	VLAN Tag Control Register
EMAC_VLANTAG_DAT	VLAN Tag Indirect Access Data Register
EMAC_WDT_TMOUT	Watchdog Timeout Register
EMAC_MMC_CTRL	MMC Control Register
EMAC_MMC_FPE_RXINT	MMC Receive FPE Interrupt Status Register
EMAC_MMC_FPE_RXIMSK	MMC Receive FPE Interrupt Mask Register
EMAC_MMC_FPE_TXINT	MMC Transmit FPE Interrupt Status Register
EMAC_MMC_FPE_TXIMSK	MMC Transmit FPE Interrupt Mask Register
EMAC_MMC_IPC_RXINT	MMC Receive IPC Interrupt Status Register
EMAC_MMC_IPC_RXIMSK	MMC Receive IPC Interrupt Mask Register
EMAC_RXFPE_FRGMNT_CNT	MMC Receive FPE Fragment Counter Register
EMAC_MMC_RXINT	MMC Receive Interrupt Status Register
EMAC_MMC_RXIMSK	MMC Receive Interrupt Mask Register
EMAC_RXPKT_ASERR_CNT	MMC Receive FPE Packet Assembly Error Counter Register
EMAC_RXPKT_ASOK_CNT	MMC Receive FPE Packet Assembly Okay Counter Register
EMAC_RXPKT_SMDERR_CNT	MMC Receive FPE Packet SMD Error Counter Register

Table 30-2: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_TXFPE_FRGMNT_CNT	MMC Transmit FPE Packet Fragment Counter Register
EMAC_TXHLDRQ_CNT	MMC Transmit FPE Hold Request Counter Register
EMAC_MMC_TXINT	MMC Transmit Interrupt Status Register
EMAC_MMC_TXIMSK	MMC Transmit Interrupt Mask Register
EMAC_EST_CTL	EST Control Register
EMAC_EST_EXT_CTL	EST Extended Control Register
EMAC_EST_FRMSZ_CAP	EST Frame Size Capture Register
EMAC_EST_FRMSZ_ERR	EST Frame Size Error Register
EMAC_EST_GCL_CTL	EST Gate Control List Control Register
EMAC_EST_GCL_DAT	EST Gate Control List Data Register
EMAC_EST_IEN	EST Interrupt Enable Register
EMAC_EST_SCH_ERR	EST Scheduling Error Register
EMAC_EST_STAT	EST Status Register
EMAC_FPE_ADV	Frame Preemption Advance Register
EMAC_FPE_CTRLSTS	MTL Frame Preemption Control and Status Register
EMAC_MTL_ISTAT	MTL Interrupt Status Register
EMAC_MTL_OPMODE	MTL Operation Mode Register
EMAC_Q0_INT_CTLSTAT	Queue 0 Interrupt Control and Status Register
EMAC_Q[n]_INT_CTLSTAT	Queue n Interrupt Control and Status Register
EMAC_RQ0_CTL	Receive Queue Control Register
EMAC_RQ0_DBG	Receive Queue 0 Debug Register
EMAC_RQ0_MSPKTOF_CNT	Receive Queue 0 Missed Packet and Overflow Counter Register
EMAC_RQ0_OPMODE	Receive Queue 0 Operation Mode Register
EMAC_RQ[n]_CTL	Receive Queue n Control Register
EMAC_RQ[n]_DBG	Receive Queue n Debug Register
EMAC_RQ[n]_MSPKTOF_CNT	Receive Queue n Missed Packet and Overflow Counter Register
EMAC_RQ[n]_OPMODE	Receive Queue n Operation Mode Register
EMAC_RXQ_DMA_MAP0	Receive Queue to DMA Channel Mapping Register 0
EMAC_RXQ_DMA_MAP1	Receive Queue to DMA Channel Mapping Register 1
EMAC_TBS_CTRL	MTL Time Based Scheduling Control Register
EMAC_TQ0_DBG	Transmit Queue 0 Debug Register

Table 30-2: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_TQ0_ETS_STAT	Transmit Queue 0 ETS Status Register
EMAC_TQ0_OPMODE	Transmit Queue 0 Operation Mode Register
EMAC_TQ0_QTMWGT	Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register
EMAC_TQ0_URFL	Transmit Queue 0 Underflow Packet Counter Register
EMAC_TQ[n]_DBG	Transmit Queue n Debug Register
EMAC_TQ[n]_ETS_CTL	Transmit Queue n ETS Control Register
EMAC_TQ[n]_ETS_STAT	Transmit Queue n ETS Status Register
EMAC_TQ[n]_HICRDT	Transmit Queue n CBS HiCredit Register
EMAC_TQ[n]_LOCRDT	Transmit Queue n CBS LoCredit Register
EMAC_TQ[n]_OPMODE	Transmit Queue n Operation Mode Register
EMAC_TQ[n]_QTMWGT	Transmit Queue n IdleSlopeCredit, Quantum or Weights Register
EMAC_TQ[n]_SSCRDT	Transmit Queue n SendSlopeCredit Register
EMAC_TQ[n]_URFL	Transmit Queue n Underflow Packet Counter Register
EMAC_RXICMP_ERR_OCNT	Receive ICMP Error Octets Counter Register
EMAC_RXICMP_ERR_PCNT	Receive ICMP Error Packets Counter Register
EMAC_RXICMP_OCNT_G	Receive ICMP Good Octets Counter Register
EMAC_RXICMP_PCNT_G	Receive ICMP Good Packets Counter Register
EMAC_RXIPV4_FRGMT_OCNT	Receive IPv4 Fragmented Octets Counter Register
EMAC_RXIPV4_FRGMT_PCNT	Receive IPv4 Fragmented Packets Counter Register
EMAC_RXIPV4_OCNT_G	Receive IPv4 Good Octets Counter Register
EMAC_RXIPV4_PCNT_G	Receive IPv4 Good Packets Counter Register
EMAC_RXIPV4_HDRERR_OCNT	Receive IPv4 Header Error Octets Counter Register
EMAC_RXIPV4_HDRERR_PCNT	Receive IPv4 Header Error Packets Counter Register
EMAC_RXIPV4_PYLD_OCNT	Receive IPv4 Payload Octets Counter Register
EMAC_RXIPV4_PYLD_PCNT	Receive IPv4 Payload Packets Counter Register
EMAC_RXIPV4_UDPCSD_OCNT	Receive IPv4 UDP Checksum Disable Octets Counter Register
EMAC_RXIPV4_UDPCSD_PCNT	Receive IPv4 UDP Checksum Disable Packets Counter Register
EMAC_RXIPV6_OCNT_G	Receive IPv6 Good Octets Counter Register
EMAC_RXIPV6_PCNT_G	Receive IPv6 Good Packets Counter Register
EMAC_RXIPV6_HDRERR_OCNT	Receive IPv6 Header Error Octets Counter Register
EMAC_RXIPV6_HDRERR_PCNT	Receive IPv6 Header Error Packets Counter Register

Table 30-2: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_RXIPV6_PYLD_OCNT	Receive IPv6 Payload Octets Counter Register
EMAC_RXIPV6_PYLD_PCNT	Receive IPv6 Payload Packets Counter Register
EMAC_RXTCP_ERR_OCNT	Receive TCP Error Octets Counter Register
EMAC_RXTCP_ERR_PCNT	Receive TCP Error Packets Counter Register
EMAC_RXTCP_OCNT_G	Receive TCP Good Octets Counter Register
EMAC_RXTCP_PCNT_G	Receive TCP Good Packets Counter Register
EMAC_RXUDP_ERR_OCNT	Receive UDP Error Octets Counter Register
EMAC_RXUDP_ERR_PCNT	Receive UDP Error Packets Counter Register
EMAC_RXUDP_OCNT_G	Receive UDP Good Octets Counter Register
EMAC_RXUDP_PCNT_G	Receive UDP Good Packets Counter Register
EMAC_RX1024TOMAX_GB	Receive 1024 - Maximum Octets Packets Good Bad Counter Register
EMAC_RX128TO255_GB	Receive 128 - 255 Octets Packets Good Bad Counter Register
EMAC_RX256TO511_GB	Receive 256 - 511 Octets Packets Good Bad Counter Register
EMAC_RX512TO1023_GB	Receive 512 - 1023 Octets Packets Good Bad Counter Register
EMAC_RX64_GB	Receive 64 Octets Packets Good Bad Counter Register
EMAC_RX65TO127_GB	Receive 65 - 127 Octets Packets Good Bad Counter Register
EMAC_RX_ALGNERR_PCNT	Receive Alignment Error Packets Counter Register
EMAC_RX_BCERR_PCNT_G	Receive Broadcast Packets Good Counter Register
EMAC_RX_CRCERR_PCNT	Receive CRC Error Packets Counter Register
EMAC_RX_CTL_PCNT_G	Receive Control Packets Good Counter Register
EMAC_RX_FIFOOVFL_PCNT	Receive FIFO Overflow Packets Counter Register
EMAC_RX_JBERR_PCNT	Receive Jabber Error Packets Counter Register
EMAC_RX_LPITRAN_CNT	Receive LPI Transition Counter Register
EMAC_RX_LPIUSEC_CNT	Receive LPI Microseconds Counter Register
EMAC_RX_LENERR_PCNT	Receive Length Error Packets Counter Register
EMAC_RX_MCST_PCNT_G	Receive Multicast Packets Good Counter Register
EMAC_RX_OCNT_G	Receive Octet Good Counter Register
EMAC_RX_OCNT_GB	Receive Octet Good Bad Counter Register
EMAC_RX_ORTYPE_PCNT	Receive Out of Range Type Packets Counter Register
EMAC_RX_OVRSZ_PCNT	Receive Oversize Packets Good Counter Register
EMAC_RX_PCNT_GB	Receive Packets Good Bad Counter Register

Table 30-2: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_RX_PAUSE_PCNT	Receive Pause Packets Counter Register
EMAC_RX_ERR_PCNT	Receive Error Packets Counter Register
EMAC_RX_RNTERR_PCNT	Receive Runt Error Packets Counter Register
EMAC_RX_UNDRSZ_PCNT_G	Receive Undersize Packets Good Counter Register
EMAC_RX_UCST_PCNT_G	Receive Unicast Packets Good Counter Register
EMAC_RX_VLAN_PCNT_GB	Receive VLAN Packets Good Bad Counter Register
EMAC_RX_WDTERR_PCNT	Receive Watchdog Error Packets Counter Register
EMAC_TX1024TOMAX_GB	Transmit 1024-Max Octets Packets Good Bad Counter Register
EMAC_TX128TO255_GB	Transmit 128-255 Octets Packets Good Bad Counter Register
EMAC_TX256TO511_GB	Transmit 256-511 Octets Packets Good Bad Counter Register
EMAC_TX512TO1023_GB	Transmit 512-1023 Octets Packets Good Bad Counter Register
EMAC_TX64_GB	Transmit 64 Octets Packets Good Bad Counter Register
EMAC_TX65TO127_GB	Transmit 65-127 Octets Packets Good Bad Counter Register
EMAC_TX_BCST_PCNT_G	Transmit Broadcast Packets Good Counter Register
EMAC_TX_BCST_PCNT_GB	Transmit Broadcast Packets Good Bad Counter Register
EMAC_TX_CRERR_PCNT	Transmit Carrier Error Packets Counter Register
EMAC_TX_DFRD_PCNT	Transmit Deferred Packets Counter Register
EMAC_TX_EXCOL_PCNT	Transmit Excessive Collision Packets Counter Register
EMAC_TX_EXDFRL_PCNT	Transmit Excessive Deferral Error Counter Register
EMAC_TX_LPITRAN_CNT	Transmit LPI Transition Counter Register
EMAC_TX_LPIUSEC_CNT	Transmit LPI Microseconds Counter Register
EMAC_TX_LTCOL_PCNT	Transmit Late Collision Packets Counter Register
EMAC_TX_MCST_PCNT_G	Transmit Multicast Packets Good Counter Register
EMAC_TX_MCST_PCNT_GB	Transmit Multicast Packets Good Bad Counter Register
EMAC_TX_MCOL_PCNT_G	Transmit Multiple Collision Packets Good Counter Register
EMAC_TX_OSIZE_PCNT_G	Transmit Osize Packets Good Counter Register
EMAC_TX_OCNT_G	Transmit Octet Good Counter Register
EMAC_TX_OCNT_GB	Transmit Octet Good Bad Counter Register
EMAC_TX_PCNT_G	Transmit Packet Good Counter Register
EMAC_TX_PCNT_GB	Transmit Packet Good Bad Counter Register
EMAC_TX_PAUSE_PCNT	Transmit Pause Packets Counter Register

Table 30-2: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_TX_SNGLCOL_PCNT_G	Transmit Single Collision Packets Good Counter Register
EMAC_TX_URFL_PCNT	Transmit Underflow Error Packets Counter Register
EMAC_TX_UCST_PCNT_GB	Transmit Unicast Packets Good Bad Counter Register
EMAC_TX_VLAN_PCNT_G	Transmit VLAN Packets Good Counter Register

ADSP-SC59x EMAC Interrupt List

Table 30-3: ADSP-SC59x EMAC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
19	EMAC0_DMA3_TX	DMA3_TX Channel Interrupt	None	
20	EMAC0_DMA4_TX	DMA4_TX Channel Interrupt	None	
21	EMAC0_DMA5_TX	DMA5_TX Channel Interrupt	None	
22	EMAC0_DMA6_TX	DMA6_TX Channel Interrupt	None	
23	EMAC0_DMA7_TX	DMA7_TX Channel Interrupt	None	
24	EMAC0_DMA0_RX	DMA0_RX Channel Interrupt	None	
31	EMAC0_DMA1_RX	DMA1_RX Channel Interrupt	None	
32	EMAC0_DMA2_RX	DMA2_RX Channel Interrupt	None	
33	EMAC0_DMA3_RX	DMA3_RX Channel Interrupt	None	
162	EMAC0_DMA4_RX	DMA4_RX Channel Interrupt	None	
163	EMAC0_DMA5_RX	DMA5_RX Channel Interrupt	None	
164	EMAC0_DMA6_RX	DMA6_RX Channel Interrupt	None	
165	EMAC0_DMA7_RX	DMA7_RX Channel Interrupt	None	
221	EMAC0_STAT	Status	None	
222	EMAC0_PWR	Power Interrupt	None	
223	EMAC0_DMA0_TX	DMA0_TX Channel Interrupt	None	
224	EMAC0_DMA1_TX	DMA1_TX Channel Interrupt	None	
225	EMAC0_DMA2_TX	DMA2_TX Channel Interrupt	None	
226	EMAC1_MAC	MAC Interrupt	None	
227	EMAC0_STAT	Status	None	
228	EMAC1_PWR	Power Interrupt	None	
229	EMAC1_DMA0_TX	DMA0_TX Channel Interrupt	None	
230	EMAC1_MAC	MAC Interrupt	None	

Table 30-3: ADSP-SC59x EMAC Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
232	EMAC1_DMA0_RX	DMA0_RX Channel Interrupt	None	

ADSP-SC59x EMAC Trigger List

Table 30-4: ADSP-SC59x EMAC Trigger List Generators

Trigger ID	Name	Description	Sensitivity
180	EMAC0_MCGR_DMA_REQ0	MCGR DMA Request	Edge
181	EMAC0_MCGR_DMA_REQ1	MCGR DMA Request	Edge
182	EMAC0_MCGR_DMA_REQ2	MCGR DMA Request	Edge
183	EMAC0_MCGR_DMA_REQ3	MCGR DMA Request	Edge

Table 30-5: ADSP-SC59x EMAC Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
206	EMAC0_DMA0_RX_STOP	DMA RX Channel Start Trigger	Edge
207	EMAC0_DMA1_RX_STOP	DMA RX Channel Start Trigger	Edge
208	EMAC0_DMA2_RX_STOP	DMA RX Channel Start Trigger	Edge
209	EMAC0_DMA3_RX_STOP	DMA RX Channel Start Trigger	Edge
210	EMAC0_DMA4_RX_STOP	DMA RX Channel Start Trigger	Edge
211	EMAC0_DMA5_RX_STOP	DMA RX Channel Start Trigger	Edge
212	EMAC0_DMA6_RX_STOP	DMA RX Channel Start Trigger	Edge
213	EMAC0_DMA7_RX_STOP	DMA RX Channel Start Trigger	Edge
214	EMAC0_DMA0_TX_STOP	DMA TX Channel Stop Trigger	Edge
215	EMAC0_DMA1_TX_STOP	DMA TX Channel Stop Trigger	Edge
216	EMAC0_DMA2_TX_STOP	DMA TX Channel Stop Trigger	Edge
217	EMAC0_DMA3_TX_STOP	DMA TX Channel Stop Trigger	Edge
218	EMAC0_DMA4_TX_STOP	DMA TX Channel Stop Trigger	Edge
219	EMAC0_DMA5_TX_STOP	DMA TX Channel Stop Trigger	Edge
220	EMAC0_DMA6_TX_STOP	DMA TX Channel Stop Trigger	Edge
221	EMAC0_DMA7_TX_STOP	DMA TX Channel Stop Trigger	Edge

Table 30-5: ADSP-SC59x EMAC Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
222	EMAC1_DMA0_RX_STOP	DMA RX Channel Stop Trigger	Edge
223	EMAC1_DMA0_TX_STOP	DMA TX Channel Stop Trigger	Edge
224	EMAC0_MCGR_DMA_ACK0	MCGR DMA Acknowledge	Edge
225	EMAC0_MCGR_DMA_ACK1	MCGR DMA Acknowledge	Edge
226	EMAC0_MCGR_DMA_ACK2	MCGR DMA Acknowledge	Edge
227	EMAC0_MCGR_DMA_ACK3	MCGR DMA Acknowledge	Edge

EMAC Definitions

The following definitions are helpful for using the EMAC.

ARI

Application Receive Interface

ATI

Application Transmit Interface

CPT

Current Presentation Time

DA

Destination Address

DCB

Data Center Bridging

EMAC CORE

CORE layer inside EMAC which performs the actual Ethernet operations, including interface with PHY through the reduced media interface(s).

EMAC DMA

DMA Controller of EMAC

EMAC MTL

MAC Transaction Layer inside EMAC

EMAC SCB

System Crossbar Interface of EMAC

EOP

End of Packet

EST

Enhancement to Scheduled Traffic

ETS

Enhanced Transmission Selection

FPE

Frame Preemption

ICMP

Internet Message Control Protocol

MAC

Media Access Control

MII

Media Independent Interface

MMC

MAC Management Counter

OST

One Step Time Stamp

PTO

PTP Time Stamp Offload

PTP

Precision Time Protocol

LPI

Low Power Interface

MDIO

Management Data Input/Output

RGMII

Reduced Gigabit Media Independent Interface

RMII

Reduced Media Independent Interface

SA

Source Address

SMI

Station Management Interface that controls PHY through MDIO and MDC signals.

SOP

Start Of Packet

TBS

Time Based Scheduling

TPT

Target Presentation Time

TSN

Time Sensitive Networking

VLAN

Virtual LAN

EMAC Block Diagram and Interfaces

The ***EMAC Simplified Block Diagram*** illustrates the overall functional architecture of the Ethernet MAC peripheral. The EMAC module is comprised of four major layers: EMAC SCB, EMAC DMA, EMAC MTL, and EMAC CORE. Each of these layers (sub blocks) is explained in depth in their respective sections in this chapter.

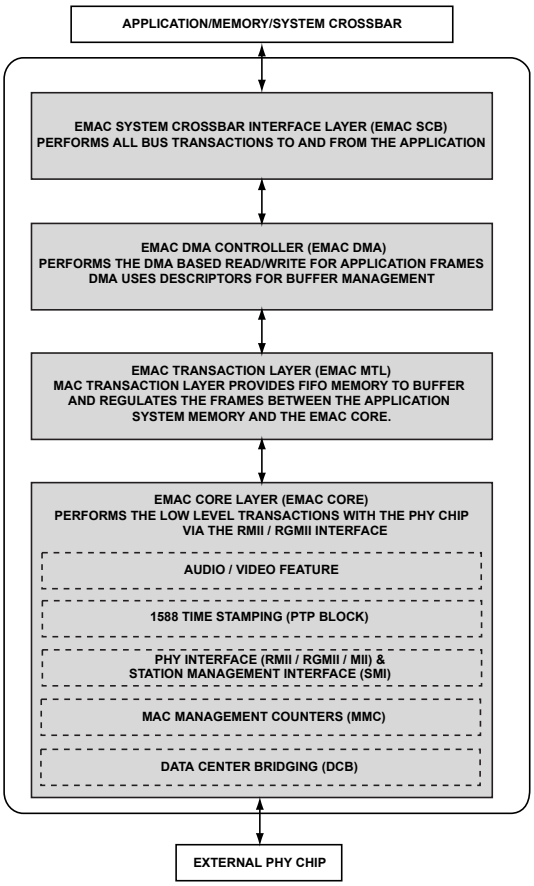


Figure 30-1: EMAC Simplified Block Diagram

A more comprehensive block diagram is shown in the ***EMAC Complete Block Diagram***. It includes most of the important blocks inside the EMAC. The EMAC is connected to processor memory and the system crossbar through

the System Crossbar Bus Interface (SCB) and System Peripheral Bus Interface (SPB). These connections are which are part of the SCB layer. The SPB interface is connected to all modules that require MMR programming.

The DMA controller performs application data transfer frame by frame, through well-defined descriptor structures. A FIFO layer acts as a buffer between the DMA controller and EMAC CORE.

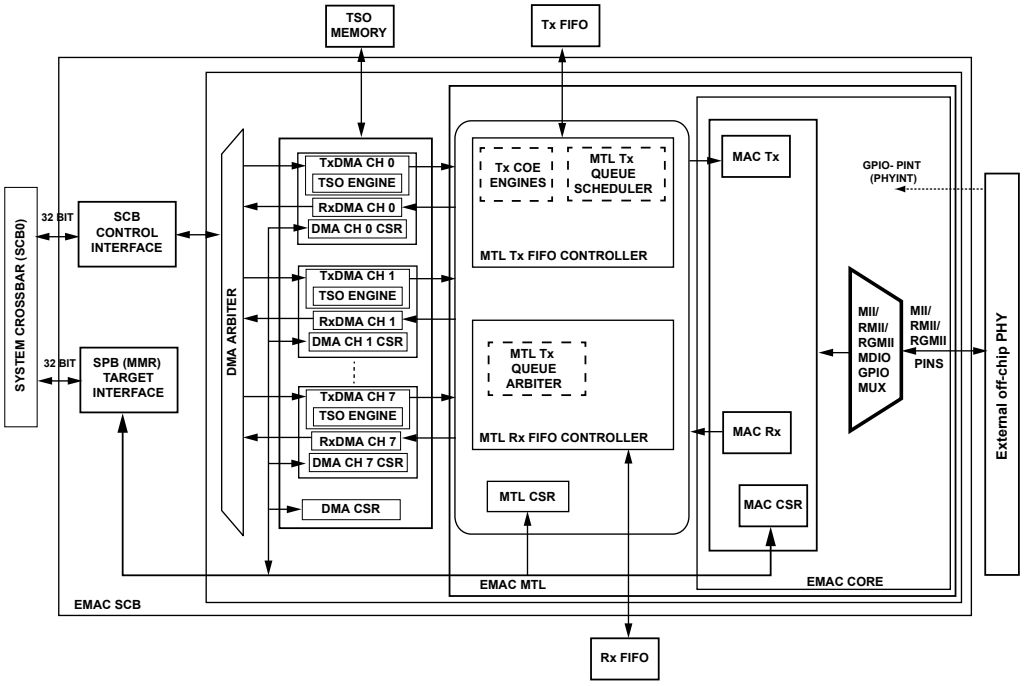


Figure 30-2: EMAC Complete Block Diagram

The *EMAC System Level Block Diagram* shows how the EMAC module interacts with rest of the system inside the processor.

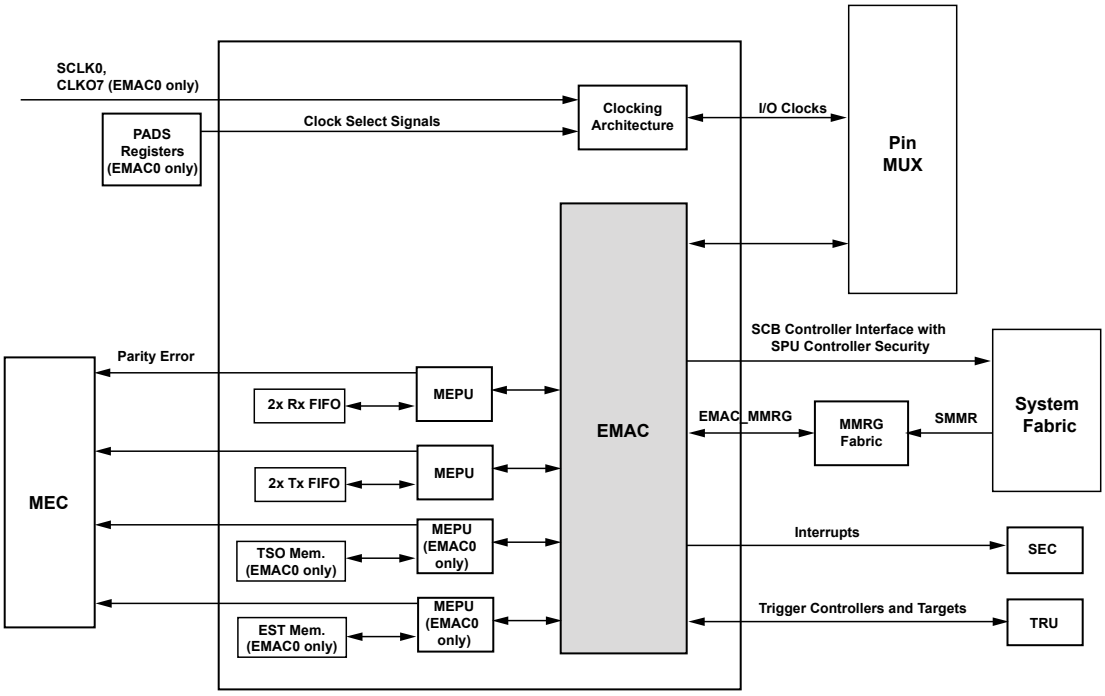


Figure 30-3: EMAC System Level Block Diagram

The *EMAC FIFO/Memory Details* table shows details about various FIFO and memory instances used inside the EMAC module. Each SPRAM/DPRAM stores data along with parity bits generated by MEPU. The parity errors are handled by the Memory Error Controller (MEC) unit.

Table 30-6: EMAC FIFO/Memory Details

EMAC Instance	FIFO/Memory Type	Single (SPRAM)/ Dual Ported (DPRAM)	Runs on	No. of Instances	Size	No. of data bits	No. of parity bits
EMAC0	Rx/Tx FIFOs (Odd/Even)	SPRAM	SCLK0	4	2048x40	35	5
	TSO Memory		SCLK0	1	512x36	32	4
	EST Memory		PTP Clock	1			
EMAC1	Rx FIFO	DPRAM	SCLK0	1	256x40	35	5
	Tx FIFO		SCLK0	1	128x40	35	5

EMAC CORE Sub Blocks

The ***Core Transmit Engine Sub Blocks*** table summarizes the core transmit engine sub blocks and their functions. Refer to the EMAC CORE section for further explanation of each of these sub blocks.

Table 30-7: CORE Transmit Engine Sub Blocks

CORE Transmit Engine Sub Block	Function
Transmit Bus Interface	Interface to the FIFO.
Transmit Packet Controller	Appends Zero-PAD data, if required, for short frames. Appends CRC for frame checksum from the CRC generator.
Transmit Protocol Controller	Generates preamble and SFD, as per 802.3 protocol. Generates jam pattern in half-duplex mode, for collisions. Jabber timeout, for excessively large frames. Flow control for half-duplex mode (back pressure). Generates transmit frame status.
Transmit Scheduler	Maintains the inter-frame gap between two transmitted frames. Follows the truncated binary exponential back-off algorithm for half-duplex mode.
Transmit CRC Generator	Generate CRC for the frame checksum field of the Ethernet frame.
Transmit Flow Control	Receives the pause frame, appends the calculated CRC, and sends the frame to the protocol engine module.
Transmit Checksum Offload Engine	Supports checksum calculation and insertion in the transmit path, for IPV4/TCP/UDP/ICMP packets.
TCP/IP Segmentation Offload (TSO) Engine	Useful in offloading the TCP segmentation functions to the hardware.
UDP/IPv4 Fragment Offload (UFO) Engine	Supports breaking a large UDP packet into smaller multiple packets for transmission.

The ***Core Receive Engine Sub Blocks*** table summarizes the core receive engine sub blocks and their function. Refer to the EMAC CORE section for more information on each of these sub blocks.

Table 30-8: Core Receive Engine Sub Blocks

CORE Receive Engine Sub block	Functionality Overview
Receive Protocol Engine	Strips the incoming preamble and SFD. Checks for correct length or type field. Performs internal loop back, if necessary. Generates receive status. Supports watchdog of received frames. Supports jumbo frames.
Receive CRC Module	Checks for CRC error, by comparing with FCS.

Table 30-8: Core Receive Engine Sub Blocks (Continued)

CORE Receive Engine Sub block	Functionality Overview
Receive Packet Controller Module	Packs incoming 8-bit input stream to 32-bit data internally. Performs frame filtering, for uni-cast, multi-cast, and broadcast frames. Attaches the calculated IP checksum input from checksum offload engine. Updates the receive status to bus interface.
Receive Flow Control Module	Detects the receiving pause frame and pauses the frame transmission for the delay specified within the received pause frame. Works in full duplex mode.
Receive IP Checksum Offload Engine	Calculates IPv4 header checksums and verify against the received IPv4 header checksums. Identifies a TCP, UDP, or ICMP payload in the received IP datagrams.
Receive Bus Interface Unit Module	Interface to the FIFO.
Address Filtering Module	Filters destination and source address based on uni-cast, multi-cast, and broadcast frames. Provides CRC hash filtering.
IPv4 ARP Offload Engine	This feature allows the processing of the IPv4 ARP request packet in the receive path and generating corresponding ARP response packet in the transmit path

EMAC PHY Interface

The EMAC can interface to the PHY through the RMII interface standard. The ***RMII Pins*** table shows the RMII pins available in the EMAC, in terms of their generic names. Refer to the data sheet for exact pin names.

Table 30-9: RMII Pins

Signal No.	Generic Signal Name (IEEE Standards)	RMII Pin Functionality
1.	TXD0	RMII transmit data pin D0 (di-bit lower)
2.	TXD1	RMII transmit data pin D1 (di-bit higher)
3.	RXD0	RMII receive data pin D0 (di-bit lower)
4.	RXD1	RMII receive data pin D1 (di-bit higher)
5.	RMII CLK	RMII common clock (for TX and RX), also called reference clock
6.	TXEN	RMII transmit enable pin (TX valid)
7.	CRS	RMII carrier sense / receive data valid
8.	MDC	Serial management clock driven by EMAC
9.	MDIO	Serial management bidirectional data

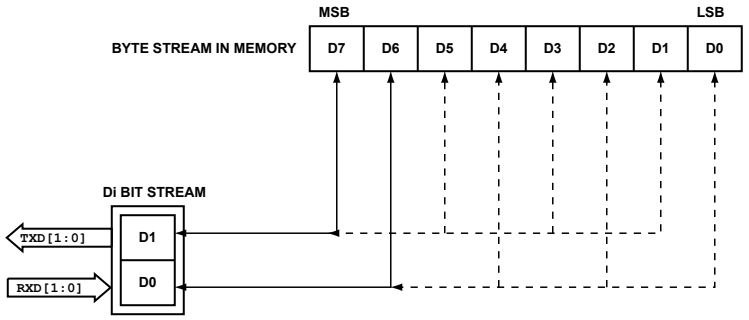


Figure 30-4: RMII Di-bit Data Transfer

RGMII Board Design Recommendations

Use the following guidelines during when performing board design when using the RGMII interface.

MAC to PHY (Transmit)

The Ethernet MAC transmits data to the Ethernet PHY. The Ethernet MAC sends data with tskewT (the timing of TXC at the MAC) that meets the RGMII specification (tskewT = –500 ps to +500 ps skew window for transmitter to drive data). The RGMII specification requires that at the PHY end, tskewR is sampled at 1.0 to 2.6 ns. According to the RGMII standard, clocks must be routed such that an additional trace delay of greater than 1.5 ns and less than 2 ns is added to the associated clock signal.

To meet this standard without adding trace delays, most of the PHYs in the industry already include delay logic that can compensate for this on-board delay. These PHY types can manage a tskewR of ±500 ps (the skew for TXC data sampling seen inside the PHY).

The PHY or the on-board delay must delay the clock signal by 1.5 to 2.0 ns so that tskewR is sampled at 1.0 to 2.6 ns. The ***MAC to PHY Delay Diagram (Transmitting Data)*** figure shows where the delays must occur.

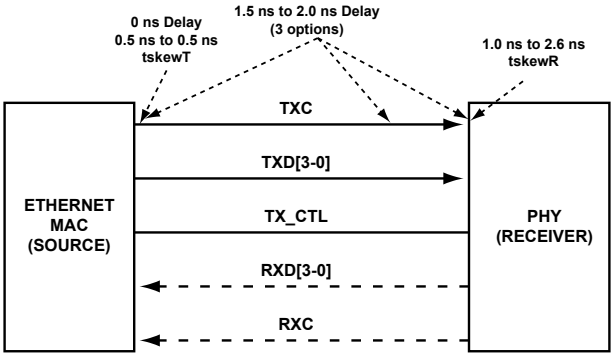


Figure 30-5: MAC to PHY Delay Diagram (Transmitting Data)

PHY to MAC (Receive)

The Ethernet MAC receives data from the Ethernet PHY. Just as in the transmit case, a trace delay of greater than 1.5 ns and less than 2 ns must be added to the associated clock signal, as required by the RGMII specification. Also similar to the transmit case, most of the PHYs in the industry already include delay logic that can compensate for the RXC clock as well.

As shown in the ***MAC to PHY Delay Diagram (Receiving Data)*** figure, a board trace or the PHY can be used to generate the required 1.5–2.0 ns delay to RXC and the 1.0–2.6 ns tskewR.

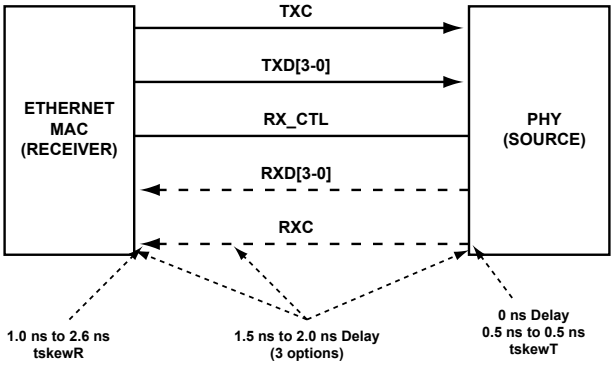


Figure 30-6: MAC to PHY Delay Diagram (Receiving Data)

There following are two options for board design.

- The on-board delay must delay the clock signal by 1.5 to 2.0 ns. In this case, no additional delay must be introduced by the PHY.
- Most of the PHYs in the industry already include a mode that can introduce a delay logic. When this PHY mode is used, then the trace lengths on the board must be matched exactly.

The second option is recommended since it is easier to implement the delay by using the mode in the PHY rather than during the board design. This is the approach followed in the ADI EZ-Kits.

NOTE: Refer to the product specific data sheet for exact processor timing.

For the trace length recommendations for the EMAC signals, see the data sheet.

For the exact requirements as recommended by the RGMII protocol, see the RGMII specification.

Clock Sources

The Ethernet MAC is clocked internally from CLKO7. Check the processor data sheet for the valid frequency range of the appropriate CLKO7 signal for Ethernet operation.

EMAC0 Clock Sources

EMAC0 supports RGMII, RMII and MII interfaces. The external PHY sources a 2.5 MHz or 25 MHz clock (for 10/100 or gigabit Ethernet respectively) to operate the EMAC RXCLK in RGMII mode. The RGMII TXCLK is driven from CLK07 of the CDU (Clock Distribution Unit) and needs to be configured to 125 MHz regardless of the EMAC0 speeds (10/100/1000 Mbit/s). The `EMAC_MAC_CFG.PS` and `EMAC_MAC_CFG.FES` bits are used to divide the clocks down.

The following tables show the clock sources for the RGMII, MII and RMII interfaces.

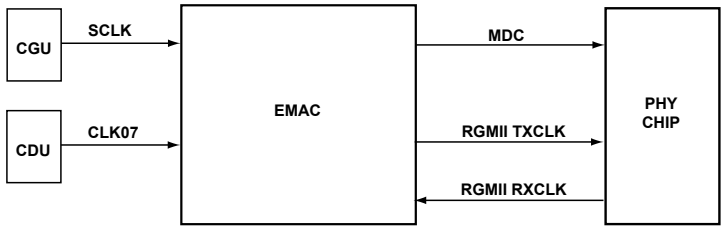


Figure 30-7: EMAC Clock Sources - RGMII PHY Interface

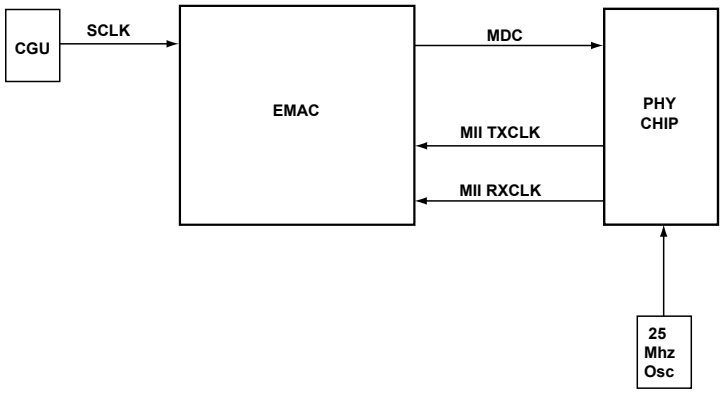


Figure 30-8: EMAC Clock Sources - MII PHY Interface

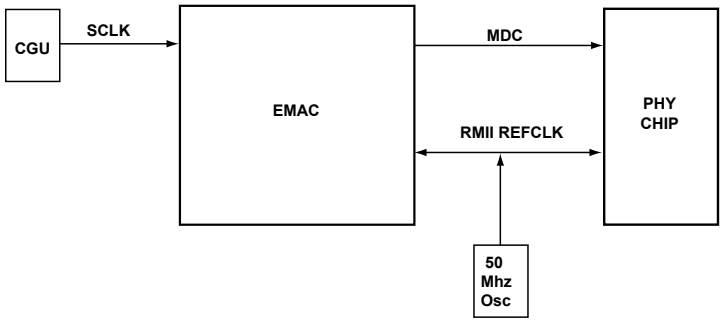


Figure 30-9: EMAC Clock Sources - RMII PHY Interface

EMAC Architectural Concepts

This section explains different architectural concepts relevant to EMAC peripheral, such as EMAC SCB, EMAC DMA, EMAC MTL, EMAC CORE, and others.

EMAC System Crossbar Interface (EMAC SCB)

The EMAC SCB bus interface provides the bus connectivity to support highly effective throughput of data traffic. System bus use is maximized by allowing simultaneous read and write transfers initiated from different DMA channels. The EMAC controller connects directly to the SCB0 crossbar. The following interfaces are available with the design.

- A 32-bit SCB controller interface for reading and writing to and from the application memory.

- A 32-bit SPB target interface for register programming.

Refer to the “System Crossbars (SCB)” chapter for more information on how the crossbar operates. This chapter details only the EMAC-specific information.

Table 30-10: EMAC-SCB Interface Data Transfer Specifications with Crossbar

Specification Term	Comments
1 beat in SCB	SINGLE burst
BLEN4 bursts	4 beats in SCB
BLEN8 bursts	8 beats in SCB
BLEN16 bursts	16 beats in SCB
Bus size	32-bit fixed bus size; equals 1 beat
INCR bursts	Incrementing Bursts
INCR ALIGNED bursts	Incrementing aligned bursts
UNDEF bursts	Undefined burst length
PBL	Programmable Burst Length for DMA

The ***EMAC DMA Read/Write channels with System Crossbar*** figure shows DMA write channel and read channel data paths and their connection to the system crossbar.

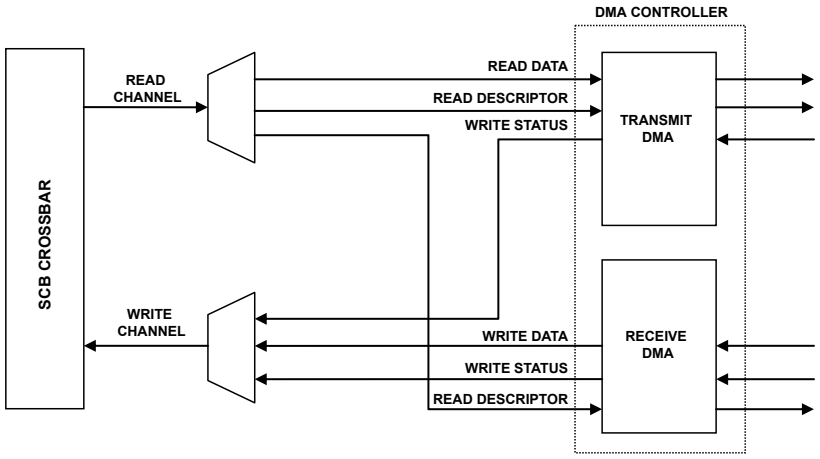


Figure 30-10: EMAC DMA Read/Write channels with System Crossbar

NOTE: Transmit descriptor read and receive descriptor write-back (status update) operations can occur simultaneously. However, transmit descriptor read and write-back operations cannot occur simultaneously. Transmit DMA (or receive DMA) does not initiate the next transfer unless the previous one is complete.

Priority of SCB Requests

The descriptor transfers have higher priority than the data transfers. For example, if there are two bus requests, such as a receive descriptor read and a transmit data read, the receive descriptor read has a higher priority. The next receive data write (after the receive descriptor read) does not depend on the completion of the transmit data-read transfer.

If there are requests for descriptor reads from both DMA channels, they are serviced based on a first-come first-serve. Receive DMA has higher priority if the descriptor-read requests are generated from both the DMA channels in the same clock cycle. Similarly, in the write channel, descriptor writes from DMA have higher priority than the data-write transfers for the receive DMA.

When programs enable the AV or DCB feature, the DMA may contain multiple channels. If the requests are generated from different channels simultaneously in the same clock cycles, the request priority among the channels is: Channel 7 (if present), Channel 6, Channel 5, Channel 4, Channel 3, Channel 2, Channel 1, and Channel 0.

SCB Interface Programming Options

The SCB bus interface supports the following programmable options for the EMAC module. These options are available using the EMAC_DMA_SYSBMODE and EMAC_DMA[n]_CTL registers.

- ***Outstanding transactions.*** The EMAC-SCB supports up to 16 (EMAC0) / 4 (EMAC1) outstanding read/write requests on the SCB bus. Software can control these requests by programming the EMAC_DMA_SYSBMODE.WR_OSR_LMT and EMAC_DMA_SYSBMODE.RD_OSR_LMT bits. Maximum outstanding requests = EMAC_DMA_SYSBMODE.WR_OSR_LMT + 1 or EMAC_DMA_SYSBMODE.RD_OSR_LMT + 1.
- ***Allowed burst sizes.*** The allowed burst sizes are 4 (EMAC_DMA_SYSBMODE.BLEN4), 8 (EMAC_DMA_SYSBMODE.BLEN8), 16 (EMAC_DMA_SYSBMODE.BLEN16) and the SINGLE burst. The EMAC-SCB uses only those burst sizes configured by the program (through the EMAC_DMA_SYSBMODE register) for data transfer through the SCB bus. Data transfers are restricted to the maximum burst size from this list of programmed burst sizes.
- ***Burst splitting and burst selection.*** The EMAC-SCB splits the DMA requests into multiple bursts on the SCB system bus. Splitting is based on DMA count and software controllable burst enable bits (EMAC_DMA_SYSBMODE.FB, BLEN16, BLEN8, BLEN4) as well as burst types (INCR and INCR_ALIGNED). Burst types are also controllable through the software. Burst length select priority is in the sequence: FB, BLEN16, BLEN8, BLEN4
- ***INCR burst type***
  - If the EMAC_DMA_SYSBMODE.FB bit is disabled, the EMAC-SCB always chooses the maximum allowed burst length based on the BLEN16, BLEN8 and BLEN4, bits. When the DMA requests are not multiples of the maximum allowed burst length, the SCB can choose a burst-length of any value less than the maximum enabled. (All lesser burst-length enables are redundant). For example, when length bits are enabled and the DMA requests a burst of 42 beats, the SCB splits it into three bursts of 16, 16 and 10 beats respectively.

- If EMAC_DMA_SYSBMODE.FB is enabled, then the burst length is based on the priority of the enabled bits in the following order: BLEN16, BLEN8 and BLEN4. When the DMA requests a burst transfer, the SCB interface splits the requested bursts into multiple transfers using only the enabled burst lengths. This splitting can occur when the requested burst is not a multiple of the maximum enabled burst. If it cannot choose any of the enabled burst lengths, then it selects the burst length as 1.

For example, the DMA requests a burst transfer of 42 beats, the SCB interface splits it into multiple bursts of size 16, 16, 8, 1 and 1 beats respectively. (In this case, the allowed burst sizes are enabled and the sequence is in decreasing burst sizes).

- ***INCR_ALIGNED burst type.*** When the address-aligned burst-type is enabled (EMAC_DMA_SYSBMODE.AAL), the SCB interface splits the DMA requested bursts. The "INCR Burst Type" section explains burst splitting conditions further. Each burst-size aligns to the least significant bits of the start address. The SCB interface initially generates smaller bursts so that the remaining transfers move with the maximum (enabled) fixed burst lengths.

  For example, in the same setting as explained earlier for the EMAC_DMA_SYSBMODE.FB bit disabled, the DMA requests a burst size of 42 beats at the start address of 0x000003A4 (EMAC_DMA_SYSBMODE.BLEN4, EMAC_DMA_SYSBMODE.BLEN8 and (EMAC_DMA_SYSBMODE.BLEN16 are enabled). The SCB starts the first transfer with size 3 such that the address of the next burst is aligned (0x000003B0) for a burst of 16. Therefore, the sequence of bursts is 3, 16, 16, and 7, respectively.

  When the EMAC_DMA_SYSBMODE.FB bit is enabled, then (having a start address of 0x000003A4 with 42 beats), the sequence of burst transfers is 1, 1, 1, 16, 16, 4, and 3 respectively. The sequence of smaller bursts at the beginning is used to align the address to the next higher enabled burst-lengths programmed in the register.

Burst Reordering and Data Interleaving

The SCB protocol allows reordering of data transfers with different SCB IDs with respect to the sequence of requests. It also allows data interleaving between transfers of different SCB IDs for read channels. In the EMAC, the requests from each internal requester (receive DMA and transmit DMA) are generated with different SCB IDs. Therefore, reordering and data interleaving can be performed as two DMAs operate independently and are allocated different address space on the target memory.

The SCB requester supports reordering and interleaving from the SCB completer on the read channel. Each DMA internally ensures that a read request and a write request is never generated at the same time to the same address (it can occur for only descriptor accesses). The transmit DMA ensures that transmit descriptor reads, transmit data reads, and transmit descriptor write-back requests are generated only after the previous requests are complete. The receive DMA generates only descriptor reads on the read channel with a different ID. Therefore, data reordering and interleaving on the read channel does not cause any problems in the MAC.

Posted Writes

In posted writes, the write channel of SCB requester transfers an OKAY response to the DMA as soon as the last cycle or beat of data is accepted by the SCB interconnect. The SCB requester sends this response without waiting for the response from the target SCB completer on the write response channel.

In non-posted writes, the SCB controller interface transfers the response received from the SCB target channel to the DMA. To transfer the Ethernet data to system memory, the receive DMA always issues posted write requests to the SCB controller interface. This enables pipelining of the data requests without delays.

The SCB controller does not support out-of-order write transfers. However, the receive DMA ensures that the sequence is maintained for descriptor writes, descriptor reads, and data transfers. For writing descriptors (status or time stamp), the DMA always issues the non-posted write requests. This is required because the transfer complete interrupt is generated based on the descriptor writes for which the DMA needs completion response from the memory target. This ensures no race condition between the hardware and the software because the interrupt is generated only after the data and descriptor are written to the target memory.

SCB Bus Transaction Status

The SCB uses the `EMAC_DMA_DBG_STAT0.AXRHSTS` and `EMAC_DMA_DBG_STAT0.AXWHSTS` bits to indicate whether the channel is active or not.

DMA Controller (EMAC DMA)

The DMA has independent transmit and receive engines, and a CSR (control and status register) space. The transmit engine transfers data from the system memory to the FIFO (MTL), whereas the receive engine transfers data from the FIFO (MTL) to the system memory. The DMA engine uses descriptors to efficiently move data from source to destination with minimal processor core intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The DMA controller can be programmed to interrupt the application for situations such as packet transmit and receive transfer completion, and other normal or error conditions.

The DMA and the application communicate through two internal data structures:

1. Control and status registers (CSR).
2. Descriptor lists and data buffers

The EMAC supports the ring structure for DMA descriptor as shown in the following figure.

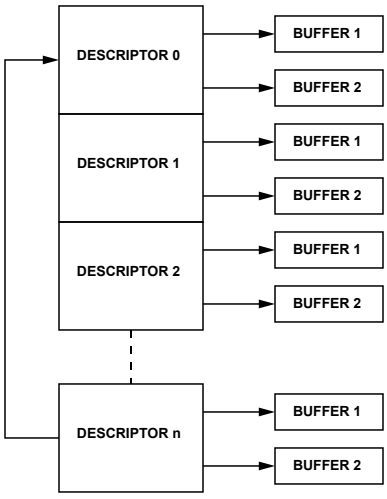


Figure 30-11: Descriptor Ring Structure

In the ring structure, descriptors are separated by the number programmed in the DSL field (32-bit unit) of the EMAC_DMA[n]_CTL registers. The application needs to program the total ring length, that is, the total number of descriptors in ring span in the following registers of a DMA channel:

- Transmit Descriptor Ring Length register (EMAC_DMA[n]_TXDSC_RLEN)
- Receive Descriptor Ring Length register (EMAC_DMA[n]_RXCTL2.RDRL)

The EMAC_DMA[n]_TXDSC_TLPTR or EMAC_DMA[n]_RXDSC_TLPTR register contains the pointer to the descriptor address (N). The base address registers (EMAC_DMA[n]_TXDSC_ADDR and EMAC_DMA[n]_RXDSC_ADDR) and the current descriptor pointer registers (EMAC_DMA[n]_TXDSC_CUR or EMAC_DMA[n]_RXDSC_CUR) determine the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (N – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA goes into the Suspend mode when this condition occurs. The application must perform a write to the Descriptor Tail pointer register and update the tail pointer so that the following condition is true:

```
Current Descriptor Pointer < Descriptor Tail Pointer;
```

The DMA automatically wraps around the base address when the end of ring is reached, as shown in the *DMA Descriptor Ring* figure.

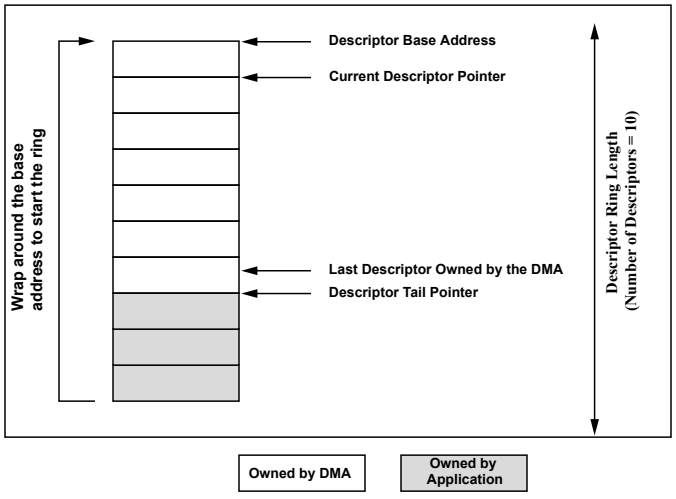


Figure 30-12: DMA Descriptor Ring

For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

The DMA supports up to 8 transmit and 8 receive descriptor lists (or DMA channels). The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers in the

system memory. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet but cannot exceed a single packet. Buffers contain only data. Buffer status is maintained in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP (End of Packet) is detected.

For more information on the descriptor, see DMA Descriptors section that describes the descriptor structure and how the DMA accesses the descriptors.

DMA Bursts Using the SCB Interface

The DMA engines attempt to transfer data in maximum size bursts that are programmed using the EMAC_DMA[n]_TXCTL and EMAC_DMA[n]_RXCTL bits. The receive and transmit descriptors are always accessed at the maximum possible burst length to read 16 bytes (limited by PBL or 32). The burst transfers initiated by the DMA can be split into multiple burst transfers as per the SCB protocol requirements and the settings of the EMAC_DMA_SYSBMODE register. The transmit DMA initiates a data transfer only when sufficient space is available in the MTL transmit queue to accommodate either of the following:

- Bytes corresponding to the configured burst (PBL x 4)
- Remaining bytes in the transmit Buffer without EOP
- Number of bytes until EOP

The receive DMA initiates a data transfer in the following conditions:

- Sufficient data is available in MTL receive queue to accommodate the configured burst
- EOP (when it is less than the configured burst length) is detected in the receive queue

The DMA indicates the start address and the number of transfers required to the SCB interface. When the SCB interface is configured for a fixed-length burst, it transfers the data by using the best combination of INCR4, INCR8, or INCR16 and SINGLE transactions. If EOP is reached before the fixed-burst ends on the SCB interface, dummy transfers are performed in-order to complete the fixed-burst. Otherwise, the DMA transfers the data using INCR (undefined length) and SINGLE transactions. When the SCB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer initiated by the SCB is less than or equal to the size of the configured PBL. Therefore, all subsequent beats start at an address that is aligned to the configured PBL.

Application Data Buffer Alignment

The transmit and receive data buffers do not have any restrictions on start address alignment. The start address for the buffers aligns to any of the 4 bytes. However, the DMA always initiates write transfers with address aligned to the bus width and dummy data (old data) in the invalid byte lanes. This typically happens during the transfer of the beginning or end of an Ethernet packet. The software driver should discard the dummy bytes based on the start address of the buffer and size of the packet.

Example for Buffer Read

If the transmit buffer address is 0x0002 and 15 bytes must transfer, the DMA reads 5 full words (5 x 32-bit data) from address 0x0000. However, when transferring data to the EMAC transmit FIFO, the extra bytes (the first 2 bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always transfers a full 32-bit data to the transmit FIFO unless it is the end-of-frame.

Example for Buffer Write

If the receive buffer address is 0x0002 and 15 bytes of a received frame must transfer, the DMA writes 5 full words (5 x 32-bit data) to address 0x0000. However, the first 2 bytes of first transfer and the last 3 bytes of the third transfer have dummy data.

Buffer Size Calculations

The DMA engines do not update the size fields in the transmit and receive descriptors alone. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver must perform the size calculations.

The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES2) towards the MAC. If a descriptor is marked as first (FD bit of TDES3 is set), the DMA marks the first transfer from the buffer as SOP. If a descriptor is marked as last (LD bit of TDES3), the DMA marks the last transfer from that data buffer as EOP to the MTL.

The receive DMA transfers data to a buffer until the buffer is full or the end of packet is received from the MTL. When the FD bit of a descriptor is set, the amount of valid data in a buffer is accurately indicated by the buffer size field (programmed in the `EMAC_DMA[n]_RXCTL` registers) minus the data buffer pointer offset. The offset is zero when the data buffer pointer is aligned to the data bus width (32 bits). If a descriptor is marked as last, the buffer may not be full (as indicated by the buffer size in Bits [14:1] of the `EMAC_DMA[n]_RXCTL` registers). To compute the amount of valid data in this final buffer, the driver must:

- Read the packet length (PL bits of RDES3 [14:0]), and
- Subtract the sum of the buffer sizes of the preceding buffers in this packet

The receive DMA always transfers the start of next packet with a new descriptor.

DMA Transmit Operation

The following sections describe how the transmission process works for direct memory access on the EMAC controller.

Default (Non-OSF) Mode

The following sequence describes the default process for DMA transmit works.

1. The application sets up the transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet packet data.
2. The application advances the descriptor tail pointer offset value of the transmit channel.

3. While in the run state, the DMA runs an arbitration cycle to select the next transmit DMA channel from which the packets requiring transmission should be processed.
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended, the `EMAC_DMA[n]_STAT.TBU` and `EMAC_DMA[n]_STAT.TEB` bits are set, and the transmit engine proceeds to step 11:
   - The descriptor is flagged as owned by the application (TDES3 [31] = 1'b0)
   - The descriptor tail pointer is equal to the current descriptor pointer in ring descriptor list mode
   - An error condition occurs
6. If the acquired descriptor is flagged as owned by DMA (TDES0 [31] = 1#b1), the DMA decodes the transmit data buffer address from the acquired descriptor.
7. The DMA fetches the transmit data from the application memory and transfers the data to the MTL for transmission.
8. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end of- Ethernet-packet data is transferred to the MTL.
9. When packet transmission is complete, if the IEEE 1588 time stamp feature was enabled for the packet (as indicated in the transmit status), the time stamp value obtained from MTL is written to the transmit descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this transmit descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step.

   If time stamp feature is not enabled for this packet, the DMA does not alter the contents of TDES0 and TDES1.

   NOTE: By default, the transmit DMA places subsequent request after the current descriptor write request is completed to the descriptor memory. If posted descriptor write is enabled by setting the `EMAC_DMA_MODE.DSPW` bit, the transmit DMA places subsequent request after the current descriptor write request is placed without waiting for its completion, thereby improving the throughput.
10. The transmit interrupt (`EMAC_DMA[n]_STAT.TI`) bits are set after completing transmission of a packet that has interrupt on completion (TDES2[31]) set in its last descriptor. The DMA engine returns to step 3.
11. In the suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the `EMAC_DMA[n]_TXDSC_TLPTR` registers when it receives a transmit poll demand and the underflow interrupt status bit is cleared. If the application stopped the DMA by clearing transmit interrupt bit, the DMA enters the stop state.

    NOTE: In non-OSF mode, it is not required for the driver/application to extract the packet status and release the descriptor for the packet, even though FD is set to 1. The driver must track the descriptor that has OWN = 0 and LD = 1, as 1; such a descriptor contains the transmission status for the packet.

This indicates that descriptors up to this descriptor can be released for reuse by the driver/application for subsequent packets.

OSF Mode

While in the run state, the transmit process can simultaneously acquire two packets without closing the status descriptor of the first packet (if the EMAC_DMA[n]_TXCTL.OSF bit is set). As the transmit process finishes transferring the first frame, it immediately polls the transmit descriptor list for the second packet. If the second packet is valid, the transmit process transfers this frame before writing the status information of the first packet.

In OSF mode, the run state transmit DMA operates in the following sequence.

1. The DMA operates as described in steps 1–7 of default (non-OSF) mode
2. Without closing the previous last descriptor of the frame, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into suspend mode and skips to Step 7.
4. The DMA fetches the transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and time stamp of previous packet. When the status is available, the DMA writes the time stamp to TDES0 and TDES1 if such time stamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES1, thus closing the descriptor.

   If the time stamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES0 and TDES1.
6. If enabled, the transmit interrupt is set; the DMA fetches the next descriptor, and then proceeds to Step 3 (when status is normal). If the previous transmission status shows an underflow error, the DMA goes into suspend mode (Step 7).
7. In suspend mode, if a pending status and time stamp are received from the MTL, the DMA:
   - Writes the time stamp (if enabled for the current packet) to TDES0 and TDES1
   - Writes the status to the corresponding TDES1
   - Sets relevant interrupts and returns to Suspend mode

   If no status is pending and the application stopped the DMA by clearing the EMAC_DMA[n]_TXCTL.ST bits, the DMA enters the Stop state.
8. The DMA can exit suspend mode and enter the run state (goes to step 1 or step 2 depending on pending status) only after receiving a transmit poll demand in transmit descriptor tail pointer register of the corresponding channel.

NOTE: The DMA fetches the next descriptor before closing the current descriptor. Therefore, the descriptor ring length must be more than 2. A minimum descriptor length of 4 is recommended.

In OSF mode, all except the last descriptor is closed immediately. The last descriptor is closed after the packet is transmitted on the line. Therefore, to minimize the complexity, the transmission status and only the required control bits are updated in the pending last descriptor of the previous packet.

Transmit Packet Processing

The transmit DMA engine expects that the data buffers contain complete Ethernet packets, excluding the preamble, pad bytes, and FCS fields. The DA, SA, and type/length fields contain valid data. When the transmit descriptor indicates that the MAC must disable CRC or PAD insertion, the buffer must have complete Ethernet packets (excluding preamble), including the CRC bytes.

Packets can be data-chained and can span several buffers. Packets must be delimited by the first descriptor (TDES3[29]) and the last descriptor (TDES3[28]). As transmission starts, the first descriptor must have TDES3[29] set. When this occurs, the packet data is transferred from the application buffer to the MTL transmit queue. Concurrently, iwhen the current packet has the last descriptor (TDES3[28]) clear, the transmit process attempts to acquire the next descriptor. The transmit process expects this descriptor to have TDES3[29] clear. When TDES3[28] is clear, it indicates an intermediary buffer. When TDES3[28] is set, it indicates the last buffer of the packet.

After the last buffer of the packet has been transmitted, the DMA writes back the final status information to the transmit descriptor 3 (TDES3) word of the descriptor that has the last descriptor bit set in transmit descriptor 3 (TDES3[28]). At this time, if interrupt on completion (TDES2[31]) is set, the `EMAC_DMA[n]_STAT.TI` bit is set, the next descriptor is fetched, and the process repeats. The actual packet transmission begins after either of the following:

- The MTL transmit queue has reached a programmable transmit threshold (programmed in the `EMAC_TQ0_OPMODE.TTC` bit field)
- A full packet is contained in the FIFO

Programs can also use the store-and-forward mode (`EMAC_TQ0_OPMODE.TSF` bit is set). In this mode, descriptors are released (OWN bit in TDES0[31] clears) when the DMA finishes transferring the packet.

NOTE: To ensure proper transmission of a packet and the next packet, specify a non-zero buffer size for the transmit descriptor that has the last descriptor (TDES3[28]) set.

Transmit Polling Suspended

Either of the following conditions suspends transmit polling:

- The DMA detects a descriptor owned by the application (TDES3[31] = 0). To resume, the driver must give descriptor ownership to the DMA and then issue a poll demand command by writing the `EMAC_DMA[n]_TXDSC_TLPTR` registers. If the DMA goes into the SUSPEND state because of this condition, the `EMAC_DMA[n]_STAT.NIS` and `EMAC_DMA[n]_STAT.TBU` bits are set.
- A packet transmission is aborted when a transmit error is detected because of underflow.

  The appropriate transmit descriptor 3 (TDES3) bit is set. When this condition occurs, the following bits are set and the information is written to transmit descriptor 0, causing the suspension:

- EMAC_DMA[n]_STAT.AIS
- Transmit Underflow bit of corresponding queue (EMAC_Q0_INT_CTLSTAT.TXUNFIS bits)

In all conditions, the position in the transmit list is retained. The retained position is that of the descriptor following the last descriptor closed by the DMA. The driver must explicitly issue a transmit poll demand command after rectifying the suspension cause.

DMA Transmit Channel Arbitration

An arbiter provides access to multiple DMAs trying to access the Bus Interface Unit (BIU).

When there is a request in the transmit DMA, the DMA arbiter checks the type of the request: packet buffer fetch or descriptor fetch request. The descriptor fetch requests have higher priority than the buffer requests. Therefore, when there is a descriptor fetch request, the DMA arbiter acknowledges the DMA channel that is requesting for a descriptor fetch. If there is no descriptor fetch request, the arbiter looks for packet buffer fetch requests.

The DMA arbiter acknowledges the descriptor fetch request of one DMA channel at a time. Descriptor fetch requests are granted using a fixed priority with the higher channel having higher priority (channel 1 having priority over channel 0, channel 2 having priority over channel 1 and so on). For packet buffer fetches, the DMA arbiter uses the programmed channel weight and priority to decide which channel to acknowledge. The DMA arbiter performs a burst-by-burst arbitration based on one of the following algorithms, which can be selected by programming the EMAC_DMA_MODE.TAA bit field.

- Weighted Strict Priority (WSP). In WSP arbitration mode, the arbiter first processes channel 7 (or the last selected channel) followed by channel 6, channel 5, and so on. If a channel does not have a frame to transmit, the weight of the channel is reassigned to channel 7 (or the last selected channel). If channel 7 has no frames to transmit, the remaining weight is assigned to channel 6 and so on.
- Weighted Round Robin (WRR). In WRR arbitration mode, the arbiter first selects the channel with the highest weight programmed, and then the channel with next highest weight, and so on. If any channel does not have a frame to transmit, the weight of that channel is equally distributed to all channels that have frames to transmit. The weight of the channel is programmed in the EMAC_DMA[n]_TXCTL.TCW (transmit channel weight) bit fields.
- Fixed priority (FP). In fixed priority mode, channel 0 has the lowest priority and the last selected channel has the highest priority. The weight programmed in the TCW bits of a channel is ignored. In WSP or WRR arbitration, the channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel. When a channel completes all the DMA burst transfers, the arbiter grants the bus to the next channel.

In WSP or WRR arbitration, the channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel. When a channel completes all the DMA burst transfers, the arbiter grants the bus to the next channel.

DMA Receive Operation

In the receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel. The ARI data at the start of the frame indicates the channel number to which the current frame must be written. If only one DMA channel is selected, this information is not provided.

The following list shows the reception sequence for the receive DMA engine:

1. The application sets up the receive descriptors (RDES0-RDES3) and the OWN bit (RDES3[31]). The application must set the correct value in the EMAC_DMA[n]_RXDSC_TLPTR register.
2. When the EMAC_DMA[n]_RXCTL.SR bit is set, the DMA enters the run state. The DMA looks for free descriptors based on the EMAC_DMA[n]_RXDSC_ADDR and EMAC_DMA[n]_RXDSC_TLPTR register values. If there are no free descriptors, the DMA channel enters the suspend state and goes to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from the acquired descriptors.
4. If IEEE 1588 time stamping is enabled and the time stamp is available for the previous packet, the DMA writes the time stamp (if available) to the RDES0 and RDES1 registers of the current descriptor and sets the CTXT bit field (RDES3[30]).
5. The DMA processes the incoming packets and places these in the data buffers of acquired descriptor.
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
7. The DMA takes the status of the receive frame from the MTL and writes the status word to current descriptor with the OWN bit cleared and the last descriptor bit set.
8. The DMA writes the frame length to RDES3 and VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. If the IEEE 1588 time stamp feature is enabled, the DMA stores the time stamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).
10. If more descriptors are available in the receive DMA descriptor ring, go to step 3; otherwise, go to the suspend state (step 11).
11. The receive DMA exits the suspend state when a receive poll demand is given, and the application advances the receive tail pointer register of a channel.

The engine proceeds to step 2 and refetches the next descriptor.

Receive Descriptor Acquisition

The receive engine always attempts to acquire an extra descriptor in anticipation of an incoming packet. descriptor acquisition is attempted if any of the following conditions is satisfied:

- The EMAC_DMA[n]_RXCTL.SR bit is set immediately after being placed in the run state.

- The EMAC_DMA[n]_RXDSC_TLPTR register value is ahead of the current descriptor acquired by the receive DMA.
- The controller has completed packet reception, but the current receive descriptor is not yet closed.
- A receive poll demand is issued (update of the tail pointer register).

Receive Packet Processing

The sequence for processing a receive packet is as follows:

1. The MAC transfers the received packets to the MTL memory only if the packet passes the address filter. If the packet fails the address filtering, it is dropped in the MAC block (unless the EMAC_MACPKT_FILT.RA bit is set).
2. If the packet size is greater than or equal to the configurable threshold bytes set for the receive queue of the MTL, or when the complete packet is written to the queue in the store-and-forward mode, the MTL block requests the DMA block to begin transferring the packet data to the receive buffer pointed to by the current descriptor.

   Because of collision or premature termination, packets smaller than 64 bytes are removed from the MTL receive queue.
3. When the DMA application interface (SCB or MDC) is ready, it transfers the data and sets the following:
   - If the packet fits in a single descriptor, the DMA sets both last descriptor (RDES3[28]) and first descriptor (RDES3[29]).
   - If the packets fits into more than one descriptor, the DMA sets the first descriptor (RDES3[29]) to delimit the packet.
4. The DMA releases the descriptors by resetting the OWN (RDES3[31]) bit to 1'b0, either because the receive is full or the last segment of the packet is transferred to the receive buffer. The received packet's status is updated in the last descriptor.
5. If the interrupt enabled on completion (RDES3[30]) bit is set in any of the descriptors between the first and last descriptor of the packet and the EMAC_DMA[n]_IEN.RIE bit is set, the DMA sets the EMAC_DMA[n]_STAT.RI bit.

The same process repeats unless the DMA encounters a descriptor flagged as being owned by the application or when there are no more descriptors in the ring. When the DMA finds a descriptor owned by the application and if the EMAC_DMA[n]_IEN.RBUE bits are set, the receive process sets the EMAC_DMA[n]_STAT.RBU bits and then enters the suspend state. The position in the receive list is retained.

DMA Error Response

For any data transfer initiated by a DMA channel, if the completer replies with an error response, the DMA stops all operations and updates the error bits and the EMAC_DMA[n]_STAT.FBE bits. The application can either perform a reset to EMAC or re-initialize the DMA descriptor list and start again. The rest of the DMA channels are not affected by such errors.

DMA Descriptors

The DMA in the EMAC transfers data based on a linked list of descriptors. The descriptor addresses must be aligned to the 32-bit bus width. The application creates the descriptors in the system memory. The EMAC supports the following two types of descriptors.

- Normal Descriptor: Normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.
- Context Descriptor: Context descriptors are used to provide control information applicable to the packet to be transmitted. Each normal descriptor contains two buffers and two address pointers. These buffers enable the EMAC to be compatible with various types of memory management schemes.

NOTE: There is no limit for the number of descriptors that can be used for a single packet.

For basic details that describe the descriptor ring structure, see DMA Controller (EMAC DMA).

Transmit Descriptor

The DMA in EMAC requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit packet basis. The transmit normal descriptor has two formats: read format and write-back format.

Transmit Normal Descriptor (Read Format)

The *Transmit Normal Descriptor (Read Format)* figure shows the read format for a transmit normal descriptor. The following tables describe the read format for the transmit normal descriptors: TDES0, TDES1, TDES2, and TDES3.

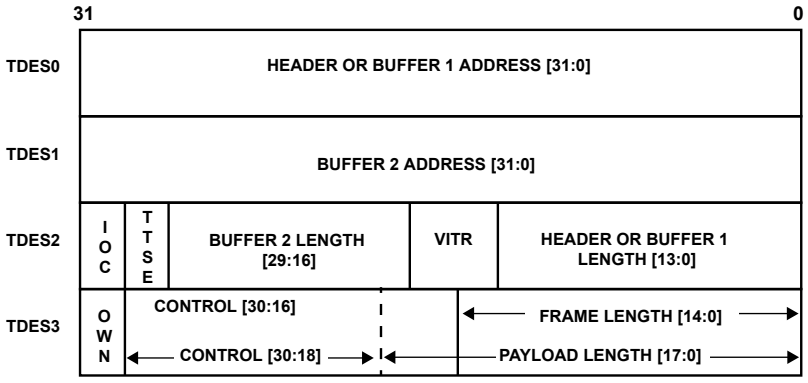


Figure 30-13: Transmit Normal Descriptor (Read Format)

Table 30-11: TDES0 Normal Descriptor (Read Format)

Bit	Name	Description
31–0	BUF1AP	Buffer 1 Address Pointer or TSO Header Address Pointer. These bits indicate the physical address of buffer 1. These bits also indicate the TSO header address pointer when the following bits are set: • TSE bit of TDES3

Table 30-11: TDES0 Normal Descriptor (Read Format) (Continued)

Bit	Name	Description
		• FD bit of TDES3

Table 30-12: TDES1 Normal Descriptor (Read Format)

Bit	Name	Description
31–0	BUF2AP	Buffer 2 or Buffer 1 Address Pointer. These bits indicate the physical address of buffer 2 when a descriptor ring structure is used. There is no limitation for the buffer address alignment.

Table 30-13: TDES2 Normal Descriptor (Read Format)

Bit	Name	Description
31	IOC	Interrupt on Completion. This bit controls the setting of TI and ETI status bits in the EMAC_DMA[n]_STAT register. When ETIC = 1 and TDES2[LD] = 0, this bit sets the ETI bit. When TDES3[LD] = 1, this bit sets the TI status bit.
30	TTSE/TMWD	Transmit Time stamp Enable or External TSO Memory Write Enable. This bit enables the IEEE1588 time stamping for the transmit packet referenced by the descriptor, if the TSE bit is not set. If the TSE bit is set and external TSO memory is enabled, setting this bit disables external TSO memory writing for this packet.
29:16	B2L	Buffer 2 Length. The driver sets this field. When set, this field indicates buffer 2 length.
15:14	VTIR	VLAN Tag Insertion or Replacement. These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC pad control bits appropriately when VLAN tag insertion, Replacement, or Deletion is enabled for the packet. The following list describes the values of these bits: • 00 = Do not add a VLAN tag • 01 = Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. • 10 = Insert a VLAN tag with the tag value programmed in the MAC_VLAN_Incl register or context descriptor • 11 = Replace the VLAN tag in packets with the tag value programmed in the MAC_VLAN_Incl register or context descriptor. This option should be used only with the VLAN packets. These bits are valid when the enable SA and VLAN insertion on transmit option is selected while configuring the core.
13:0	HL or B1L	Header Length or Buffer 1 Length. For header length only bits [9:0] are taken. The size 13:0 is applicable only when interpreting buffer 1 length. If the TCP segmentation offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length in bytes from the Ethernet source address until the end of the TCP header. The maximum header length supported for the TSO feature is 1023 bytes. If the TCP segmentation offload feature is not enabled, this field is equal to buffer 1 length

Table 30-14: TDES3 Normal Descriptor (Read Format)

Bit	Name	Description
31	OWN	Ownership. When set, this bit indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit either when it completes the transfer of data given in the associated buffers.
30	CTXT	Context Type. This bit should be set to 1'b0 for normal descriptor.
29	FD	First Descriptor. When set, this bit indicates that the buffer contains the first segment of a packet.
28	LD	Last Descriptor. When this bit is set, it indicates that the buffer contains the last segment of the packet. When this bit is set, the B1L or B2L field should have a non-zero value.
27:26	CPC	Disable CRC. When this bit is set, the EMAC does not append a cyclic redundancy check (CRC) to the end of the transmitted frame. • 00 = CRC and Pad Insertion. The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packet of length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. • 01 = CRC Insertion (Disable Pad Insertion). The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the transmit buffer, that is, the packet being transferred from the transmit buffer is of length greater than or equal to 60 bytes. • 10 = Disable CRC Insertion. The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the transmit buffer. • 11 = CRC Replacement. The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the transmit buffer. This field is valid only for the first descriptor. Note: When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.

Table 30-14: TDES3 Normal Descriptor (Read Format) (Continued)

Bit	Name	Description
25:23	SAIC	SA Insertion Control. When set, these bits request the MAC to add or replace the source address field in the Ethernet packet with the value given in the MAC address 0 register. The application must set the CRC pad control bits appropriately when SA insertion control is enabled for the packet. Bit 25 specifies the MAC address register (1 or 0) value that is used for source address insertion or replacement. Bits [24:23] encodings are as follows • 00 = Do not include the source address • 01 = Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. • 10 = Replace the source address. For reliable transmission, the application must provide frames without source addresses. • 11 = Reserved These bits are valid when the Enable SA and VLAN Insertion on Tx option is selected while configuring the core and when the First descriptor (FD) bit (TDES3 [29]) is set.
22:19	SLOTNUM or THL	SLOTNUM: Slot Number Control bits in AV mode. These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field in the `EMAC_DMA[n]_SFN_CTLSTAT` register. Fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels. THL: TCP/UDP Header Length. If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. The value must be equal to 2 for UDP header. This field is valid only for the first descriptor
18	TSE	TCP Segmentation Enable. When this bit is set, the DMA performs the TCP/UDP segmentation or UDP fragmentation for a packet depending on the TSE_MODE[1:0] bit of the `EMAC_DMA[n]_TXCTL` register. This bit is valid only if the FD bit is set.
17:16	CIC/TPL	Checksum Insertion Control or TCP Payload Length. These bits control the checksum calculation and insertion. Bit encodings are as follows: • 00 = Checksum insertion disabled. • 01 = Only IP header checksum calculation and insertion are enabled • 10 = IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware • 11 = IP header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware When the TSE bit is set, this field contains the upper bits [17:16] of the TCP payload (or IP payload for UDP fragmentation). This allows the TCP/UDP packet length field to be spanned across TDES3[17:0] to provide 256 KB packet length support. This field is valid only for the first descriptor

Table 30-14: TDES3 Normal Descriptor (Read Format) (Continued)

Bit	Name	Description
15	TPL	Reserved or TCP Payload Length. When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is bit 15 of the TCP payload length [17:0]. This field is valid only when the enable TCP segmentation offloading for TCP/IP packets option is selected while configuring the core
14:0	FL/TPL	Frame Length or TCP Payload Length. This field is equal to the length of the packet to be transmitted in bytes. When the TSE bit is not set, this field is equal to the total length of the packet to be transmitted: `Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length` When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length in case of segmentation and IP payload in case of UDP fragmentation. In case of segmentation, this length does not include Ethernet header or TCP/UDP/IP header length. In case of fragmentation, this length does not include Ethernet header and IP header. When DWRR/WFQ algorithm is NOT enabled, value written into this field is not used when TSE = 0.

Transmit Normal Descriptor (Write-Back Format)

The write-back format of the transmit descriptor includes time stamp low, time stamp high, OWN, and status bits.

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and time stamp information for the corresponding transmit packet.

The *Transmit Normal Descriptor (Write-Back Format)* figure illustrates the write-back format of the transmit descriptor.

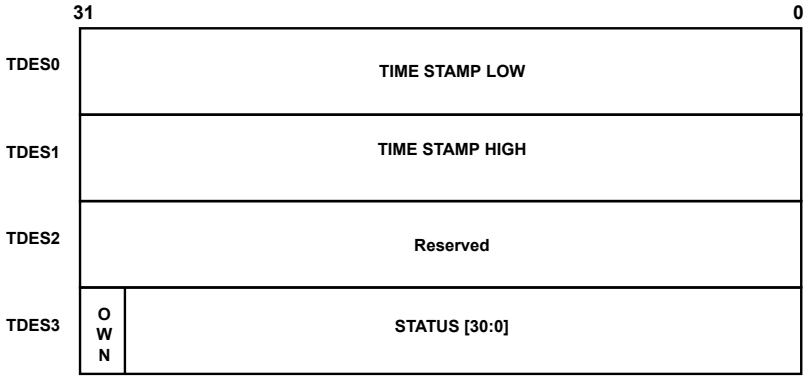


Figure 30-14: Transmit Normal Descriptor (Write-Back Format)

TDES0 Normal Descriptor (Write-Back Format)

As described in the following table, this format is only applicable to the last descriptor of a packet.

Table 30-15: TDES0 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–0	TTSL	Transmit Packet Time Stamp Low. The DMA updates this field with least significant 32 bits of the time stamp captured for the corresponding transmit packet. The DMA writes the time stamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the time stamp only if the last segment bit (LS) in the descriptor is set and the time stamp status (TTSS) bit is set.

TDES1 Normal Descriptor (Write-Back Format)

As described in the following table, this format is only applicable to the last descriptor of a packet.

Table 30-16: TDES1 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–0	TTSH	Transmit Packet Time Stamp High. The DMA updates this field with the most significant 32 bits of the time stamp captured for corresponding transmit packet. The DMA writes the time stamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the time stamp only if the last segment bit (LS) in the descriptor is set and time stamp status (TTSS) bit is set.

TDES2 Normal Descriptor (Write-Back Format)

This format is only applicable to the last descriptor of a packet.

Table 30-17: TDES2 Normal Descriptor (Write-Back Format)

Bit	Name
31–0	Reserved

TDES3 Normal Descriptor (Write-Back Format)

This format is applicable only to the last descriptor of a packet.

Table 30-18: TDES3 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31	OWN	Ownership. When this bit is set, it indicates that the EMAC DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.
30	CTXT	Context Type. This bit should be cleared (=0) for normal descriptor.
29	FD	First Descriptor. When set, this bit indicates that the buffer contains the first segment of a packet.
28	LD	Last Descriptor. This bit is set (=1) for the last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.

Table 30-18: TDES3 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
27–24	Reserved	
23	DE	Descriptor Error. When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the descriptor. Descriptor errors include: • Incorrect sequence from the context descriptor. For example, a location after the first descriptor for a packet. • All 1s • CTXT is set to 1 and the LD or FD bits are set to 1. **NOTE:** When a descriptor error occurs due to all 1s or CTXT, LD, and FD bits set to 1, the transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When the IOC bit in the TDES2 register of the corresponding first descriptor is set to 1, the transmit DMA sets the TI bit in the `EMAC_DMA[n]_STAT` register. **NOTE:** Based on the CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the first descriptor (even if the FD bit is not set) and partial packet is sent.
22–18	Reserved	
17	TTSS	Transmit Time Stamp Status. This status bit indicates that a time stamp has been captured for the corresponding transmit packet. When this bit is set, the TDES0 and TDES1 registers have time stamp values that were captured for the transmit packet. This field is valid only when the last segment control bit (TDES3 [28]) in a descriptor is set. This bit is valid only when the IEEE1588 time stamping feature is enabled; otherwise, it is reserved.
16	EUE	ECC Uncorrectable Error Status. Indicates an ECC uncorrectable error in the TSO memory. **NOTE:** An uncorrectable error in transmit FIFO memory is reported when the packet flushed bit (bit 13, FF) = 1. This is because all such packets are flushed by the EMAC module.
15	ES	Error Summary. Indicates the logical OR of the following bits: TDES3[0]: IP Header Error DES3[14]: Jabber Timeout TDES3[13]: Packet Flush TDES3[12]: Payload Checksum Error TDES3[11]: Loss of Carrier TDES3[10]: No Carrier TDES3[9]: Late Collision TDES3[8]: Excessive Collision TDES3[3]: Excessive Deferral TDES3[2]: Underflow Error This bit is also set when the EUE (bit 16) is set.

Table 30-18: TDES3 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
14	JT	Jabber Timeout. This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the EMAC_MAC_CFG register is not set.
13	FF	Packet Flushed. This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	Payload Checksum Error. This bit indicates that the checksum offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure is caused by: • insufficient bytes in the payload, as indicated by the payload length field of the IP header. • the MTL starts to forward the packet to the MAC transmitter in store-and-forward mode without the checksum being calculated. This error condition only occurs when the transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock. In this case the MTL starts forwarding the packet when the FIFO is full, even in store-and-forward mode. • a bus error is detected during packet transfer. When the full checksum offload engine is not enabled, this bit is reserved.
11	LoC	Loss of Carrier. This bit indicates that a loss of carrier occurred during packet transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the half-duplex mode.
10	NC	No Carrier. This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	LC	Late Collision. This bit indicates that packet transmission was terminated because a collision occurred after the collision window (64 byte times including preamble in MII mode and 512 byte times including preamble and carrier extension in GMII mode). This bit is not valid if underflow error is set.
8	EC	Excessive Collision. This bit indicates that the transmission was terminated after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after first collision and the transmission of the packet is terminated.
7–4	CC	Collision Count. This 4-bit counter value indicates the number of collisions that occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	Excessive Deferral. This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000 Mbps mode or jumbo packet enabled mode) if the DC bit is set in the MAC_Configuration register. When TBS is enabled in full duplex mode and this bit is set, it indicates that the frame has been dropped after the expiry time has reached.

Table 30-18: TDES3 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
2	UF	Underflow Error. This bit indicates that the MAC terminated the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: • The DMA encountered an empty transmit buffer while transmitting the packet • The application filled the MTL transmit FIFO slower than the MAC transmit rate The transmission process enters the suspended state and sets the underflow bit corresponding to a queue in the EMAC_MTL_ISTAT register.
1	DB	Deferred Bit. This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the half-duplex mode.
0	IHE	IP Header Error. When this bit is set, it indicates that the checksum offload engine detected an IP header error. This bit is valid only when transmit checksum offload is enabled. Otherwise, it is reserved. If the COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet type field indicates an IPv4 payload. In full duplex mode, when EST/Qbv is enabled and this bit is set, it indicates the frame drop status due to frame size error or schedule error.

Transmit Context Descriptor

The context descriptor is used to provide the time stamps for one-step time stamp correction, VLAN tag ID for VLAN insertion feature.

The transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the time stamps for one-step time stamp correction and VLAN Tag ID for VLAN insertion feature. Write back is done on a context descriptor only to reset the OWN bit.

NOTE: The VLAN Tag IDs and MSS values, provided by the application in a context descriptor with their corresponding valid bits set, are stored internally by the DMA. When the outer or inner VLAN tag is provided with the valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The inner VLAN tag control input is used only for the next packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

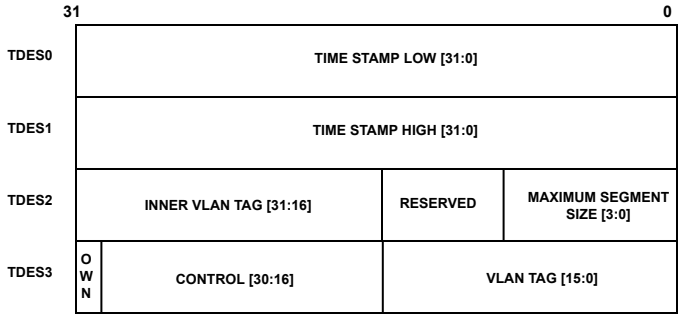


Figure 30-15: Transmit Context Descriptor Format

Table 30-19: TDES0 Context Descriptor

Bit	Name	Description
31–0	TTSL	Transmit Packet Time Stamp Low. For one-step correction, the driver can provide the lower 32 bits of time stamp in this descriptor word. The DMA uses this value as the low word for doing one-step time stamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

Table 30-20: TDES1 Context Descriptor

Bit	Name	Description
31–0	TTSH	Transmit Packet Time Stamp High. For one-step correction, the driver can provide the upper 32 bits of time stamp in this descriptor. The DMA uses this value as the high word for doing one-step time stamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

Table 30-21: TDES2 Context Descriptor

Bit	Name	Description
31–16	IVT	Inner VLAN Tag. When the IVLTV bit of the TDES3 context descriptor is set and the TCMSSV and OSTC bits of the TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN tag that is inserted in the subsequent transmit packets.
15–14	Reserved	
13–0	MSS	Maximum Segment Size. When the enable TCP segmentation offloading for TCP/IP packets option is selected, the driver provides maximum segment size in this field. This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of the TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

Table 30-22: TDES3 Context Descriptor

Bit	Name	Description
31	OWN	Own Bit. When this bit is set, it indicates that the EMAC DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit immediately after the read.
30	CTXT	Context Type. This bit should be cleared (=0) for normal descriptor.

Table 30-22: TDES3 Context Descriptor (Continued)

Bit	Name	Description
29–28	Reserved	
27	OSTC	One-Step Time stamp Correction Enable. When this bit is set, the DMA performs a one-step time stamp correction with reference to the time stamp values provided in TDES0 and TDES1.
26	TCMSSV	One-Step Time stamp Correction Input or MSS Valid. When this bit and the OSTC bit are set, it indicates that the time stamp correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25–24	Reserved	
23	CDE	Context Descriptor Error. When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the context descriptor. Context descriptor errors can be: • Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet. • All 1s • CD, LD, and FD bits set to 1. **NOTE:** When the context descriptor error occurs due to all 1s or the CTXT, LD, and FD bits set to 1, the transmit DMA closes the transmit descriptor with the DE and LD bits set to 1. When the IOC bit in TDES2 of the corresponding first descriptor is set to 1, transmit DMA sets the TI bit in the DMA_CH#_Status register. **NOTE:** Based on the CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the first descriptor (even if the FD bit is not set) and a partial packet is sent. **NOTE:** This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported).
22–20	Reserved	

Table 30-22: TDES3 Context Descriptor (Continued)

Bit	Name	Description
19–18	IVTIR	Inner VLAN Tag Insert or Replace. When these bits are set, the MAC is requested to perform inner VLAN tagging or un-tagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. The following list describes the values of these bits: • 00 = Do not add the inner VLAN tag. • 01 = Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. • 10 = Insert an inner VLAN tag with the tag value programmed in the MAC_Inner_VLAN_Incl register or context descriptor. • 11 = Replace the inner VLAN tag in packets with the tag value programmed in the MAC_Inner_VLAN_Incl register or context descriptor. This option should be used only with the VLAN frames. These bits are valid when the Enable SA and VLAN Insertion on transmit and Enable Double VLAN Processing options are selected.
17	IVLTV	Inner VLAN Tag Valid. When this bit is set, it indicates that the IVT field of TDES2 is valid.
16	VLTV	VLAN Tag Valid. When this bit is set, it indicates that the VT field of TDES3 is valid.
15–0	VT	VLAN Tag. This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the MAC_VLAN_Incl register is reset.

Receive Descriptor

The DMA in the EMAC module attempts to read a descriptor only if the tail pointer is different from the base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC. Otherwise, the performance of the DMA is impacted greatly because of the unavailability of the descriptors. In such situations, the receive FIFO in the MTL becomes full and starts dropping packets.

The following receive descriptors are present:

- Normal descriptors
- Context descriptors

All receive descriptors are prepared by the software and given to the DMA as “Normal” descriptors with the content as shown in Receive Normal Descriptor (Read Format). The DMA reads this descriptor and after transferring a received packet (or part of) to the buffers indicated by the descriptor, the receive DMA closes the descriptor with the corresponding packet status. The format of this status is given in Receive Normal Descriptor (Write-Back Format).

For some packets, the normal descriptor bits are not enough to write the complete status. For such packets, the receive DMA writes the extended status to the next descriptor (without processing or using the buffers/pointers embedded in that descriptor). The format and content of the descriptor write back is described in Receive Context Descriptor.

Receive Normal Descriptor (Read Format)

The read format for a receive normal descriptor is made up of a header or buffer 1 address, reserved field, payload or buffer 2 or next descriptor address, a 30-bit reserved filed, OWN bit, and an interrupt bit.

The *Receive Normal Descriptor Read Format* figure shows the read format for a receive normal descriptor.

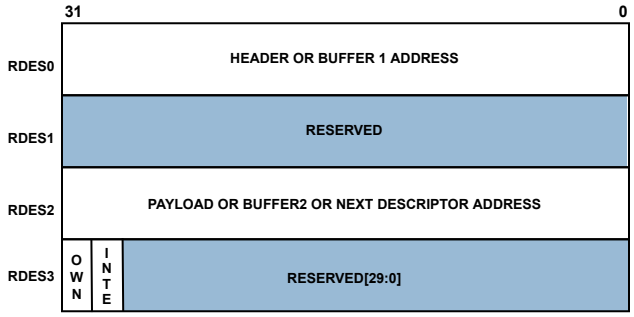


Figure 30-16: Receive Normal Descriptor Read Format

RDES0 Normal Descriptor (Read Format)

The following table explains the read format of the RDES0 normal descriptor.

Table 30-23: Receive Descriptor Fields 0 (RDES0)

Bit	Name	Description
31–0	BUFF1AP	Header or Buffer 1 Address Pointer. These bits indicate the physical address of buffer 1. There are no limit when the SPH bit of the control register of a channel is reset, these bits indicate the physical address of buffer 1. When the SPH bit is set, these bits indicate the physical address of header buffer, where the receive DMA writes the L2/L3/L4 header bytes of the received packet. The application can program a byte-aligned address for this buffer which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a write operation with RDES0[1:0] as zero. However, the packet data is shifted as per actual offset as given by buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data width.

RDES1 Normal Descriptor (Read Format)

The following table describes the read format of the RDES1 normal descriptor.

Table 30-24: Receive Descriptor Fields 1 (RDES1)

Bit	Name	Description
31–0	Reserved	Reserved

RDES2 Normal Descriptor (Read Format)

The following table describes the read format of the RDES2 normal descriptor.

Table 30-25: Receive Descriptor Fields 2 (RDES2)

Bit	Name	Description
31–0	Reserved or BUF1AP	Buffer 2 Address Pointer. These bits indicate the physical address of buffer 2. When the `EMAC_DMA[n]_CTL.SPH` bit is set, the buffer address pointer must be bus width-aligned, that is, RDES2[3:0, 2:0, or 1:0] = 0 corresponding to 128, 64, or 32 bus width. LSBs are ignored internally. When the `EMAC_DMA[n]_CTL.SPH` bit is reset, there is no limitations on the RDES2 value. However, the receive DMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores BUF2AP and writes to the complete location.

RDES3 Normal Descriptor (Read Format)

The following table describes the read format of the RDES3 normal descriptor.

Table 30-26: Receive Descriptor Fields 3 (RDES3)

Bit	Name	Description
31	OWN	Own Bit. When this bit is set, it indicates that the EMAC module owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full
30	IOC	Interrupt Enabled on Completion. When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29–26	Reserved	
25	BUF2V	Buffer 2 Address Valid. When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address, to which the buffer 2 address in RDES2 is pointing, to write received packet data.
24	BUF1V	Buffer 1 Address Valid. When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address pointed to by Buffer 1 address in RDES1 can be used by the DMA to write received packet data.
23–16	Reserved	
15–0	Reserved	Reserved

Receive Normal Descriptor (Write-Back Format)

The *Receive Normal Descriptor Write-Back Format* figure illustrates the write-back format for a receive normal descriptor.

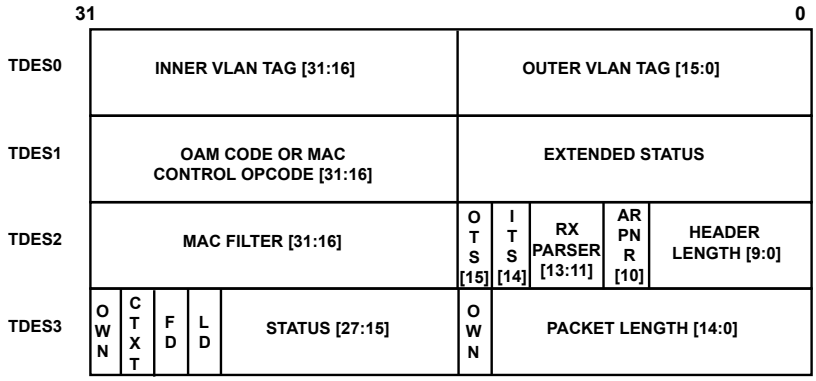


Figure 30-17: Receive Normal Descriptor Write-Back Format

RDES0 Normal Descriptor (Write-Back Format)

The following table describes the write-back format for the RDES0 normal descriptor.

Table 30-27: RDES0 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–16	IVT	Inner VLAN Tag. This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when double VLAN tag processing and VLAN tag stripping are enabled.
15–0	OVT	Outer VLAN Tag. This field contains the outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

RDES1 Normal Descriptor (Write-Back Format)

The following table describes the write-back format for the RDES1 normal descriptor.

Table 30-28: RDES1 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–16	OPC	OAM Subtype Code, or MAC Control Packet opcode. OAM subtype code: If bits [18:16] of RDES3 are set to 3'b111, this field contains the OAM subtype and code fields. MAC Control Packet opcode: The bits [15:8] of RDES3 contains the subtype and bits [7:0] contains the code.
15	TD	Time Stamp Dropped. This bit indicates that the time stamp was captured for this packet but was dropped in the MTL receive FIFO because of overflow. This bit is available only when the time stamp feature is selected. Otherwise, this bit is reserved.
14	TSA	Time Stamp Available. When time stamp is present, this bit indicates that the time stamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the last descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.

Table 30-28: RDES1 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
13	PV	PTP Version. This bit indicates that the received PTP message has the IEEE 1588 version 2 format. When this bit is reset, it indicates the IEEE 1588 version 1 format. This bit is available only when time stamp feature is selected. Otherwise, this bit is reserved.
12	PFT	PTP Packet Type. This bit indicates that the PTP message is sent directly over Ethernet. This bit is available only when time stamp feature is selected. Otherwise, this bit is reserved.
11–8	PMT	PTP Message Type. These bits are encoded to give the type of the message received: • 0000 = No PTP message received • 0001 = SYNC (all clock types) • 0010 = Follow_Up (all clock types) • 0011 = Delay_Req (all clock types) • 0100 = Delay_Resp (all clock types) • 0101 = Pdelay_Req (in peer-to-peer transparent clock) • 0110 = Pdelay_Resp (in peer-to-peer transparent clock) • 0111 = Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) • 1000 = Announce • 1001 = Management • 1010 = Signaling • 1011–1110 = Reserved • 1111 = PTP packet with Reserved message type These bits are available only when the time stamp feature is selected.
7	IPCE	IP Payload Error. When this bit is set, it indicates either of the following: • The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. • The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. • The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.
6	IPCB	IP Checksum Bypassed. This bit indicates that the checksum offload engine is bypassed. This bit is available when the enable receive TCP/IP checksum check feature is selected.
5	IPV6	IPv6 Header Present. This bit indicates that an IPV6 header is detected. When the enable split header feature option is selected and the SPH bit of control register of a channel is set, the IPV6 header is available in the header buffer area to which RDES0 is pointing.

Table 30-28: RDES1 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
4	IPV4	IPV4 Header Present. This bit indicates that an IPV4 header is detected. When the SPH bit of RDES3 is set, the IPV4 header is available in the header buffer area to which RDES0 is pointing.
3	IPHE	IP Header Error. When this bit is set, it indicates either of the following: • The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. • The IP datagram version is not consistent with the Ethernet type value. • Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set. This bit is available when the enable receive TCP/IP checksum check feature is selected.
2–0	PT	Payload Type. These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): • 000: Unknown type or IP/AV payload not processed • 001 = UDP • 010 = TCP • 011 = ICMP • 110 = AV Tagged Data Packet • 111 = AV Tagged Control Packet • 101 = AV Untagged Control Packet • 100 = IGMP if IPV4 Header Present bit is set else DCB (LLDP) Control Packet If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 000.

RDES2 Normal Descriptor (Write-Back Format)

The following table describes the write-back format for the RDES2 normal descriptor.

Table 30-29: RDES2 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31–29	L3L4FM	Layer 3 and Layer 4 Filter Number Matched. These bits indicate the number of the layer 3 and layer 4 filter that matched the received packet: • 000 = Filter 0 • 001 = Filter 1 • 010 = Filter 2 • 011 = Filter 3 • 100 = Filter 4 • 101 = Filter 5 • 110 = Filter 6 • 111 = Filter 7 This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter. Note: This status is not available when flexible RX parser is enabled.
28	L4FM	Layer 4 Filter Match. When this bit is set, it indicates that the received packet matches one of the enabled layer 4 port number fields. This status is given only when one of the following conditions is true: • Layer 3 fields are not enabled and all enabled layer 4 fields match • All enabled layer 3 and layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31–29]. Note: This status is not available when the flexible receive parser is enabled
27	L3FM	Layer 3 Filter Match. When this bit is set, it indicates that the received packet matches one of the enabled layer 4 port number fields. This status is given only when one of the following conditions is true: • All enabled layer 3 fields match and all enabled layer 4 fields are bypassed • All enabled filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31–29]. Note: This status is not available when the flexible receive parser is enabled
26–19	MADRM	MAC Address Match or Hash Value. When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the hash value computed by the MAC. A packet passes the hash filter when the bit corresponding to the hash value is set in the hash filter register. Note: This status is not available when the flexible receive parser is enabled
18	HF	Hash Filter Status. When this bit is set, it indicates that the packet passed the MAC address hash filter. Bits[26:19] indicate the hash value. Note: This status is not available when the flexible receive parser is enabled.

Table 30-29: RDES2 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
17	DAF/RXPI	Destination Address Filter Fail. When flexible RX parser is disabled, and this bit is set, it indicates that the packet failed the DA filter in the MAC. When flexible RX parser is enabled, this bit is set to indicate that the packet parsing is incomplete (RXPI) due to ECC error. Note: When this bit is set, ES bit of RDES3 is also set.
16	SAF/RXPD	SA Address Filter Fail. When the flexible receive parser is disabled, and this bit is set, it indicates that the packet failed the SA filter in the MAC. When the flexible receive parser is enabled, this bit is set to indicate that the packet is dropped (RXPD) by the parser. Note: When this bit is set, ES bit of RDES3 is also set.
15	OTS	Outer VLAN Tag Filter Status. This bit is valid for both single and double VLAN tagged frames.
14	ITS	Inner VLAN Tag Filter Status (ITS). This bit is valid only for double VLAN tagged frames, when double VLAN processing is enabled. For more information, see the Filter Status topic.
13–11	RxParser	Rx Parser • 000 = Rx Parser filter passed • 001 = Packet rejected in Rx Parser • 010 = Rx Parser overflow errors • 011 = Incomplete Rx parsing of the packet • 100 = AF = 1, RF = 1, Bypass • 101 = RA = 1 and L2 filter fails (Rx parser bypass) • 110 = Reserved • 111 = Incomplete parsing due to ECC error
10	ARPNR	ARP Reply Not Generated. When this bit is set, it indicates that the MAC did not generate the ARP reply for received ARP request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time). This bit is reserved when the Enable IPv4 ARP Offload option is not selected.
9–0	HL	L3/L4 Header Length. This field contains the length of the header of the packet split by the MAC at L3 or L4 header boundary as identified by the MAC receiver. This field is valid only when the first descriptor bit is set (FD = 1). The header data is written to the buffer 1 address of corresponding descriptor. If header length is zero, this field is not valid. It implies that the MAC did not identify and split the header. This field is valid when the enable split header feature option is selected.

RDES3 Normal Descriptor (Write-Back Format)

The following table describes the write-back format for the RDES3 normal descriptor.

Table 30-30: RDES3 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31	OWN	Ownership. When this bit is set, it indicates that the EMAC DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full
30	CTXT	Receive Context Descriptor. When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 1'b0 to this bit for normal receive descriptor. When CTXT and FD bits are used together, {CTXT, FD} • 00 = Intermediate Descriptor • 01 = First Descriptor • 10 = Reserved • 11 = Descriptor Error (due to all 1s) **NOTE:** When a descriptor error occurs, the receive DMA closes the receive descriptor indicating descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. Receive DMA sets the CDE field of the `EMAC_DMA[n]_STAT` register but does not the RI field even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.
29	FD	First Descriptor. When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. See the CTXT bit description for details of using the CTXT bit and FD bit together.
28	LD	Last Descriptor. When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
27	RS2V	Receive Status RDES2 Valid. When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	Receive Status RDES1 Valid. When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	Receive Status RDES0 Valid. When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
24	CE	CRC Error. When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
23	GP	Giant Packet. When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). Note: Giant packet indicates only the packet length. It does not cause any packet truncation.
22	RWT	Receive Watchdog Timeout When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.

Table 30-30: RDES3 Normal Descriptor (Write-Back Format) (Continued)

Bit	Name	Description
21	OE	Overflow Error. When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. Note: This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.
20	RE	Receive Error. When this bit is set, it indicates that the gmii_rxer_i signal is asserted while the gmii_rxdv_i signal is asserted during packet reception. This error also includes carrier extension error in the GMII and half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
19	DE	Dribble Bit Error. When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
18–16	LT	Length/Type Field. This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: • 000 = The packet is a length packet • 001 = The packet is a type packet • 011 = The packet is a ARP Request packet type • 100 = The packet is a type packet with VLAN Tag • 101 = The packet is a type packet with Double VLAN Tag • 110 = The packet is a MAC Control packet type • 111 = The packet is a OAM packet type • 010 = Reserved
15	ES	Error Summary. When this bit is set, it indicates the logical OR of the following bits: • RDES3[24]: CRC Error • RDES3[19]: Dribble Error • RDES3[20]: Receive Error • RDES3[22]: Watchdog Timeout • RDES3[21]: Overflow Error • RDES3[23]: Giant Packet • RDES2[17]: Destination Address Filter Fail, when flexible RX parser is enabled • RDES2[16]: SA Address Filter Fail, when flexible RX parser is enabled This field is valid only when the LD bit of RDES3 is set.
14–0	PL	Packet Length. These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when the LD bit of RDES3 is set and Overflow Error bits are reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. This field is valid when the LD bit of RDES3 is set. When the last descriptor and error summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current packet.

Receive Context Descriptor

This descriptor is read-only for the application. Only the DMA can write to this descriptor. The context descriptor provides information about the extended status related to the last received packet. The bit 30 of RDES3 indicates the context type descriptor.

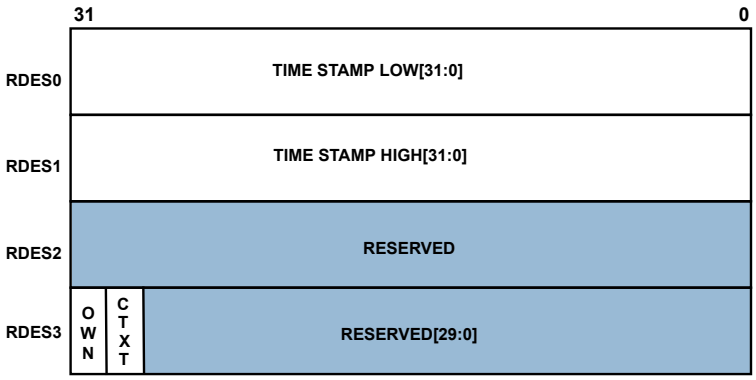


Figure 30-18: Receive Context Descriptor

RDES0 Context Descriptor

The following table describes the format for the RDES0 context descriptor.

Table 30-31: RDES0 Context Descriptor

Bit	Name	Description
31–0	RTSL	Receive Packet Time Stamp Low. The DMA updates this field with least significant 32 bits of the time stamp captured for corresponding receive packet. When this field and the RTSH field of RDES1 show all-one value, the time stamp must be considered as corrupt.

RDES1 Context Descriptor

The following table describes the format for the RDES1 context descriptor.

Table 30-32: RDES1 Context Descriptor

Bit	Name	Description
31–0	RTSH	Receive Packet Time Stamp High. The DMA updates this field with most significant 32 bits of the time stamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the time stamp must be considered as corrupt.

RDES2 Context Descriptor

The following table describes the format for the RDES2 context descriptor.

Table 30-33: RDES2 Context Descriptor

Bit	Name	Description
31–0	Reserved	

RDES3 Context Descriptor

The following table describes the format for the RDES3 context descriptor.

Table 30-34: RDES3 Context Descriptor

Bit	Name	Description
31	OWN	Ownership. When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full
30	CTXT	Receive Context Descriptor. When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 10 to this bit for context descriptor. DMA writes 11 to indicate a descriptor error due to all 1s. When CTXT and DE bits are used together, {CTXT, DE} • 00 = Reserved • 01 = Reserved • 10 = Context Descriptor • 11 = Descriptor Error **NOTE:** When a descriptor error occurs, the receive DMA closes the receive descriptor indicating a descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the `EMAC_DMA[n]_STAT.CDE` bit but does not set the `EMAC_DMA[n]_STAT.RI` bit field even when IOC is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.
29	DE	Descriptor Error. See the CTXT bit description for details of using the DE bit along with the CTXT bit.
28–0	Reserved	

Descriptor Structure for Split Header Support

EMAC supports splitting the incoming receive packet header such that the header and the payload are stored in different buffers (buffer-1 and buffer-2) in the system memory. For more details about splitting header on receive packets, see Header-Payload Split.

The *Descriptors without Split Header Feature* figure shows the descriptor structure without the split header feature.

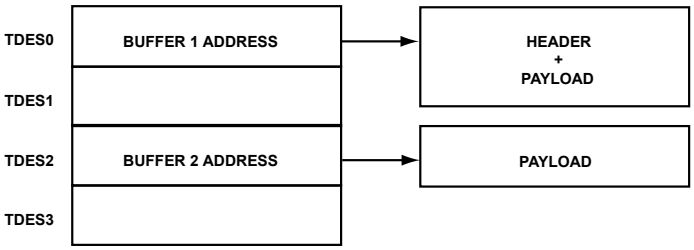


Figure 30-19: Descriptors without Split Header Feature

The *Descriptors with Split Header Feature* figure shows the descriptor structure with the split header feature.

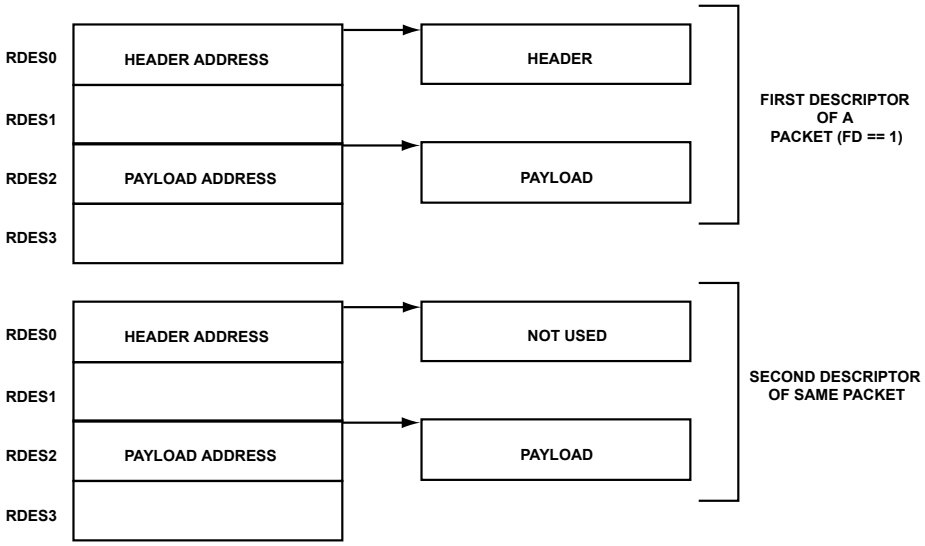


Figure 30-20: Descriptors with Split Header Feature

The DMA writes the header of the received packet by using the header address to which the RDES0 in the first descriptor is pointing (FD bit of RDES3 is set). The DMA writes the payload of the received packet into the buffer address to which the RDES2 is pointing. For subsequent descriptors (FD is set to 0), the address to which RDES0 (Header address) is pointing is not used. The payload is written only to buffers to which the RDES2 (payload address) is pointing.

The `EMAC_MAC_EXT_CFG.HDSMS` bit field indicates the maximum header size allowed for splitting the header data in the received packet based on the value programmed. When the `EMAC_DMA[n]_RXCTL2.ARBS` bit field is programmed to a non-zero value, it indicates the receive buffer size for buffer-1 and the `EMAC_DMA[n]_RXCTL.RBSZ_X_0` and `EMAC_DMA[n]_RXCTL.RBSZ_13_Y` indicates the receive buffer size for buffer-2. The `EMAC_DMA[n]_RXCTL2.ARBS` field can be set to 0, for backward compatibility.

The value in the `EMAC_MAC_EXT_CFG.HDSMS` field should be greater than or equal to the Buffer-1 size indicated by the `EMAC_DMA[n]_RXCTL2.ARBS` field for all the channels.

The following table shows how DMA processes a packet based on the packet type.

Table 30-35: Split Header Support Depending on the Packet Type

Packet Type	Name	Description
TCP or UDP	00 (L3L4 split	The DMA writes the Ethernet header + IP header + TCP or UDP header into the header buffer.
IP (not TCP/UDP)		The DMA writes the Ethernet header + IP header into the header buffer.
Non-IP		The DMA does not split the header and payload
Any packet	01 (L2 split)	The DMA writes the Ethernet header based on split offset (SPLOFST) into the header buffer.
IP	10 (Combination of L2 or L3/L4 split)	L3/L4 split
Non-IP		L2 split
NA	11	Reserved

The IP header includes IPv4 options in case of IPv4 packet and IPv6 extension headers in case of IPv6 frames. The points at which the header is split are shown in the *L3/L4 Header Split Points* figure.

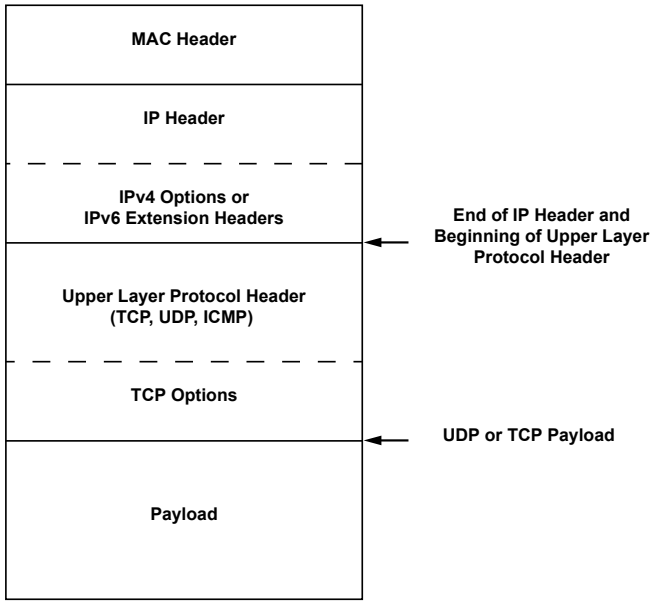


Figure 30-21: L3/L4 Header Split Points

NOTE:
- The VLAN tag stripping must be set for the split function. For instance, the DMA separates the header and payload of an untagged packet only. So, when a tagged packet is received, then program EMAC such that the VLAN tags are deleted/stripped from the received packets.
- L3/L4 split is applicable for IP packets that are either untagged or VLAN stripped. If VLAN tag is retained in the packet forwarded to the DMA, L3-L4 split is not performed. However, if the `EMAC_SPLM_OFST_CFG.SPLM` bit field is set to 2, L2 split is performed for VLAN tagged IP packets.

- For AV packets, to specify an alternative L2 split value, use the EMAC_SPLM_OFST_CFG.SAVE and EMAC_SPLM_OFST_CFG.SAVO bit fields.
- L2 Split is not supported for packets with variable preamble.

The following table shows the header length availability in RDES2.

Table 30-36: Header Length Availability in RDES2

First Descriptor (FD) Value	Last Descriptor (LD) Value	Header Length Availability in RDES2
0	1	Header length not available
1	0	HL[9:0] for all packets
1	1	For IP packets, HL[9:0] For non-IP packets, HL[9:2]

The DMA writes the header length in RDES2 of the first receive descriptor (RDES3[29] FD bit is set) for the packet. The packet length is written in RDES3 of the last receive descriptor (RDES3[28] LD bit set).

The buffer length for the payload is set by the driver through the EMAC_DMA[n]_RXCTL.RBSZ_X_0 and EMAC_DMA[n]_RXCTL.RBSZ_13_Y fields.

The DMA fills receive buffers fully in all except the last descriptor. The header length is taken to be the value based on the bits programmed in the EMAC_MAC_EXT_CFG.HDSMS field.

Enhanced Descriptor for Time-Based Scheduling

The time-based scheduling feature needs Enhanced Descriptors (that are 32 Bytes) to be enabled on all the DMA channels that intend to use the feature (by setting the EMAC_DMA[n]_TXCTL.EDSE bit).

The structure of the 32-byte descriptor for the context and the normal descriptor in read and write formats are described in the following sections.

Enhanced Normal Descriptor - Read

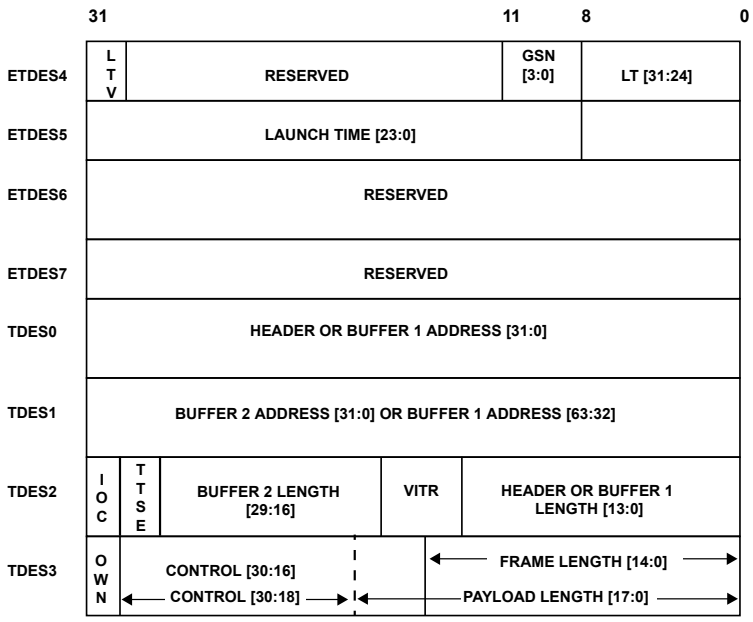


Figure 30-22: Enhanced Normal Descriptor - Read (32-bit Mode)

Table 30-37: Enhanced Normal Descriptor - Read

Bit	Name	Description
31–12	LTV	Launch Time Valid indicates the Launch Time (LT) and GSN fields present in the descriptor are valid. The LTV must be set only if the FD bit of the descriptor is not set.
11–8	GSN	GCL slot number associated with the packet
7–0	LT	Launch Time associated with the packet.

For details of the other fields, see Transmit Normal Descriptor (Read Format).

Enhanced Normal Descriptor - Write

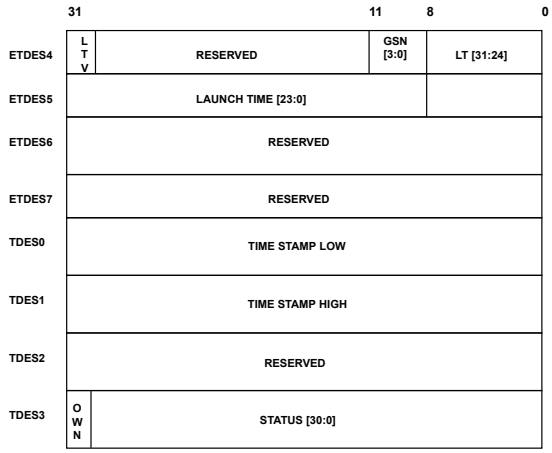


Figure 30-23: Enhanced Normal Descriptor - Write (32-Bit Mode)

For details of LTV, GSN, LT fields, see Enhanced Descriptor for Time-Based Scheduling. For details of the other fields, see Transmit Normal Descriptor (Write-Back Format).

NOTE: In both the write back formats, no modifications can be done to the extended 16 bytes. The rest of the 16 bytes (TDESC0 to TDESC3) are written back as per the previous 16 bytes Descriptor format.

NOTE: Fetch time when enabled overrides the AV slot function

NOTE: When an unaligned new GCL list (with a different CTR) is installed it is recommended not to have traffic during the installation of the new list. Any traffic during the switching of the lists might have unpredictable behavior regarding Fetch, Launch, and Launch expiry as the CTR and BTR values get updated while the frame is being processed.

Enhanced Context Descriptor (Read)

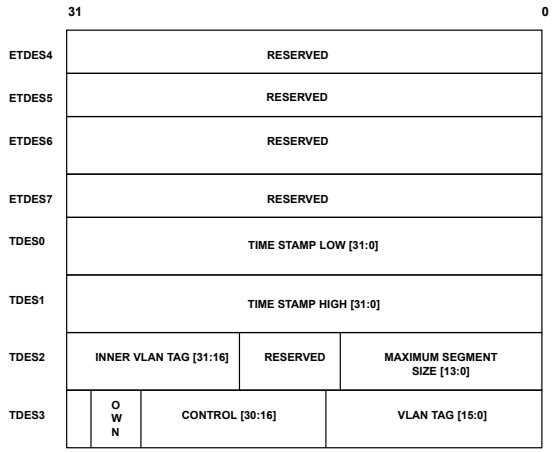


Figure 30-24: Enhanced Context Descriptor - Read

For details of the various fields, see Transmit Context Descriptor.

Enhanced Context Descriptor (Write)

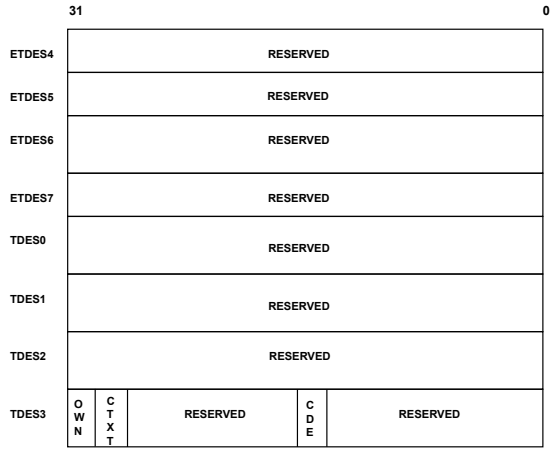


Figure 30-25: Enhanced Context Descriptor (Write)

For details of the various fields, see Receive Context Descriptor.

MAC Transaction Layer (MTL)

The MAC Transaction Layer (MTL) provides the FIFO memory interface to the buffer and regulates the packets between the application system memory and the MAC. It also enables the data to transfer between the application and MAC clock domains. The MTL layer has transmit and receive data paths. The data path for both directions is 32 bits wide and each transmit/receive DMA channel has a dedicated FIFO.

The transmit FIFO size for EMAC0 is 2048 bytes and for EMAC1 is 1024 bytes. The receive FIFO size for EMAC0 is 2048 bytes and for EMAC1 is 512 bytes.

Transmit Path and Operation

The DMA engine controls all transactions for the transmit path with the application. The DMA pushes the Ethernet packets read from the application or system memory into the corresponding queue. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or the complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the DMA.

The queue fill level is indicated to DMA (using PBL and watermark) so that it can initiate a data fetch in required bursts from the application or system memory. The DMA indicates the SOP and EOP as packet delimiters through the Application Transmit Interface (ATI). In configurations with multiple queues, the DMA should also indicate the queue number for which the transaction is being addressed.

The following modes of operation trigger reading the data towards the MAC.

1. Threshold mode. When the number of bytes in the FIFO queue crosses the configured threshold level, (or when the end of packet is written before the threshold is crossed), the data is ready to be popped out and

forwarded to the MAC. The threshold level is configured using the EMAC_TQ0_OPMODE.TTC bit field that corresponds to an MTL queue.

2. Store-and-forward mode. In this mode, the MTL pops the packet towards the MAC only when one or more of the following conditions are true:
   - A complete packet is stored in the queue
   - The transmit FIFO becomes almost full
   - The ATI watermark becomes low

   The watermark becomes low when the requested queue does not have space to accommodate the requested burst length on the ATI. Therefore, the MTL, when operating in the store-and-forward mode, allows the packet transmission even if the packet length is bigger than the transmit queue size. The application can flush the complete content of the transmit queue using the EMAC_TQ0_OPMODE.FTQ bit that corresponds to a MTL queue. This bit is self-clearing and initializes the queue pointers to the default state.

   If the EMAC_TQ0_OPMODE.FTQ bit is set during a packet transfer from the MTL to the MAC, the MTL stops further transfer because the queue is empty. Therefore, an underflow event occurs at the MAC transmitter. For information about the initialization and transmit operations for the MTL layer, see the following sections.

Initialization Flow

At reset, the MTL is ready to manage the flow of data to and from the application or DMA, and MAC. For configurations with a single transmit queue, there are no initialization requirements for enabling the MTL. For configurations with multiple transmit queues, the program must initialize the queue size for each of the queues using the EMAC_TQ0_OPMODE.TQS bits that correspond to a transmit queue. The program must also initialize the MAC block. The DMA controllers must be individually enabled through their respective registers.

Transmit DMA Channels and Queues

The number of transmit DMA channels is equal to the transmit queues. This is because the DMA is designed to arbitrate among multiple channels (to fetch the descriptor and packet data from the system memory) in terms of PBLs (Programmable Burst Lengths). However, the transmit queues are packet-based storage memory. Therefore, one to one mapping is required between the DMA channels and transmit queues to maintain the packet-level coherency.

Transmit Operation with Multiple Packets in the Buffer

The transmit FIFO can be configured to accept up to two packets at a time.

Retransmission During Collision

While a packet is being transferred from the MTL to the MAC, a collision event can occur on the MAC line interface in half-duplex mode. The MAC indicates a retry attempt to the MTL by giving the status even before the EOP is transferred from the MTL. The MTL then enables the retransmission by popping out the packet again from the queue.

After more than 96 bytes (or 548 bytes in 1000-Mbps mode) are read out towards the MAC, the queue controller frees up that space and makes it available to the application or the DMA to push in more data. This means that the retransmission is not possible after this threshold is crossed or when the MAC indicates a late-collision event.

Transmit Queue Flush Operation

The MTL allows a transmit Queue to be flushed at any moment using the EMAC_TQ0_OPMODE.FTQ bit. The application can read back the same bit (cleared after flush is complete) to check the status of the flush operation.

Receive Path and Operation

The MTL receive module receives packets from the MAC and pushes them into the receive queue. The status (fill level) of the queue is indicated to the DMA when it crosses the configured receive threshold (EMAC_RQ0_OPMODE.RTC bits of the corresponding MTL queue), or the complete packet is received. The MTL also indicates the fill level of the queue so that the DMA can initiate preconfigured burst transfers towards the SCB interface.

The receive operation occurs as follows:

1. When the MAC receives a packet, it indicates the availability of receive data.
2. The MAC indicates the SOP and EOP delimiters.
3. The MTL accepts the data and pushes it into corresponding receive queue.
4. After the EOP is transferred, the MAC drives the status word which is also pushed into the corresponding receive queue by the MTL.

   NOTE: In Threshold (cut-through) mode, the status words are stored after the packet EOP. In store-and-forward mode, the location for the maximum status words are reserved before writing the SOP and the status is written to reserved locations after writing the EOP.

5. If IEEE 1588 time stamp feature is enabled and the 64-bit time stamp is available along with the packet status, it is pushed into the receive queue as a part of the status word. Therefore, two additional locations are taken per packet to store the time stamp in the receive queue.
6. The MTL takes the data out of the FIFO and sends it to the DMA.
   - Threshold mode (default). In this mode, the MTL reads the data and indicates its availability to the application or DMA when one of the following occurs:
     - Data bytes equal to the threshold amount are written to the receive queue (EMAC_RQ0_OPMODE.RTC bit field value)
     - A full packet of data is received into the queue
   - Receive FIFO Store-and-forward mode. This mode is configured using EMAC_RQ0_OPMODE.RSF bit. The initial receive queue locations are reserved for the status words before writing the SOP. A packet is read out only after it is completely written into the receive queue. In this mode, all error packets are dropped (if configured through the EMAC_RQ0_OPMODE.FEP of a queue) such that only valid packets are read and forwarded to the application.

Multi-Packet Receive Operation

In Threshold mode, the packet status is available immediately after the packet data. In store-and-forward mode, the packet data is available after the packet status. The MTL can store any number of packets into the queue when it is not full. When the MAC receives a packet and the corresponding receive queue is full, the MTL ignores that packet and an overflow event occurs. Additionally, the MTL increments the overflow counter (`EMAC_RQ0_MSPKTOF_CNT.OVFPKTCNT`) of the corresponding queue.

Receive Error Handling

If the MTL receive queue is full before it receives the EOP data from the MAC, the following happens:

- An overflow is declared
- The whole packet (including the status word) is dropped
- The overflow counter in the DMA (`EMAC_RQ0_MSPKTOF_CNT.OVFPKTCNT` bits of corresponding MTL queue) is incremented.

This is true even if the `EMAC_RQ0_OPMODE.FEP` bit of the corresponding MTL queue is set.

If the start address of such a packet has already been transferred to the read controller, the rest of the packet is dropped and a dummy EOP is written to the queue along with the status word with overflow status. The status indicates a partial packet because of overflow. In such packets, the packet length field is invalid. If the MTL receive queue is configured to operate in the store-and-forward mode and the length of the received packet is more than the queue size, overflow occurs, and all such packets are dropped.

The MTL receive control logic can filter error and undersized packets using the `EMAC_RQ0_OPMODE.FEP` and `EMAC_RQ0_OPMODE.FUP` bits of the corresponding MTL queue. If the start address of such a packet has already been transferred to the receive queue read controller, that packet is not filtered. The start address of the packet is transferred to the Read Controller after the packet crosses the receive threshold set by the `EMAC_RQ0_OPMODE.RTC` of the corresponding MTL queue.

EMAC CORE

The EMAC CORE is the lowest block in the EMAC peripheral and it performs all operations with the external world (PHY chip). It has independent transmit and receive modules. The modules interact with the EMAC Transaction Layer (MTL) at one end and interacts with the PHY chip through the RGMII/RMII/MII interface at the other end. Both modules have several sub blocks which are discussed in subsequent sections.

Transmission is initiated when the MTL pushes in data with the start of packet (SOP). The CORE then transmits to the RGMII/RMII/MII. After the end of packet transfers out, the CORE gives the status of the transmission back to the MTL. The MTL forwards the transmission to the application through DMA.

A receive operation initiates when the EMAC detects an SFD on the RGMII/RMII/MII. The CORE strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field used to verify the CRC for the frame. The frame drops in the core when it fails the address filter.

NOTE: The term *CORE* (written in capitals) refers to the internal block of Ethernet peripheral. Do not confuse the term with the *processor core*.

EMAC CORE Transmission Engine

The following modules constitute the transmission function (transmission engine components) of the EMAC:

- Transmit Bus Interface Module (TBU)
- Transmit Packet Controller Module (TPC)
- Transmit Protocol Engine Module (TPE)
- Transmit Scheduler Module (STX)
- Transmit CRC Generator Module (CTX)
- Transmit Flow Control Module (FTX)

Transmit Bus Interface Module (TBU)

This module interfaces the transmit path of the EMAC CORE with the MAC Layer FIFO interface. This module outputs the transmit status to the application at the end of normal transmission or collision.

Transmit Packet Controller Module (TPC)

The Transmit Packet Controller (TPC) module consists of eight registers to hold the data and the last data control received from the TBU.

The register provides a buffer between the application and the Transmit Protocol Engine (TPE) to regulate data flow.

When the number of bytes received from the application are less than 60 (DA+SA+LT+DATA), the state machine that interfaces with TBU automatically appends zeros to the packet being transmitted. This is done to make the data length exactly 46 bytes to meet the minimum data field requirement of IEEE 802.3. The EMAC module can also be programmed to not append any padding.

The cyclic redundancy check (CRC) for the Frame Check Sequence (FCS) field is calculated before transmission to the TPE module. This value is computed by the CTX module. The TPC module receives the computed CRC and appends it to the data being transmitted to the TPE module. When the MAC is programmed to not append the CRC value to the end of Ethernet packets, the TPC module ignores the computed CRC and transmits only the data received from the TBU module to the TPE module. An exception to this rule is that when the MAC is programmed to append pads for packets (DA+SA+LT+DATA) less than 60 bytes sent by the TBU module, the TPC module always appends the CRC at the end of padded packet.

Transmit Protocol Engine Module (TPE)

The Transmit Protocol Engine (TPE) module consists of a Transmit State Machine that controls the operation of Ethernet packet transmission. The Transmit State Machine of this module contains the following features to meet the IEEE 802.3/802.3z specification:

- Generates preamble and SFD
- Generates jam pattern in half-duplex mode after normal collision

- Generates carrier extension in the half-duplex (only in the RGMII) mode when packet is smaller than 512 bytes
- Supports packet bursting in the half-duplex (only in the RGMII) mode
- Jabber timeout
- Flow control for half-duplex mode (back pressure)
- Generates transmit packet status
- Contains time stamp snapshot logic for IEEE 1588 support

When the TPC module requests the TPE module for a new packet transmission, the Transmit State Machine sends out the preamble and SFD, followed by the data received. The preamble is defined as 7 bytes of 10101010 pattern and the SFD is defined as 1 byte of 10101011 pattern.

The collision window is defined as 1 slot time (512-bit times for 10/100 Mbps Ethernet and 4096-bit times for 1,000 Mbps Ethernet). The jam pattern generation is applicable only to half-duplex mode, not to full duplex mode.

In MII/RMII mode, if a collision occurs any time from the beginning of the packet to the end of the CRC field, the Transmit State Machine sends a 32-bit jam pattern of 0x55555555 on MII/RMII to inform all other stations that a collision has occurred. If the collision is seen during the preamble transmission phase, the Transmit state machine completes the transmission of preamble and SFD, and then sends the jam pattern.

In RGMII mode, if a collision occurs any time between the beginning of the packet and the end of the extension field, the Transmit State Machine sends a 32-bit jam pattern of 0x55555555 on RGMII to inform all other stations of the collision. If the collision is seen during the preamble transmission phase, the Transmit State Machine completes the transmission of preamble and SFD, and then sends the jam pattern. If a collision occurs during the extension field, the Transmit State Machine sends a 32-bit jam pattern of 0x1F1F1F1F.

If the collision occurs after the collision window and before the end of the FCS field (or the end of burst if the packet burst mode is enabled), the Transmit State Machine sends a 32-bit jam pattern and sets the late collision bit in the transmit packet status.

In RGMII half-duplex mode (1000 Mbps), the Transmit State Machine ensures that all valid carrier events exceed a slot time of 4,096-bit times. To accomplish this, any transmit packet shorter than 512 bytes from the TFC module is extended using a carrier extension.

When the packet burst mode is enabled, only the first packet of the burst is carrier extended if it is shorter than 512 bytes. The carrier extension is not applicable for MII/RMII half-duplex and RGMII or MII/RMII full-duplex modes. When the packet burst mode is enabled, the Transmit State Machine transmits a burst of packets (when packets are available from the TFC module) without releasing the carrier of the PHY. To accomplish this, the state machine inserts the carrier extension for a minimum IPG period (96-bit times) between the packets. The Transmit State Machine continues to burst packets when additional packets are available from the TPC module and a burst limit of 8192-byte times is not exceeded. When an additional packet is not available at the end of the IPG period in the middle of the burst, the Transmit State Machine releases the carrier RGMII.

Packet bursting is applicable only for the RGMII half-duplex mode. It is not applicable in the MII/RMII and RGMII full-duplex modes. In the RGMII half-duplex mode, the size of the first packet in a packet burst should at least be equal to the slot time. If the first packet (including carrier extension) is less than the slot time, the MAC considers all data bytes, received for the first packet and subsequent packets, which end immediately after the slot time is reached as first packet in the burst. This may result in CRC error. However, if the packet burst ends (both RXDV and RXER go low) before the slot time, the subsequent packet is considered as a new packet. This behavior is according the IEEE 802.3.

The TPE module maintains a jabber timer to stop the transmission of Ethernet packets if the TFC module transfers more than 2,048 (default) bytes. The timeout is changed to 10,240 bytes when the jumbo packet is enabled.

The Transmit State Machine uses the deferral mechanism for flow control (backpressure) in the half-duplex mode. When the application requests to stop receiving packets, the Transmit State Machine sends a JAM pattern of (8'h55) 32 bytes whenever it senses a reception of a packet, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. The application requests the flow control through by setting the `EMAC_Q[i]_TXFLOW_CTL.FCB_BPA` bit of the corresponding MTL queue. If the application requests a packet to transmit, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort their transmissions because of excessive collisions.

If IEEE 1588 time stamp is enabled for the transmit packet, this block takes a snapshot of the system time when the SFD is put onto the transmit RGMII or MII/RMII bus. The system time source is either an external input or it is internally generated according to the configuration selected.

Transmit Scheduler Module (STX)

The transmit scheduler is responsible for scheduling the frame transmission on the RGMII or MII/RMII. The two major functions of this module are:

- Maintains the inter-packet gap between two transmitted packets. The STX module maintains an idle period of the configured inter-packet gap (`EMAC_MAC_CFG.IPG`) between any two transmitted packets. If packets from the TPC arrive at the TPE module sooner than the configured IPG time, the TPE module waits for the enable signal from the STX module before starting the transmission on GMII or MII. The STX module starts its IPG counter as soon as the carrier signal of RGMII or MII/RMII goes inactive. At the end of the programmed IPG value, the module issues an enable signal to the TPE module in the full-duplex mode.

  In the half-duplex mode and when IPG is configured for 96-bit times, the STX module follows the rule of deference specified in the IEEE 802.3, Section 4.2.3.2.1. The module resets its IPG counter if a carrier is detected during the first two-thirds (64-bit times for all IPG values) of the IPG interval. If the carrier is detected during the final one-third of the IPG interval, the STX module continues the IPG count and enables the transmitter after the IPG interval.

- Implements the Truncated Binary Exponential Back-off algorithm in half-duplex mode.

Transmit CRC Generator Module (CTX)

The MAC Transmit CRC Generator (CTX) module interfaces with the TFC module to generate the CRC for the FCS field of the Ethernet packet.

The TPC module sends the packet data and any necessary padding to the CTX module through an 8-bit interface.

The CTX module calculates the 32-bit CRC for the FCS field of the Ethernet packet. The encoding is defined by the following generating polynomial:

G (x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

The CTX module gets the byte data of Ethernet packet from the TPC module (DA + SA + LT + DATA + PAD) qualified with a data valid signal. The TPC also indicates to the CTX when to reset the previously calculated CRC and to start the new CRC calculation for the coming packet. The TPC module issues the start command before sending the new packet data for calculation. The calculated CRC is valid on the next clock after the data is received.

In RGMII mode, the data valid signal is valid for every clock, from the first data byte through the last data byte. In MII/RMII mode, this signal is valid every alternate clock.

Transmit Flow Control Module (FTX)

The Transmit Flow Control (FTX) module generates and transmits the pause packets to the TFC module based on the flow control triggers in full-duplex mode. The TFC module receives the pause packet from the FTX module, appends the calculated CRC, and sends the packet to the TPE module.

EMAC Core Reception

The following are the functional blocks (reception engine components) in the receive path of the EMAC core.

- Receive Protocol Engine Module (RPE)
- Receive CRC Module (CRX)
- Receive Packet Controller Module (RPC)
- Receive Flow Control Module (FRX)
- Receive Bus Interface Unit Module (RBU)
- Address Filtering Module (AFM)

Receive Protocol Engine Module (RPE)

The Receive Protocol Engine (RPE) consists of the Receive State Machine which strips the preamble, SFD, and carrier extension of the received Ethernet packet (in half-duplex 1000-Mbps mode). The sequence is as follows:

1. When the RX_DV signal of GMII or MII becomes active, the Receive State Machine of RPE starts looking for the SFD field (byte 0x5D in GMII mode; 0xD nibble in MII mode). The state machine drops received packets until it detects SFD.
2. When SFD is detected, the state machine begins sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).

3. If the IEEE 1588 time stamp feature is enabled, the RPE takes a snapshot of the system time at which SFD of any packet is detected on GMII or MII. If this packet is not dropped during MAC filtering, the time stamp is passed to the application. In MII mode, the RPE converts the received nibble data into bytes and forwards the valid packet data to the RFC module.
4. The Receive State Machine of the RPE module decodes the length/type field of the receiving Ethernet packet.

   If the length/type field is less than 1,536 and if the MAC is programmed for the Auto CRC/pad stripping (EMAC_MAC_CFG.ACS bit), the state machine sends the data of the packet up to the count specified in the length/type field and starts dropping bytes (including the FCS field). The state machine of the RPE module decodes the length/type field and checks for the length interpretation.

   NOTE: In Audio Video (AV) or Data Center Bridging (DCB) mode when additional receive queues are selected, the packets that are less than or equal to 16 bytes in length after pad stripping, always get dropped inside the MAC receiver. This happens even if the packets have passed the address filter and have no CRC error.

   If the length/type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if the CRC stripping for type packet has not been enabled with the EMAC_MAC_CFG.CST bit. However, if CRC stripping for type packets is enabled and the Receive Checksum Offload Engine is not enabled, the MAC strips and drops the last 4 bytes of all packets of Ether type before forwarding the packets to the application.
5. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets larger than 2,048 (10,240 if Jumbo Packet is enabled) bytes, (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. In addition, a programmable watchdog timer can be used (bit 16 of the EMAC_WDT_TMOUT register) to override the fixed timeout of 2,048 or 10,240 bytes. Programs can disable the watchdog timer using the EMAC_MAC_CFG.WD bit. However, even if the watchdog timer is disabled, a packet greater than 32 KB size is cut off and a watchdog timeout status is given.

At the end of every received packet, the RPE module generates received packet status and sends it to the RPC module. Control, missed packet, and filter fail status are added to the receive status in the RPC module.

Receive CRC Module (CRX)

The receive CRC module checks for any CRC errors in the receiving frame.

This module calculates the 32-bit CRC for the received frame that includes the destination address field through the FCS field (DA+SA+LT+DATA+PAD+FCS). The following generating polynomial defines the encoding.

G (x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

Irrespective of the auto pad or CRC strip, the CRC module receives the entire frame to compute the CRC check for received frame.

Receive Packet Controller Module (RPC)

The MAC Receive Packet Controller (RPC) receives the Ethernet packet data and status from the RPE module.

The RPC module consists of a FIFO of parameterized depth (default set to 4 deep and 33-bits wide) and two state machines for writing and reading the FIFO. The FIFO holds the received Ethernet packet data and byte enables, along with a control bit to indicate the last data. The state machines manage the FIFO and provide a packet buffering for the Ethernet packet being received from the RPE module. The following are main functions of the RPC module:

- Converting Data path — converts 8-bit data to 32-bit data to the RBU module
- Packet filtering
- Attaching the calculated IP checksum input form the IPC
- Updating the receive status and forwarding it to RBU

If the `EMAC_MACPKT_FILT.RA` bit is set, the RPC module initiates the data transfer to the RBU module when 4 bytes of Ethernet data are received from the RPE module. At the end of the data transfer, the RPC module sends out the received packet status that includes the packet filter bits (SA Filterfail and DA Filterfail) and status from the RPC module. These bits are generated based on the filter-fail signals from the AFM module. This status bit indicates to the application whether the received packet has passed the filter controls (both address filter and Packet Filter controls from CSR). The RPC module does not drop any packet on its own in this mode.

If the `EMAC_MACPKT_FILT.RA` bit is reset, the RPC module performs packet filtering based on the destination or source address. If the application does not want to receive any bad packets such as runt, CRC error packets, the application still needs to perform another level of filtering. The RPC module waits to receive the first 14 bytes of received data (type field) from the RPE module. Until then, the module does not initiate any transfers to the RBU module. After receiving the destination or source address bytes, the RPC checks the filter-fail signal from the AFM module for an address match. On detecting filter-fail from AFB, the packet is dropped at the RPC module and not transferred to the application.

On a delayed filter response from the AFM (this can only occur if the AFM logic is changed), the RPC module waits until the FIFO is full, and then proceeds with the packet transfer to the RBU module. However, the RPC module still takes the delayed response from the AFM module and if it is a (DA or SA) filter failure, it drops the rest of the packet and sends the receive status word (with zero packet-length, CRC error and runt error bits set) immediately indicating the filter fail. If there is no response from the AFM until EOP is transmitted, the filter fail status in the receive status word is updated accordingly.

Receive Flow Control Module (FRX)

The Receive Flow Controller (FRX) detects the pause packet being received and pauses the packet transmission for the delay specified within the received pause packet. The FRX module is enabled only in the full-duplex mode.

Receive Bus Interface Unit Module (RBU)

The Receive Bus Interface Unit (RBU) converts the 32-bit data received from the RPC module into a 32-bit protocol on the Application side.

Address Filtering Module (AFM)

The Address Filtering Module (AFM) performs the destination and source address checking function on all received packets and reports the address filtering status to the RPC module.

The address checking is done based on different parameters (EMAC_MACPKT_FILT register) chosen by the application. These parameters are inputs to the AFM module as control signals, and the AFM module reports the status of the address filtering based on the combination of these inputs. The AFM module does not filter the receive packets but reports the status of the address filtering (whether to drop the packet or not) to the RFC module. The AFM module also reports address filter status and whether the received packet is a multicast packet or a broadcast packet.

The AFM module probes the 8-bit receive data path between the RPE module and the RFC module and checks the destination and source address field of each incoming packet. In GMII mode, the module takes 8/14 clocks (from the start of packet) to compare the destination/source address of the packet being received. Similarly, in MII mode, the module takes 14/26 clocks (from the start of packet) to compare the destination or source address of the receiving packet. The AFM module gets the physical (MAC) address of the station and the Multicast Hash table from CSR module for address checking. The CSR module provides the packet filter register parameters to AFM.

Source Address, VLAN, CRC Insertion/Replacement

The following sections provide information on Source Address and VLAN Insertion, Replace, or Delete, Queue/Channel Based VLAN Tag Insertion on Transmission, and CRC Insertion and Replacement.

Double VLAN Processing

The EMAC module supports two VLAN tags, namely inner and outer, for processing double VLANs.

This feature is referred called double VLAN tagging where the MAC can process two VLAN tags. With this feature, the EMAC supports:

- Insertion, replacement, or deletion of up to two VLAN tags in the transmit path.
- Packet filtering and stripping based on any one of the two VLAN tags in the receive path. Stripping and providing up to two VLAN tags in the receive path as a part of the receive status.

Transmit Path

The following table describes the double VLAN features supported by the MAC on the transmit side.

Table 30-38: Double VLAN Processing Features in Transmit Path

Feature	Description
Support for C-VLAN and S-VLAN Tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the EMAC_VLAN_INCL.CSVL bit and the EMAC_INVLAN_INCL.CSVL bit. The EMAC supports processing of any sequence of outer and inner VLAN tags. **NOTE:** The EMAC does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN tag types or whether the insertion or replacement operation results in an invalid sequence of VLAN tag type. Therefore, the application must provide a correct sequence of VLAN tag types and program the MAC so that the VLAN tag types in the transmitted packet contains the correct sequence. The application must ensure the following: • The inner tag should not be S-VLAN when the outer C-VLAN tag insertion is enabled. • The outer tag should not be C-VLAN when the inner S-VLAN tag insertion is enabled. • The inner tag cannot be S-VLAN when the outer tag is replaced with C-VLAN. • The outer tag cannot be C-VLAN when the inner tag is replaced with S-VLAN.
VLAN Tag deletion	Programs can enable the VLAN tag deletion for outer or inner tag using the EMAC_VLAN_INCL.VLC and EMAC_INVLAN_INCL.VLC bit fields, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN Tag Insertion or Replacement	Programs can enable the VLAN tag insertion or replacement for outer or inner tag using the EMAC_VLAN_INCL.VLC EMAC and EMAC_INVLAN_INCL.VLC bit fields, respectively. When VLAN tag insertion or replacement is enabled, the EMAC_VLAN_INCL.VLTI bit and the EMAC_INVLAN_INCL.VLTI bit are used to determine whether the VLAN tag is taken from the register or the control word.

Receive Path

The following table describes the features supported by the MAC on the receive side and the corresponding bits in the EMAC_VLANTAG_CTL register.

Table 30-39: Double VLAN Processing in Receive Path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the EMAC_VLANTAG_CTL.ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the EMAC_VLANTAG_CTL.ERSVLM bit.
Outer and Inner VLAN Tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EMAC_VLANTAG_CTL.EVLS and EMAC_VLANTAG_CTL.EIVLS bits.
16-bit outer and inner VLAN Tag and Type in Rx status	The MAC can provide the 16-bit outer and inner VLAN tag and type in the Rx status based on the EMAC_VLANTAG_CTL.EVLRXS and EMAC_VLANTAG_CTL.EIVLRXS bits, respectively.

Table 30-39: Double VLAN Processing in Receive Path (Continued)

Feature	Description
Disabling or skipping checking of outer VLAN Tag type	The MAC can disable or skip checking of outer VLAN tag type to match C-VLAN or S-VLAN based on the EMAC_VLANTAG_CTL.DOVLTC bit.

Source Address and VLAN Insert, Replace, or Delete

The EMAC supports inserting or replacing the source address based on the information in the MAC address registers. The MAC can also insert, replace, or delete the VLAN fields (VLAN type and VLAN tag) based on the setting of the EMAC_VLAN_INCL.VLTI bit. Programs can enable the SA insert or replace feature for all transmit packets or selective packets. Similarly, programs can enable the VLAN insert, replace, or delete feature for all transmit packets or selective packets.

Programming Source Address Insert or Replace

The software can use the SA insert or replace feature to instruct the MAC to do the following for transmit packets:

- Insert the content of the MAC address registers in the SA field
- Replace the content of the SA field with the content of the MAC address registers

When SA insert is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the transmit packet and it inserts the content of MAC address registers in the SA field. Similarly, when SA replace is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the destination address field in the transmit packet with the content of the MAC address registers.

The program can enable the SA insert or replace feature for all transmit packets or selective packets:

- Enable SA insert or replace for all packets: program the EMAC_MAC_CFG.SARC bit field
- Enable SA insert or replace for selective packets: program the SA insertion control field (bits [25:23] of TDES3) in the first transmit descriptor of the packet. When bit 25 of TDES3 is set, the SA insertion control field indicates insertion or replacement by the MAC address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by the MAC address0 registers.

If the MAC address1 registers are not enabled, the MAC address0 registers are used to insert or replace irrespective of the value of the most-significant bit of the SA insertion control field.

Programming VLAN Insert, Replace, or Delete

The software can use the VLAN insert, replace, or delete feature to instruct the MAC to do the following for transmit packets:

- Delete the VLAN type and VLAN tag fields
- Insert or replace the VLAN type and VLAN tag fields using the EMAC_VLAN_INCL.VLTI bit as described in the following table.

Table 30-40: Double VLAN Processing in Receive Path

Condition	Description
EMAC_VLAN_INCL.VLTI bit is set	The MAC inserts or replaces the following: • VLAN type field (C-VLAN or S-VLAN as indicated by the EMAC_VLAN_INCL.CSVL bit • VLAN Tag field content of the VT field of transmit context descriptor of the packet
EMAC_VLAN_INCL.VLTI bit is reset	The MAC inserts or replaces the following: • VLAN Type field (C-VLAN or S-VLAN as indicated by the EMAC_VLAN_INCL.CSVL bit • VLAN tag field with the EMAC_VLAN_INCL.VLT bit field

When VLAN replace or delete is enabled, the MAC checks if the VLAN type field (0x8100 or 0x88a8) is present after the DA and SA fields in the transmit packet. The replace or delete operation does not occur if the VLAN type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check for the presence of the VLAN type field in the transmit packet and just inserts the VLAN type and VLAN tag fields.

Programs can enable the VLAN insert, replace, or delete feature for all transmit packets or selective packets:

- Enable VLAN insert, replace, or delete for all packets: program the EMAC_VLAN_INCL.VLC and EMAC_VLAN_INCL.VLP bit fields
- Enable VLAN insert, replace, or delete for selective packets: program the VTIR field of the TDES2 normal descriptor

Queue/Channel Based VLAN Tag Insertion on Transmit

The EMAC supports channel/queue based VLAN tag insertion on all transmitted packets. Queue/channel specific VLAN tag registers are accessed using indirect addressing through the EMAC_VLAN_INCL register. VLAN type and tag values can be independently programmed for each queue/channel. See Programming Queue/Channel Based VLAN Inclusion Registers for more details.

CRC Insert and Replace

The software can use the CRC replace feature to instruct the MAC to replace the FCS field in the transmit frame with the CRC computed by the MAC. This feature works on per-frame basis. To enable the CRC replacement feature, program the CRC replacement control field (bit 26 of TDES3) in the first transmit descriptor of the frame.

Packet Filtering

the *Packet Filtering Sequence* figure shows the filtering sequence for receive packets. The sequence shown in the figure is valid when all the filters (L2, VLAN, L3, L4) are active. If any of the layer filters are not enabled, that filter is bypassed, and the subsequent filter is applied. A packet that fails any of the filters is discarded. However, the discarded packet can be forwarded to the host based on the register control. For example, when the EMAC_MACPKT_FILT.RA bit is set to 1, all discarded packets are forwarded to the host but with its packet status indicating the specific filter-failure. If RA=0, the EMAC_MACPKT_FILT.VTFE and

EMAC_MACPKT_FILT.IPFE bits of control whether the packets that fail the VLAN filter and Layer 3-4 filter should be discarded or forwarded to the host.

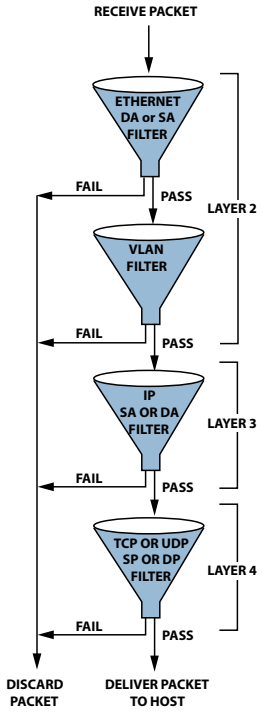


Figure 30-26: Packet Filtering Sequence

Source Address or Destination Address Filtering

The Address Filtering Module of the MAC checks the source address (SA) and destination address (DA) fields of each incoming packet.

Unicast Destination Address Filtering

The MAC supports 32 MAC addresses for unicast perfect filtering. If perfect filtering is selected (the EMAC_MACPKT_FILT.HUC bit is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default address0 register (EMAC_ADDR0_HI, EMAC_ADDR0_LO) is always enabled.

The address1 through 31 register addresses (both HIGH and LOW) are selected with an individual enable bit. For these addresses, programs can mask each byte during comparison with corresponding received DA byte by setting the corresponding mask byte control bit in the register. This enables group address filtering for the DA.

In hash filtering mode (when the EMAC_MACPKT_FILT.HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a hash table (256 bits for EMAC0 and 64 bits for EMAC1). For hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the hash table. A value of 00000 selects bit 0 of the selected register, and a value of 11111 selects bit 63 of the hash table register (registers 7-0). When the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the hash filter; otherwise, the packet is considered to have failed the hash filter.

Multicast Destination Address Filtering

To program the MAC to pass all multicast packets, set the EMAC_MACPKT_FILT.PM bit. If the EMAC_MACPKT_FILT.PM bit is reset, the MAC performs the filtering for multicast addresses based on the EMAC_MACPKT_FILT.HMC bit.

The multicast address is compared with the programmed MAC destination address registers (1–31). Group address filtering is also supported. In hash filtering mode, the MAC performs imperfect filtering using a 64-bit hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the hash table. A value of 000000 selects bit 0 of the selected register and a value of 111111 selects bit 63 of the hash table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the hash filter. Otherwise, the packet is considered to have failed the hash filter.

Hash or Perfect Address Filtering

To configure the DA filter to pass a packet when its DA matches either the hash filter or the perfect filter, set the EMAC_MACPKT_FILT.HPF bit and the corresponding EMAC_MACPKT_FILT.HUC or EMAC_MACPKT_FILT.HMC bits. This is applicable to both unicast and multicast packets. If the EMAC_MACPKT_FILT.HPF bit is reset, only one of the filters (hash or perfect) is applied to the received packet.

Broadcast Address Filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the EMAC_MACPKT_FILT.DBF bit.

Unicast Source Address Filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. Programs can configure the MAC address registers [1–31] to use SA instead of DA for comparison by setting bit 30 of corresponding register.

The MAC also supports group filtering with SA. Programs can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the EMAC_MACPKT_FILT.SAF bit is set. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word. When the EMAC_MACPKT_FILT.SAF bit is set, the SA filter and DA filter result is AND'ed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in order.

Inverse Filtering

For DA and SA filtering, programs can invert the filter-match result at the final output by setting the EMAC_MACPKT_FILT.DAIF and EMAC_MACPKT_FILT.SAIF bits. The EMAC_MACPKT_FILT.DAIF bit is applicable for both unicast and multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

The following tables summarize the DA and SA filtering based on the type of packets received.

NOTE: When the EMAC_MACPKT_FILT.RA bit is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx status.

Table 30-41: Destination Address Filtering

Packet Type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA Filter Operation
Broad-cast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match
Multi-cast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if `EMAC_MACPKT_FILT.PCF` = 0x

Table 30-42: Source Address Filtering

Packet Type	PR	SAIF	SAF	DA Filter Operation
Unicast	1	X	X	Pass all packets
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

VLAN Filtering

The following sections provide information on EMAC VLAN filtering.

VLAN Tag Perfect Filtering

In VLAN tag perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. Programs can enable the inverse matching for VLAN packets by setting the `EMAC_VLANTAG_CTL.VTIM` bit. In addition, programs can enable processing of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the `EMAC_VLANTAG_CTL.ESVL` bit. The VLAN packet status bit (bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

NOTE: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results. By default, the VLAN tag-based perfect filter is available in all configurations.

VLAN Tag Hash Filtering

The 16-bit VLAN hash table is used for group address filtering based on the VLAN tag. The MAC performs the VLAN hash matching using the `EMAC_VLANTAG_CTL.VTHM` (VLAN tag hash table match enable) bit. The MAC provides VLAN tag hash filtering with a 16-bit hash table.

If the `EMAC_VLANTAG_CTL.VTHM` bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the `EMAC_VLAN_HASHTBL` register. A value of 1 in the register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet is forwarded. A value of 0 indicates that VLAN-tagged packet is dropped.

NOTE:
- The 16 or 12 bits of the VLAN tag are considered for CRC-32 computation based on the `EMAC_VLANTAG_CTL.ETV` bit.
- When the bit is reset, most significant four bits of CRC-32 of VLAN tag are inverted and used to index the content of `EMAC_VLANTAG_CTL` register.
- When the bit is set, most significant four bits of CRC-32 of VLAN tag are directly used to index the content of the `EMAC_VLANTAG_CTL` register.

If the VLAN tag of a packet matches the perfect or hash filter, the packet is dropped. If the VLAN perfect and VLAN hash match are enabled, a packet is considered as matched if either the VLAN hash or the VLAN perfect filter matches. When inverse matching is set, a packet is forwarded only when both perfect and hash filters indicate mismatch.

The *VLAN Match Status* table shows the different possibilities for VLAN matching and the final VLAN match status. When the `EMAC_MACPKT_FILT.RA` bit is set, all packets are received, and the VLAN match status is indicated in the VF bit of RDES2 normal descriptor (write-back format). When the RA bit is not set and the

EMAC_MACPKT_FILT.VTFE bit is set, the packet is dropped if the final VLAN match status is fail. In the following table, X signifies that entry can have any value.

When VLAN VID is programmed to 0 in the EMAC_VLANTAG_DAT.VID bit field, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN hash match depends on the EMAC_VLANTAG_CTL.VTHM and EMAC_VLANTAG_CTL.VTIM bits.

Table 30-43: VLAN Match Status

VID	VLAN Perfect Filter Match Result	VTHM Bit	VLAN Hash Filter Match Result	VTIM Bit	Final VLAN Match Status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID! = 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

Extended Receive VLAN Filtering and Routing

The MAC receiver can classify the received packets based on a VLAN tag and steer them to a specific Rx DMA channel. The MAC compares the received frame's VLAN tag with all the enabled and relevant filters and provides a filtering result. If any of the perfect filters give a pass result and if the respective filter's DMA channel number is enabled, the frame is routed to that DMA channel.

In addition to filtering, routing can also be done. For more details about routing, see Extended VLAN Based DMA Selection in Dynamic Mapping.

Comparison Modes

For each VLAN tag filter, the application has the following comparison options:

- It can program the MAC to compare an outer VLAN tag or an inner VLAN tag with the programmed VID.
- It can choose if 12 or 16 bits of the VID field need to be compared.
- Type check can be disabled or enabled for each filter; if enabled, the application can choose if the VID comparison is for SVLAN or CVLAN type frames only. For example, if a filter is enabled for 16-bit comparison,

SVLAN type, and outer VLAN tag, any single or double VLAN tagged frames with outer SVLAN tags are compared with this filter, and a pass or fail result is obtained.

NOTE: The inner VLAN tag comparison is applicable only if double VLAN tag processing is enabled using the `EMAC_VLANTAG_CTL.EDVLP` bit.

Filtering

When extended receive VLAN filtering and routing is enabled, the application can enable both perfect and hash filtering. The overall VLAN filter result is based on the perfect filter result and the hash filter result (if enabled). The filter result is passed to the application as part of the status bits.

Perfect filtering is done using the EMAC_MAC_VLAN_TAG_FILTER[n] registers (these registers are accessed indirectly using the `EMAC_VLANTAG_DAT` and `EMAC_VLANTAG_CTL` registers). For each VLAN tag filter, the MAC compares the relevant VLAN tag ID and gives a result. If any one of the VLAN tag filters gives a match, the frame is considered to have passed the VLAN tag filters. If the frame mismatches all the filters, the frame is considered to have failed the VLAN filter. This behavior is applicable only when the inverse filtering is not enabled in the `EMAC_VLANTAG_CTL` register.

If inverse filtering is enabled and the frame has mismatched all the relevant filters, then it is considered to have passed the VLAN filter. If the frame matches any one of the relevant filters, then it is considered as a fail. If none of the enabled filters can perform a comparison or if none of the filters are enabled, then the frame is bypassed to the application.

The overall filter result and the values programmed in the `EMAC_MACPKT_FILT.VTFE` and `EMAC_MACPKT_FILT.RA` fields determine if the frame must be dropped or forwarded to the application.

- When RA = 1 or VTFE = 0, The MAC forwards the frame, irrespective of the filter result.
- When RA = 0 and VTFE = 1, The MAC forwards the frame only if the VLAN tag filter status is pass. After forwarding a frame to the application, the relevant filter result is indicated through the status bits.

See Programming Extended VLAN Filtering and Routing on Receive.

VLAN Filter Status

The extended receive VLAN filtering and routing feature provides two status bits to indicate the comparison result of the VLAN tags.

By default, the MAC indicates the VLAN filter status through one bit in the status – VF in RDES2. When extended receive VLAN filtering and routing is enabled, two status bits are used to indicate the comparison result of VLAN tags. The outer VLAN tag filter pass and inner VLAN tag filter pass bits are defined in the following positions in various configurations. The status indicated through these bits is updated in RDES2 as follows:

- Bit 15 – Outer VLAN tag filter status
- Bit 14 – Inner VLAN tag filter status

Outer VLAN Tag Filter Status (OTS)

The outer VLAN tag filter pass bit provides the following status:

- In perfect filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame's outer VLAN tag has matched one of the VLAN tag filters.
- When this bit is reset, it indicates that the frame's outer VLAN tag has either failed the relevant outer VLAN tag filters or bypassed them.
- When none of the filters are enabled for outer VLAN tag comparison, then this bit is reset.
- When Inverse filtering is enabled and this bit is set, then the frame's VLAN tag has passed all the relevant VLAN tag filters. If it is reset, then it has failed at least one of or bypassed all the filters programmed for outer VLAN tag comparison.
- This bit is valid for both single and double VLAN tagged frames.

Inner VLAN Tag Filter Status (ITS)

The inner VLAN tag filter pass bit provides the following status:

- In perfect filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame's inner VLAN tag has matched one of the VLAN tag filters.
- When this bit is reset, it indicates that the frame's inner VLAN tag has either failed the relevant inner VLAN tag filters or bypassed them.
- When none of the filters are enabled for inner VLAN tag comparison, then this bit is reset.
- When inverse filtering is enabled and this bit is set, then the frame's VLAN tag has passed all the relevant VLAN tag filters. When it is reset, then it has failed at least one of or bypassed all the filters programmed for inner VLAN tag comparison.
- This bit is valid for only double VLAN tagged frames, when double VLAN processing is enabled.

The application must look at the status bits and the programming to determine if the frame has passed or failed the VLAN filter.

The *OTS and ITS Bit Values with At Least 1 Perfect Filter Enabled* table and *OTS and ITS Bit Values with Only VLAN Hash Filter Enabled* table show the possible filter combinations and the corresponding filter results. These tables explain the scenarios when double VLAN processing and hash VLAN filter are enabled in the design.

In the following tables:

- `EMAC_VLANTAG_CTL.VTIM`: VLAN tag inverse match enable – bit 17.
- HFO: Hash filter enabled for outer VLAN tag comparison – bit 25 (VHTM) in the `EMAC_VLANTAG_CTL` register.
- HFI: hash filter enabled for inner VLAN tag comparison – bit 27 (ERIVLT) in the `EMAC_VLANTAG_CTL` register.

- PFO – Perfect filter comparison enabled for outer VLAN tag – Any of the EMAC_MAC_VLAN_TAG_FILTER registers is enabled (bit 16 is set) and programmed for outer VLAN tag comparison (bit 20 is set to 0).
- PFI – Perfect filter comparison enabled for inner VLAN tag – Any of the EMAC_MAC_VLAN_TAG_FILTER registers is enabled (bit 16 is set) and programmed for inner VLAN tag comparison (bit 20 is set to 1).
- OTS – Outer VLAN tag filter status
- ITS – Inner VLAN tag filter status

The following table shows the possible values of status bits (OTS and ITS) when at least one perfect filter is enabled.

Table 30-44: OTS and ITS Bit Values with At Least 1 Perfect Filter Enabled

VTIM	HFO	HFI	PFO	PFI	OTS	ITS
0	0	0	0	1	0	1/0
0	0	0	1	0	1/0	0
0	0	0	1	1	1/0	1/0
0	1	0	1	1	1/0	1/0
0	1	0	1	0	1/0	0
0	1	0	0	1	1/0	1/0
0	0	1	1	1	1/0	1/0
0	0	1	1	0	1/0	1/0
0	0	1	0	1	0	1/0
1	0	0	0	1	0	1/0
1	0	0	1	0	1/0	0
1	0	0	1	1	1/0	1/0
1	1	0	1	1	1/0	1/0
1	1	0	1	0	1/0	0
1	1	0	0	1	1/0	1/0
1	0	1	1	1	1/0	1/0
1	0	1	1	0	1/0	1/0
1	0	1	0	1	0	1/0

The following table shows the possible values of status bits (OTS and ITS) when none of the perfect filters are enabled and only the VLAN hash filter is enabled.

Table 30-45: OTS and ITS Bit Values with Only VLAN Hash Filter Enabled

VTIM	HFO	HFI	OTS	ITS
0	0	0	0	0

Table 30-45: OTS and ITS Bit Values with Only VLAN Hash Filter Enabled (Continued)

VTIM	HFO	HFI	OTS	ITS
0	1	0	1/0	0
0	0	1	1/0	1/0
1	0	0	1/0	0
1	1	0	1/0	0
1	0	1	1/0	1/0

- When no perfect filters are enabled, any VLAN packet is considered to have bypassed the perfect filter.
- When VLAN hash filter is enabled for one of the tags, the respective status bit depends on the result of the filter. Status bits are set to 0 when VLAN hash filter is not enabled.
- The value 1/0 for the ITS/OTS field indicates that the final result is dependent on the result of the enabled relevant filter.

Example 1: The second row of the above table indicates that at least one perfect filter is enabled for outer VLAN tag comparison and none of the filters are enabled for Inner VLAN tag comparison. Inverse VLAN filtering is not enabled. The bit OTS is given as 1/0. When the received frame passes at the least one of the enabled Outer VLAN Tag filters, then the bit is set to 1. When the frame does not pass any of the enabled Outer VLAN Tag filters, then the bit is set to 0.

Example 2: Last row of the above table indicates that inverse filtering is enabled, Hash filter and at least one perfect filter is enabled for inner VLAN tag comparison, then when the received frame's inner VLAN tag mismatches with both the hash filter and all the enabled perfect filters, then the frame has the ITS bit set to 1. Otherwise, the bit is set to 0. The OTS bit is set to 0 as the comparison is not performed.

VLAN Stripping

Each of the VLAN tags has individual control over stripping. The programming options of always strip, never strip, strip on pass and strip on fail are available. Inner or outer VLAN tag stripping is based on the pass or fail results of the individual tag. If a tag is bypassed by all the relevant filters, stripping is not applicable for the tag.

- If strip on pass is enabled for the outer VLAN tag, stripping is done only if the outer VLAN tag has passed the relevant filters. The outer VLAN tag filter result bit is set.
- If strip on fail is enabled for the outer VLAN tag, stripping is done only if the outer VLAN tag has failed relevant filters. The Outer VLAN tag filter result bit is reset.
- If the Outer VLAN tag of the received frame is bypassed by the entire filter (no comparison has been made), the tag is not stripped, though the status bit is still 0.
- As multiple filters are enabled, it is possible that the received VLAN frame matches more than one filters. The VLAN tag's value is not always deterministic from the filter status bits.

- If the application strips the VLAN tag based on the filter result, it might lose the VID. If stripping is enabled for any of the tags, the tag can be put in the status. The application must enable the respective EMAC_VLANTAG_CTL.EVLRXS or EMAC_VLANTAG_CTL.EIVLRXS bits.

VLAN Filter Fail Packets Queue

When VLAN filtering is enabled, the VLAN filter fail packets can be routed to a programmable queue (EMAC_RXQ_CTL4.VFFQ) when the EMAC_MACPKT_FILT.RA bit = 1 or EMAC_MACPKT_FILT.VTFE = 0 and the enable bit (EMAC_RXQ_CTL4.VFFQE) for the queue is set.

The packets that pass the VLAN filtering are routed based on the VLAN TAG priority field. The VLAN tag priorities can be assigned to receive queues by programming the PSRQ field in the corresponding EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. The packets that fail the VLAN filter are discarded if EMAC_MACPKT_FILT.RA=0 or EMAC_MACPKT_FILT.VTFE =1. However, when RA=1 or VTFE=0, the VLAN filter fail packets are still forwarded to the application. In such a scenario, when the VLAN filter fail queue enable (EMAC_RXQ_CTL4.VFFQE) bit is set, the VLAN filter fail packets are forwarded to the receive queue number programmed in the EMAC_RXQ_CTL4.VFFQ bit. If VFFQE=0, the receive queue number is determined by the VLAN priority mapping as per the PSRQ fields.

The *Receive Queue Routing Table for Unicast Tagged Packets* shows the receive queue routing table for Unicast tagged packets, with the DA/SA filter enabled.

In the following table X = don't care; * = when UFFQE is enabled else PSRQ

Table 30-46: Receive Queue Routing Table for Unicast Tagged Packets

RA	VFTE	SA/DA Filter Result	VLAN Filter Result	VFFQE	Queue Routing
X	X	Pass	Pass	X	PSRQ
0	0	Pass	Fail	0	PSRQ
0	0	Pass	Fail	1	VFFQ
0	X	Fail	X	X	Dropped
0	1	Pass	Fail	X	Dropped
1	X	Fail	X	0	UFFQ*/PSRQ
1	X	Fail	X	1	UFFQ*/PSRQ
1	X	Pass	Fail	0	PSRQ
1	X	Pass	Fail	1	VFFQ

Layer 3 and Layer 4 Frame Filtering

The MAC supports layer 3 and layer 4 based frame filtering. The layer 3 filtering refers to the IP source or destination address filtering in the IPv4 or IPv6 frames whereas layer 4 filtering refers to the source or destination port number filtering in TCP or UDP.

When layer 3 and layer 4 filtering is enabled, the frames are filtered in the following way:

Matched Packets. The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the `EMAC_MAC_CFG.IPC` bit is set and one of the following conditions is true:

- All enabled layer 3 and layer 4 fields match
- At least one of the enabled fields match and other fields are bypassed or disabled

When multiple layer 3 and layer 4 filters are enabled, any filter match is considered as a match. IWhen more than one filter matches, the MAC provides the status of the lowest filter where filter 0 is the lowest filter and filter 3 is the highest filter. For example, when filter 0 and filter 1 match, the MAC gives the status corresponding to filter 0.

NOTE: The source or destination address and VLAN tag filters (if enabled) have precedence over layer 3 and layer 4 filter. This means that a packet which fails the source or destination address, or VLAN tag filter is dropped irrespective of the layer 3 and layer 4 filter results.

Unmatched Packets. The MAC drops the packets that do not match any of the enabled fields. Programs can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets. When a packet is dropped, the aborted or partial packets can be dropped in the MTL Rx FIFO. If the Rx FIFO operates in the threshold (cut-through) mode and the threshold is programmed to a small value, such that packet transfer to application starts before the failed layer 3 and layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.

Non-TCP or UDP IP Packets. By default, all non-TCP or UDP IP packets are bypassed from the layer 3 and layer 4 filters. The program can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 Filtering

The EMAC supports perfect matching or inverse matching for IP source address and destination address. The matching compares all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, the program can enable the last four data registers of a register set to contain the 128-bit IP source address or IP destination address. The IP source address or destination address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP source address or destination address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, the program can enable the second and third data registers of a register set to contain the 32-bit IP source address and IP destination address. The remaining two data registers are reserved. The IP source address or destination address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP source address and destination address in the received packet in the higher byte of the respective register.

Layer 4 Filtering

The EMAC supports perfect matching or inverse matching for TCP or UDP source and destination port numbers. However, only one type (TCP or UDP) can be programmed at a time. The first data register contains the 16-bit source and destination port numbers of TCP or UDP, that is, the lower 16 bits for source port number and higher 16 bits for destination port number.

The TCP or UDP source and destination port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP source and destination port number in the received packet is in the higher byte of the register.

Layer 3 and Layer 4 Filters Registers

The MAC implements a set of registers for layer 3 and layer 4 based frame filtering. In a register set, there is a control register, such as the EMAC_L3L4_CTL[i] register (layer 3 and layer 4 control register), to control the frame filtering. In addition, there are five address registers to program the layer 3 and layer 4 fields to be matched, which are:

- EMAC_L4_ADDR[i] (layer 4 Address Register)
- EMAC_L3_ADDR0_REG[i] (layer 3 address 0 register)
- EMAC_L3_ADDR1_REG[i] (layer 3 address 1 register)
- EMAC_L3_ADDR2_REG[i] (layer 3 address 2 register)
- EMAC_L3_ADDR3_REG[i] (layer 3 address 3 register)

EMAC Station Management Interface (SMI)

The IEEE 802.3 MII station management interface, also known as the MDIO management interface, allows the processor to monitor and control one or more external Ethernet physical-layer transceivers. (Physical-layer transceivers are commonly called PHYs). The management interface physically consists of a 2-wire serial connection composed of the MDC (management data clock) output signal and the MDIO (management data input/output) bidirectional data signal. The IEEE 802.3 MII station management interface applies to both MII/RMII.

The application can address only one register in the PHY in any given time and send control data or receive status information. All the transfers are initiated by the EMAC CORE, and the PHY chip only acts as a target device.

Standard PHY control and status registers typically provide

- Device capability status bits (for example: auto-negotiation, duplex modes, 10/100 speeds, and protocols)
- Device status bits (for example: auto-negotiation complete, link status, remote fault)
- Device control bits (for example: reset, speed selection, loop back, and auto-negotiation start)

Upon power-up, an MDIO read access (at default rates) of device capabilities in PHY status registers can determine the supported PHY features.

The MII management logical interface specifies:

- A set of 16-bit device control or status registers within the PHYs, including both required registers with standardized bit definitions as well as optional vendor-specified registers.
- A 5-bit device addressing scheme which allows the MAC to select one of up to 32 externally connected PHY devices.

- A 5-bit register addressing scheme for selecting the target register within the addressed device.
- A transfer frame protocol for 16-bit read and write accesses to PHY registers through the MDC and MDIO signals under control of the MAC.

Table 30-47: Station Management Interface pins

Station Management Interface Pins	Pin Description
MDIO – Management Data I/O	A periodic clock that runs at a maximum period of 400 ns. Always driven by the EMAC to PHY.
MDC – Management Data Clock	Data signal driven by EMAC or PHY, depending on write or read access based on EMAC; synchronous to MDC.

MDC Clock Frequency

The EMAC uses the `EMAC_MDIO_ADDR.CR` bit field to determine the frequency of MDC as shown in the *MDC Clock Frequency Selection* table. The clock range selection determines the frequency of the clock relative to the CLKO7 frequency. The table shows the suggested range of CLKO7 frequency applicable for each value of the `EMAC_MDIO_ADDR.CR` field. The programmability based on CLKO7 frequency range ensures that the MDC clock frequency range is within the IEEE specifications of 1.0 MHz to 2.5 MHz. However, the EMAC MDC can also support higher frequencies for PHY devices that support the frequencies.

Table 30-48: MDC Clock Frequency Selection

Selection	Programmed Frequency Range	Frequency of MDC	Min and Max MDC Freq (Per Specifications)
0000	60–100 MHz	CLKO7/42	MIN = 1.43 MHz and MAX = 2.39 MHz
0001	100–150 MHz	CLKO7/62	MIN = 1.61 MHz and MAX = 2.42 MHz
0010	20–35 MHz	CLKO7/16	MIN = 1.25 MHz and MAX = 2.19 MHz
0011	35–60 MHz	CLKO7/26	MIN = 1.35 MHz and MAX = 2.31 MHz

The ***MDIO Frame Parameters*** table provides MDIO data transfer parameters. The write and read sequences provided in the tables, ***MDIO Write Data Sequence*** and ***MDIO Read Data Sequence***, are based on these parameters.

Table 30-49: MDIO Frame Parameters

Parameter	Description
IDLE	The MDIO line is three-state (noted as Z in sequence); there is no clock on MDC.
PREAMBLE	32 continuous bits, each of value 1
START	Start of frame is 01
OPCODE	10 for read and 01 for write
PHY ADDR	5-bit address select for one of 32 PHYs (noted as AAAAA in sequence)
REG ADDR	Register address in the selected PHY (noted as RRRRR in sequence)

Table 30-49: MDIO Frame Parameters (Continued)

Parameter	Description
TA	Turnaround is Z0 for read and 10 for write (Z = high impedance)
DATA	Any 16-bit value. Driven by MAC or PHY based on direction (noted as DDD...DDD).

Table 30-50: MDIO Write Data Sequence

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	111...111	01	01	AAAAA	RRRRR	10	DDD... DDD	Z

Table 30-51: MDIO Read Data Sequence

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	111...111	01	10	AAAAA	RRRRR	Z0	DDD... DDD	Z

MDIO Frame Structure

Both Clause 22 and Clause 45 MDIO frame structures are supported by the EMAC. The `EMAC_MDIO_ADDR.C45E` bit can be programmed to enable Clause 22 or Clause 45 mode of operation. The *MDIO Clause 45 Frame Structure* table shows the Clause 45 frame format and the *MDIO Clause 22 Frame Structure* table shows the Clause 22 frame format. In addition to normal read and write operations, the SMA also supports post-read increment address while operating in Clause 45 mode.

Table 30-52: MDIO Clause 45 Frame Structure

Field	Description
IDLE	The MDIO line is in three-state; there is no clock on MDC
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 2’b00
OPCODE	00/01/10/11
PHY ADDR	5-bit address select for one of 32 PHYs
DEV ADDR	5-bit address select for one of 32 devices
TA	Turnaround is Z0 for read and 10 for write (Z = high impedance)
DATA/ADDRESS	16-bit value. For an address cycle (OPCODE = 2'b00), this frame contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, this field contains the data that is written to the register. For read or post-read increment address frames, this field contains the contents of the register read from the PHY. • In address and data write cycles, the EMAC drives the MDIO line during the transfer of these 16 bits.

Table 30-52: MDIO Clause 45 Frame Structure (Continued)

Field	Description
	• In read and post-read increment address cycles, the PHY drives the MDIO line during the transfer of these 16 bits.

Table 30-53: MDIO Clause 22 Frame Structure

Field	Description
IDLE	The MDIO line is in tri-state; there is no clock on MDC
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 2'b01
OPCODE	10 for read and 01 for write
PHY ADDR	5-bit address select for one of 32 PHYs
DEV ADDR	5-bit address to select the register within each MMD
TA	Turnaround is Z0 for read and 10 for write (Z = high impedance)
DATA	Any 16-bit value. Driven by MAC or PHY based on direction (noted as DDD...DDD).

Table 30-54: MDIO Write Data Sequence

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	111...111	01	01	AAAAA	RRRRR	10	DDD...DDD	Z

Table 30-55: MDIO Read Data Sequence

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	111...111	01	01	AAAAA	RRRRR	Z0	DDD...DDD	Z

SMI Write Operation

When programs set the EMAC_MDIO_ADDR.GOC_1/EMAC_MDIO_ADDR.GOC_0 bit field to 2'b01 and the EMAC_MDIO_ADDR.GB bit to 1, the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (EMAC_MDIO_DATA register) to the SMI to initiate a write operation into the PHY registers. At this point, the SMI module starts a write operation on the GMII Management Interface using the management packet format specified in the GMII specifications (as per IEEE 802.3-2002, Section 22.2.4.5).

When the SMA module starts a write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The EMAC_MDIO_ADDR.GB bit is set high until the write operation is complete. The CSR ignores the write operations performed to the EMAC_MDIO_ADDR register or the EMAC_MDIO_DATA register during this period (the EMAC_MDIO_ADDR.GB bit is high). When the write operation is complete, the SMI module indicates this to the CSR, and the CSR resets the EMAC_MDIO_ADDR.GB bit.

The *SMI Write Operation through MDIO/MDC Pins* figure shows this operation.

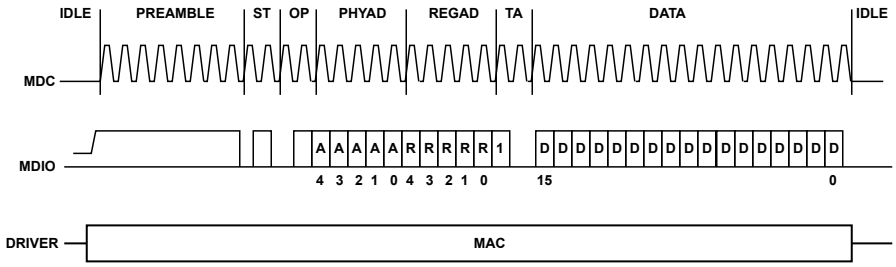


Figure 30-27: SMI Write Operation through MDIO/MDC Pins

SMI Read Operation

When programs set the EMAC_MDIO_ADDR.GOC_1/EMAC_MDIO_ADDR.GOC_0 bit field to 2'b11 and the EMAC_MDIO_ADDR.GB bit to 1, the MAC CSR module transfers the PHY address and the register address in PHY to the SMI to initiate a read operation in the PHY registers. The SMI starts a read operation on the GMII Management Interface using the management packet format specified in the GMII specifications (as per IEEE 802.3-2002, Section 22.2.4.5).

When the SMI starts a read operation on the MDIO, the CSR ignores the write operations to the EMAC_MDIO_ADDR or EMAC_MDIO_DATA registers during this period (the EMAC_MDIO_ADDR.GB bit is high) and the transaction is completed without any error. When the read operation is complete, the SMI indicates this to the CSR. The CSR resets the GB bit and updates the EMAC_MDIO_DATA register with the data read from the PHY.

The *SMI Read Operation through MDIO/MDC Pins* figure shows this operation.

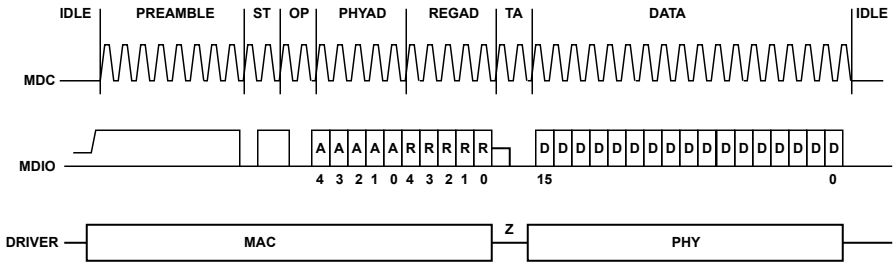


Figure 30-28: SMI Read Operation through MDIO/MDC Pins

Preamble Suppression

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The EMAC provides controls to support preamble suppression. EMAC transmits MDIO frames with only 1 preamble bit. The preamble suppression can be enabled by setting the EMAC_MDIO_ADDR.PSE bit.

Trailing Clocks and Back-to-Back Transactions

The EMAC drives MDC clock on the MDC port for duration of the MDIO frame. There is no clock driven during the idle period. The trailing clocks feature can be used if the PHY needs the MDC clock to be active for some cycles after the MDIO frame. The EMAC_MDIO_ADDR.NTC bit field allows programming of trailing clocks from 0 to 7.

The EMAC supports back to back transactions which allows start of next MDIO frame even before the trailing clocks are completed for previous MDIO frame. This feature can be enabled by setting the EMAC_MDIO_ADDR.BTB bit when trailing clocks feature is also enabled. When back to back transactions is enabled, the GMII busy is cleared immediately after MDIO frame completion. This allows the software to issue the next command, which is executed by EMAC while trailing clocks are still on for the previous MDIO frame. When back to back transactions is not enabled, the GMII busy is cleared after the trailing clocks are completed for MDIO frame.

Interrupt for MDIO Transaction Completion

The EMAC can generate an interrupt on completion of MDIO read or write transactions. Therefore, the application need not poll the GMII busy field of the EMAC_MDIO_ADDR register to know the completion of MDIO commands.

EMAC Management Counters (MMC)

The EMAC provides a comprehensive set of 32-bit MAC management counters. It uses these counters for gathering statistics on the received and transmitted frames. The MMC sub block also includes

- A control register (EMAC_MMC_CTRL) for managing the behavior of the counters
- Two 32-bit registers containing interrupts generated (EMAC_MMC_RXINT and EMAC_MMC_TXINT)
- Two 32-bit registers containing masks for the interrupt register (EMAC_MMC_RXIMSK and EMAC_MMC_TXIMSK)

The MMC receive counters are updated for frames passed by the address filtering sub block in the EMAC CORE. Statistics of frames dropped by the AFM module are not updated unless they are runt frames of less than 6 bytes. (Destination address bytes are not received fully.) The module is also capable of gathering statistics on encapsulated IPv4, IPv6, and TCP, UDP, or ICMP payloads in received Ethernet frames.

The MMC register naming conventions are as follows:

- Tx as a prefix or suffix indicates counters associated with transmission.
- Rx as a prefix or suffix indicates counters associated with reception.
- G as a suffix indicates registers that count good frames only.
- GB as a suffix indicates registers that count frames regardless of whether they are good or bad.

Transmitted frames are considered *good* when transmitted successfully. In other words, a transmitted frame is good if the frame transmission does not abort due to any of the following errors:

- Jabber timeout
- No carrier or loss of carrier
- Late collision
- Frame underflow

- Excessive deferral
- Excessive collision

Received frames are good when none of the following errors exists:

- CRC error
- Runt frame (shorter than 64 bytes)
- Alignment error
- Length error (non-type frames only)
- Out-of-range (non-type frames only, longer than maximum size)

The maximum frame size depends on the frame type, as follows:

- Untagged frame maximum size = 1518
- VLAN frame maximum size = 1522
- Jumbo frame maximum size = 9018
- Jumbo VLAN frame maximum size = 9022

The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the *Size of the Maximum Receive Packet* table.

Table 30-56: Size of the Maximum Receive Packet

JE	S2KP	GPSLCE	EDVLP	Untagged Frame maximum size in bytes	Single VLAN Frame maximum size in bytes	Double VLAN Frame maximum size in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

The EMAC_MMC_CTRL register also contains bits that control preset, freeze and roll-over of counters. The EMAC uses the EMAC_MMC_CTRL.RSTONRD bit to enable an auto-reset feature whenever the counters are read. The EMAC uses the EMAC_MMC_CTRL.CNTRST bit to reset all the counters.

The MMC can trigger an interrupt when the corresponding bits are enabled in the transmit, receive, and IPC mask registers, and when the counter reaches half or full. The status is also updated in the corresponding interrupt register.

MMC Receive Interrupt Register

The EMAC_MMC_RXINT register maintains the interrupts that are generated when the receive statistic counters reach half their maximum values (0x80000000), and when they cross their maximum values (0xFFFFFFFF). When EMAC_MMC_CTRL.CNTSTOPRO bit is set, then interrupts are set, but the counter remains at all ones. The EMAC_MMC_RXINT register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits 7:0) of the respective counter must be read to clear the interrupt bit.

MMC Transmit Interrupt Register

The EMAC_MMC_TXINT register maintains the interrupts generated when the transmit statistic counters reach half their maximum values (0x80000000), and when they cross their maximum values (0xFFFFFFFF). When EMAC_MMC_CTRL.CNTSTOPRO is set (= 1), then interrupts are set, but the counter remains at all ones. The EMAC_MMC_TXINT register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits 7:0) of the respective counter must be read to clear the interrupt bit.

MMC Receive Checksum Offload Interrupt Register

The EMAC_MMC_IPC_RXINT register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x80000000), and when they cross their maximum values (0xFFFFFFFF). When the EMAC_MMC_CTL.NOROLL is set, then interrupts are set, but the counter remains at all ones. The EMAC_MMC_IPC_RXINT register is 32 bits wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The least-significant byte lane (bits 7:0) of the counter must be read to clear the interrupt bit.

EMAC Precision Time Protocol (PTP) Engine

The following sections describe the precision time protocol (PTP) engine.

IEEE1588 and the PTP Engine

The Ethernet MAC peripheral includes a PTP engine to assist applications requiring time synchronization. The PTP module is tightly integrated with the EMAC CORE to aid hardware time stamping defined in the IEEE1588 2002/2008 standards. Applications can use accurate hardware time stamps through TCP/IP stacks (if using network layer communication) to exchange time information across devices connected over network. Applications can also use accurate hardware time stamps through Ethernet device drivers (if using MAC layer communication) to exchange time information.

PTP Engine

For calculation of drift in time between two Ethernet devices, the device records its system time whenever a timing message is sent or received (IEEE 1588 protocol). Due to the indeterministic delay of a software system for a node, the software is unable to capture an accurate time when the message is sent or received. However, the hardware can monitor the signal on the communication media and get an accurate message of arrival and departure time.

The PTP (precision time protocol) module is closely integrated with the EMAC module. It provides hardware assistance to implement both the IEEE 1588–2002 and IEEE 1588–2008 standards on Ethernet (IEEE 802.3). It takes

one input clock signal as its PTP clock and maintains the timing information (called *system time*) at the nanosecond level.

The PTP module includes hardware for clock and system time adjustment. The pulse-per-second (PPS) signals physically represent the system time. PPS can be programmed to a fixed frequency or provide flexibility to the signal in terms of pulse width, interval, start, and stop time of the signals. The PTP module can be programmed to trigger an alarm interrupt when system time reaches specified time.

The PTP module can be programmed to detect different types of received frames, capture the system time, and time stamp those frames with the captured system time. Programs can configure any frame so that the PTP module capture the system time when it is transmitted. The PTP module can also capture the system time when an event is detected on the auxiliary snapshot trigger input pins (`EMAC_PTPAUX_MCG_IN[n]`).

IEEE 1588 Standard

Many systems require two independent devices to operate in a time synchronized fashion. If each system relied solely on its oscillator, differences between the characteristics and operating conditions of each oscillator would limit the ability of the clocks to operate synchronously. To serve applications requiring synchronized clocks, the system uses a periodic correction mechanism.

A simple way to synchronize multiple systems is to choose one system (with the best clock) as a requester. The system requester broadcasts the clock and timing information to other systems (completers); subsequently, the completers adjust their clocks and timing according to that of requester. However, this method has limitations. The requester cannot broadcast the time at infinitesimal intervals. Path delay (propagation delay) exists between a requester and a completer and the delay varies between each completer and requester.

IEEE 1588 is also known as precision time protocol or PTP. The standard specifies a protocol used to synchronize the time and clock of multiple devices, dispersed but interconnected by any communication, (for example, Ethernet IEEE 802.3). According to the protocol, timing messages are exchanged between two devices. Then, one of the devices calculates its drift from other device and corrects its system time. (Both devices must have the same representation of their system time). The protocol resolves path delay between devices. It also helps synchronize the clocks of multiple devices and all of the limitations mentioned are resolved.

IEEE 1588 was published in 2002 where four types of timing messages were defined: Sync, Follow_Up, Delay_Req, and Delay_Resp. Here the protocol synchronizes two or more devices where one is a requester and others are completers. The requester device sends Sync, Follow_Up, and Delay_Resp messages to the completer device in the system. The target sends the Delay_Req messages to the requester device. A following section provides more information on IEEE 1588–2002.

In 2008, a newer version of IEEE 1588 was introduced which provides further mechanisms to measure the peer-to-peer delay. Three more timing messages (PdelayReq, PdelayResp, and PdelayRespFollowup) were added to implement peer-to-peer synchronization. The following section provides more information on IEEE 1588–2008.

IEEE 1588–2002

The IEEE 1588–2002 standard defines the precision time protocol (PTP). The protocol allows precise synchronization of clocks in measurement and control systems that use network communication, local computing, and distributed objects. The protocol applies to systems that communicate by local area networks that support multicast

messaging, including (but not limited to) Ethernet. This protocol also allows heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range using minimal network and local clock computing resources.

The PTP is transported over UDP/IP. The system or network is classified into requester and completer nodes for distributing the timing or clock information. The ***IEEE 1588–2002 PTP Process*** figure shows the PTP process used for synchronizing a target node to a controller node by exchanging PTP messages.

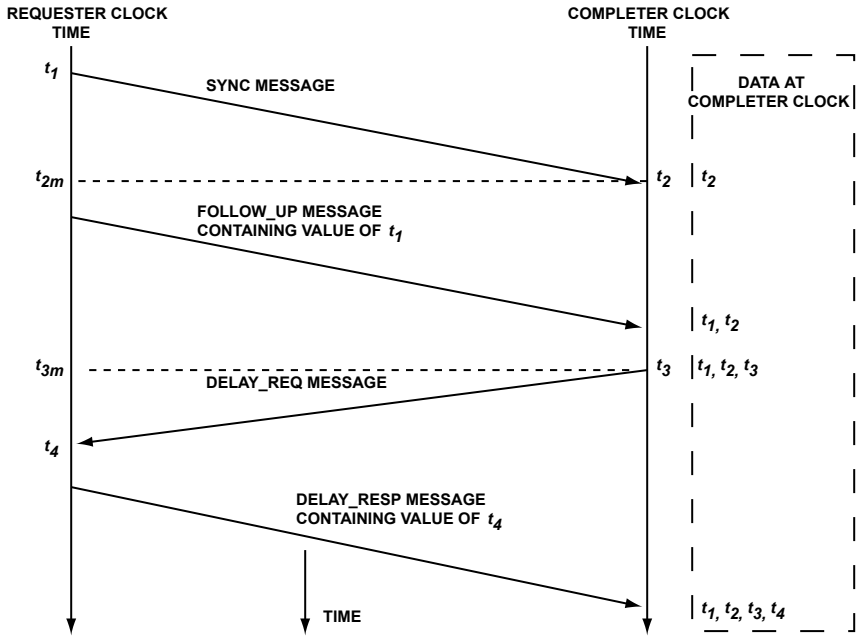


Figure 30-29: IEEE 1588–2002 PTP Process

As shown in the figure, the PTP uses the following process:

1. The requester broadcasts the PTP sync messages to all its nodes. The sync message contains the reference time information of the requester. The time at which this message leaves the system of the requester is t_1. The requester must capture this time for Ethernet ports, at RMII.
2. The completer receives the Sync message and also captures the exact time, t_2, using its timing reference.
3. The requester sends a Follow_up message to the completer, which contains t1 information for later use.
4. The completer sends a Delay_Req message to the requester, noting the exact time, t_3, at which this frame leaves the RMII.
5. The requester receives the message, capturing the exact time, t_4, at which it enters its system.
6. The requester sends the t_4 information to the completer in the Delay_Resp message.
7. The completer uses the four values of t_1, t_2, t_3, and t_4 to synchronize its local timing reference to the timing reference of the requester.

Most of the PTP implementation occurs in the software above the UDP layer. However, the hardware support must capture the exact time when specific PTP packets enter or leave the Ethernet port at the RMII. Hardware must capture this timing information and return it to the software for the proper implementation of PTP with high accuracy.

IEEE 1588–2008 Advanced Time Stamps

In addition to the basic time stamp features mentioned in IEEE 1588–2002 time stamps, the EMAC supports the following advanced time stamp features defined in the IEEE 1588–2008 standard.

- Support for the IEEE 1588–2008 (Version 2) time stamp format.
- Provides an option to take snapshot of all frames or only PTP type frames.
- Provides an option to take snapshot of only event messages.
- Provides an option to select the node to be a requester or completer.
- Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status.
- Provides an option to run nanoseconds time in digital or binary format.

Peer-to-Peer (P2P) PTP Message Support

The IEEE 1588–2008 version supports Peer-to-Peer PTP (Pdelay) message in addition to SYNC, Delay Request, Follow-up, and Delay Response messages. Refer to the ***Propagation Delay Calculation between Nodes Supporting P2P Path Correction*** figure. The figure shows the method to calculate the propagation delay between nodes supporting peer-to-peer path correction.

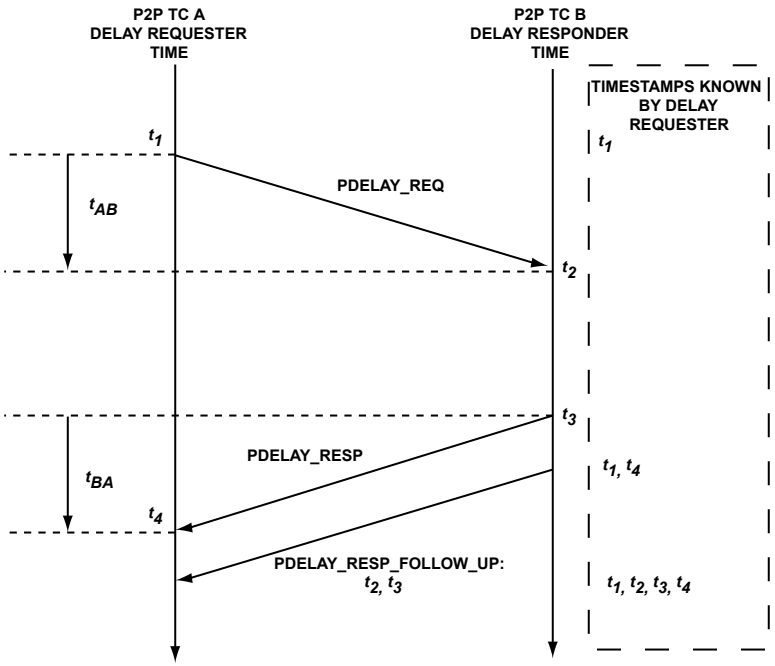


Figure 30-30: Propagation Delay Calculation between Nodes Supporting P2P Path Correction

As shown in the figure, the propagation delay is calculated in the following way:

1. Port 1 issues a Pdelay_Req message and generates a time stamp, t1, for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a time stamp, t2, for this message.
3. Port 2 returns a Pdelay_Resp message and generates a time stamp, t3, for this message. To minimize errors due to frequency offset between the two ports, port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. The port 2 returns any one of the following:
   - The difference between the time stamps t2 and t3 in the Pdelay_Resp message.
   - The difference between the time stamps t2 and t3 in the Pdelay_Resp_Follow_Up message.
   - The time stamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.
4. Port 1 generates a time stamp, t4, on receiving the Pdelay_Resp message.

Port 1 uses all four time stamps to compute the mean link delay.

Block Diagram

The ***PTP Block Diagram*** figure shows the functional block diagram of PTP module.

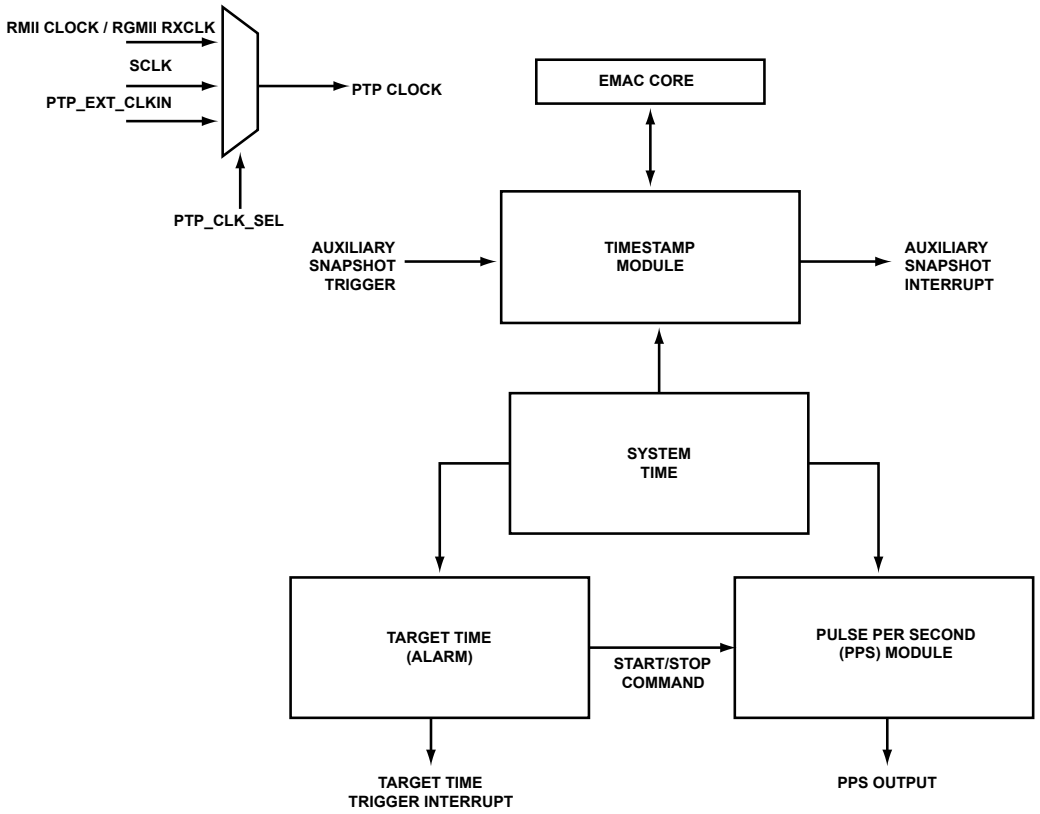


Figure 30-31: PTP Block Diagram

A system time module is present which keeps the time of PTP module. It consists of hardware which can be programmed for time initialization, time correction, and clock correction.

The time stamp module captures the time (provided by the system time module) at various conditions. For example, when the EMAC sends or receives a frame or during the rising edge of the auxiliary snapshot trigger `EMAC_PTPAUX_MCG_IN[n]` pins. When system time is captured after detection of a frame, the time stamp module automatically includes the time information in the frame descriptor. Time stamping on the detection of a frame can be programmed on a per frame basis.

The PTP clock drives the PTP module. This clock can be selected from three different clock sources.

The Pulse per Second (PPS) module generates a pulse or train of pulse on the PPS output pins, (`EMAC_PTPPPS[n]`). It is the physical representation of system time. PPS can be fixed (where only frequency varies) or flexible (where width, interval, start time, and stop time can be programmed).

The target time module acts as an alarm for the PTP module. Whenever system time reaches a value equal to programmed target time, the target time trigger interrupt is generated. By appropriate programming, The target time trigger can also start or stop flexible PPS output at specific time.

PTP Module Clock

The PTP module clock features include Clock Source Selection and Clock Frequency Range .

Clock Source Selection

The PTP module can take one of these clock sources as its input clock — CLKO7, RMII reference RGMII Rx clock, or PTP external clock.

As shown in the ***PTP Clock Source Selection*** table, the `PADS_PCFG0` register [bits 1:0] selects the PTP clock source.

Table 30-57: PTP Clock Source Selection

PADS_PCFG0 [1:0]	PTP Clock Source	Clock Description
00 – RMII or MII	ETH0_RXCLK_REFCLK/ ETH1_REFCLK	MII/RMII reference clock
00 – RGMII	CLKO7	Processor Ethernet clock
10	PTP external clock	Clock available on EMAC_PTPCLKIN[n] pin
01 or 11	SCLK0	Processor system clock

Clock Frequency Range

The resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance. The timing constraints achievable for logic operating on the selected PTP clock source limit the maximum PTP clock frequency.

The minimum PTP clock frequency depends on the time required between two consecutive frames. The IEEE specification determines the RMII clock frequency. The minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC. See the ***Minimum PTP Clock Frequency*** table.

A minimum delay required between two consecutive time stamp captures is 8 clock cycles of RMII and 3 clock cycles of PTP clocks. If the delay between two time stamp captures is less than this delay, the EMAC does not take a time stamp snapshot for the second frame.

The ***Minimum PTP Clock Frequency*** table assumes:

- Minimum Ethernet packet size = 64 bytes
- Mininum IFG = 96 bit times or 12 bytes
- Preamble = 8 bytes
- 3 PTP clock + 4 GMII/MII clock minimum gap between two SFDs (start of frame delimiters)
- 3 PTP clock + 8 RMII clock minimum gap between two SFDs

Table 30-58: Minimum PTP Clock Frequency

Mode (Full Duplex in all cases)	Minimum Gap between two SFDs	PTP clock	Comments
10 Mbps RMII	64 bytes of data + 8 bytes of preamble + min IFG	3 PTP clock cycle + 8RMII clock cycle 3360 RMII clock cycles	In RMII @ 10 Mbps, 1 byte transmitted in 40 clock cycles
	(2560 + 320 + 480) RMII clock cycles	3 PTP clock cycle 3352 RMII clock cycles	RMII clock = 50 MHz
	3360 RMII clocks	PTP clock cycle 1117.33 20 ns = 22346 ns	1 RMII clock cycle = (1/50 MHz) = 20 ns
		PTP frequency min = 0.045 MHz	
100 Mbps RMII	64 bytes of data + 8 bytes of preamble + min IFG	3 PTP clock cycle + 8RMII clock cycle 336 RMII clock cycles	In RMII, 1 byte transmitted in 4 clock cycles
	(256 + 32 + 48) RMII clock cycles	3 PTP clock cycle 328 RMII clock cycles	RMII clock = 50 MHz
	336 RMII clocks	PTP clock cycle 109.33 20 ns = 2186 ns	1 RMII clock cycle = (1/50 MHz) = 20 ns
		PTP frequency min = 0.45 MHz	

The minimum PTP clock frequency also depends on the maximum value of the `EMAC_TM_SUBSEC` register. Even at the highest `EMAC_TM_SUBSEC.SSINC` value, the `EMAC_TM_SEC` register value can be incremented every second. Since the `EMAC_TM_SUBSEC.SSINC` is an 8-bit field, the minimum PTP clock frequency allowed is approximately 4 MHz.

Time Stamp Module

The time stamp module captures time in seconds and nanoseconds maintained as system time. The time stamp module also captures time when specific events occur. Events include detection of a frame transmitted or received

over the EMAC and a rising edge on the EMAC_PTPAUX_MCG_IN[n] pins. The time stamp module does not need to time stamp all of the transmitted or received frames over the EMAC. The PTP module can be programmed to detect specific kinds of frames for time stamping.

Frame Detection and Time Stamping

The PTP module automatically monitors all received and transmitted IEEE 1588 event messages on the Ethernet. If the module detects an event message, it takes a snapshot of the system time. The PTP module stores the value to the 64-bit fields in transmit or receive descriptor.

The time stamping occurs at the EMAC RMII/RGMII interface when the module sees the start of frame of an event message packet.

Transmit Path Time Stamping

The MAC captures a time stamp when the Start Packet Delimiter (SFD) of a packet is sent on the MII/RMII/RGMII interface. The packets for which the time stamps must be captured, can be controlled on per-packet basis. Each transmit packet can be marked to indicate whether a time stamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. The program needs to specify the packets to capture time stamps.

The program can specify the packets by using the control bits in the transmit descriptor. The MAC returns the time stamp to the software inside the corresponding transmit descriptor, connecting the time stamp automatically to the specific PTP packet. The 64-bit time stamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the time stamp.

Receive Path Time Stamping

The MAC can be programmed to capture the time stamp of all packets received on the GMII or MII interface or to process packets to identify the valid PTP messages. Use the following options of the EMAC_TM_CTL register to control the snapshot of the time to be sent to the application:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 time stamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable time stamp snapshot for the received packet for IPv4 or IPv6
- Enable time stamp snapshot only for event messages (Sync, Delay_Req, Pdelay_Req, or Pdelay_Resp)
- Enable the node to be a main or target and select the snapshot type

This feature controls the type of messages for which snapshots are taken.

Refer to the ***PTP Frame Type Selections*** table.

Table 30-59: PTP Frame Type Selections

TSENALL (bit 8)	SNAPTYPSEL (bits [17:16])	TSMSTRENA (bit 15)	TSEVNTENA (bit 14)	Frames
1	X	X	X	All
0	00	X	0	Sync, Follow_Up, Delay_Req, Delay_Resp
0	00	0	1	Sync
0	00	1	1	Delay_Req
0	01	X	0	Sync, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_ Follow_Up
0	01	0	1	Sync, Pdelay_Req, Pdelay_Resp
0	01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
0	10	X	X	Sync, Delay_Req
0	11	X	X	Pdelay_Req, Pdelay_Resp

Time Stamp Correction

According to the IEEE 1588 specification, a time stamp must be captured when the PTP message time stamp point (leading edge of the first bit of the octet immediately following the SFD octet) crosses the boundary between the node and the network.

Because the MAC takes the time stamp at an internal point far from the actual boundary of the node and network, this captured time stamp is corrected/updated for the ingress/egress path latency (including the delay in the PHY layers). Further correction is done for the inaccuracies/errors introduced due to the clock (RGMII Tx, Rx clock). The clock can be different at the capture point when compared with the PTP clock that is used to generate the time. The resultant Clock Domain Crossing (CDC) circuits add error, depending on the clock period of the GMII and PTP clocks.

Ingress Correction

In the receive side, the time stamp captured at the internal snapshot point is delayed (later in time) as compared to the time at which that packet's SFD bit is received at the port's boundary. Therefore, the captured time stamp must be reduced by the ingress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the `EMAC_TM_INGCORR_NSEC` and `EMAC_TM_INGCORR_SNSEC` registers.

The correction value consists of the following components:

1. External latency in the PHY layer between boundary point and the input of the core. If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has a register indicating the maximum and minimum ingress latency. The software can read these registers and determine the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), the ingress latency must be determined from its data sheet or timing characteristics.

2. Internal latency from the input of the core to the internal capture point. The internal ingress latency can be read from the EMAC_TM_ING_LTNCY register. This read-only register gives the latency in scaled nanoseconds format, as defined in IEEE 1588 Clause 5.3.2. The latency differs based on the active PHY interface (RGMII, RMII, so on) and the operating speed. Therefore, the software must read this register after any speed change in the MAC to determine the current internal latency.
3. CDC Synchronization. The CDC synchronization error is approximately equal to two times the clock period of the PTP clock.

Software should add the values determined from these components and write them into the EMAC_TM_INGCORR_NSEC.TSIC and EMAC_TM_INGCORR_SNSEC.TSICSNS bit fields.

NOTE: The value written to the register must be negative (two's complement), because it must be subtracted from the captured time stamp. The MAC receiver adds the value in this register to the captured time stamp and then gives the resultant value as the time stamp of the received packet.

When the EMAC_TM_CTL.TSCTRLSSR bit is set, the nanoseconds field of the captured time stamp is in decimal format with a granularity of 1 ns. So, bit 31 of EMAC_TM_INGCORR_NSEC.TSIC must be set (= 1) (for negative value). Bits[30:0] must be written with "109 - total ingress_correction_value[nanosecond part]" represented in binary. For example, if the required correction value is -5 ns, the value is 0xBB9A_C9FB.

When EMAC_TM_CTL.TSCTRLSSR bit is reset, the nanoseconds field of the captured time stamp is in binary format with a granularity of ~0.466 ns. Therefore, bits [30:0] must be written with "231 - total ingress_correction_value" represented in binary with bit[31] = 1.

Egress Correction

In the transmit side, the time stamp captured at the internal snapshot point is earlier (advanced in time) when compared with the time at which that packet's SFD bit is output at the port's boundary. Therefore, the captured time stamp must be compensated by the egress latency and the errors in CDC sampling. Software determines/calculates this correction value and writes it into the EMAC_TM_EGCORR_NSEC and EMAC_TM_EGCORR_SNSEC registers.

The correction value consists of the following components:

1. External latency in the PHY layer between the output of the core and the boundary of the port and the network. If the PHY is compliant to the IEEE 802.3 Clause 45 MMD registers, it has a register indicating the maximum and minimum egress latency. The software can read these registers and determine the average egress latency in the PHY. Otherwise, (if the PHY does not support these registers), the egress latency must be determined from its data sheet or timing characteristics.
2. Internal latency from the internal capture point and the output of the core. This internal egress latency can be read from the EMAC_TM_EG_LTNCY register. This read-only register gives the latency in scaled nanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs based on the active PHY interface (RGMII, RMII, so on) and the operating speed. Therefore, the software must read this register after any speed change in the MAC to determine the current internal latency.

3. CDC synchronization error. The CDC synchronization error value depends on the one-step time stamping mode. When one-step time stamping is enabled, the value = (1 x period of PTP clock + 4 x period of Tx clock).

The ***PHY Interface Latency*** table lists the egress and ingress latency for various PHY interfaces.

Table 30-60: PHY Interface Latency

PHY Interfaces		Egress Latency	Ingress Latency
RGMII	1.0 Gbps	12	12
RGMII	100 Mbps	40	40
RGMII	10 Mbps	400	400
RMII	100 Mbps	40	40
RMII	10 Mbps	400	400

PTP Processing and Control

When the EMAC module receives a frame, frame detection and time stamping are based on some of the PTP fields in the frame. The ***PTP Message Format (IEEE 1588–2008)*** table shows the common message header for the PTP messages. This format is derived from the IEEE standard 1588–2008. When the EMAC module sends a PTP frame, the frame follows this format.

When a frame is received, the PTP module compares these fields with standard values and finds out the type of PTP frame and other information (for example, PTP version, IP version, and others). The module then updates the related fields in RDES4. When a frame is transmitted, programs must ensure that all the fields are appropriate. The PTP module on the other end of a communication must correctly detect and decode the frame.

Table 30-61: PTP Message Format (IEEE 1588–2008)

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30

Table 30-61: PTP Message Format (IEEE 1588–2008) (Continued)

Bits	Octets	Offset
controlField (used in version 1. For version 2, messageType field is used for detecting different message types.)	1	32
logMessageInterva	1	33

There are some fields in the Ethernet payload that can be used to detect the PTP packet type and control the snapshot taken. These fields are different for the following PTP frames:

- PTP Frames Over IPv4
- PTP Frames Over IPv6
- PTP Frames Over Ethernet

For these PTP frames, EMAC does not consider the PTP version 1 messages as valid when the frame consists of peer delay multicast address as destination address (DA).

PTP Frame Over IPv4

The ***IPv4-UDP PTP Frame Fields Required for Control and Status*** table provides information about the fields that are matched to control snapshot for the PTP packets. The packets are sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. The positions are based on IEEE 1588–2008 standards and the message format. The format is defined in the ***PTP Message Format (IEEE 1588–2008)*** table in the PTP Processing and Control section.

Table 30-62: IPv4-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	0x0800	IPv4 datagram
IP Version and Header Length	14	0x45	IP version is IPv4
Layer 4 Protocol	23	0x11	UDP
IP Multicast Address (IEEE 1588 Version 1)	30, 31, 32, 33	0xE0,0x00, 0x01,0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast Address (IEEE 1588 Version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP-Primary multicast address: 224.0.1.129 PTP-Peer delay multicast address: 224.0.0.107

Table 30-62: IPv4-UDP PTP Frame Fields Required for Control and Status (Continued)

Field Matched	Octet Position	Matched Value	Description
UDP Destination Port	36, 37	0x013F 0x0140	0x013F - PTP Event Messages. These are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only). 0x0140 - PTP general messages
PTP Control Field (IEEE version 1)	74	0x00/0x01/0x02/ 0x03/0x04	0x00 - SYNC 0x01 - Delay_Req 0x02 - Follow_Up 0x03 - Delay_Resp 0x04 - Management
PTP Message Type Field (IEEE version 2)	42 (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xA/0xB/ 0xC/0xD	0x0 - SYNC 0x1 - Delay_Req 0x2 - Pdelay_Req 0x3 - Pdelay_Resp 0x8 - Follow_Up 0x9 - Delay_Resp 0xA - Pdelay_Resp_Follow_Up 0xB - Announce 0xC - Signaling 0xD - Management
PTP Version	43 (nibble)	0x1 or 0x2	0x1 - Supports PTP version 1 0x2 - Supports PTP version 2

PTP Frame Over IPv6

The ***IPv6-UDP PTP Frame Fields Required for Control And Status*** table provides information about the fields that are matched to control the snapshots for the PTP packets. The packets are sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. The positions are based on IEEE 1588–2008 standards and the message format defined in PTP Message Format (IEEE 1588–2008).

Table 30-63: IPv6-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	0x86DD	IP datagram
IP Version	14 (bits [7:4])	0x06	IP version is IPv6
Layer 4 Protocol	20 (IPv6 extension header not defined for PTP packets)	0x11	UDP

Table 30-63: IPv6-UDP PTP Frame Fields Required for Control and Status (Continued)

Field Matched	Octet Position	Matched Value	Description
PTP Multicast Address	38–53	FF0x:0:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:0:6B (Hex)	PTP - primary multicast address: FF0x: 0:0:0:0:0:0:0:181 (Hex) PTP - Peer delay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP Destination Port	56, 57 (IPv6 extension header not defined for PTP packets)	0x013F, 0x0140	0x013F - PTP event messages 0x0140 - PTP general messages
PTP Control Field (IEEE 1588 version 1)	93 (IPv6 extension header not defined for PTP packets)	0x00/0x01/0x02/ 0x03/0x04	0x00 - SYNC 0x01 - Delay_Req 0x02 - Follow_Up 0x03 - Delay_Resp 0x04 - Management (version1)
PTP Message Type Field (IEEE version 2)	74 (nibble) (IPv6 extension header not defined for PTP packets)	0x0/0x1/0x2/0x3/0x8/0x9/0xA/0xB/ 0xC/0xD	0x0 - SYNC 0x1 - Delay_Req 0x2 - Pdelay_Req 0x3 - Pdelay_Resp 0x8 - Follow_Up 0x9 - Delay_Resp 0xA - Pdelay_Resp_Follow_Up 0xB - Announce 0xC - Signaling 0xD - Management
PTP Version	75 (nibble)	0x1 or 0x2	0x1 - Supports PTP version 1 0x2 - Supports PTP version 2

PTP Frame Over Ethernet

Refer to the ***Ethernet PTP Frame Fields Required for Control and Status*** table. The table provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. The positions are based on IEEE 1588–2008 standards and the message format defined in the table.

Table 30-64: Ethernet PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched value	Description
MAC Destination Multicast Address (The address match of destination address (DA) programmed in MAC address 0 is used when the EMAC_TM_CTL.TSENMACADDR bit is set)	0–5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses: 01-1B-19-00-00-00 01-80-C2-00-00-0E
MAC Frame Type	12, 13	0x88F7	PTP Ethernet frame
PTP control field (IEEE Version 1)	45	0x00/0x01/0x02/ 0x03/0x04	0x00 - SYNC 0x01 - Delay_Req 0x02 - Follow_Up 0x03 - Delay_Resp 0x04 - Management
PTP Message Type Field (IEEE version 2)	14 (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xA/0xB/ 0xC/0xD	0x0 - SYNC 0x1 - Delay_Req 0x2 - Pdelay_Req 0x3 - Pdelay_Resp 0x8 - Follow_Up 0x9 - Delay_Resp 0xA - Pdelay_Resp_Follow_Up 0xB - Announce 0xC - Signaling 0xD - Management
PTP Version	15(nibble)	0x1 or 0x2	0x1 - Supports PTP version 1 0x2 - Supports PTP version 2

Auxiliary Time Stamp Snapshot

The auxiliary snapshot feature stores snapshots of the system time whenever a rising edge is detected on the EMAC_PTPAUX_MCG_IN[n] pins.

The PTP stores 64 bits of the captured time stamp in a 16-deep FIFO. Only 64 bits of the time stamp are stored in the FIFO. The upper 16 bits of seconds can be read from the EMAC_TM_HISEC register when it is present. When a snapshot is stored, the PTP indicates this event to the EMAC with the auxiliary snapshot interrupt. The EMAC_TM_STAT.AUXTSTRIG bit is set (=1). The value of the snapshot is read through the EMAC_TM_AUXTS_SEC and EMAC_TM_AUXTS_NSEC registers. If the FIFO becomes full and an external trigger to take the snapshot is asserted, then the snapshot trigger missed (EMAC_TM_STAT.ATSSTM) bit is set (=1). The latest snapshot is not written to the FIFO when it is full.

When a host reads the 64-bit time stamp from the FIFO through the EMAC_TM_AUXTS_SEC and EMAC_TM_AUXTS_NSEC registers, the space becomes available to store the next snapshot.

NOTE: A space in the FIFO is created whenever the EMAC_TM_AUXTS_SEC register is read. Therefore, read the EMAC_TM_AUXTS_NSEC register before reading the EMAC_TM_AUXTS_SEC register.

The program can clear the FIFO by setting the EMAC_TM_AUX_CTL.ATSFC bit. When multiple snapshots are present in the FIFO, the EMAC_TM_STAT.ATSNS bits indicate the count.

NOTE: The minimum gap between two events on the EMAC_PTPAUX_MCG_IN[n] pin must be 4 cycles of PTP_CLK + 3 cycles of CLKO7). Otherwise, the logic misses the rising edge of the trigger.

System Time

To get a snapshot of the time, the EMAC requires a reference time in 64-bit format as defined in the IEEE 1588 specification. The PTP module maintains 80-bit time, known as system time. The PTP clock updates system time.

The 80-bit timing reference is split into the following registers:

- EMAC_TM_NSEC–32-bit nanoseconds register which provides time in nanoseconds
- EMAC_TM_SEC–32-bit seconds register which provides time in seconds
- EMAC_TM_HISEC–16-bit high seconds register which provides time beyond the seconds register. The IEEE 1588 standard does not include this register. Its use is application specific.

The 64-bit system time (seconds and nanoseconds) is the source for taking time stamps for Ethernet frames being transmitted or received at the RMII.

Since the PTP clock frequency does not correspond to a 1 ns period, the EMAC_TM_NSEC register is incremented with a value equal to the PTP clock period for every PTP clock cycle. The function uses the EMAC_TM_SUBSEC register. The EMAC_TM_NSEC value is incremented with the value programmed in EMAC_TM_SUBSEC register every PTP clock cycle.

Whenever the EMAC_TM_SEC register overflows from 0xFFFFFFFF to 0x00000000, the seconds overflow interrupt is triggered. The EMAC uses the EMAC_TM_STAT.TSSOVF bit to indicate the event. After a seconds overflow, the EMAC_TM_HISEC register increments by one.

The system time module supports the following types of rollover modes for the EMAC_TM_NSEC register.

- Digital rollover mode. The maximum value in the nanoseconds field is 0x3B9AC9FF, that is, 10^9 nanoseconds. After it reaches this value, the EMAC_TM_SEC register increments and the EMAC_TM_NSEC register restarts counting from zero. Accuracy in digital rollover mode it is 1 ns per bit.
- Binary rollover mode. The nanoseconds field rolls over and increments the seconds field after the value reaches 0x7FFFFFFF. Accuracy in binary rollover mode is ~0.466 ns per bit.

System Time Adjustment

The following sections describe the process for system time adjustment.

System Time Initialization

System time can be initialized with 64-bit time when the PTP module is enabled. The initial value is written to the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT system time update registers. The system time counter is written with the value in the registers when EMAC_TM_CTL.TSINIT =1.

Coarse Correction Method

If the complete system time has an offset based on the system time of the requester, the coarse correction method can correct it. The time offset value is written to the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT registers. The offset value is then added to or subtracted from the system time when the EMAC_TM_CTL.TSUPDT bit is set (=1). Use the EMAC_TM_NSECUPDT.ADDSUB bit to choose addition or subtraction. System time correction occurs in one clock cycle using the coarse correction method.

NOTE: During subtraction, the EMAC_TM_SECUPDT register value must be less than the value of the EMAC_TM_SEC register. Check the value prior to subtracting using coarse correction.

Fine Correction Method

If a target PTP clock frequency has a drift based on the requester PTP clock (as defined in IEEE 1588), it can be corrected using the fine correction method. Using this method, system time is corrected over a period (unlike coarse correction where it happens in one clock cycle). This correction helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP sync message intervals.

Using this method, an accumulator sums the contents of the EMAC_TM_ADDEND register. The ***System Time Update, Fine Correction Method*** figure shows the method. The arithmetic-carry that the accumulator generates acts as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency divider.

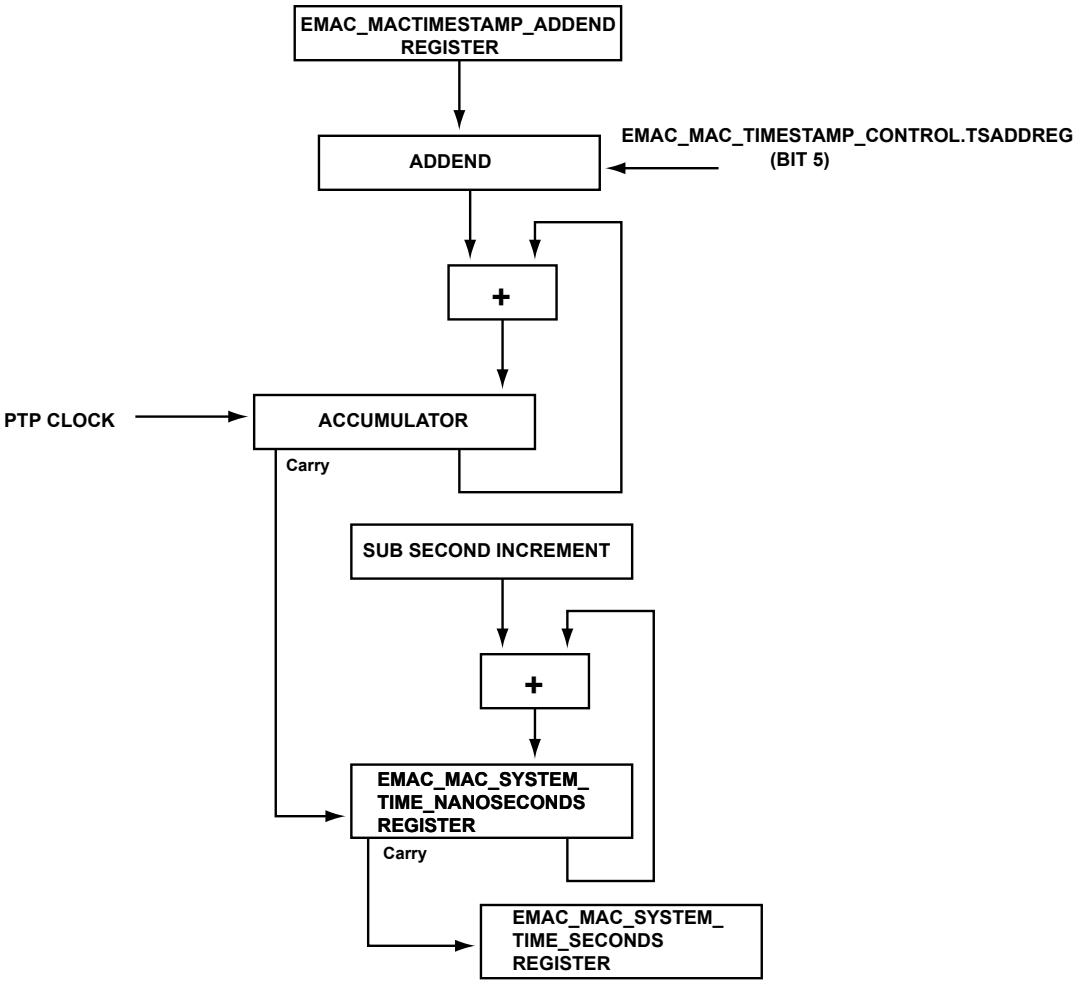


Figure 30-32: System Time Update, Fine Correction Method

Calculating Addend Value

The example in this section uses the fine correction process for system time.

In this example, the requester clock runs at 50 MHz and the completer clock has drifted to 66 MHz. The goal is to adjust the completer system time to 50 MHz, so that the completer PTP module synchronizes with the requester. Using the figure in Fine Correction Method, the nanoseconds increment signal runs at 50 MHz. The nanoseconds increment is the carry from accumulator register. The addend value increments the carry from accumulator at the rate of the completer clock (66 MHz).

The accumulator overflows and generates a carry every N addend value, so N × Addend = 2^{32}.

The accumulator increments at 66 MHz. This addition brings the carry to 50 MHz N = 66/50 = 1.32.

Hence, the addend = 2^{32}/1.32 = 0xC1F07C1F.

Therefore, if the addend is programmed with 0xC1F07C1F, the completer system time runs at 50 MHz, which synchronizes with the requester.

In the Fine Correction Method figure, the sub second increment is the value programmed in the EMAC_TM_SUBSEC register. It increments the EMAC_TM_NSEC register according to the frequency of the nanoseconds increment signal.

In the example, the sub second increment is 20 (for digital rollover) or 43 (for binary rollover). This addition increments the EMAC_TM_NSEC register by 20 ns (1/50 MHz).

The software must calculate the drift in frequency and update the EMAC_TM_ADDEND register accordingly.

NOTE: The PTP reference clock is the clock at which the system time is updated. When the EMAC_TM_CTL.TSCFUPDT bit is cleared (= 0), this clock equals the PTP clock. Using fine correction, the PTP reference clock is generated on the nanoseconds increment signal at which the system time updates.

Target Time Trigger (Alarm)

The PTP module provides an alarm function by triggering an alarm at a preset time. It sets the EMAC_TM_STAT.TSTARGT0 bit when the system time matches the value of the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers. This trigger can generate an interrupt and command the flexible PPS module to start or stop PPS output, depending on value programmed in the EMAC_PPS_CTL.TRGTMODSEL0 field.

The trigger is enabled by programming the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_SEC registers. Once an alarm has occurred, if the PTP needs another alarm, the software must:

- Clear the status bit,
- Reprogram the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers to a future value, and
- If the time programmed in the target time registers has elapsed, indicate a target time programming error by setting the EMAC_TM_STAT.TSTRGTERR0 bit.

The alarm time is represented in absolute units, not relative units. For example, if the software must generate an alarm after 10 seconds, it must read the current system time value. Then, the software must add the number corresponding to 10 seconds, and write the result back to the target time registers.

Pulse-Per-Second (PPS)

Pulse-per-second (PPS) is a physical representation of system time. It consists of a single pulse or train of pulses. The PTP uses PPS for extra synchronization or to monitor the synchronization performance between clocks. With proper configuration, the PTP module can generate PPS signals that are output on the EMAC_PTPPPS[n] pins. The PTP supports two kinds of PPS output: fixed and flexible.

Fixed Pulse-Per-Second Output

The EMAC module supports a fixed pulse-per-second (PPS) output that indicates 1 second intervals (default). Change the frequency of the PPS output by configuring the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits. The default value for these bits is 0000, which configures a 1 Hz signal with a pulse width equal to the period of the PTP clock.

The ***PPS Output Frequencies*** table shows various PPS output frequencies.

Table 30-65: PPS Output Frequencies

PPSCTL Bit Setting	Binary Rollover	Digital Rollover
0001	2 Hz	1 Hz
0010	4 Hz	2 Hz
0011	8 Hz	4 Hz
...	...	...
1111	32.768 kHz	16.384 kHz

In binary rollover mode, the PPS output (ETH)_PTPPS0) has a duty cycle of 50% with these frequencies.

In digital rollover mode, the PPS output frequency is an average number. The actual clock is a different frequency that is synchronized every second. PPS output pulses have different periods and duty cycles. This behavior is because of the non-linear toggling of the bits in digital rollover mode. For example:

- When `EMAC_PPS_CTL.PPSCTRL_PPSCMD` = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.
- When `EMAC_PPS_CTL.PPSCTRL_PPSCMD` = 0010, the PPS (2 Hz) is a sequence of:
  - One clock of 50 % duty cycle and 537 ms period
  - Second clock of 463 ms period (268 ms low and 195 ms high)
- When `EMAC_PPS_CTL.PPSCTRL_PPSCMD` = 0011, the PPS (4 Hz) is a sequence of:
  - Three clocks of 50 % duty cycle and 268 ms period
  - Fourth clock of 195 ms period (134 ms low and 61 ms high)

Flexible Pulse-Per-Second Output

The EMAC module also provides the flexibility to program the start or stop time, width, and interval of the pulse generated on the PPS output. Enable this feature, called flexible PPS, by setting the `EMAC_PPS_CTL.PPSEN0` bit.

The flexible PPS output options are:

- Program the start point of the single pulse and start and stop points of the pulse train in terms of system time. The target time registers program the start and stop time.
- Program the stop time in advance. Programs can configure the stop time before the actual start time has elapsed.
- Program the width, between the rising edge and corresponding falling edge of the PPS signal output, in terms of number of units of subsecond increment. This value is configured in the `EMAC_TM_SUBSEC` register.

- Program the interval, between the rising edges of PPS signal, in terms of number of units of subsecond increment. This value is configured in the EMAC_TM_SUBSEC register.
- Cancel the programmed PPS start or stop request.
- Indicate an error if the start or stop time programmed has already elapsed.

PPS Start or Stop Time

Start time can initially be programmed in the target time registers. If necessary, the start or stop time can be programmed again, but, only after the earlier programmed value is synchronized to the PTP clock domain. The EMAC_PPS[n]_TGTM_NSEC.TRGTBUSY bit indicates the status of synchronization. Programs can configure the start or stop time in advance, even before the earlier stop or start time has elapsed.

Program the start or stop time with advanced system time to ensure proper PPS signal output. If the application programs a start or stop time that has already elapsed, the EMAC module sets the EMAC_TM_STAT.TSTRGTERR0 bit indicating the error. If enabled, the EMAC module also sets the target time trigger (alarm) interrupt event. The application can cancel the start or stop request only when the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

PPS Width and Interval

The PPS width and interval are programmed in terms of granularity of system time, that is, the number of the units of subsecond increment value. For example, with the PTP reference clock of 50 MHz, a PPS pulse width of 40 ns, and an interval of 100 ns, program the width and interval to 2 and 5, respectively.

Use a faster PTP reference clock to achieve smaller granularity. Before commanding to trigger a pulse or pulse train on the PPS output, programs must configure or update the interval and width of the PPS signal output.

PPS Command

When the PPS module has a flexible PPS output configuration, the PTP can use the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits to command the PPS module to use any of the flexible PPS features. Programming these bits with a non-zero value instructs the PPS module to initiate an event. Once the command transfers or synchronizes to the PTP clock domain, these bits clear automatically. Software must ensure that these bits are programmed only when they are all-zero.

The ***Flexible PPS Output Commands*** table describes the different commands and their functions.

Table 30-66: Flexible PPS Output Commands

PPSCTL (Bits 3–0)	Command	Description
0000	No Command	
0001	Start Single Pulse	Generates a single pulse rising at the start point defined in the target time registers and of the duration defined in the EMAC_PPS[n]_WID register.

Table 30-66: Flexible PPS Output Commands (Continued)

PPSCTL (Bits 3–0)	Command	Description
0010	Start Pulse Train	Generates a train of pulses rising at the start time configured in the target time registers and of the duration configured in the EMAC_PPS[n]_WID register. The train of pulses repeats at the interval configured in the EMAC_PPS[n]_INTVL register. By default, the PPS pulse train is runs freely unless shalted by a Stop Pulse Train at Time command or a Stop Pulse Train Immediately command.
0011	Cancel Start	Cancels the Start Single Pulse and Start Pulse Train commands when the system time has not crossed the programmed start time.
0100	Stop Pulse Train at Time	Stops the train of pulses initiated by the command for start pulse train after the time programmed in the target time registers elapses.
0101	Stop Pulse Train Immediately	Immediately stops the train of pulses initiated by the command for start pulse train.
0110	Cancel Stop Pulse Train	Cancels the stop pulse train at time command when the programmed stop time has not elapsed. The PPS pulse train switches to free-running on the successful execution of this command.
0111-1111	Reserved	

PTP Interrupts

Set the EMAC_MAC_IEN.TSIE bit to enable interrupts from the PTP module. The EMAC module uses the EMAC_MAC_ISTAT.TSIS bit to indicate the status of the interrupt. The PTP supports the following three types of interrupts.

Auxiliary Snapshot Trigger

When an external event occurs on the EMAC_PTPAUX_MCG_IN[n] pins and a time stamp snapshot occurs, an auxiliary snapshot interrupt is triggered. The EMAC module uses the EMAC_TM_STAT.AUXTSTRIG bit to indicate the interrupt.

Target Time Reached

This interrupt is triggered when the system time becomes equal to the value written in the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers. This interrupt can be used as an alarm and is indicated on the EMAC_TM_STAT.TSTARGT0 bit.

System Time Seconds Register Overflow

The system time seconds register overflow interrupt is triggered when the EMAC_TM_SEC register overflows from 0xFFFF FFFF to 0x0000 0000. This interrupt is indicated on the EMAC_TM_STAT.TSSOVF bit. As soon as EMAC_TM_SEC register overflows, the EMAC_TM_HISEC register increments by one.

Media Clock Generation and Recovery (MCGR)

The EMAC module supports dedicated hardware for media clock recovery. This hardware generates a reference clock, which is multiplied to get the desired media clock. The hardware clock multiplier is external to the EMAC.

The media clock generation and recovery process is mainly hardware-based and involves insignificant software.

NOTE: The PPS and MCGR modes are mutually exclusive in that all the instances selected in the configuration must operate either in PPS mode or in MCGR mode. Mixed operation mode, where a few instances operate in PPS mode and others in MCGR mode, is not supported. Therefore, program all the instances to the same mode, irrespective of whether they are used or not.

The media clock generation and recovery support involve a dedicated external hardware and a thin layer of software. The dedicated hardware generates the reference clock (for example, 6 kHz or 8 kHz for AVB class A stream) which is then multiplied to get the desired media clock (for example an audio MCLK of 12.288 MHz).

The media clock recovery process involves host software extracting the presentation time from the 1722 frames and programming them into a CSR register, which is read by the EMAC to generate a reference clock. When the timer expires, the previous value is either toggled or a pulse is sent. The output at the recovered clock pin is generally a wave of 750 Hz ranging up to 8 kHz depending on the application that the clock is synced with, at the other end of the network.

As shown in the *Media Clock Generation and Recovery Block Diagram*, the time stamp array and the DMA module are external to the EMAC block.

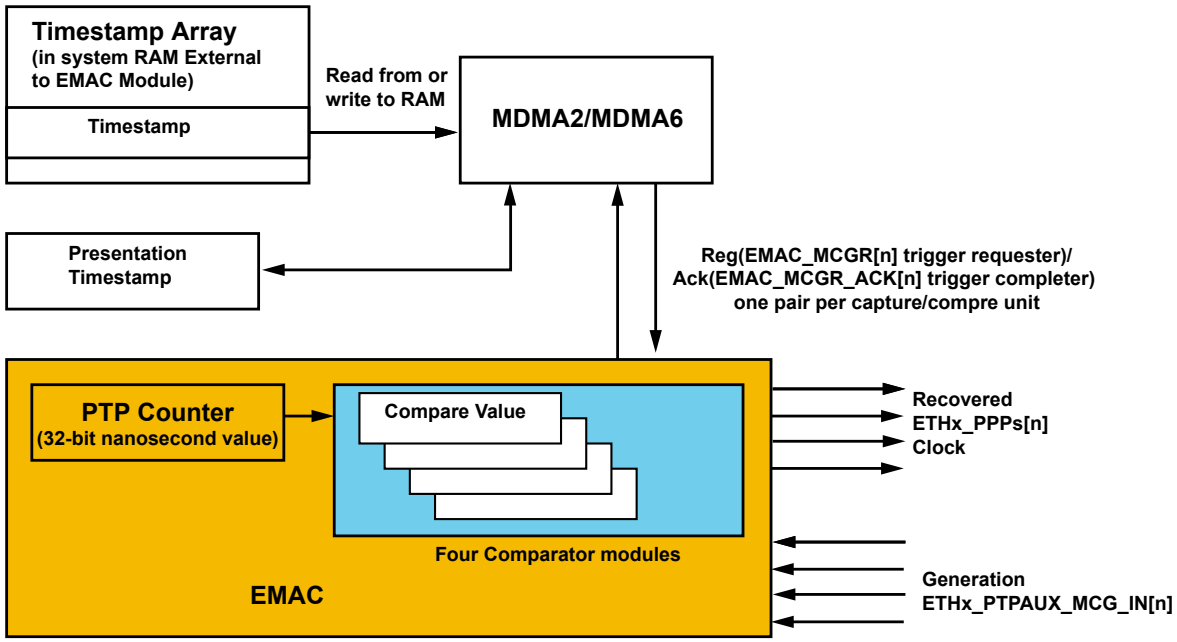


Figure 30-33: Media Clock Generation and Recovery Block Diagram

For the operation of the media clock generation and recovery, the EMAC supports:

- A 32-bit presentation time nanosecond counter that wraps at the full 32-bit value (and not after one second) to match up with the 1722 presentation time format.
- A mechanism to handshake with an external application (such as DMA) to program the presentation time into the CSR register. The hardware compares the programmed time stamp value with the presentation time and generates a wave based on the programmable settings.

Presentation Time Counter

The presentation time counter follows the 1722 presentation time format. In this format, time is represented as a 32-bit counter in nanoseconds that does a binary rollover on reaching the maximum value of 32'hFFFF_FFFF. This counter is another view of the PTP system time (1588 timer). For a given PTP, system time (PTP[63:32] represents time in seconds; PTP[31:0] represents time in nanoseconds. There exists a corresponding presentation time (32-bit value in nanosecond). The presentation time computed is referred to as the Current Presentation Time (CPT).

The CPT can be viewed as derived from 64-bit PTP system time as:

PTPNS [63:0] = (PTP[63:32] × 32'd1,000,000,000) + PTP[31:0]

where,

PTPNS is the PTP system time converted into a 64-bit nanosecond value

CPT [31:0] = PTPNS [31:0]

Media clock recovery is possible when the system time is internally generated in the EMAC module. CPT is computed in the same way as the internal PTP system time generation. Because CPT and system time values are in nanoseconds, and must be synchronous, the increment cycle of both the timers are same.

The timer is updated at the same instance; however, the updated value could be different. The software must compute a separate 32-bit update value in nanoseconds for the CPT update. A CPT sampled at different edges of a triggering input results in the generation of media clock time stamps which are inserted in 1722-based AVBTP packets to recover the clock at the destination.

NOTE:
- The value of CPT can be obtained by reading the `EMAC_PRSNTM_NSEC` register.
- The EMAC module supports both fine grain and coarse grain correction modes.
- Only the digital rollover mode must be used for the PTP time. The CPT also uses the same increment value and CPT requires the time to be in nanoseconds.

Comparator Modules

As shown in Figure 30-33 Media Clock Generation and Recovery Block Diagram, the EMAC module supports four comparator modules to handle multiple media clocks that might be required.

The comparator modules handshake with the application to:

- Program the time stamps into the `EMAC_PPS[n]_TGTM_SEC` register when `EMAC_TM_CTL.PTGE` =1.
- Set the presentation time control bits of the instance for recovery mode

The time stamps received from the application are referred as Target Presentation Time (TPT).

The MCGR and PPS modes are mutually exclusive. When the `EMAC_PPS_CTL.MCGREN3` through `EMAC_PPS_CTL.MCGREN0` bits are set (= 1), the corresponding PPS instance[n] operates in MCGR mode. Otherwise, by default, the PPS instance[n] operates in PPS mode.

The comparator module sends up to two requests to the application for TPT write, when EMAC_TM_CTL.PTGE = 1, and the presentation time control bits of the instance are configured for MCGR mode. Subsequent requests are generated each time a presentation time match occurs. (CPT transitions from a value less than TPT, to a value greater than or equal to CPT). The first request is asserted when a specific comparator instance is configured to MCGR mode with a non-zero presentation time control; an additional request is made when the first data is received into the comparator. This sequence allows the application to write the next TPT value while the EMAC module processes the previous TPT value for a match.

The TPT read from the EMAC_PPS[n]_TGTM_SEC register is considered as a future time. A toggle/pulse is generated in the next cycle when a match is detected. When a match is detected, the EMAC_PTPAUX_MCG_IN[n] trigger requester request for that comparator is asserted (to obtain next TPT) until the corresponding application acknowledgment is set.

The presentation control bits (for example, EMAC_PPS_CTL.PPSCMD1) determine the shape of the generated waveform. Also, these bits can be programmed to either toggle or generate a high/low pulse for one PTP clock cycle, upon match.

Clock Recovery

A match is when the free-running CPT value matches the received TPT value. An output signal, EMAC_PTPPPS[n], is asserted (toggle, low pulse, or high pulse) based on the programmed presentation control value.

Clock Generation

Based on the presentation control value programmed in the EMAC_PPS_CTL register, the comparator captures the presentation time and programs it into the EMAC_PPS[n]_TGTM_SEC register. A request is raised to the application to read the captured time stamp. Until the acknowledgment that the read operation is complete is received from the application, no new time stamps are captured.

Media Clock Generation and Recovery Flow

The *Media Clock Generation and Recovery Flow* figure shows the media clock generation and recovery flow.

For both media clock generation and recovery, set the EMAC_TM_CTL.PTGE field for CPT generation and set the EMAC_PPS_CTL.MCGREN3 through EMAC_PPS_CTL.MCGREN0 field for the corresponding instance to enable MCGR mode.

For media clock generation, the generated presentation time is sampled at positive, negative or at both the edges of trigger inputs (EMAC_PTPAUX_MCG_IN[n]) based on the mode programmed in the EMAC_PPS_CTL register. The sampled value is captured in the EMAC_PPS[n]_TGTM_SEC registers. After these registers are updated with the presentation time, the EMAC asserts the EMAC_PTPAUX_MCG_IN[n] trigger to request the application to read the presentation time. The application acknowledges that it has read the captured time stamp by asserting the corresponding EMAC_PTPAUX_MCG_IN[n] trigger completer.

For media clock recovery, the PPS instance asserts the EMAC_PTPAUX_MCG_IN[n] trigger to tell the application to program the target time seconds registers with the TPT value and acknowledge the corresponding request using the EMAC_PTPAUX_MCG_IN[n] trigger completer. When the application programs the target time seconds registers with the TPT value, the particular comparator compares the programmed TPT value with the CPT and

generates the waveform on EMAC_PTPPPS[n] based on the presentation time control value programmed in the EMAC_PPS_CTL register.

NOTE:
- The consecutive triggers to sample the presentation time should always be asserted after a few PTP clock cycles to allow for synchronization delays. (This is not an issue when the input trigger maximum frequency is 8 kHz and the PTP clock is running at least at 1 MHz).
- DMA acknowledgment can be both posted as well as non-posted, in the media clock recovery mode.

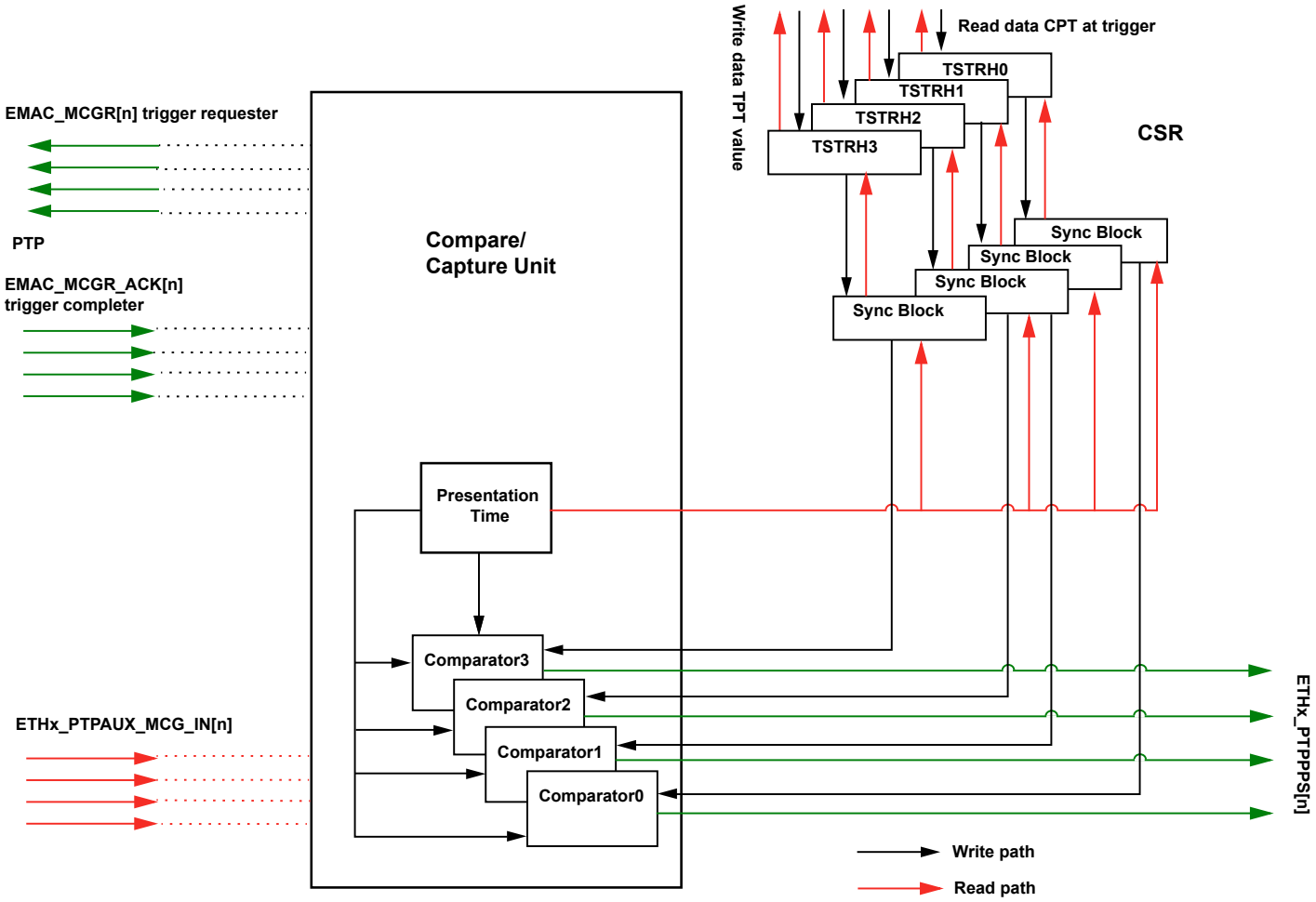


Figure 30-34: Media Clock Generation and Recovery Flow

PTP Time Stamp Offload Function

The PTP time stamp offload function enables the automatic generation of specific PTP packets when the MAC is working as a specific node in the PTP network. These packets may be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. This functionality helps in offloading certain functions of a PTP node with better accuracy and lower latency of responses.

Based on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application or based on reception of a particular PTP message.

The ***PTP Message Generation Criteria*** table indicates the PTP message generation criteria. Note that all other programming combinations are invalid for the PTP offload feature.

Table 30-67: PTP Message Generation Criteria

Programming (EMAC_TM_CTL)			Mode	Criteria for Generation of PTP Messages	PTP Message Type Generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
2'b00	0	1	Ordinary or boundary completer	SYNC message reception	Delay_Req
2'b00	1	1	Ordinary or boundary requester	Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
2'b01	0	1	Transparent completer	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				SYNC message reception	Delay_Req
2'b01	1	1	Transparent requester	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
2'b11	X	X	Peer-to-peer transparent	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp

NOTE: Clocks that support the peer delay mechanism must not generate delay request/delay response messages, according to the IEEE 1588-2008 specification. However, the EMAC supports this for flexibility, with a programmable control bit (EMAC_PTO_CTL.DRRDIS). Programs can use this bit to control the response generation for delay request/delay response message. For example, in transparent completer mode, delay request is generated in response to received sync only when the bit is reset.

For example, when the MAC is set as an ordinary or boundary completer clock in the PTP network, it can respond to the reception of SYNC messages with an automatic generation and transmission of the corresponding Delay_Req

message. Similarly, various other modes of operation are explained in the *PTP Message Generation Criteria* table. The MAC supports the multicast communication model for the generation of SYNC and Pdelay_Req PTP messages. For instance, the destination address field of the generated PTP over Ethernet packet is the defined special multicast addresses (0x011B19000000 for all except peer delay mechanism messages and 0x0180C200000E for peer delay mechanism messages).

When the MAC responds to received SYNC, Delay_Req, and Pdelay_Req PTP messages with a special multicast destination address, it also uses the corresponding special multicast address in the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

When the MAC responds to received SYNC, Delay_Req, and Pdelay_Req PTP messages with a unicast destination address, it takes the SA field of the received packets and makes them as the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

At the same time, all received PTP messages are forwarded to the application along with the receive status, indicating whether the response was generated by the MAC and whether it satisfies the packet filtering logic of the MAC receiver.

When the MAC automatically generates a PdelayReq or responds with a Delay_Req, the egress time stamp of these two PTP messages are provided in the transmit TS status (`EMAC_TM_STAT`) register; an interrupt is generated.

In addition to matching messageType and versionPTP fields for basic PTP over Ethernet message detection, the following fields are matched to qualify the received PTP message type:

1. The domainNumber field is checked for a match against the value programmed in the CSR.
2. The twoStepFlag in flagField field is checked for one step indication (1'b0).

The transportSpecific field is checked for the default PTP over Ethernet (4'h0) or 802.1AS mode (4'h1), when enabled.

Packet Generation

When enabled, the MAC can automatically generate PTP packets. The *Common PTP Message Header Fields* table describes the fields of specific PTP packets generated. The format and content of the packets are described in the following sections.

Table 30-68: Common PTP Message Header Fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5

Table 30-68: Common PTP Message Header Fields (Continued)

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

messageType

The following encoded values are used for PTP message types:

- SYNC – 4'h0
- Delay_Req – 4'h1
- Pdelay_Req – 4'h2
- Pdelay_Resp – 4'h3
- Delay_Resp – 4'h9

transportSpecific

The following transport protocol encoding is used:

- Default PTP over Ethernet – 4'h0
- 802.1AS mode – 4'h1

versionPTP

It is always configured to 2 because PTP version 2 is supported.

domainNumber

This contains the value from the `EMAC_PTO_CTL` register.

flagField

The following values are used:

- alternateMasterFlag (Octet 0 bit 0) – 1b0 for SYNC and Delay_Resp
- twoStepFlag (Octet 0 bit 1) – 1'b0 for SYNC and Pdelay_Resp
- unicastFlag (Octet 0 bit 2) – 1'b0 for Multicast Address, 1'b1 for Unicast Address

correctionField

For more information, see the Table 30-69 MAC Transmit PTP Modes table.

sourcePortIdentity

This field takes the value programmed in the EMAC_SRCPRT_IDNTY0 - EMAC_SRCPRT_IDNTY2 registers.

sequenceId

Pdelay_Resp and Delay_Resp use the same sequenceId field from received Pdelay_Req and Delay_Req PTP messages. For SYNC/Delay_Req, Pdelay_Req, a separate sequenceId counter is maintained. These sequenceId counters get incremented by 1 every time the corresponding message is generated and transmitted.

controlField

The following encoded values are used for controlField:

- SYNC – 8'h00
- Delay_Req – 8'h01
- Pdelay_Req – 8'h02
- Pdelay_Resp – 8'h05
- Delay_Resp – 8'h03

logMessageInterval

- SYNC – This contains logSyncInterval from the corresponding EMAC_LOG_MSG_INTVL register.
- Delay_Resp – This contains the sum of DRSYNCR and logSyncInterval value taken from the EMAC_LOG_MSG_INTVL register for a multicast PTP message and 8'h7F for unicast PTP message.
- Delay_Req, Pdelay_Req and Pdelay_Resp – 8'h7F where logSyncInterval = log2 (Mean Value of Interval in seconds)

The MAC supports values of -15 to 15 for logSyncInterval fields, which translates to a range from 32.768 microsecond (2-15) to 215 second. For a given value of log sync interval (N), the time interval between two SYNC packets is given by the following:

- 2(30+N) ns, when N is negative (-1 to -15)
- 2N seconds, when N is positive (0 to 15)

For example:

- When logSyncInterval is programmed to 1, the interval is 21; therefore, the SYNC message is sent once every 2 seconds.
- When logSyncInterval is programmed to -1, the interval is 2-1 = 0.536 seconds; therefore, the SYNC message is sent once every 536 milliseconds. The value is 0.536 seconds, because 2-30 = 1 ns.
- When logSyncInterval is programmed to -5, the interval is 2-5 = 33.55 ms; therefore, the SYNC message is sent once every 33.55 milliseconds.

NOTE: The MAC uses the PTP system time to generate the intervals for periodic packet transmission. For negative values of log message interval programmed, the generated period may deviate from the value given by the equation 2(30+N), because of the non-binary nature of the nanoseconds field of the system time.

PTP Message Specific Fields

The PTP includes the following fields for messaging:

messageLength

There is no suffix supported; therefore, this field contains the length of the PTP message that includes 34 byte PTP common header and the body specific to the message type.

For SYNC and Delay_Req packets, this field contains 44, whereas for Delay_Resp, Pdelay_Req and Pdelay_Resp, it contains 54.

originTimestamp

This field is the captured egress time stamp for SYNC, Delay_Req, and Pdelay_Req PTP messages receiveTimestamp. For Delay_Resp PTP message, this field is the ingress time stamp of the corresponding received Delay_Req PTP message.

requestingPortIdentity

For Delay_Resp and Pdelay_Resp PTP messages, this field is the sourcePortIdentity field taken from the corresponding received Delay_Req and Pdelay_Req PTP messages.

requestReceiptTimestamp

For the Pdelay_Resp PTP message, this field = 0.

One Step Time Stamp

The EMAC supports the one step time stamp feature. When the feature is enabled, the MAC identifies the offset in the packet and inserts the time stamp received from the application at that offset.

Depending upon the type of message and its mode, the MAC updates the following fields of transmit PTP packets:

- correctionField in the PTP header of messages.

- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

The *MAC Transmit PTP Modes* table shows how the PTP mode is selected for the one step time stamp operation based on the configuration of `EMAC_TM_CTL.SNAPTYPSEL`, `EMAC_TM_CTL.TSMSTRENA`, and `EMAC_TM_CTL.TSEVNTENA` bits. The bits are updated for the incoming PTP packets based on the message type in that mode, during the one step time stamping operation.

Table 30-69: MAC Transmit PTP Modes

Programming			Mode	Per Packet Control[*1]			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE[*2]	OSTC[*3]	TTS[*4]	
X	X	X	N/A	1	X	X	Time stamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)
2'b00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
							Delay_Req (correction field for residence time and Egress Asym Cor
2'b00	0	1	Ordinary or boundary completer	1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for Egress Asym Cor)
2'b00	1	1	Ordinary or boundary requester	0	1	X	Sync (originTimestamp field) sync (correction field for sub nanosecond cor)

Table 30-69: MAC Transmit PTP Modes (Continued)

Programming			Mode	Per Packet Control[*1]			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE[*2]	OSTC[*3]	TTS[*4]	
2'b01	X	0	End-to-end transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym Cor)
						Ingress TS	Pdelay_Req (correction field for residence time and Egress Asym Cor)
						Ingress TS	Pdelay_Resp (correction field for residence time and Ingress Asym Cor)

Table 30-69: MAC Transmit PTP Modes (Continued)

Programming			Mode	Per Packet Control[*1]			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE[*2]	OSTC[*3]	TTS[*4]	
2'b01	0	1	Ordinary or boundary slave with support for peer delay mechanism or peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym Cor) (applicable only for peer-to-peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for Egress Asym Cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)

Table 30-69: MAC Transmit PTP Modes (Continued)

Programming			Mode	Per Packet Control[*1]			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE[*2]	OSTC[*3]	TTS[*4]	
2'b01	1	1	Ordinary or boundary requestor with support for peer delay mechanism	0	1	X	Sync (originTimestamp field) sync (correction field for sub nanosecond cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)
2'b10	X	X	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress Asym cor)
						Ingress TS	Delay_Req (correction field for residence time and Egress Asym Cor)

Table 30-69: MAC Transmit PTP Modes (Continued)

Programming			Mode	Per Packet Control*1			Messages Processed on Tx
SNAPTYPSE	TSMSTRENA	TSEVNTENA		TTSE*2	OSTC*3	TTS*4	
2'b11	X	X	Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym Cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)

*1 The per packet control values provided here are the recommended settings used by devices in typical PTP operation, for the programmed mode.

*2 TTSE represents TTSE bit of transmit descriptor. The TTSE function is independent of the OST function and the programmed operation mode for OST. The MAC captures and returns the time stamp when the TTSE bit is set.

*3 OSTC represents OSTC bit of transmit descriptor.

*4 TTS represents the time stamp value provided in the TTSH, TTSL fields of transmit descriptor.

NOTE: Residence time/ turnaround time is calculated as the difference between the captured time stamp (egress time stamp) and the ingress time stamp. When the sub nanosecond feature is enabled, the residence time calculation includes sub nanosecond accuracy. Clocks supporting peer delay mechanism do not use delay request or response; however, it is included in OST for flexibility.

Enabling One Step Time Stamp

Programs can enable the one step time stamp feature for a packet by setting bit 20 (OSTC) in the transmit descriptor. For updating the correction field in certain PTP packets, the ingress time stamp must be given in the TSSL and TSSH fields. When the added IEEE 1588 higher word register option is selected, the time stamp is 80 bits. It also contains the content of the higher 16 bits of the `EMAC_TM_HISEC` register.

One Step Time Stamping for PTP Over UDP

The EMAC supports one step time stamp operation for PTP messages sent over UDP/IPv4 and UDP/IPv6. Because origin time stamp and/or correction field (based on the PTP message type and the programmed PTP mode) in the

PTP message are updated as part of the one-step time stamp operation, it also requires update to the checksum field of the PTP message sent over UDP/IP.

Checksum Update for One-Step Timestamping for PTP Over UDP

In the EMAC, checksum is updated for a packet, in the MTL before the packet is sent to the MAC where the PTP message is updated for one step time stamp operation. PTP requires the time stamp to be captured while the packet (SFD of the packet) is on the line and one step time stamp requires the origin time stamp and/or correction field to be updated in the PTP header while the packet is being transmitted on the line. Because the PTP message is present in the UDP payload, it is not possible to update the checksum field in the UDP header for modifications made to the PTP message, because it is already transmitted by then. Instead, the following options are supported that are in line with specification mentioned in Annex D and E of IEEE 1588-2008:

- When PTP message is sent over UDP/IPv4, the UDP checksum may be set to 0. The application must set the UDP checksum to 0 and set CIC to 0 or 1 while providing the packet to the MAC and the MAC does not update checksum for changes in origin time stamp or correction field.
- When PTP message is sent over UDP/IPv6, a transmitting node extends the UDP payload of all PTP messages by 2 octets beyond the end of the PTP message. The contents of the UDP checksum field or the final 2 octets of the UDP payload might be modified to ensure that the UDP checksum remains intact after modification of PTP fields. The application must add the two extra bytes at the end of the PTP message while providing the packet to the MAC and the MAC updates these two bytes to keep the checksum correct.

A programmable option (`EMAC_TM_CTL.CSC` field) supports UDP with a non-zero checksum for IPv4. When set, the two bytes at the end of the PTP message are updated to keep the UDP checksum correct as with PTP over UDP/IPv6. The application must add two bytes at the end of the PTP message sent to the EMAC and the EMAC updates these two bytes to keep the checksum correct.

Updating the pad bytes to keep the checksum correct requires the old value of correction field and/or origin time stamp in the packet. To get better latency and area, the application can set the origin time stamp field to zero, for PTP message in particular modes in which EMAC overwrites the origin time stamp. For example, EMAC operating in ordinary/boundary requester mode overwrites the origin time stamp field of the sync message, when OSTC is set. Because the sync message is originated by the port operating in ordinary/boundary requester mode, it is possible for the application generating the message to set the origin time stamp field to zero.

The transmit checksum engine is automatically enabled when this feature is selected. The EMAC uses the transmit checksum engine to identify the start of a PTP message in the UDP/IP packet. The checksum engine has the capability to detect errors in the header (L3) and payload (L4). These errors are not computed for some type of packets, which is indicated by "No" in the *Transmit Checksum Offload Engine Functions for Different Packet Types* table in the *Synopsis DesignWare® Cores Ethernet Quality-of-Service Databook*.

For one step, operation the packet is free of those errors and have "Yes/Not applicable" for the corresponding packet type in the *Transmit Checksum Offload Engine Functions for Different Packet Types* table. If an error is reported by the checksum engine or there is a "No" in For PTP messages sent over UDP/IP, no update is performed on the PTP message. See “Transmit Checksum Offload Engine” of the *Synopsis DesignWare® Cores Ethernet Quality-of-Service Databook* for more details about the errors detected by transmit checksum engine.

NOTE:
- The packet is sent with specific IP address and UDP port numbers and does not have IPv4 options/ IPv6 extension headers. Applications must take care of these requirements before giving the packet to the EMAC. EMAC does not validate them. As PTP is identified by specific value in the UDP destination port and as EMAC does not investigate these fields, it assumes an IP packet with OSTC set as a PTP packet, and it is the responsibility of the application to set the OSTC bit only for IP packets which contains PTP messages.
- For PTP over UPD/IPv6 and PTP over UDP/IPv4 with CSC set, the EMAC does not verify the presence of the two pad bytes, instead it overwrites the two bytes after the PTP message.
- When the enable one step time stamp for PTP over UDP/IP feature is selected, the enable transmit TCP/IP checksum offload feature is automatically enabled, and this is enabled only for queue 0 in multiple transmit queue configurations. If the application requires OST support for PTP messages sent over UDP/IP for packets sent from queues other than 0, transmit checksum engine must be manually enabled for those transmit queues. If checksum engine is not enabled, no updates are done for PTP messages sent over UDP/IP even though OSTC is set.
- One step time stamp operation is not supported for tunneled packets

Multiple Channels and Queues Support

The EMAC supports up to eight queues and channels on transmit and receive paths. The EMAC Block Diagram and Interfaces figure shows the architecture of EMAC with multiple queues and channels.

Multiple Queues Support in the Transmit Path

The EMAC supports up to eight queues and channels on transmit and receive paths.

NOTE: EMAC0 supports eight transmit DMA channels and queues. EMAC1 supports one transmit DMA channel and queue.

The number of transmit DMA channels is equal to the number of transmit queues and is direct one-to-one mapping. This enables the interleaving of packet/data transfer of multiple transmit DMA on the host/system bus and each transmit queue is reserved for the corresponding transmit DMA. Therefore, the integrity of packet transfer by each transmit DMA is maintained in the transmit queue and enables efficient use of the host bus for data transfers among multiple transmit DMAs without depending on the packet length, transmit buffer availability and so on.

When two transmit DMAs can transfer data into the same transmit queue, then a transmit DMA starts a packet transfer, the other transmit DMA cannot transfer data to the same transmit queue unless the previous packet transfer is complete. This means that the second DMA remains idle until the first packet transferred is complete, which is effectively a one-to-one mapping of the transmit DMA to the transmit queue.

This section describes how multiple queues and channels are operated in the EMAC.

Fixed Priority Scheme

The fixed priority scheme does not necessarily cause “starving” of lower priority channels for the SCB bus because of the capabilities of the SCB protocol and characteristics of the transmit DMA, described as follows:

- Read command requests from a transmit DMA for a burst of data transfers is driven on a separate SCB channel and is executed in one clock cycle itself. The actual data transfer transfers start on a separate SCB channel only after the request is accepted and it takes multiple clock cycles to complete with relatively large read access latencies.
- SCB requester is free to drive out subsequent requests even before the previous data transfers are completed.
- Transmit DMA requests for another data transfer only after the data transfer for its previous request is completed and transferred to the MTL TxQ, and space is available in TxQ to take in the next request.
- The EMAC can be programmed to control the maximum number of outstanding requests, requested burst sizes by DMA, and the burst sizes of each data transfer on the SCB with the help of the `EMAC_DMA_SYSBMODE` register.

When all transmit DMAs place a request simultaneously and are serviced by the fixed priority algorithm, all the requests are placed on the SCB bus in a window of 32 clocks (two clocks per SCB request, two SCB burst requests per transmit DMA request). The data transfer of the first request takes 32 clock cycles + the read-access latencies (when there are tens of cycles). Therefor the transmit DMA arbiter is already IDLE and has completed servicing all the transmit DMA requests before this data transfer completes.

Effectively, all the transmit DMA channels get equal access to SCB bus and the corresponding TxQs are filled up at the same rate. The data is read out from the TxQs as per the traffic class scheduler in the MTL. At a steady state (assuming that SCB bandwidth available to fetch data from system memory is always more than the operating line rate), the bandwidth for each channel is controlled by the priorities and settings of the traffic scheduler in the MTL.

Multiple Queues in the Receive Path

The EMAC supports up to eight receive channels. The number of receive channels is independent of the receive queues.

NOTE: EMAC0 supports eight receive DMA channels and queues. EMAC1 supports 1 receive DMA channel and queue.

The EMAC supports independent selection of the number of receive DMA channels and the number of receive queues in the MTL receive buffer. Therefore, the packets from any receive queue can be serviced by any of the receive DMA. As all the packets of receive queues are stored in a shared RXFIFO buffer in the MTL, the receive scheduler selects the receive queue whose packet data is to be transferred to the destination receive DMA, which then forwards the packet data to the host memory. The sequence of events in time are as follows:

1. Each receive DMA indicates whether it is ready to transfer data, that is, when the receive DMA has fetched the descriptor and has a receiver buffer ready, to accept the packet data. Each receive DMA also indicates the number of words that it can transfer, based on the space available in the receive buffer and the value programmed in the `EMAC_DMA[n]_CTL.PBLX8` bit field.
2. The MTL receive scheduler matches the number of bytes/packets available for transfer at the top of all receive queues with the corresponding receive DMA ready status and the number of bytes requested by it.
3. If there are multiple successful matches, the scheduler selects the receive queue to be serviced, based on the arbitration schemes (strict priority or weighted strict priority) decided by the `EMAC_MTL_OPMODE.RAA` bit

field. In the weighted strict priority scheme, the weights of each receive queue is set in the EMAC_RQ[n]_CTL.RXQ_WEGT bit field.

4. The MTL receive controller fetches the data of the selected receive queue from the shared RXFIFO memory and starts the transfer of the requested PBL number of bytes to the DMA block or until the end of packet and the corresponding receive status is transferred (whichever is first).
5. After completing the current request, the scheduler repeats the process by starting a new arbitration cycle. However, it selects the same queues queue for servicing, if the end of packet is not transferred and the EMAC_RQ[n]_CTL.RXQ_FRM_ARBIT bits are 1, for the receive queue.

Receive Queue to DMA Mapping

The packets in the MTL receive queues can be routed to any one of the multiple DMA channels by programming the following registers:

- EMAC_RXQ_DMA_MAP0 register (for queues 0, 1, 2 and 3)
- EMAC_RXQ_DMA_MAP1 register (for queues 4, 5, 6 and 7)

The following types of receive queue to DMA mapping is possible EMAC_DMA[n]_IEN.RSE through programming:

- Static mapping
- Dynamic (per packet) mapping

Static Mapping

In static mapping mode, all packets of a receive queue are connected to a specific DMA channel. For example, all the packets from receive queue 0 can be routed to a DMA channel by programming the EMAC_RXQ_DMA_MAP0.Q0MDMACH and EMAC_RXQ_DMA_MAP0.Q0DDMACH bit fields.

Similarly, packets from other receive queues can be routed to any DMA channel by programming register fields corresponding to each queue.

Dynamic Mapping

In Dynamic (per packet) mapping mode, the destination DMA channel is chosen by the MAC core receiver for each packet.

In this mode, the destination DMA channel of a packet being read from a receive queue is not constant but decided independently for each packet. For example, if the EMAC_RXQ_DMA_MAP0.Q1MDMACH bit is set, the static mapping is disabled for receive queue 1 and the value in the EMAC_RXQ_DMA_MAP0.Q1MDMACH bit is ignored. The destination DMA channel is chosen by the MAC receiver for each packet, depending on the following in decreasing order of priority.

1. L3-L4 filter Based DMA Selection. When the enable layer 3 and layer 4 packet filter feature is present and enabled in the program configuration, the TCP/UDP and IP header fields of the received packet are matched against the corresponding values programmed and enabled for comparison in the EMAC_L3L4_CTL[i] register. If the match is successful, the DMA channel number programmed in the EMAC_L3L4_CTL[i].DMCHN bit field is selected as the destination DMA channel number provided the

EMAC_L3L4_CTL[i].DMCHEN bit of the same register is set. If none of the L3-L4 registers give a comparison match, the EMAC proceeds to the next step.

2. Extended VLAN Based DMA Selection. Extended routing is applicable only if the VLAN filter has passed. Routing is only based on perfect filter result. Each perfect filter has a programmable DMA channel enable and a DMA channel number field. Routing is applicable for a filter when the DMA channel enable bit is set.

   The frame is routed to the smallest matching filter's DMA channel provided it is enabled. If that filter's DMA channel number is not enabled, the frame gets routed to channel 0. For example, if a frame's VLAN tag matches filters 7, 3, and 1. Then the MAC checks if filter 1's DMA channel number is enabled through programming. If yes, the frame is routed to the programmed value; otherwise it is routed to channel 0. When the inverse filter is enabled, the frame is routed to the least mismatched filter's DMA channel number provided it is enabled. If the DMA channel enable bit is not set, then the frame is routed based on DA based addressing or to channel 0.

   If the hash filter is also enabled, it is used to determine the filter result only. Routing still depends on the enabled perfect filters. If none of the perfect filters are enabled or if all of them are bypassed, the VLAN filter based routing is not performed. The frame is routed through DA based addressing or to channel 0. If all the perfect filters give a fail result and the hash filter has passed, the VLAN filter result is a pass. However, routing is based on DA based addressing or to channel 0. The behavior is similar when inverse filtering is enabled.

3. Ethernet DA-Based DMA Selection. The DA address of the received packet is compared against the programmed DA values in MAC address registers. If the address matches any of the programmed values, the corresponding EMAC_ADDR[i]_HI.DCS bit field (when enabled) determines the destination DMA channel number. If none of the previous operations can make a successful match/decision, then the packet is routed to DMA channel 0 by default.

Broadcast/Multicast Packet Duplication

The EMAC supports the Broadcast/Multicast Packet Duplication feature to send the received broadcast/multicast packets to multiple DMA channels.

In a virtual system that supports multiple guest OS's (for instance, each OS owning a DMA channel), where each DMA could act as an independent Ethernet node, it may be required to route the received multicast/broadcast packets to all/multiple DMA channels. This is implemented by using the MTL RX FIFO read controller to read the packet from the same queue multiple times and route the same packet to the specified multiple DMA channels.

To maintain backward compatibility, through software control, the DMA channel selection (EMAC_ADDR[i]_HI.DCS) field are redefined to denote multiple DMA channels.

The EMAC_ADDR[i]_HI.DCS bit field determines the DMA channel number to which the received packet (that matches the MAC address present in that register) must be routed.

The EMAC_ADDR[i]_HI.DCS is a programmable field that supports more than one DMA channel number to be selected.

The EMAC_ADDR[i]_HI.DCS field can hold any one of the following values:

- Binary value of the channel number when the EMAC_MAC_EXT_CFG.PDC bit is set to 0

- One-hot value of the channel number when the EMAC_MAC_EXT_CFG.PDC bit is set to 1

Use DA based routing to enable the multicast/broadcast packets to route to multiple DMA channels. Packet duplication is not supported in other DMA channel routing mechanisms such as extended VLAN, or L3-L4 based routing. In other words if the destination DMA channel is decided by the VLAN or L3-L4 filter result, it is routed only to the specific channel even if the DA of the packet matches the multicast address in the EMAC_ADDR[i]_HI/EMAC_ADDR[i]_LO registers.

The packet duplication feature is supported only on the highest MTL queue configured. Therefore, packet duplication for broadcast/multicast packets requires that the EMAC_RXQ_CTL1.MCBCQ bit field is programmed to the highest RxQ present in the configuration.

In packet duplication, the MTL read controller sequentially transfers the same packet to all the required receive DMA channels. This packet is removed from the MTL FIFO only after the packet transfer completes. Therefore, this feature is suitable for smaller packet sizes or very low bandwidth traffic. This ensures that the MTL RxQ does not overflow due to the long period when the packet remains in the queue during the packet duplication.

NOTE: When more than one bit is set for the DMA channel number, a packet that is not routed to the highest receive queue is routed to only one DMA channel as indicated by the lowest bit set in the EMAC_ADDR[i]_HI.DCS bit field.

Assigning Tag Priorities to Transmit and Receive Queues

The EMAC supports assigning tag priorities to transmit and receive queues through programming registers.

The VLAN tag priorities can be assigned to transmit queues by programming the EMAC_TXQ_PRTY_MAP0.PSTQ0 through EMAC_TXQ_PRTY_MAP1.PSTQ7 bit fields in the following registers:

- EMAC_TXQ_PRTY_MAP0 (for transmit queues 0 to 3)
- EMAC_TXQ_PRTY_MAP1 (for transmit queues 4 to 7)

For example, if the program needs to assign a VLAN tag priority of 3 to transmit queue 0, set bit [3] in the EMAC_TXQ_PRTY_MAP0.PSTQ0 bit field.

The same VLAN tag priority can be set for multiple transmit queues. The transmit packet association to transmit queue is governed by the application, so the MAC does not route the packet based on transmit queue priority mapping; it is only used for Pause Flow Control (PFC). The settings in the PSTQ[n] field determines the transmit queues blocked for transmission when the PFC packet is received with corresponding VLAN tag priorities enabled.

The VLAN tag priorities can be assigned to receive queues by programming the EMAC_RXQ_CTL2.PSRQ0 through EMAC_RXQ_CTL3.PSRQ7 bit fields. The bit corresponding to the VLAN tag priority can be set in the PSRQ field for assigning that priority to the receive queue. For example, to assign a VLAN tag priority of 3 to receive queue 0, set bit [3] in the EMAC_RXQ_CTL2.PSRQ0 bit field. The VLAN tag priority assigned to the receive queue must be unique, that is, more than one receive queue cannot be assigned the same VLAN tag priority. However, more than one VLAN tag priority can be assigned to same receive queue.

The settings in the PSRQ bit field is used for VLAN tagged receive packet routing to receive queues as well as for PFC based transmit flow control. The received VLAN tagged receive packet is routed to the receive queue that has the VLAN tag priority match. In PFC based transmit flow control, the PSRQ bit field corresponding to a particular receive queue is used for enabling VLAN tag priorities in the PFC packet transmitted when corresponding receive queue threshold levels are reached.

Distribution of Receive Packets from MAC to Receive Queues

The EMAC provides multiple receive queues to support classification and distribution of ingress packets. The MAC receiver pushes the ingress packets to the receive queues based on packet type (tagged, untagged, multicast, PTP, AVB, DCB, and so on), tag priorities, MAC packet filter results, preemptable or express packet, and so on. The distribution is controlled by the settings of the `EMAC_RXQ_CTL0` through `EMAC_RXQ_CTL4` and `EMAC_MACPKT_FILT` register fields.

EMAC additionally supports distribution of ingress packets based on the user-defined type field of the ingress packets when the `EMAC_RXQ_CTL1.TBRQE` bit is enabled. Up to 8 types for matching are programmed in the MAC_TMRQ_Regs(n) registers which are accessible through the `EMAC_INDR_ACC_CTL` and `EMAC_INDR_ACC_DAT` registers. Each MAC_TMRQ_Regs(n) contains

- TYP field: contains 16-bit type value of the packet that needs to be compared with the received Ethernet packet. This field is valid when programmed to a value greater than or equal to 0x0600.
- TMRQ field: contains the receive queue number to which ingress packet is pushed when its type field matches TYP.
- PFEX field: This bit indicates whether the TYP is applicable for express packets or preemptable packets.

Based on the packet filter settings, the MAC receiver decides to either drop or forward the ingress packet. However, the results of layer 3 and layer 4 filtering, mentioned in Layer 3 and Layer 4 Frame Filtering are not considered for the distribution of ingress packets into the receive queues. Only the results of DA/SA/VLAN filters mentioned in Source Address or Destination Address Filtering and VLAN Filtering are applicable.

Filtering of the broadcast packets is exclusively controlled by the `EMAC_MACPKT_FILT.DBF` bit. The filtering of multicast packets is controlled by the `EMAC_MACPKT_FILT.PM` bit, in addition to the DA/SA match.

By default, packets that fail any of the DA/SA filters, are dropped by the MAC receiver. However, packets that fail the DA/SA/VLAN filter can still be forwarded to the application when the `EMAC_MACPKT_FILT.RA` bit is 1. Even when the RA field is 0, VLAN filter fail packets can be forwarded if the `EMAC_MACPKT_FILT.VTFE` bit is 0.

The ingress packets go through the following operations in the decreasing order of priority.

RA = 0: Ingress packets are dropped when any one of the following conditions is true.

- DA/SA filter fail
- VLAN filter fail and `EMAC_MACPKT_FILT.VTFE` bit is 1

`EMAC_MACPKT_FILT.RA` = 1: Ingress packets that fail the DA/SA filter are also forwarded to the receive queue as mentioned in the *Routing DA/SA Failed Packets to Receive Queue* table.

Table 30-70: Routing DA/SA Failed Packets to Receive Queue

Packet Type	TSN Type	Destination Receive Queue Determined By
Broadcast/Multicast	Express	if MFFQE = 1, MFFQ; else if VFFQE = 1, VFFQ; (only for tagged packets) else RxQ0
Unicast	Express	if MFFQE = 1, MFFQ; else if VFFQE = 1, VFFQ; (only for tagged packets) else RxQ0
All	Preemptable	FPRQ field

`EMAC_MACPKT_FILT.RA` = 1 and `EMAC_MACPKT_FILT.VTFE` = 0: Ingress tagged packets that pass the DA/SA filter but fail the VLAN filter, are forwarded to the receive queue as mentioned in the *Routing DA/SA Pass But VLAN Filter Fail Packets to Receive Queue* table.

Table 30-71: Routing DA/SA Pass But VLAN Filter Fail Packets to Receive Queue

TSN Packet Type	Destination Receive Queue Determined By
Express	if VFFQE = 1, VFFQ; else RxQ0
Preemptable	RQ field

Ingress packets that pass the DA/SA/VLAN filters are forwarded to the receive queue. The *Priority Routing of Filter Pass Packets When OMCBCQ = 0* table shows the routing priority in the decreasing order when the `EMAC_RXQ_CTL1.OMCBCQ` bit field is 0.

Table 30-72: Priority Routing of Filter Pass Packets When OMCBCQ = 0

Packet Type	TSN Type	Tagged	Destination Receive Queue Determined By
Broadcast/Multicast	Express	Yes	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else PSRQ
		No	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else UPQ
Unicast AVTP Control	Express	No	AVCPQ
		Yes	if TACPQE = 1, AVCPQ; else PSRQ
Unicast PTP over Ethernet	Express	No	PTPQ
		Yes	PTPQ or PSRQ based on the value of TPQC
Unicast DCB/LLDP Control	Express	No	DCBPQ

Table 30-72: Priority Routing of Filter Pass Packets When OMCBCQ = 0 (Continued)

Packet Type	TSN Type	Tagged	Destination Receive Queue Determined By
Remaining Unicast	Express	No	If TBRQE = 1, TBRQ; else UPQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ
All Packets	Preemptable	No	If TBRQE = 1, TBRQ; else FPRQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ

The *Priority Routing of Filter Pass Packets When OMCBCQ = 1* table shows the routing priority in the decreasing order when the `EMAC_RXQ_CTL1.OMCBCQ` bit field is 1.

Table 30-73: Priority Routing of Filter Pass Packets When OMCBCQ = 1

Packet Type	TSN Type	Tagged	Destination Receive Queue Determined By
All AVTP Control	Express	No	AVCPQ
		Yes	if TACPQE = 1, AVCPQ; else PSRQ
All PTP over Ethernet	Express	No	PTPPQ
		Yes	PTPQ or PSRQ based on TPQC
Remaining Broadcast/ Multicast	Express	Yes	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else PSRQ
		No	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else UPQ
Unicast DCB/LLDP Control	Express	No	DCBPQ
Remaining Unicast	Express	No	If TBRQE = 1, TBRQ; else UPQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ
All Packets	Preemptable	No	If TBRQE = 1, TBRQ; else FPRQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ

NOTE: If the RxQ pointed to by the various fields in tables above is not enabled in the `EMAC_RXQ_CTL0` register as per the packet type, the packet is written to

- RxQ0 for express packets
- FPRQ field for preemptable packets
- If RxQ0/FPRQ is not enabled, the packet is dropped.

If the `EMAC_TM_CTL.AV8021ASMEN` bit is 1, the MAC receiver checks only for untagged PTP packets. Tagged packets with a PTP payload are considered as remaining (Unicast/Broadcast/Multicast) tagged packets in the above tables.

Traffic Shaper

Traffic shaping is the process of distributing frames/packets evenly in time to smooth out network traffic. This reduces traffic bursts which can overwhelm the network. The following sections describe how to implement and manage this feature.

Data Center Bridging

The data center bridging feature supports three types of algorithms: weighted round robin, deficit weighted round robin, and weighted fair queuing.

Weighted Round Robin

In the Weighted Round Robin algorithm used in the Data Center Bridging feature, all queues are serviced in round-robin order based on weight setting.

In Weighted Round Robin (WRR) algorithm, each queue is assigned a weight based on the percentage of configured bandwidth. All queues are serviced in the round-robin order according to the weight settings. This algorithm is less complex and requires less hardware as compared to other algorithms. Large packets get more bandwidth in this algorithm.

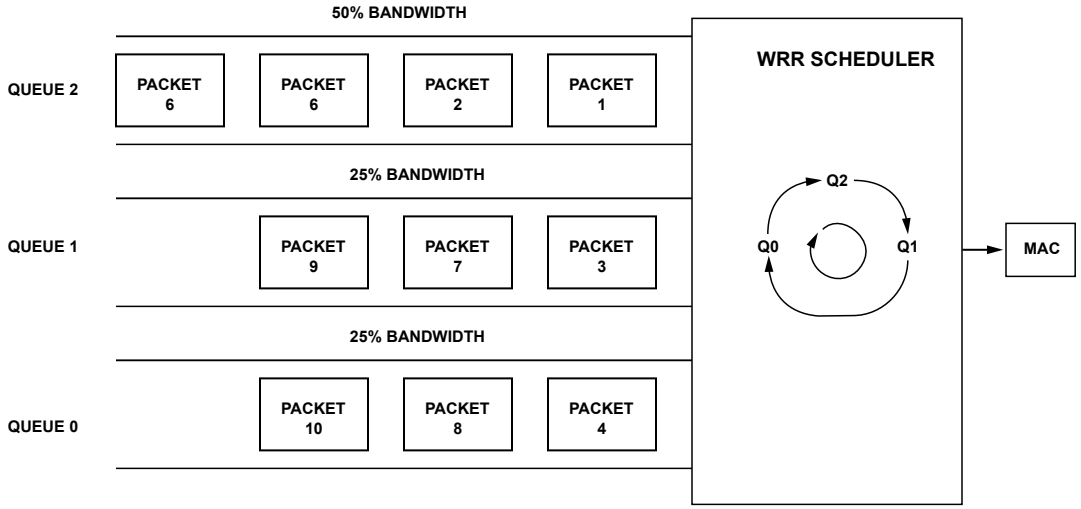


Figure 30-35: Weighted Round Robin Example

In the *Weighted Round Robin Example* figure, Queue 2, Queue 1, and Queue 0 are configured for 50%, 25%, and 25% bandwidth, respectively. The corresponding weights are two, one, and one. Before the WRR scheduler is started, Queue 2 has four packets; Queue 1 has three packets; and Queue 0 has three packets. The packets are transmitted in the following order:

Table 30-74: Weighted Round Robin Sequence Example

Packet Sequence	Queue Number
Packet 1	Queue 2
Packet 2	Queue 2
Packet 3	Queue 1
Packet 4	Queue 0

Table 30-74: Weighted Round Robin Sequence Example (Continued)

Packet Sequence	Queue Number
Packet 5	Queue 2
Packet 6	Queue 2
Packet 7	Queue 1
Packet 8	Queue 0
Packet 9	Queue 1
Packet 10	Queue 0

Deficit Weighted Round Robin

When the Data Center Bridging feature is supported, programs can use the deficit weighted round robin algorithm to assign a weight based on the percentage of configured bandwidth. A deficit counter holds the credit for transmission.

In Deficit Weighted Round Robin (DWRR) algorithm, each queue is assigned a weight based on the percentage of configured bandwidth. A deficit counter holds the credit for transmission. The quantum value, proportional to the bandwidth, is added to the deficit counter every time a queue is scanned. The packet in the queue is selected for transmission only if the credit in the deficit counter is less than or equal to the size of the packet available at the head of the queue. A selected queue is serviced until the outstanding credit is lesser than the packet size at the head of the queue during a particular queue scan cycle. After this, the next lower priority queue is serviced. When the packets in a queue are over, the credit becomes zero for that queue scan cycle. This algorithm is less complex, and it provides more fairness in scheduling as compared to other algorithms.

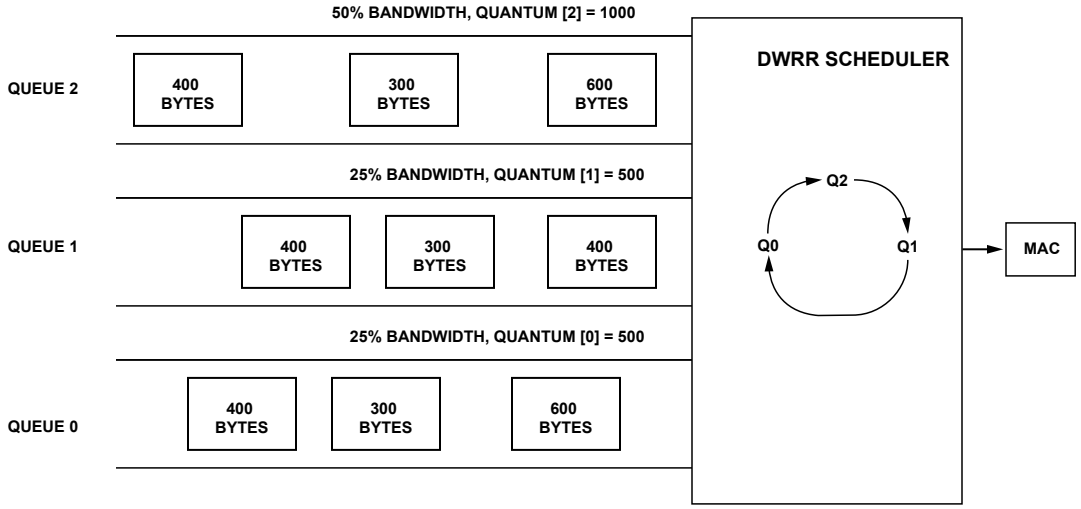


Figure 30-36: Deficit Weighted Round Robin Example

In the *Deficit Weighted Round Robin Example* figure, Queue 2, Queue 1, and Queue 0 are configured for 50%, 25%, and 25% bandwidth, respectively. The corresponding quantum values are 1000, 500, and 500. Before the DWRR scheduler is started, each queue has three packets. The *Deficit Weighted Round Robin* table shows the queues scan

cycle number and the packet size at the head of queues, deficit counter value before and after packet in selected queue is transmitted, and queue scheduled for transmission.

Table 30-75: Deficit Weighted Round Robin

Queues Scan Cycle Number	Packet Size at the Head of Queue			Deficit Counter Value Before the Packet is Transmitted in Selected Queue			Deficit Counter Value after the Packet is Transmitted in Selected Queue			Queue Scheduled for Transmission
Queue	Q2	Q1	Q0	Q2	Q1	Q0	Q2	Q1	Q0	
1	600	400	600	1000	500	500	400	500	500	Q2
	300	400	600	400	500	500	100	500	500	Q2
	400	400	600	100	500	500	100	100	500	Q1
	400	300	600	100	100	500	100	100	500	None
2	400	300	600	1100	600	1000	700	600	1000	Q2
	N/A	300	600	0	600	1000	0	300	1000	Q1
	N/A	400	600	0	300	1000	0	300	400	Q0
	N/A	400	300	0	300	400	0	300	100	Q0
	N/A	400	400	0	300	100	0	300	100	None
3	N/A	400	400	0	800	600	0	400	600	Q1
	N/A	N/A	400	0	N/A	600	0	0	200	Q0
	N/A	N/A	N/A	0	N/A	N/A	0	0	0	None

Weighted Fair Queuing

In the Weighted Fair Queuing (WFQ) algorithm used in the Data Center Bridging feature, each queue is assigned a weight based on the percentage of configured bandwidth. The algorithm computes the finish time of packets at the head of the queues based on the packet size and weights for the queue. The packets with earliest finish time is scheduled or transmission first. The algorithm is more complex and less scalable as compared to other algorithms.

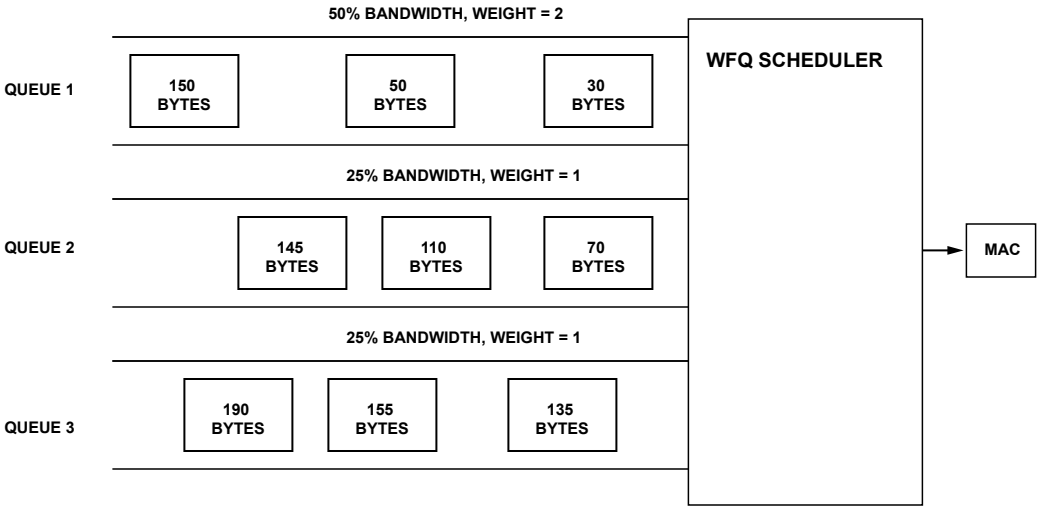


Figure 30-37: Weighted Fair Queuing Example

In the *Weighted Fair Queuing Example* figure, Queue 2, Queue 1, and Queue 0 are configured for 50%, 25%, and 25% bandwidth, respectively. The corresponding weights are 2, 1, and 1. Before the WFQ scheduler is started, each queue has three packets. The *Weighted Fair Queuing Algorithm Example Values* table shows the packet size at the head of the queue, computed finish time, and the queue scheduled for transmission.

Table 30-76: Weighted Fair Queuing Algorithm Example Values

Packet size at the Head of the Queue			Computed Finish Time in Clock Cycles (packet size or weights)			Queue Scheduled for Transmission
Q2	Q1	Q0	Q2	Q1	Q0	
30	70	135	15	70	135	Q2
50	70	135	25	70	135	Q2
150	70	135	75	70	135	Q1
150	110	135	75	110	135	Q2
N/A	110	135	N/A	110	135	Q1
N/A	145	135	N/A	145	135	Q0
N/A	145	155	N/A	145	155	Q1
N/A		155	N/A	N/A	155	Q0
N/A		190	N/A	N/A	190	Q0

Audio Video Bridging

The EMAC supports the AV data transfer in 100 Mbps and 1000 Mbps modes. The AV feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

- IEEE 802.1Qav-2009: Allows the bridges to provide time-sensitive and loss-sensitive real-time audio video data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources.
- IEEE 802.1Qat-2009: Allows the network resources to be reserved for specific traffic streams traversing a bridged local area network.
- IEEE 802.1AS-2011: Specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration.

While the AVB standards were originally designed for audio and video over standard Ethernet, extending these to control streams additionally benefits the industrial and automotive sectors. Time Sensitive Networking (TSN) is a set of standards developed by the Time Sensitive Networking Task Group with the intent to extend the future AVB enhancements to other sectors that could benefit.

Bridges are increasingly used to interconnect devices that support scheduled applications (for example, industrial automation, process control and vehicle control). The IEEE 802.1Qbv-2015 (Enhancements to Scheduling Traffic) provides performance assurances of latency and delivery variation to enable these applications in an engineered LAN while maintaining the existing guarantees for the credit-based shaper and best-effort traffic.

The IEEE 802.3br (Interspersing Traffic) and IEEE 802.1Qbu (Frame Preemption) enables the suspension of a large preemptable packet being transmitted by the MAC layer to allow one or more express packets to be transmitted before the transmission of the preemptable packet is resumed. This provides the capability to schedule express traffic packets with minimal delays/latencies or at predictable times with efficient utilization of the line bandwidth.

Transmit Path Functions

Transmit paths of queue 0 and the highest enabled queue are enabled by default.

The transmit path of queue 0 supports the strict priority algorithm, and it is used for best-effort traffic. For a queue, the strict priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for the MTL Tx FIFO is enabled, the strict priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of a size equal to the programmed threshold limit.

The transmit paths of additional queues support traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

- The queue contains one or more packets
- The credit for the queue is positive as per the algorithm

The credit-based shaper algorithm can be disabled for all queues or lower-priority queues. For example, the program can either disable the credit-based shaper algorithm for queue 1 and queue 2 or only for queue 1. The program

should not disable the credit-based shaper algorithm for queue 2 and enable it for queue 1. When the credit-based shaper algorithm for a queue is disabled, the channel uses the default strict priority algorithm.

Each transmit DMA has a separate descriptor chain for fetching the transmit data. The transmit channel that gets the access to the system bus depends on the DMA arbiter.

The transmit path has a shared FIFO (MTL layer) for each queue. The data fetched by the DMA is put in the respective part of the FIFO. The MTL transmit queue scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for a queue, the corresponding queue is selected for transmission if the following conditions are true:

- If the packet is available in the channel and has a positive or zero credit
- If the higher priority queue has no packet waiting in the FIFO

If the credit-based shaper algorithm is disabled for all queues, the packet transmitted from a queue is selected based on the priority scheme described in the following table.

Table 30-77: Weight for DMA Channels

`EMAC_DMA[n]_TXCTL.TCW` Field	**Transmit Channel Weight**
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

Receive Path Functions

The receive path of Queue 0 is enabled by default. All traffic is received on this channel. Programs can enable the receive paths of additional queues. By enabling the receive paths of multiple channels, the received data is demultiplexed and sends the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if it is an AV packet and its corresponding VLAN priority tag value. This status is updated in the extended status field of the receive descriptor as explained in Receive Descriptor.

All received packets with the EtherType field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

- AV data packets: The AV data packets are always tagged. The tagged AV data packets are received based on the programmed priority value. To specify the channel to which an AV packet with a given priority must be sent, program bits[15:8] in the `EMAC_Q[i]_TXFLOW_CTL` register of the corresponding queue.

- AV control packets: The AV control packets can be either tagged or untagged. The untagged AV control packets are received on Queue 0, by default. To receive these packets on any other queue, program`EMAC_RXQ_CTL1.AVCPQ` bit field. Like the AV data packets, the tagged AV control packets are received based on the programmed priority value.

In addition to the AV packets, the untagged PTP packets on any queue ca also be received. By default, the PTP packets (tagged or untagged) are received on queue 0. To receive these packets on any other queue, program the `EMAC_RXQ_CTL1.PTPQ` bits.

Credit-Based Shaper Algorithm

The MTL Queue Scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in queue 0. The additional queues can be programmed to use the credit-based shaper algorithm.

The following sections provide information for implementing the credit-based shaper algorithm:

- Credit value
- idleSlopeCredit and sendSlopeCredit values
- Bandwidth status

Credit Value

The credit value is part of the credit-based sharper algorithm used by the MTL queue scheduler. The credit value is accumulated every transmit clock cycle, that is, 40 ns for 100 Mbps and 8 ns for 1000 Mbps. The credit to add or subtract per cycle can be fractional based on the required idleSlope and sendSlope values, as described in the following table.

Table 30-78: Credit Value per Transmit Cycle Example

Mode	Values	Description
100 Mbps	portTransmitRate = 100 Mbps idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class) sendSlope = 30 Mbps	credit = 2.8 bits accumulates per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted. credit = 1.2 bits drains per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO based on the slot number in the transmit descriptor (if enabled) or the bandwidth availability on the application bus.

The credit for a queue builds up only when the packet is available, but it cannot be transmitted because the MAC is sending a packet from another queue. The EMAC supports another mode in which the credit can build up in advance for a queue in which no packet is available in respective part of the FIFO. This enables sending a burst of high priority traffic in a queue as soon as the data is available. You can enable this mode is enabled using the `EMAC_TQ[n]_ETS_CTL.AVALG` bit of corresponding queues.

When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no packet

waiting in a queue and other queues are transmitting. When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

idleSlopeCredit and sendSlopeCredit Values

The software must program the idleSlopeCredit and sendSlopeCredit values. The programmed values should be the credit accumulated or drained per clock cycle scaled by 1024, such as, 2.8 x 1024 = 2867 and 1.2 x 1024 = 1229. In addition, the software must program the hiCredit and loCredit values, scaled by 1024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12,000 bits and 3,036 bits respectively, the values programmed in the hiCredit and loCredit registers of the corresponding channel are 12000 x 1024 bits and two's complement of 3036 x1024, respectively.

Bandwidth Status

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the `EMAC_TQ[n]_ETS_STAT` registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The `EMAC_TQ[n]_ETS_STAT` registers show the average number of bits transmitted during the previous programmed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programmed in the `EMAC_TQ[n]_ETS_CTL.SLC` bit field of the respective queue. For example, when programming two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programmed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue.

A lower priority queue that uses the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue that uses the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, queue 1 and queue 2 use the credit-based shaper algorithm (with reserved bandwidth of 50% and 25%, respectively) and queue 0 uses the strict-priority algorithm. If queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by queue 0. Queue 2 cannot exceed the reserved bandwidth of 25%.

Slot Number Function

The slot number function is used to schedule the data fetch from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals.

The slot number at which the DMA fetches the data from system memory is programmed in the Transmit Descriptor Word 3 (TDES3) register. This 4-bit field allows the application to schedule data up to 16 slots; a programmable slot interval that ranges from 1µs to 4096 µs with a granularity of 1µs. This field is applicable only for the AV channels.

When the DMA fetches a transmit descriptor, it compares the slot number of the transmit descriptor with the internally generated reference slot interval. The slot interval is a counter that is updated based on a programmable slot interval (with a range of from 1µs to 4096 µs) of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the seconds field of the system time is incremented, that is, the sub-second counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state until there is a match.

To enable the DMA to fetch the data only if it matches the current slot or the next two slots, program the EMAC_DMA[n]_SFN_CTLSTAT.ASC bit of the corresponding DMA channel.

NOTE: If the slot number in the descriptor is less than the reference slot number, the DMA takes it as a future slot.

The program also enables the check for slot number by setting EMAC_DMA[n]_SFN_CTLSTAT.ASC bit. When this check is not enabled, the packets are fetched immediately after the descriptor is read. In addition, the EMAC_DMA[n]_SFN_CTLSTAT.RSN bits indicate the value of the reference slot number in DMA.

Queue Modes

Programs can enable a transmit queue for generic DCB and AV traffic. When the program selects multiple transmit queues without selecting the DCB feature, the queues are enabled for generic queuing using the WRR or WSP algorithms.

When the program selects the enable data center bridging feature, DCB is enabled for all selected transmit queues and queuing is based on the WRR, WSP, DWRR, or WFQ algorithms. To enable a transmit queue for AV, select the enable audio video bridging option and then the enable support for AV in transmit Queue # option. The queuing is based on the CBS or SP algorithms.

The program can enable a receive queue for generic, DCB, or AV traffic. A particular queue enabled for generic DCB or AV based routing is determined by the RXQ#EN bit field (EMAC_RXQ_CTL0 register) of the corresponding queue. The following sections explain how receive queues are enabled based on the selected features.

Multiple Receive Queues without the DCB Feature

When this mode is selected, all queues are enabled for generic queuing based on the VLAN tag priority. The VLAN tag priority should match the PSRQ field of the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. By default, untagged packets are routed to the receive queue specified in the EMAC_RXQ_CTL1.UPQ bit field. Queue 0 is the default value of the UPQ field. The program can override the default value with any other value for the UPQ field. The receive packets can also be routed to a particular DMA channel based on the EMAC_ADDR[i]_HI.DCS bit field.

Multiple Receive Queues with DCB Feature

When this mode is selected, DCB is enabled for all selected receive queues. The queuing is done based on the VLAN tag priority for DCB data packets. The VLAN tag priority should match the PSRQ field of the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. The receive packets can also be routed to a particular DMA

channel based on the EMAC_ADDR[i]_HI.DCS bit field. The DCB control packets are routed based on the EMAC_RXQ_CTL1.DCBCPQ bit field.

Multiple Receive Queues with AV Feature

When this mode is selected, AV is enabled for all selected receive queues. The queuing is done based on the VLAN tag priority for DCB data packets. The VLAN tag priority should match the PSRQ field of the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. The receive packets can also be routed to a particular DMA channel based on the EMAC_ADDR[i]_HI.DCS bit field. The AV control packets (tagged or untagged) are routed based on the EMAC_RXQ_CTL1.AVCPQ bit field. The PTP over Ethernet packets are routed based on the EMAC_RXQ_CTL1.PTPQ bit field.

Queue Priorities

The transmit queue priorities are programmed using the EMAC_TXQ_PRTY_MAP0 and EMAC_TXQ_PRTY_MAP1 registers. The receive queue priorities are programmed using the EMAC_RXFLOW_CTL register.

The priority of a receive queue is programmed using the corresponding field of the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. Assign the priority in the following order:

1. AV queue (high priority)
2. DCB queue (medium priority)
3. Best-effort queue (low priority)

The software should put packets with correct priorities in the respective programmed queue on transmit side. The MAC uses the programmed priorities for blocking the transmit queues when a PFC packet is received. If a single queue is selected for multiple priorities and PFC is enabled, the entire queue is paused if one or more priorities in the queue are paused.

TCP/IP Offloading Features

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. Therefore, the MAC has an Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path.

The TCP Segmentation Offload (TSO) engine is useful in offloading the TCP segmentation functions to the hardware.

This section contains the following sub-sections:

- Transmit Checksum Offload Engine
- Receive Checksum Offload Engine
- TCP/IP Segmentation Offload (TSO) Engine

- UDP/IPv4 Fragment Offload (UFO) Engine
- IPv4 ARP Offload Engine

Transmit Checksum Offload Engine

The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path where the software can offload the checksum insertion task to the hardware. In the transmit path the MAC calculates the checksum and inserts it in the transmit packet. This feature reduces the load on the software and can improve the overall throughput of the system. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 Bits[17:16]).

NOTE: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, when this function is enabled, the transmit FIFO automatically operates in the store-and-forward mode even if the EMAC is configured for threshold (cut-through) mode.

NOTE: The application must ensure that the transmit FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter. The reason being that when space is not available to accept the programmed burst length of data, then the MTL transmit FIFO starts reading to avoid deadlock. In such a case, the COE fails as the start of the packet header is read out before the payload checksum can be calculated and inserted. Therefore, the program must enable the checksum insertion only in the packets that are less than the number of bytes, given by the following equation:

Packet size < TxQSize() – (PBL + 7) x 4,

Where, TxQSize is indicated by the `EMAC_TQ[n]_OPMODE.TQS` bit field and PBL by the `EMAC_DMA[n]_TXCTL.TXPBL` bit field.

IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit header checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the type field of IPv4 datagrams. The integrity of the header fields is indicated by the 16-bit header checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the type field of Ethernet packet has the value 0x0800 and the version field of an IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

NOTE: IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP header error status bit in the transmit status (bit 0 in RDES3 Normal Descriptor (Write-Back Format)). This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the version field of IP header is not equal to 0x4.

  - The IPv4 header length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 header length field.
- For IPv6 datagrams:
  - The Ethernet type is 0x86dd but the IP header version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding header length field in an extension header) is completely received.

TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The transmit COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the checksum field of the input packet. This engine includes the checksum field in the checksum calculation, and then replaces the checksum field with the final calculated checksum.
- The engine ignores the checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

NOTE: For ICMP-over-IPv4 packets, the checksum field in the ICMP packet must always be 16'h0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 16’h0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the payload checksum error status bit in the transmit status vector (bit 12 of TDES3). This engine sets the payload checksum error status bit when it detects that the packet has been forwarded to the MAC transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the payload length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

The following table describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Table 30-79: Transmit Checksum Offload Engine Functions for Different Packet Types

Packet Type	Hardware IP Header Checksum Insertion	Hardware TCP/UDP Checksum Insertion
Non-IPv4 or IPv6 packet	No	No
IPv4packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine.	Yes	Yes

Table 30-79: Transmit Checksum Offload Engine Functions for Different Packet Types (Continued)

Packet Type	Hardware IP Header Checksum Insertion	Hardware TCP/UDP Checksum Insertion
IPv6packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine.	Not Applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6packet with the following next header fields in main or extension headers: • Hop-by-hop options (in IPv6 main header) • Hop-by-hop options (in IPv6 extension header) • Destinationsoptions • Routing (with segment left = 0) • Routing (with segment left > 0) • TCP, UDP, orICMP • Authentication • Any other next header field in main or extension headers	 Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable	 Yes No Yes No No Yes No No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: • IPv4 packet in an IPv4 tunnel • IPv6 packet in an IPv4 tunnel	 Yes (IPv4 tunnel header) Yes (IPv4 tunnel header)	 No No
IPv6 Tunnels: • IPv4 packet in an IPv6 tunnel • IPv6 packet in an IPv6 tunnel	 Not applicable Not applicable	 No No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Receive Checksum Offload Engine

The EMAC provides the Checksum Offload Engine that is used to detect any error in an IPv4 or IPv6 packet in the receive path. The MAC verifies the checksum field of the received packet with the internally calculated checksum

and provides the status. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

Here, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the enable double VLAN processing option is selected and the `EMAC_VLANTAG_CTL.EDVLP` bit is set.

The receive COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

Packets with TCP/IP errors (header or payload) are dropped in MTL when the `EMAC_RQ0_OPMODE.DIS_TCP_EF` bit is reset and the `EMAC_RQ0_OPMODE.FEP` bit is set.

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a payload checksum error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

The following table describes the functions supported by the receive COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

NOTE: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 30-80: Transmit Checksum Offload Engine Functions for Different Packet Types

Packet Type	Hardware IP Header Checksum Checking	Hardware TCP/UDP Checksum Checking
Non-IPv4 or IPv6 packet	No	No
IPv4packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC)	Yes	Yes
IPv6packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in MAC)	Not Applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes

Table 30-80: Transmit Checksum Offload Engine Functions for Different Packet Types (Continued)

Packet Type	Hardware IP Header Checksum Checking	Hardware TCP/UDP Checksum Checking
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: • Hop-by-hop options (in IPv6 mainheader) • Hop-by-hop options (in IPv6 extensionheader) • Destinations options • Routing (with segment left 0) • Routing (with segment left >0) • TCP, UDP, or ICMP • Any other next header field in main or extension headers	 Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable Not Applicable	 Yes No Yes Yes No Yes No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: • IPv4 packet in an IPv4 tunnel • IPv6 packet in an IPv4 tunnel	 Yes (IPv4 tunnel header) Yes (IPv4 tunnel header)	 No No
IPv6 Tunnels: • IPv4 packet in an IPv6 tunnel • IPv6 packet in an IPv6 tunnel	 Not applicable Not applicable	 No No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Header-Payload Split

The MAC can identify the boundary between header and payload of the received packet and store them into separate buffers. This feature is useful for multiple reasons:

- Header buffers can be lin faster memory/cache while payload buffers are in slower DRAM memory. This enables the software stack to process the headers faster.
- Buffers used for payload can be directly forwarded to the application layer without the need to copy the payload into application buffer.

- When the Large Receive Offload (LRO) function is implemented in the software/driver layer, it is easy to link the buffers of multiple and contiguous payload data and form a bigger packet with a new header. This reduces the number of packets forwarded to the upper layer and thus improves system software performance.

The EMAC supports multiple methods and packet types for header-payload splitting. This is controlled by the setting the `EMAC_SPLM_OFST_CFG.SPLM` bit field and described in the following table.

Table 30-81: Split Header Support Depending on the Packet Type

Packet Type	SPLM	Description
TCP or UDP packet	00 (L3/L4 split)	The DMA writes the Ethernet header + IP header + TCP or UDP header into the header buffer.
IP packet (not TCP/ UDP)		The DMA writes the Ethernet header + IP header into the header buffer.
Non-IP packet		The DMA does not split the header and payload
Any packet	01 (L2 split)	The DMA writes the Ethernet header • For AV type packets using AV type Split Offset (SAVO) when split AV Enable (SAVE) field is set to 1. • For non-AV type packets using SPLOFST
IP packet	10 (combination of L2 or L3/L4 split)	L3/L4 split
Non-IP packet		L2 split
N/A	11	Reserved

NOTE:
- L3/L4 split is applicable for IP packets that are either untagged or VLAN stripped. If VLAN tag is retained in the packet forwarded to the DMA, L3-L4 split is not performed. When SPLM field is set to 2, L2 Split is performed for IP packets which are VLAN tagged.
- For AV packets, program the `EMAC_SPLM_OFST_CFG.SAVE` and `EMAC_SPLM_OFST_CFG.SAVO` bit fields to specify an alternative L2 split value.

The IP header includes IPv4 options in case of a IPv4 packet, and IPv6 extension headers in case of IPv6 frames. The points at which the L3/L4 header is split are shown in the following figure.

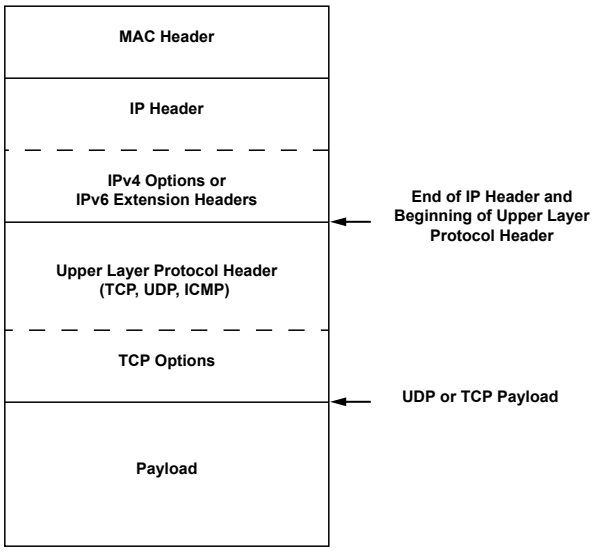


Figure 30-38: L3/L4 Header Split Points

The Figure 30-39 L2 Header Split figure shows the L2 split boundary. In this mode, the header payload boundary is determined for received packets at a fixed length offset starting from the length/type field, as indicated by the `EMAC_SPLM_OFST_CFG.SPLOFST` bit field. This mode is applicable for all received packets when `EMAC_SPLM_OFST_CFG.SPLM` = 01 and only for non-IP packets when SPLM = 10.

The following table shows the split offset for various packet types, when SPLOFST field is set to 2.

Table 30-82: Split Offset Calculation Example: SPLOFST = 2

Packet Type	Split Offset (From the Beginning of LT Field)
Untagged packet	DA + SA + LT = 14
Single VLAN tagged packet with VTAG stripped	
Double VLAN tagged packet with both VTAGs stripped	
Double VLAN tagged packet with both VTAGs stripped	DA + SA + VTAG + LT = 18
Double VLAN tagged packet with only one VTAG stripped	
Double VLAN tagged packet with no stripping	DA + SA +2 x VTAG + LT = 22

The split offset reference is always from the beginning of the LT field, whether the VLAN tag is present or stripped. For more details about the descriptor structure for the split header feature, see

Descriptor Structure for Split Header Support

.

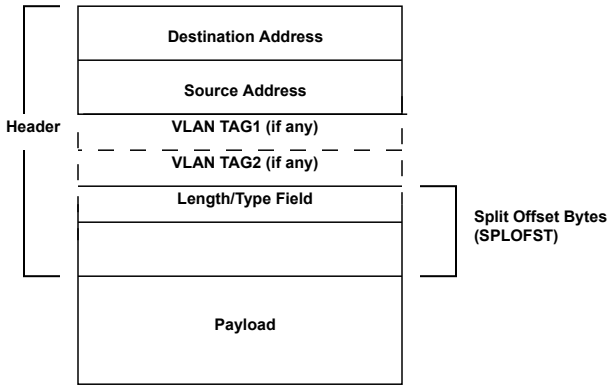


Figure 30-39: L2 Header Split

For programming details see the Programming Header Payload Split Receive section.

TCP/IP Segmentation Offload (TSO) Engine

The TCP Segmentation Offload (TSO) engine is useful in offloading the TCP segmentation functions to the hardware.

It also supports UDP Segmentation Offload (USO) in which the UDP payload is segmented in the hardware. There are no sequencing/ordering controls available or updated in the segments, so it can be used in point to point applications in which out of order packets are not expected by the receiver. The description and flow of TSO mentioned in this section is same as USO, any deviation is provided as notes.

In the TCP segmentation offload (TSO) feature, the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in the TCP Segmentation Offload Overview figure.

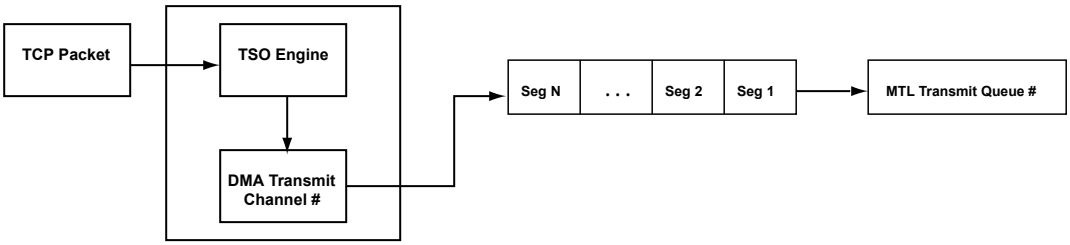


Figure 30-40: TCP Segmentation Offload Overview

DMA Operation with TSO Feature

The TCP Segmentation Offload Flow figure shows the TSO flow.

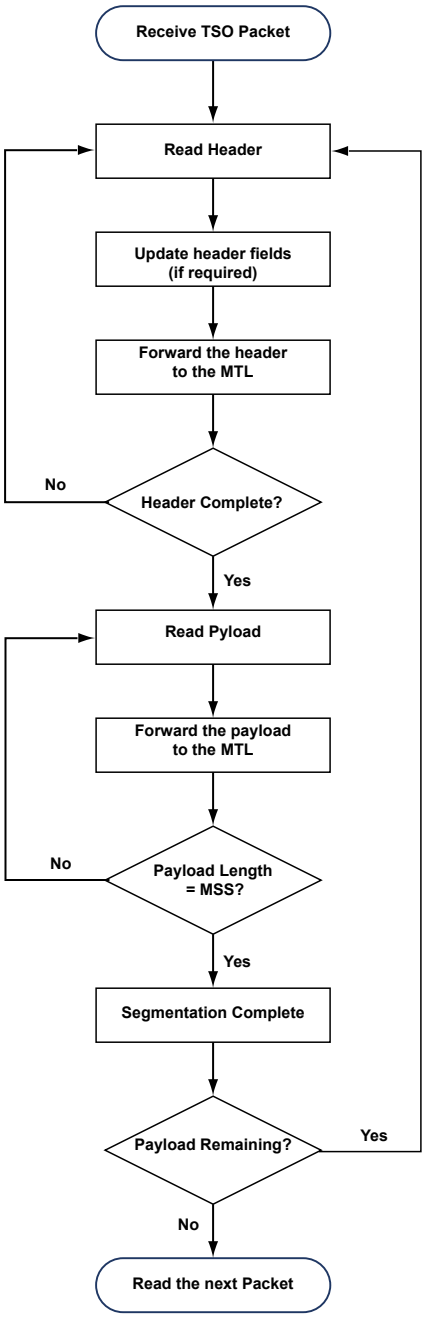


Figure 30-41: TCP Segmentation Offload Flow

For the TSO feature, the transmit DMA operation is as follows:

1. The application sets up the transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application increases the offset value of the descriptor tail pointer of the DMA transmit channel.
3. While in the Run state, the DMA fetches the next available descriptor and does one of the following:

a. If the descriptor is a context descriptor and the context is not between the first and last descriptors of a packet, the DMA stores the context values.

b. If the descriptor is a context descriptor and the context is between the first and last descriptors of a packet, the DMA closes the context descriptor indicating a context descriptor error (TDES3[23]) and fetches the next descriptor.

c. If the descriptor is a normal descriptor, the DMA checks for the TSE bit. If the TSE bit is not set, the DMA continues with the default mode of operation or OSF operation (if enabled).

4. The DMA calculates the number of segments from the TCP payload length (TDES3[17:0]) and the MSS value.
5. The DMA goes through channel arbitration to fetch the data buffers. The DMA fetches the header and payload separately.
6. For the first segment, the DMA fetches the header from the system memory and stores it in the TSO memory (when the length of header is not greater than the TSO memory size). If the current segmented packet is not the first segment, the DMA fetches the header from the local TSO memory. Otherwise, it fetches the header buffer in system memory again, as done for the first segment. In such cases (header not available is TSO memory), the DMA does not close the first descriptor containing the header buffer, until the header for last segment is fetched.
7. The required fields in the header bytes are modified/updated as per the segmentation requirements and written into the corresponding MTL TxQ.
8. The DMA then takes the payload buffer pointer, fetches the MSS number of payload bytes from the system memory and directly pushes it into the MTL TxQ. In case the buffer(s) in the descriptor do not have enough data for the MSS payload (except for the last segment), the DMA closes this descriptor.
9. The DMA jumps to Step 3 and repeats the process until the last segment is written into the TxQ.
10. The DMA closes the last descriptor and the first descriptor (containing the header buffer when it is not stored in TSO memory) and then moves on to the next packet transfer.

The DMA repeats all these steps if more descriptors are available. When no descriptor is available, the DMA enters the suspend state.

NOTE:
- The TSO memory size on ADSP-SC59x processor is 256 bytes per channel.
- The TSO engine determines whether to perform TSO or USO operation based on the THL field (TCP Header Length) in TDES3 of first normal transmit descriptor for the packet. The value of 2 indicates USO and any value greater than or equal to 5 indicates TSO.

The DMA repeats all these steps if more descriptors are available. When no descriptor is available, the DMA enters the suspend state.

TCP/IP Header Fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. The following table describes how the TCP and IP headers are updated.

Table 30-83: TSO: TCP and IP Header Fields

Packet Sequence	TCP Header	IP Header
First packet	1. The sequence number is not updated. The value provided in the header is used. 2. The TCP checksum is calculated again. 3. If set, the FIN and PSH flags are cleared.	IPv4 Header: • Total Length = MSS + TCP Header Length + IP Header Length • Identification field is not modified. It is sent as per the header provided by the software • IPv4 Header Checksum is recalculated. IPv6 Header: Payload Length = MSS + TCP Header Length + IPExtension Header Length
Subsequent packets	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header: • Total Length = MSS + TCP Header Length + IP Header Length • Identification field = Previous Identification Field + 1 • IPv4 Header Checksum is recalculated. IPv6 Header: Payload Length = MSS + TCP Header Length + IPExtension Header Length
Last packet	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If FIN and PSH flags were set in original header, these flags are set. 3. The TCP checksum is calculated again.	IPv4 Header: • Total Length = Remaining Payload + TCP Header Length + IP Header Length • Identification Field = Previous Identification Field + 1 • IPv4 Header Checksum is recalculated. IPv6 Header: Payload Length = Remaining Payload Length + TCP Header Length + IP Extension Header Length

NOTE: In case of USO, the engine updates only the following fields in the IP-UDP headers:

- Total length, identification field, and header checksum field in IPv4 header
- Payload length field in IPv6 header
- Length and checksum field in UDP header

Header and Payload Fields of Segmented Packets

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in the Table 30-83 TSO: TCP and IP Header Fields table. The *Header and Payload Fields of Segmented Packets* figure shows how the same header is used for the header fields of segmented packets.

The application must create the header in buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

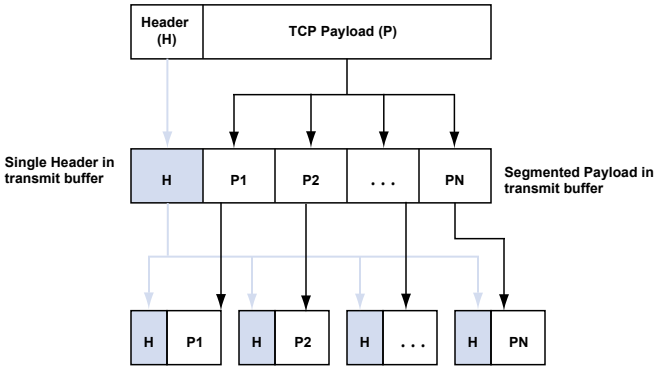


Figure 30-42: Header and Payload Fields of Segmented Packets

Context Descriptor Sequence

Programs can use the context descriptor to provide the MSS value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application wants to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset.

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this MSS value for all subsequent packets that have the TSO Enabled through the TSE bit of TDES3 normal descriptor.

If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor
2. The DMA indicates the error in descriptor status
3. The DMA generates an interrupt if the `EMAC_DMA[n]_IEN.CDEE` bit is set corresponding to a DMA channel

The application can read the interrupt status through the CDE bit of the status register corresponding to a DMA channel.

Building the Descriptor and the Packet for the TSO Feature

Set the TSE bit of TDES3 of the first normal descriptor. If the TSE bit is set in TDES3 for a non TCP/IP packet, the DMA behavior is unpredictable.

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 KB.

NOTE: The TDES3[22:19] field is used as slot number in case of AV configurations. If both TSO and AV are enabled for the same DMA transmit channel (the `EMAC_DMA[n]_TXCTL.TSE` bit is set), then slot checking for that DMA transmit channel must not be enabled (do not set the `EMAC_DMA[n]_SFN_CTLSTAT.ESC` bit).

The header of the packet including the Ethernet header, L3 header and L4 header should be provided in buffer 1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer 1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to and be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the maximum segment size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the `EMAC_DMA[n]_CTL.MSS` value or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes, otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in the `EMAC_DMA[n]_TXCTL` register should be lesser than the Tx Queue size programmed in the `EMAC_TQ0_OPMODE.TQS` bit field. It is recommended that MSS + header is equal to half the programmed transmit queue size.

The DMA also supports segmentation of VLAN tagged TCP/IP frames, so if the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.

If the double VLAN feature is selected, then the DMA passes both the tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits for both tags is applicable for all segments.

If the double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

If the TSE bit is set in TDES3 for the packet and TCP header length provided is less than 5 (meaning an invalid TCP header because it is less than 20 bytes), the DMA does not perform segmentation, instead it transmits the entire packet as a single packet. In this scenario, the CRC pad control bits are forced by DMA to 2'b00 (MAC does CRC and padding) and checksum insertion control bits are forced to 2'b11 (hardware does the checksum calculation for both header and payload).

See the Programming TSO section for more details on how to program the TSO engine.

UDP/IPv4 Fragment Offload (UFO) Engine

The EMAC supports breaking a large UDP packet into smaller and multiple packets for transmission, using IPv4 Fragmentation. Each IPv4 fragment contains the following header or payload information:

- Ethernet Header
- IP Header
- UDP Header (only in the first fragment)
- UDP Payload

NOTE:
- Fragmentation is supported only for UDP over IP. Fragmentation is not supported for TCP packets.
- Fragmentation of UDP is supported only over IPv4. This support is not available over IPv6.

The UDP Fragment Offload (UFO) Engine is an add-on function of the TSO engine. Therefore, both UFO and TSO functions can be enabled on the same transmit DMA channel. In this mode, the UDP packets are fragmented and TCP packets are segmented. If the UFO function is not enabled, then both TCP packets and UDP packets are segmented into smaller packets as described in TCP/IP Segmentation Offload (TSO) Engine.

The UFO breaks down a large UDP packet into smaller packets using IPv4 fragmentation. In this mode, the MSS field in the context Descriptor (TDES2[17:0]) and the `EMAC_DMA[n]_CTL.MSS` bit are interpreted as Maximum Fragmentation Size (MFS). This is the maximum size of the payload after the IPv4 header in each fragment. When UFO is enabled, the MSS field must be programmed in multiples of 8 bytes because the IP fragments offset is defined in terms of 8 bytes.

The UFO functions in two modes:

- UFO with valid UDP checksum
- UFO with no UDP checksum

UFO with Valid UDP Checksum

In this mode, the UFO engine calculates the UDP payload checksum and inserts it in the respective UDP header field. However, as the checksum must be calculated over the complete payload ending in the last fragment and then inserted in the UDP header encapsulated in the first IPv4 fragment, the fragments are not transmitted in order. The first IPv4 fragment containing the UDP header is sent last. The *Fragmentation Flow With Checksum* figure shows this sequence.

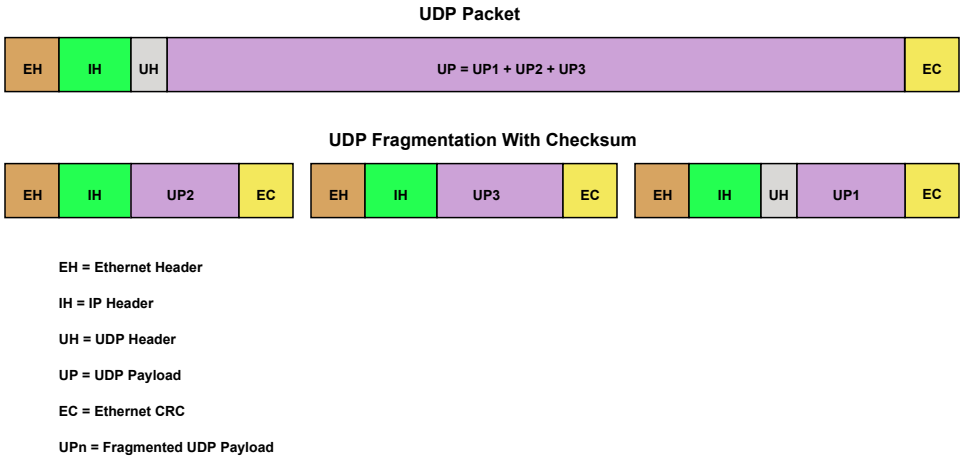


Figure 30-43: Fragmentation Flow With Checksum

The following table shows the details of the IP/UDP Header updates performed in each fragment by the UFO Engine in this mode.

Table 30-84: Details of IPv4 Fragmentation with UDP Checksum

Packet Sequence	UDP Header	IPv4 Header	Comments
First fragment (This segment is sent last)	Calculate the UDP header checksum	1. Total length = 32 bytes + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = 0 5. Update the IPv4 header checksum	The header (and 32 bytes IPv4 fragment payload) data is stored in TSO memory in this step.
Second fragment	NA (No UDP header)	1. Total length = MFS + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = previous fragment offset + (32/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.

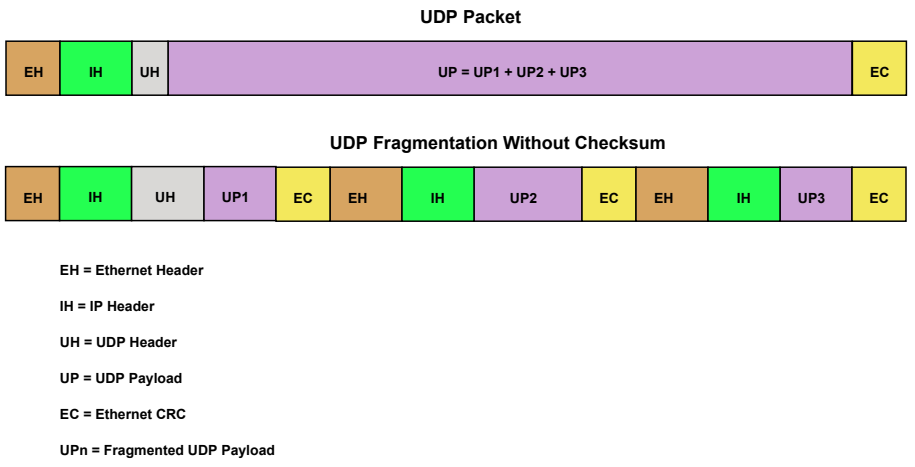
Table 30-84: Details of IPv4 Fragmentation with UDP Checksum (Continued)

Packet Sequence	UDP Header	IPv4 Header	Comments
Intermediate (non-first and non-last) fragment	NA (No UDP header)	1. Total length = MFS + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = previous fragment offset + (MFS/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.
Last fragment	NA (No UDP header)	1. Total length = remaining payload + IPv4 header Length 2. DF = 0 (do not Fragment) 3. MF = 0 4. Fragment offset = previous fragment offset + (MFS/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.

UFO with no UDP Checksum

In this mode, the UDP payload checksum is not calculated and an all-zero value is inserted in the corresponding checksum field of the UDP header and all the IP fragment packets are transmitted in sequence. the *Fragmentation Flow Without UDP Checksum* figure shows the sequence of fragmented packets created and transmitted by the UFO engine.

Figure 30-44: Fragmentation Flow Without UDP Checksum



The following table shows the details of IPv4/UDP header updates performed in each fragment of the UFO engine, in this mode.

Table 30-85: Details of IPv4 Fragmentation without UDP Checksum

Fragment Sequence	UDP Header	IPv4 Header	Comments
First fragment	Replace the UDP checksum field with all 0s	1. Total length = MFS (MSS field) + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = 0 5. Update the IPv4 header checksum	The header data is stored in the TSO memory in this step.
Intermediate (non-first and non-last) fragment	NA (No UDP header)	1. Total length = MFS (MSS field) + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 1 4. Fragment offset = previous fragment offset + (MSS/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.
Last fragment	NA (No UDP header)	1. Total length = remaining payload + IPv4 header length 2. DF = 0 (do not fragment) 3. MF = 0 4. New fragment offset = previous fragment offset + (MSS/8) 5. Update the IPv4 header checksum	The header data is read from the TSO memory. Only the Ethernet + IPv4 header is read.

Segmentation Versus Fragmentation

The following table shows the differences between UDP packet fragmentation performed by the UFO and UDP segmentation performed by the TSO engine.

Table 30-86: Segmentation Versus Fragmentation

Segmentation	Fragmentation
The UDP header is replicated for each segment. The length field in the UDP header is updated for each segment	The UDP header is not updated; however, checksum field in the UDP header is updated.
MSS field indicates the size of the maximum segment after the L4(TCP/UDP) header	MFS field indicates the size of the fragment after the L3 (IPv4) header
The TCP packet is broken down into multiple chunks by keeping L2+L3+L4 header	The UDP packet is broken down into multiple chunks by keeping the L2 + L3 header

IPv4 ARP Offload Engine

The EMAC supports the Address Recognition Protocol (ARP) offload for IPv4 packets. This feature allows the processing of the IPv4 ARP request packet in the receive path and generating corresponding ARP response packet in the

transmit path. EMAC generates the ARP reply packets for appropriate ARP request packets. The ARP packet for IPv4 is a L2 layer packet with length/type of 0x0806.

The ARP offloading process is as follows:

1. The MAC receiver gets an ARP request if the target protocol address of request matches the IPv4 address programmed in the EMAC_ARP_ADDR register of the MAC.
2. The MAC generates an ARP reply packet.
3. The MAC copies the sender hardware address field in the ARP request to the following fields:
   - DA field of the Ethernet packet header
   - Target hardware address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the target protocol address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   - SA field of the Ethernet packet header
   - Sender Hardware Address field of the ARP reply packet
6. The MAC copies the target protocol address field in the ARP request to the sender protocol address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for generated ARP reply packet.
9. The MAC transmitter sends the ARP reply.

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to application with ARP reply not generated (bit 34) status bit set. However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request.

If the disable CRC check bit of the MAC extension configuration bit is set, then the MAC does not check for valid CRC of a ARP request packet. It can generate an ARP response packet if the other conditions are valid.

NOTE: When the received ARP request is less than 64 bytes packet length, EMAC does not send a ARP response. It is treated as normal packet and forwarded to the application based on the EMAC filter settings.

Energy Efficient Ethernet (EEE) Mode

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps, 1000 Mbps, and 10 Gbps. The EMAC supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows power saving by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functions to save power during the periods of low-link use. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Transmit Path Functions

The transmit path functions include tasks that the MAC must perform to make the PHY to enter the LPI state.

In the transmit path, the software must set the `EMAC_LPI_CTLSTAT.LPIEN` bit to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the `EMAC_LPI_TMRSCTL.LST` bit field. The PHY link status (`EMAC_LPI_CTLSTAT.PLS` bit) indicates the link status of the PHY.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. Deasserts TXEN
2. Asserts TX_ER
3. Sets `EMAC_TXD[n]`[3:0] to 0x1 (for 100 Mbps) or TXD[7:0] to 0x01 (for 1,000 Mbps)
4. Updates the status (`EMAC_LPI_CTLSTAT.TLPIEN` bit) and generates an interrupt.

To bring the PHY out of the LPI state, that is, when the software resets the TLPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern
2. Starts the LPI TW TIMER. The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the `EMAC_LPI_TMRSCTL.TWT` bit field.
3. Updates the LPI exit status (`EMAC_LPI_CTLSTAT.TLPIEX` bit) and generates an interrupt

The *LPI Transitions (Transmit)* figure shows the behavior of the TXEN, TX_ER, and `EMAC_TXD[n]` [3:0] signals during the LPI mode transitions.

NOTE:
- Gating of EMAC_TXCLK is not supported on EMAC.
- If the MAC is in the LPI mode and the application issues a soft or hard reset, the MAC transmitter comes out of the LPI mode.

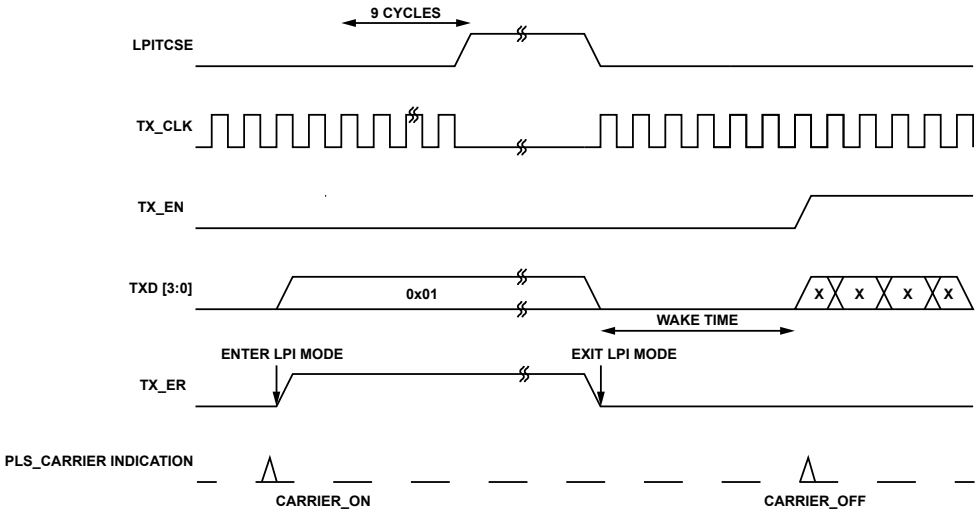


Figure 30-45: LPI Transitions (Transmit)

Automated Entry/Exit of LPI Mode

The MAC transmitter can be programmed to enter and exit LPI IDLE mode automatically based on whether it is IDLE for a specific period or has a packet to transfer. These modes are enabled and controlled by the `EMAC_LPI_CTLSTAT` register.

When the `EMAC_LPI_CTLSTAT.LPITXA` and `EMAC_LPI_CTLSTAT.LPIEN` are set, the MAC transmitter enters LPI IDLE state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI IDLE state and clear the LPIEN bit as soon as any of functions in the transmit path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition, when (`EMAC_LPI_CTLSTAT.LPIATE`) is also set, the MAC transmitter enters LPI IDLE state only when the Transmit path remains in idle state (no activity) for the period indicated by the value in the `EMAC_LPI_ENTR_TMR` register. In this mode also, the MAC transmitter exits the LPI IDLE state as soon as any of the functions becomes non-idle. However, the LPIEN bit is not cleared but remains active so that reentry to LPI IDLE state is possible without any software intervention when the MAC becomes idle again.

When both the `EMAC_LPI_CTLSTAT.LPIATE` and `EMAC_LPI_CTLSTAT.LPITXA` bits are cleared, the program can directly control the entry and exit of LPI IDLE state by programming the LPIEN bit.

Receive Path Functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts `EMAC_RXERR`
2. The PHY sets `EMAC_RXD[n]`[7:0] to 0x01
3. The PHY deasserts RX_DV

4. The MAC updates the EMAC_LPI_CTLSTAT.RLPIEN bit and immediately generates an interrupt

NOTE:
- The PHY maintains the same state of the EMAC_RXERR, EMAC_RXD[n], and RX_DV signals for the entire duration during which it remains in the LPI state.
- If the LPI pattern is detected for a very short duration (that is, less than two cycles of receive clock), the MAC does not enter the receive LPI mode.
- If the duration between end of the current receive LPI pattern and start of the next receive LPI pattern, is very short (that is, less than two cycles of receive clock), then the MAC exits and again enters the receive LPI mode. The MAC does not give the receive LPI exit and entry interrupts

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY deasserts EMAC_RXERR and returns to a normal inter-packet state.
2. The MAC updates the EMAC_LPI_CTLSTAT.RLPIEX bit and generates an interrupt immediately.

The following figure shows the behavior of the EMAC_RXERR, RX_DV, and EMAC_RXD[n][3:0] signals during the LPI mode transitions.

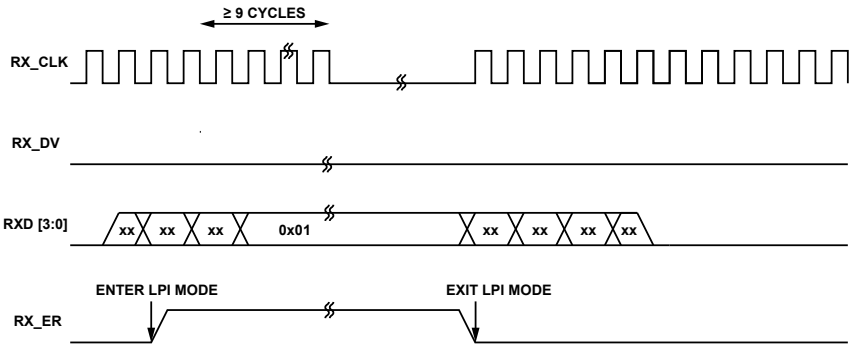


Figure 30-46: LPI Transitions (Receive)

NOTE:
- If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in the *LPI Transitions (Receive)* figure.
- If the MAC is in the LPI mode and the application issues a soft or hard reset, the MAC receiver comes out of the LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver again enters the LPI state.
- If the receive clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the receive clock domain.
- When the PHY sends the LPI pattern and if the EMAC_MAC_CFG.RE bit is set, the MAC automatically enters the LPI state. There is no other software control to prevent the MAC from entering the LPI state.

LPI Timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers. The following sections describe the LPI timers that are loaded with the respective values from the EMAC_LPI_TMRSCTL and EMAC_LPI_ENTR_TMR registers:

LPI LS Timer

The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. Programs can enable the monitoring of the EMAC_PHYIF_CTLSTAT.LUD bit by setting the EMAC_LPI_CTLSTAT.PLSEN bit.

The link status is indicated to the MAC by the EMAC_PHYIF_CTLSTAT.LNKSTS bit or the value programmed by the software in the EMAC_LPI_CTLSTAT.PLS bit. If the link status is not available in the EMAC_PHYIF_CTLSTAT, the software can get the PHY link status by reading the PHY register and accordingly update the PLS bit.

This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in the EMAC_LPI_TMRSCTL.LST bits. The RGMII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. Therefore, the software can program up to 1023 milliseconds.

LPI TW Timer

The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count is programmed using the EMAC_LPI_TMRSCTL.TWT bit field. The terminal count of the timer is the value of resolved transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count. The MAC supports the LPI TW TIMER in units of microseconds. The LPI TW TIMER is 16 bits wide. Therefore, the software can program up to 65535 µs.

LPI Auto Entry Timer

This timer counts in steps of eight microseconds, the time for which the MAC transmit path must remain in idle state (no activity), before the MAC transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when the EMAC_LPI_CTLSTAT.LPIATE bit is set.

NOTE: Program the EMAC_LPI_CTLSTAT.PLS bit to 1'b0 before switching between the GMII and MII modes. This resets the internal timers. If the mode is changed after the LPI LS TIMER or LPI TW TIMER starts, the change in the transmit clock frequency can result in incorrect timeout

Low Power Interrupt (LPI)

The MAC generates the LPI interrupt when the transmit or receive side enters or exits the LPI state. The LPI interrupt can be cleared by reading the EMAC_LPI_CTLSTAT register.

For detailed programming guidelines see Programming Energy Efficient Ethernet.

Time-Sensitive Networking

The Time-Sensitive network (TSN) is a set of standards developed by the IEEE 802.1 work group. The standards define mechanisms for the time-sensitive transmission of data over the Ethernet.

This section contains the following topics:

- Enhancements to Scheduled Traffic (EST)
- Frame Preemption (FPE)
- Time-Based Scheduling

Enhancements to Scheduled Traffic (EST)

The IEEE 802.1Qbv-2015 specification defines the schedule for each of the queues on every egress port which makes the implementation aware of traffic arrival schedule. This information can be used to block the lower priority traffic from transmission in this time window/slot. This ensures that scheduled traffic is forwarded from sender to receiver through all the network nodes with a deterministic delay. The ***Time Aware Shaper Implementation*** figure shows the scheduled traffic windows for periodic priority data.

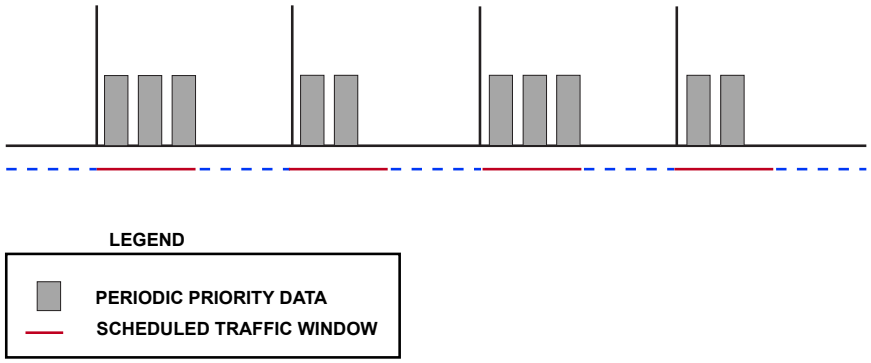


Figure 30-47: Time Aware Shaper Implementation

To achieve a low latency, ensure there are no interfering frames during the scheduled windows that are reserved for high priority traffic. The use of scheduled traffic imposes limitations while starting a transmission.

Refer to the ***Implementing a Guard Band*** figure. IWhenan interfering frame begins transmission just before the start of a reserved period, transmission can be extended into the reserved window, and potentially interfere with higher priority traffic. Therefore, a guard band whose size is equal to the largest possible interfering frame is required before the window starts.

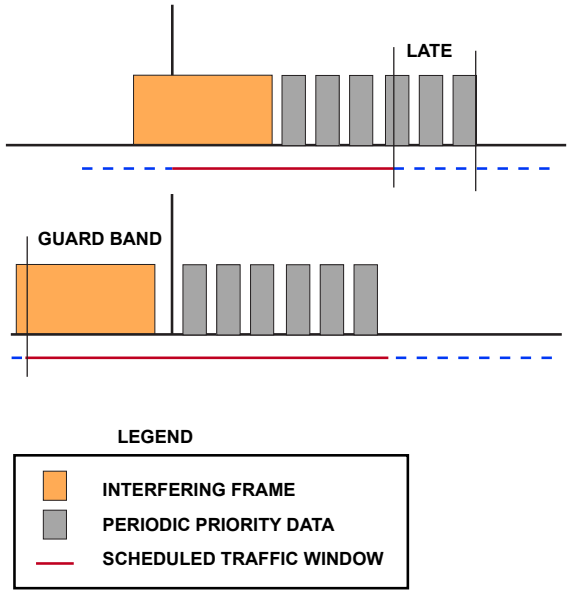


Figure 30-48: Implementing a Guard Band

A larger guard band equates to a less efficient use of the network bandwidth. However, the implementation of IEEE802.1Qbu (frame preemption) addresses this inefficiency. Frame preemption breaks the interfering frame into smaller fragments. Therefore, the guard band must only be as large as the largest possible interfering fragment (instead of the largest possible interfering frame).

During the guard band, only the frames that can complete the transmission of the entire frame before the next gate close event are permitted. This ensures that the high priority traffic can always start at the beginning of the specified window .

Definitions

The following terms are frequently used in the EST support:

- *Gate Control List* - The list in the hardware memory that holds the gate controls and the associated time intervals.
- *Gate Controls* – For a given schedule (row in gate control list), there is a gate open (O) and a gate close (C) state associated with each traffic control (TC). Gate controls are the set of O or C values, whose width is same as the configured TCs. For example, CCOCOOCO means TC7 = C, TC6 = C, TC5 = O and so forth.
- *Time Interval* – Time interval (in nano seconds). It is a 16-, 20-, or 24-bit configured field in the gate control list that indicates the time for which the associated gate controls are valid.
- *Base Time Register* – Each gate control list is associated with a 64-bit base time register that holds the start time (in PTP format) for the list.

Transmit Scheduling Updates

To support EST, the following updates are required for transmit scheduling:

- Implementation of the Gate Control List (GCL)
- Enforcing gate controls in the scheduler
- Accounting for gate open duty cycle in the computation of idleSlope (CBS)

Gate Control List (GCL) Implementation

The ***Gate Close and Open Events*** figure shows a block diagram based on the IEEE 802.1Qbv specification. It illustrates how the gate control list governs the gate close (C) and open (O) events based on the schedule provided for each event. GCL includes a time interval and gate control.

Time Interval – Defines the time in nano seconds for which the gate controls are valid. It should be applied before reading the next gate control from the list. It has a configurable width of 16, 20 or 24 to represent a maximum of 64 us, 1 ms, or 16 ms schedule intervals, respectively. The GLC implementation supports a left shift of a time interval up to 7 bits to apply a multiplication factor from 1 to 128 ns (in steps of 2^n). The maximum value (post shifting) of this field must be 999,999,999 ns.

Gate Control – Defines the open event (represented by a logic 1) or the close event (represented by logic 0) state for the gate of each TC.

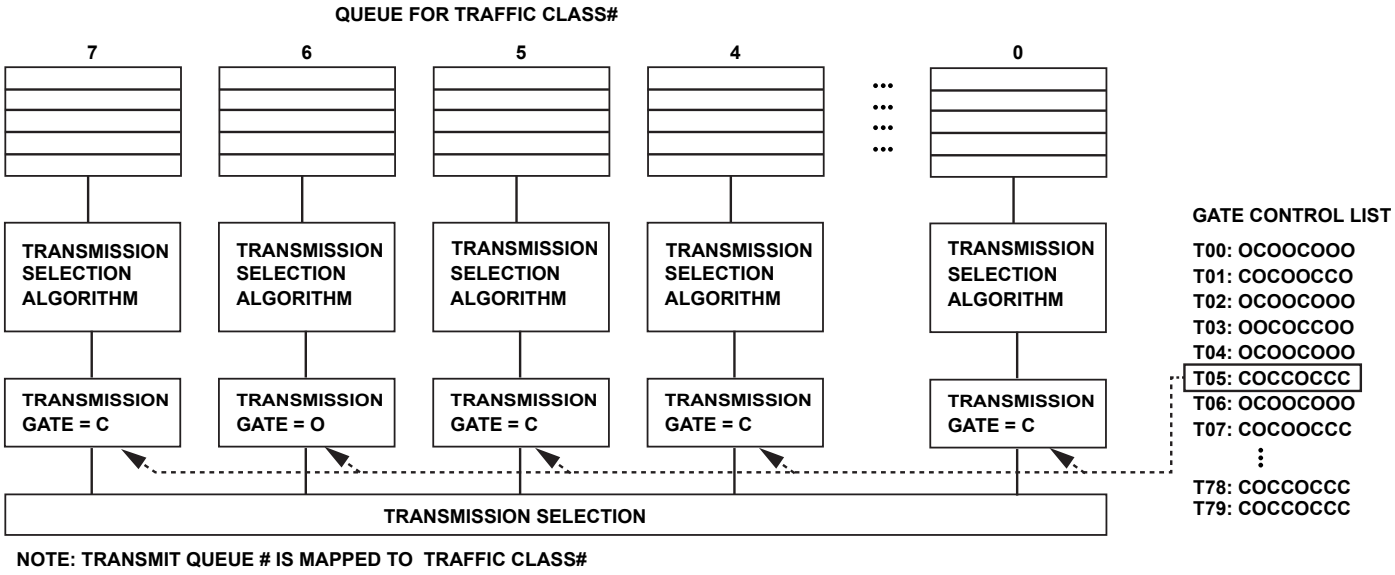


Figure 30-49: Gate Close and Open Events

The implementation of GCL consists of the following gate control lists:

- Hardware Owned List (HWOL) - a list for hardware access
- Software Owned List (SWOL) - a list for software access

The access to these lists is mutually exclusive. Hardware configures the ownership to the list in the `EMAC_EST_STAT` register. The ***External Memory Used for Holding the GCL*** table provides the implementation details of GCL.

Table 30-87: External Memory Used for Holding the GCL

Gate Control (up to 8 Bits)	Time Interval (ns) 16, 20 or 24 Bits	
OOCCCCCC	T0	HWOL or SWOL
OOOOCCCC	T1	
CCCCOOCC	T2	
CCCCCCOO	T3	
…	…	
…	…	
OCOCOCOO		
OOOOCCCC	T last	
OOOOCCCC	T0	SWOL or HWOL
OOOOCCCC	T1	
OOCCCCCC	T2	
OOOOCCCC	T3	
…	…	
…	…	
OCOCCCCC		
OOCCCCCC	T last	

GCL uses the following registers (one for each GCL). These registers are implemented through indirect addressing using the `EMAC_EST_GCL_CTL` and `EMAC_EST_GCL_DAT` registers.

Base Time Register (BTR) – A 64-bit register that holds the start time to execute the GCL. The format of the BTR is the same as the PTP format (the upper 32-bits holds time in seconds and the lower 32 bits hold time in nano seconds). Once the execution of a given list begins, the implementation can update the value in the BTR to indicate the next list execution time.

Cycle Time Register (CTR) – A 40-bit register that holds the time at which the execution of the GCL should be repeated. The cycle time register consists of an 8-bit value in seconds, and a 32-bit value in nano seconds (similar to the PTP time format with a truncated seconds register).

For a given gate control list:

Start time = Base Time + N * Cycle Time

where N is an integer value representing the iteration number starting with 0 for first iteration

If the gate control list execution takes longer than the cycle time, the list is truncated at the cycle time and the subsequent loop begins at cycle time.

Time Extension Register (TER) – An m-bit (where m = configured time interval width + 7) register that holds the amount of time (in nano seconds) that the current gate control list can be extended before switching to the new gate control list. This is useful to avoid small fragments of the current list before switching to a new list.

List Length Register (LLR) – An n-bit gate control register (when n is 7, 8, 9, 10, or 11 for a GCL configured depth of 64, 128, 256, 512, or 1024, respectively) that holds the integer value of the length of the GCL (the number of valid rows in each GCL). The processing of the GCL stops after the number of rows read equals the LLR value.

Transmission Gating Implementation

A bridge or an end station can be enhanced to allow transmission from each TC that is yet to be scheduled relative to a known timescale. To achieve this, a transmission gate is associated with each TC; the state of the transmission gate determines whether the queued frames can be selected for transmission.

For a TC, the transmission gate can be in one of the following states:

1. Open-queued frames are selected for transmission in accordance with the definition of the transmission selection algorithm associated with the TC
2. Closed-queued frames are not selected for transmission

A frame on a traffic class queue is not available for transmission when the transmission gate is in the closed state or when there is not enough time to transmit the entire frame before the next gate-close event associated with the queue.

The implementation has visibility into the current schedule of gate controls and the immediate next schedule of the gate controls. So, the maximum gate open period does not exceed the sum of the two time intervals. A frame is selected for transmission only when the gate is currently open and the duration of gate open interval is greater than the time taken to transfer the entire frame.

The implementation must have the frame size before the transmission to avoid transmission overruns and schedule only frames that can complete. To implement transmission gating:

- Program the `EMAC_EST_EXT_CTL.OVHD` bit field with a fixed overhead of the IPG or EIPG, preamble, and scheduler delay
- Enable the automatic prediction of variable overhead due to the offloads for a packet in the MAC transmitter. Consider the following while programming the EST-related fields for correct scheduling of packets:
    - Speeds: 1Gpbs/100 Mbps
    - PTOV in ns: 6 PTP clock cycles
    - CTOV in ns: 3 PTP + 3 transmit clock cycles
    - Per packet slot interval in ns: (packet size + overhead + IPG or EIPG + preamble) * (time to transmit 1 byte)
    - Overhead, scheduler delay in bytes is calculated as:

      17 Tx + (6 + X + Y + Z) CLKO7 clock cycles

where

- If Tx COE or TSO is selected, X = 4

  If TBS or OST is selected, Y = 4

  If VLAN tag insertion is enabled, Z = 2

The implementation compensates for the data transfer delays from the buffer to the line by offsetting the current time with all the relevant delays (provided by the `EMAC_EST_CTL.CTOV` field). This configuration ensures that the schedule provided is always accurately implemented at the line.

NOTE:
- An overhead is added to the packet size to account for the scheduler delay, IPG or EIPG, and preamble overheads on the line. Therefore, for a 128 byte frame to be transmitted, the gate open window must be able to accommodate at least 128 overhead bytes.
- In 100 Mbps mode of operation, the rounding down error is about 0.4%. For example, the gate open duration must be at least 1035 bytes for transmitting 1000 byte frames at a speed of 100 Mbps (31 bytes for scheduler delay, line overheads for two-port SRAM configurations, and 4 bytes for rounding down error)
- The minimum time interval for GCL entry must be equal to the time required to transmit fragment size + scheduler delay + IPG/EIPG + preamble overheads

Idle Slope Computation

When EST is enabled, credit is accumulated only when the gate is open; therefore, the effective data rate of the idle slope must be increased to reflect the duty cycle for the transmission gate associated with the queue.

The idle slope is computed based on GateOpenTime and OperCycleTime values. Program the idleSlope registers (implemented one per CBS-enabled TC) based on the following equation. The existing MTL register has sufficient bit width to accommodate the new values for idleSlope.

idleSlope = (operIdleSlope(N) * OperCycle/GateOpenTime)

Operational Details of GCL

Set (=1) the switch to the software list using the `EMAC_EST_CTL.SSWL` bit. Hardware can subsequently access the programmed gate control list. The first configuration of gate controls are applied when the current time is equal to the value in the BTR. It is held for the programmed time interval value.

To avoid transmission overruns, one additional gate control event is read ahead from the list. This operation permits the GCL to determine the next gate close events (if any) for the TCs that are open.

The scheduling is based on the gate open state and time interval of only the current and subsequent schedule. An internal accumulator is used to add the time intervals when gate controls are applied. The BTR and accumulator hold the time at which the next set of gate controls will be applied.

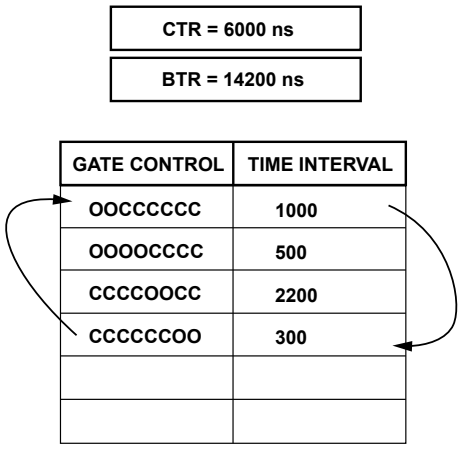


Figure 30-50: GCL Example - Base Time and Cycle Time

Refer to the ***GCL Example - Base Time and Cycle Time*** figure. The GCL is read progressively from the first row adhering to the schedule. The read operations continue until the list length (from LLR register) is reached and the execution of the list restarts at BTR + CTR time. At this point, the BTR is incremented by value in CTR to mark the beginning of a new cycle. In the absence of any gate controls, all gates are deemed to be in the open state during the execution of the list.

In cases where the execution time of the list is greater than the cycle time, the list is truncated and the next iteration starts when the current time equals BTR + CTR.

Table 30-88: GCL Execution Example - BTR and CTR

Current Time	Gate Control Applied	Accumulator Value (ns)	BTR (with updates)
14200	OOCCCCCC	1000	14200
15200	OOOOCCCC	1500	14200
15700	CCCCOOCC	3700	14200
17200	CCCCCCOO	4000	14200
18200	OOOOOOOO	0	20200
20200	OOCCCCCC	1000	20200
21200	OOOOCCCC	1500	20200

Refer to the ***GCL Execution Example - BTR and CTR*** table. The execution starts at 14200 and the first set of gate controls OOCCCCCC are applied immediately. When the time interval is 1000 ns, the next set of gate controls are applied at 14200 (BTR) + 1000 (Accumulator) = 15200 ns. Since there are no gate controls available after the execution of the last gate control and before the next iteration of the loop, the gates are deemed to be in open state during that period as depicted at time 18200.

Refer to the ***GCL Example - Base Time and Cycle Time, Execution Time > Cycle Time*** figure. Because the list execution takes longer than the allocated cycle time, the list is truncated.

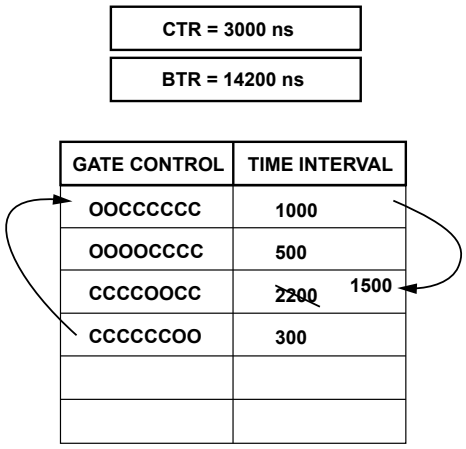


Figure 30-51: GCL Example - Base Time and Cycle Time, Execution Time > Cycle Time

Refer to the ***GCL Execution Example - BTR and CTR , Execution Time > Cycle Time*** table. The list is started from the BTR + CTR. While applying the third set of gate controls, the BTR + Cycle Time (17200) < BTR + Accumulator (17900). Therefore, the list is truncated and execution switches to a new iteration at 17200.

Table 30-89: GCL Execution Example - BTR and CTR, Execution Time > Cycle Time

Current Time	Gate Control Applied	Accumulator Value (ns)	BTR (with updates)
14200	OOCCCCCC	1000	14200
15200	OOOOCCCC	1500	14200
15700	CCCCOOCC	3700	14200
17200	OOCCCCCC	1000	17200
18200	OOOOCCCC	1500	17200
18700	CCCCOOCC	3700	17200

Installing a New GCL

When a new software programmed GCL is available and must be executed at the new BTR value, the switch to the new GCL can happen as a:

- New base time aligned with current schedule
- New base time unaligned with current schedule

New Base Time Aligned with Current Schedule

If the cycle time for the new gating cycle is the same as the cycle time for the current gating cycle, and if the BTR chosen for the new gating cycle is an integer multiple of the current cycle time (+ current BTR), the new and old gating cycle are aligned. This would be the ideal case; it allows the new gating cycle to be installed and executed with no timing issues. The implementation completes the execution of an iteration of the current list, and switches to the new list at the beginning of the base time listed in the new list.

When (New BaseTime ≥ Current Time), then ConfigChangeTime = New BaseTime, Else If (New BaseTime < Current Time):

1. Set the BTRError.
2. ConfigChangeTime = (New BaseTime + N* New CycleTime).

   where N is the smallest integer for which the relationship ConfigChangeTime ≥ CurrentTime and (N =< 8) = TRUE.

When N > 8, the hardware cannot auto recover and the loop count value in BTR error reporting is set to 1111. The software must reprogram the new base time.

The ***Switching to a New List Example - Aligned*** figure illustrates the installation of a new GCL that is aligned with the existing configuration. In the example, after the sixth iteration of the first GCL, the BTR values of the old and new GCL are equal. At that point, the new GCL is processed as a natural extension to the existing GCL.

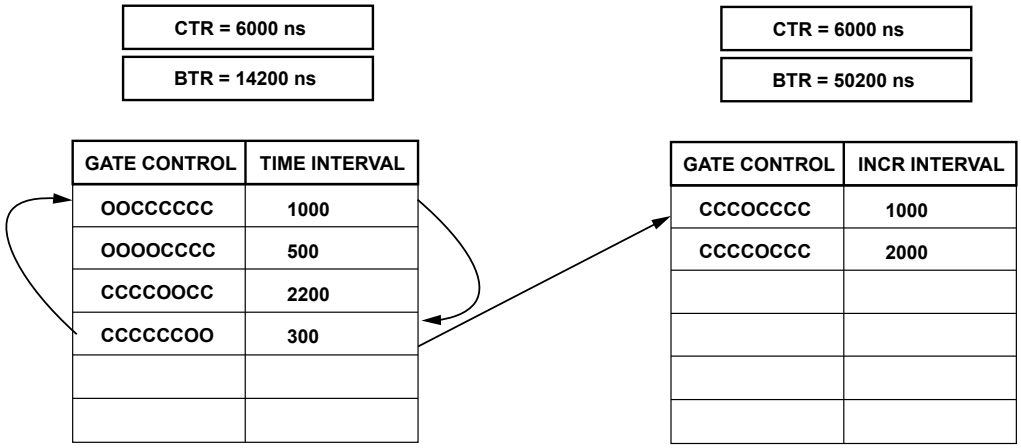


Figure 30-52: Switching to a New List Example - Aligned

Table 30-90: GCL Execution Example - BTR and CTR

Current Time	Gate Control Applied	Accumulator Value	BTR (with updates)
44200	OOCCCCCC	1000	44200
45200	OOOOCCCC	1500	44200
45700	CCCCOOCC	3700	44200
47900	CCCCCCOO	4000	44200
48200	OOOOOOOO	0	50200
50200	CCCOCCCC	1000	50200
51200	CCCCOCCC	3000	50200
53200	OOOOOOOO	0	56200

New Base Time Unaligned with Current Schedule

If a new cycle time differs from current cycle time or the new base time is in the future and not an integer multiple of the current cycle time, the old and new cycles are unaligned. When new base time is reached (when the new configuration is installed and starts to execute), the last old cycle is normally truncated to start the first new cycle. This could be undesirable if it results in a very short last old cycle. Arguably, it would be better to extend the old cycle by that small amount, rather than start a short cycle. The cycle time extension register (related to the current configuration list) allows this extension of the last old cycle. If the last complete old cycle ends normally in less than current Cycle Time Extension (TER) ns before the new base time, then the last complete cycle (before new base time is reached) is extended so that it ends at the new base time.

The ***Switching to a New List Example - Unaligned*** figure shows how the installation of a new GCL that is unaligned with the current schedule truncates the current list.

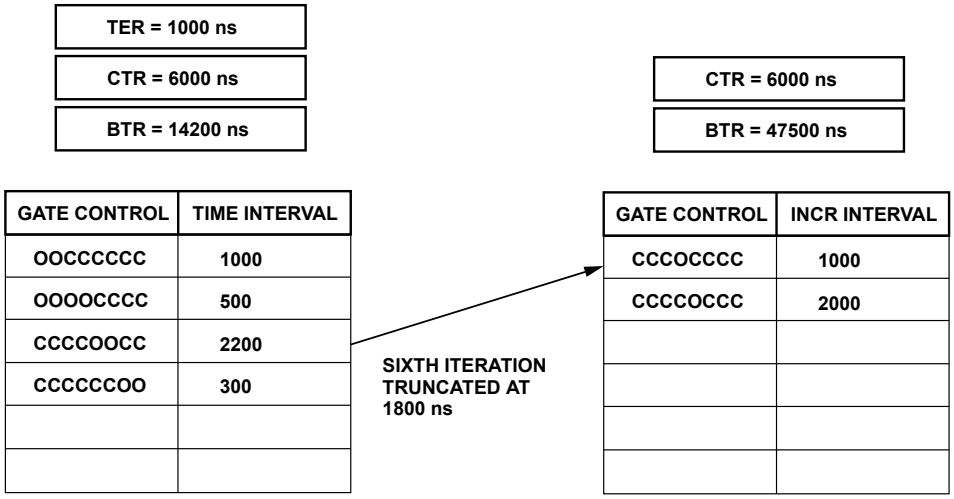


Figure 30-53: Switching to a New List Example - Unaligned

At the end of the fifth iteration the Current time + Cycle time extension (TER) < New BTR. So, the sixth iteration of current configuration is started. During the sixth iteration of the current list when the new BTR value is smaller than the next schedule in the current list, it switches to the new list.

Table 30-91: Extending to New List by Truncating the Current List

Current Time	Gate Control Applied	Accumulator Value	BTR (with updates)
44200	OOCCCCCC	1000	44200
45200	OOOOCCCC	1500	44200
45700	CCCCOOCC	3700	44200
47500	CCCOCCCC	4000	44200
48500	CCCCOCCC	0	50200
50500	OOOOOOOO	1000	50200

The *Switching to a New List Example - Extended* figure shows where the current configuration list is extended (instead of starting a new iteration). The extension time of 800 ns is less than the allowed cycle extension time (TER) of 1000 ns.

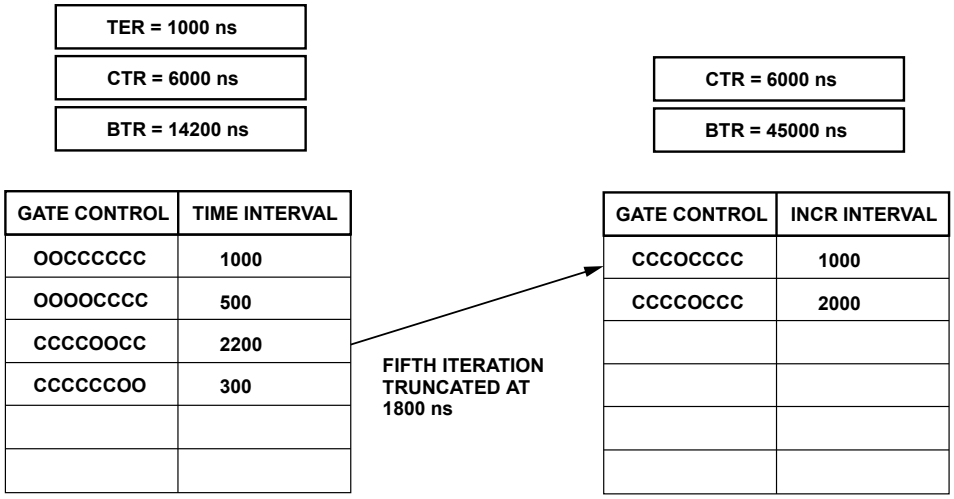


Figure 30-54: Switching to a New List Example - Extended

Table 30-92: Switching to New List by Extending the Current List

Current Time	Gate Control Applied	Accumulator Value	BTR (with updates)
38200	OOCCCCCC	1000	38200
39200	OOOOCCCC	1500	38200
39700	CCCCOOCC	3700	38200
41900	CCCCCCOO	4000	38200
42200	OOOOOOOO	0	44200
45000	CCCOCCCC	1000	45000
46000	CCCCOCCC	3000	45000
48000	OOOOOOOO	0	51000

Impact of Transmit Scheduling Algorithms on EST

When EST is used in isolation, the gate control list manages the final open/close state of the queues along with the checks carried out by the transmission selection algorithm in MTL. Because the gate controls operate on a predefined repetitive schedule, use the strict priority or Credit Based Shaper (CBS) scheduling algorithms.

Other algorithms such as the Weighted Round Robin (WRR), Deficit Weighted Round Robin (DWRR) and Weighted Fair Queuing (WFQ) implement masking of the queues based on the current winning queue. The algorithm is applied only among the group of queues that open simultaneously. To ensure queues whose gates are open get priority, these algorithms are modified to treat gate open queues and gate closed queues as separate groups giving priority to gate open queues.

For example, consider four queues (Q3, Q2, Q1, Q0) with weights 4:3:2:1; Q3 and Q2 are in open state in slot one slot, while Q1 and Q0 are in open state in another slot. In this case, the scheduler works as follows:

1. In the first slot, the Q3 and Q2 are serviced in the ratio of 4:3 for the duration that the slot is open.
2. In the second slot, the queues Q1 and Q0 are serviced in the ratio of 2:1.
3. Fresh arbitration is started every time a slot is opened.

The traffic does not get distributed in the intended ratio of 4:3:2:1, but rather as two groups with different ratios and only for the duration of the slot when the gates are open continuously.

Frame Preemption (FPE)

The IEEE 802.1Qbv-2015 specification defines the schedule for each queue on every egress port that makes the implementation aware of the traffic arrival schedule. This information can be used to block the lower priority traffic from transmission in this time window. Scheduled traffic is forwarded from the sender to the receiver through all the network nodes with a deterministic delay.

To achieve a low latency, ensure that there are no interfering frames during the scheduled windows that are reserved for high priority traffic. The use of scheduled traffic imposes limitations while starting a transmission.

If an interfering frame begins transmission just before the start of a reserved time period, it can extend transmission into the reserved window, and potentially interfere with higher priority traffic. Therefore, a guard band whose size is equal to the largest possible interfering frame is required before the window starts.

A larger guard band equates to a less efficient use of network bandwidth. However, with the implementation of IEEE802.1Qbu (frame preemption), this issue is addressed. Frame preemption breaks the interfering frame into smaller fragments. Therefore, the guard band needs to be only as large as the largest possible interfering fragment instead of the largest possible interfering frame.

During the guard band, only the frames that can complete the transmission of the entire frame before the next gate close event are permitted. This ensures that the high priority traffic can always start at the beginning of the window reserved for it.

NOTE: Internal loopback is not supported for preempted frames.

Preemption allows one or more higher priority (express) frames to interrupt the transmission of a lower priority (preemptable) frame; the preemptable frame transmission is resumed and completed after the express frame transmission is complete. To support frame preemption, the following abstractions of the MAC are used:

- A preemptable MAC, called pMAC, which carries the preemptable traffic
- An express MAC, called eMAC, which carries the express traffic

In the implementation, only the parts of the MAC that hold the state during preemption are replicated and represented as pMAC and eMAC. The MAC uses the following ways to halt the transmission of preemtable traffic in the presence of express traffic:

- The MTL scheduler interrupts the preemptable traffic that is currently being transmitted. When the preemption capability is active, the MAC interrupts the transmission and reception of preemptable frames. A preempted fragment can be followed by zero or more express frames before the continuation fragments. The preemptable frame can be fragmented any number of times. However, the minimum final and non-final fragment length criterion must be met.
    - Interleaving of more than one preemptable packet is not permitted. IWhen a preemptable packet is fragmented by an express packet, another preemptable packet cannot be transferred until all the remaining fragments of the first preempted packet are transferred.
- The MTL scheduler prevents starting the transmission of preemptable traffic. When the preemption capability is inactive, the pMAC entity is disabled and only express traffic is transmitted or received.

NOTE: Queue 0 can be an express queue when FPE is enabled, and EST is disabled.

Enabling Frame Preemption

Enable the frame preemption feature by setting the `EMAC_FPE_CTLSTAT.EFPE` bit.

GCL Modification to Support FPE

In the EST only configuration, the GCL entry has up to 24 bits of time interval, and up to 8 high order bits representing the gate open and close state requirements as shown in the ***Gate Control*** table.

Table 30-93: Gate Control - FPE Disabled

Gate Control (up to 8 Bits)	Time Interval (ns) 16, 20 or 24 Bits
OOCCCCCC	T0
OOOOCCCC	T1
CCCCOOCC	T2
CCCCCCOO	T3
…	…
…	…
OCOCOCOO	T126
OOOOCCCC	T127

EST only supports the *SetGate* operation, which implies that the gates are set to either open or close at a given time interval.

However, when both FPE and EST are enabled, the GCL also supports *Set-and-Hold-MAC* and *Set-and-Release-MAC* operations, in addition to the *SetGate* operation. To enable the support of hold and release operations, the format of the GCL is slightly changed while configuring and enabling the FPE. The first queue (Q0) is always programmed to carry preemption traffic. Therefore, it is always open. The bit corresponding to Q0 is renamed as hold and release MAC control.

The TX queues whose packets are preemptable are indicated by the EMAC_FPE_CTRLSTS.PEC field. The GCL bit of the corresponding queue is ignored and considered as always open. So, even if the software writes a 0 (C), it is ignored for such queues.

Table 30-94: Gate Control, FPE Enabled

Gate Control (up to 7 Bits)	Release/Hold Indication	Time Interval (ns) 16, 20 or 24 Bits
CCCOOOO	0	T0
CCCCOOO	0	T1
OCCCOOO	1	T2
COCCOOO	1	T3
CCOCOOO	1	T4
CCCOOOO	0	T5
CCCCOOO	0	T6

When the hold and release bit transitions from a 0 to 1, a *Set-and-Hold-MAC* operation is performed. This operation indicates the halting of the preemptable traffic. A hold request is sent to MTL in advance (the time interval is given in the EMAC_FPE_ADV.HADV field). When the hold and release bit transitions from a 1 to 0, a *Set-and-Release-MAC* operation is performed. This operation marks the resumption of the preemptable traffic. A release request is sent to MTL in advance (the time interval is given in the EMAC_FPE_ADV.RADV field). The preemptable traffic is blocked for the duration that the hold and release bit is set to a 1 in the GCL.

NOTE: The EMAC_FPE_ADV.HADV field is the advance time to initiate the hold state. Ideally, it should be programmed to a value of minimum fragment size. The size should be smaller than the time interval (minimum fragment size + IPG + preamble overheads in nanoseconds).

Impact of Preemption on CBS

In CBS, the definition of *transmit* is:

- True for the duration of frame transmission from the queue.
- False when frame transmission from the queue is complete.

When the CBS algorithm is used along with frame preemption, the value of *transmit* is true only while the MAC is transmitting the frame. If the frame transmission is delayed or interrupted, the value of *transmit* is false until transmission of the frame commences or is resumed. For example, a preemptable frame transmission is interrupted to allow the transmission of an express frame from a different queue. Or the start of express frame is delayed because a preemptable frame is being transmitted)

The value of *transmit* is false during the transmission of any overhead that is a consequence of frame preemption. For example, an additional frame overhead (mCRC, fragment count) that is added to the preemptable frame.

At any given time, if there are no frames in the queue, and the value of *transmit* is false, and credit is a positive value. The credit value is set to zero if there is no preemptable frame from the queue for which transmission is in-progress but has been interrupted.

mPacket Format

When the preemption capability is active, the MAC sends mPackets to the PHY. An mPacket can be one of the following:

1. An express packet
2. A preemptable packet
3. An initial fragment of a preemptable packet
4. A continuation fragment of a preemptable packet

The ***mPacket Format*** figure shows the format of the mPacket. It contains an express packet, a complete preemptable packet, or the initial fragment of a preemptable packet. Format B shows the format of an mPacket containing a continuation fragment of a packet.

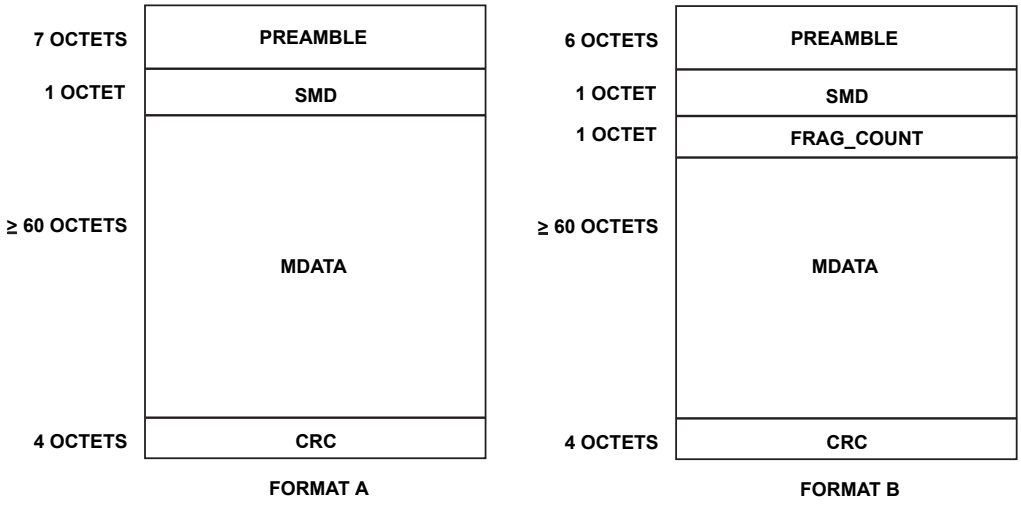


Figure 30-55: mPacket Format

Preamble

The preamble in the mPacket shown in ***mPacket Format*** contains seven octets. The preamble in the mPacket shown in ***mPacket Format*** (Format B) contains six octets. Each octet contains the value of 0x55 (transmitted in order from left to right 10101010).

Start mPacket Delimiter (SMD)

The value of the SMD indicates whether the mPacket contains an express packet, the start of a preemptable packet (initial fragment or complete packet), or any of continuation fragments of a preemptable packet. The ***SMD Values*** table shows the valid SMD values for different mPacket types.

Table 30-95: SMD Values

mPacket Type	Notation	Frame Count	Value
Verify packet	SMD-V	N/A	0x07
Respond packet	SMD-R	N/A	0x19
Express packet	SMD-E	N/A	0xD5
Preemptable packet start	SMD-S0	0	0xE6
	SMD-S1	1	0x4C
	SMD-S2	2	0x7F
	SMD-S3	3	0xB3
Continuation fragment	SMD-C0	0	0x61
	SMD-C1	1	0x52
	SMD-C2	2	0x9E
	SMD-C3	3	0x2A

frag_count

A frag_count is a modulo 4 counter that increments for each continuation fragment of the preemptable packet. The frag_count protects against mPacket reassembly errors by enabling the detection of the loss of up to three packet fragments.

The frag_count field is present only in mPackets with SMD-C notation (continuation fragment). The frag_count is zero in the first continuation fragment of each preemptable packet. The ***frag_count Values*** table shows value associated with the frag_count.

Table 30-96: frag_count Values

frag_count	Value
0	0xE6
1	0x4C
2	0x7F
3	0xB3

mData

The contents of the packet from the MAC. It includes the first byte after the SFD to the last byte before the FCS. The contents are sent in the mData fields of one or more mPackets for that frame. The minimum size of the mData field is 60 bytes.

CRC

The CRC field contains a Cyclic Redundancy Check (CRC) and has an indication of the final mPacket of a frame. In the final mPacket of a frame, the CRC field contains the last four octets of the MAC frame (the FCS field).

For other mPackets, the CRC field contains an mCRC value. The mCRC is calculated on the octets of the packet from the first octet of the frame (the octet following the SFD of preemption frames) to the last octet of the packet transmitted in that mPacket. The check by performing an XOR of the calculated 32 bit CRC value of the fragment and a value of 0x0000FFFF.

Summary of Packet Formats

- Express packet – 7 bytes of preamble, SMD-E, data, and CRC
- Complete preemptable packet – 7 bytes of preamble, current preemptable packet SMD, data, CRC
- Initial fragment (non-final) of preemptable packet– 7 bytes of preamble, current preemptable packet SMD, data, mCRC
- Continuation fragments (non-final) of preemptable packet – 6 bytes of preamble, current preemptable continuation fragment SMD, current preemptable continuation fragment FC, data, mCRC
- Final fragment of preemptable packet – 6 bytes of preamble, current preemptable continuation fragment SMD, current preemptable continuation fragment FC, data, CRC

When frame preemption is enabled, the MAC receiver communicates the exceptions related to received preempted fragments (incorrect fragment sequencing, missed fragment, and so on) to MTL layer by reporting it as a CRC error. Software should not set (=1) the `EMAC_MAC_EXT_CFG.DCRCC` bit. Otherwise, the MTL layer might forward the unintended preempted fragments to the application.

Table 30-97: Current and Previous SMD Values

Previous Preemptable Packet SMD	Current Preemptable Packet SMD
SMD-S0	SMD-S1
SMD-S1	SMD-S2
SMD-S2	SMD-S3
SMD-S3	SMD-S0

Table 30-98: Current and Previous SMD Values

Previous Preemptable Fragment SMD	Previous Preemptable Fragment FC	Current Preemptable Continuation Fragment SMD	Current Preemptable Continuation Fragment FC
SMD-S0	N/A	SMD-C0	FC0
SMD-S1	N/A	SMD-C1	FC0
SMD-S2	N/A	SMD-C2	FC0

Table 30-98: Current and Previous SMD Values (Continued)

Previous Preemptable Fragment SMD	Previous Preemptable Fragment FC	Current Preemptable Continuation Fragment SMD	Current Preemptable Continuation Fragment FC
SMD-S3	N/A	SMD-C3	FC0
SMD-C0	FC0	SMD-C0	FC1
SMD-C0	FC1	SMD-C0	FC2
SMD-C0	FC2	SMD-C0	FC3
SMD-C0	FC3	SMD-C0	FC0
SMD-C1	FC0	SMD-C1	FC1
SMD-C1	FC1	SMD-C1	FC2
SMD-C1	FC2	SMD-C1	FC3
SMD-C1	FC3	SMD-C1	FC0
SMD-C2	FC0	SMD-C2	FC1
SMD-C2	FC1	SMD-C2	FC2
SMD-C2	FC2	SMD-C2	FC3
SMD-C2	FC3	SMD-C2	FC0
SMD-C3	FC0	SMD-C3	FC1
SMD-C3	FC1	SMD-C3	FC2
SMD-C3	FC2	SMD-C3	FC3
SMD-C3	FC3	SMD-C3	FC0

Transmit Preemption

To support preemption, the MAC should have more than one transmit queue with at least one queue designated as express queue (and one queue designated as preemption queue). When FPE is enabled (`EMAC_FPE_CTLSTAT.EFPE` =1), the MTL preempts a preemptable frame. When a hold request is asserted (EST/Qbv configured and enabled), express frames are available for transmission. The frame is present in the MTL FIFO and qualified for arbitration after ensuring that the minimum mPacket mData field size is met.

Therefore, preemption occurs only when at least 60 bytes of the preemptable frame have been transmitted and at least 64 bytes (including the frame CRC) remain.

The earliest starting position of preemption is controlled by the `EMAC_FPE_CTRLSTS.AFSZ` bit. Preemption does not occur until at least 64 x (1 + AFSZ) - 4 bytes of the preemptable frame have been sent.

When preemption occurs, all of the preemptable queues are blocked; only the express queues can arbitrate (if more than one express queue has traffic) and transmit.

The continuation fragment of the preempted frame is the first frame to be transmitted after the release request is asserted (EST/Qbv configured and enabled) and all the express traffic transmission completes.

NOTE: All the PTP packets should be transmitted as express packets.

MTL communicates the following frame-type information to the MAC using a 2-bit preemption control signal on the MTI interface (qualified with SoF and EoF). Refer to the ***MTI Preemption Control Values*** table.

- Express frame
- Preemption frame (full or fragment)
- Continuation fragment (non-final or final)

Table 30-99: MTI Preemption Control Values

Qualifier	Preemption Control Values	Frame Types
SoF	00	Start express
SoF	01	Start preemption
SoF	10	Continuation fragment
SoF	11	Reserved
EoF	00	End of frame
EoF	01	End of fragment

The MTL must wait for the previous fragment status to be received before resuming the continuation fragments of a preempted frame.

MAC Transmit Preemption

The MAC supports preemption by implementing the functionality needed to generate the mPackets. The functionality is based on the preemption control value received on the MTI interface (qualified with SoF and EoF). The MAC determines the frame type (see the ***MTI Preemption Control Values*** table) and generates mPackets accordingly. When the preemption capability is active, the MAC replaces the SFD of a preemption packet with an SMD-S value. A 2-bit rolling frame count is encoded in the SMD-S value. The SMD-E value is the same as the SFD value. So, the SFD of an express packet does not need to be replaced.

Transmit Fragment Status

The MAC sends the transmit fragment status to indicate a successful transmission of fragmented mPackets. To indicate a fragment status, bit 31 of the mti_txstatus_o field is set (=1) and all other values in the status field are zero.

For a transmission error (for example, an underflow or jabber), the frame status is sent (and not a fragment status) with an error indication along with all other relevant status fields. For a receiving an error status, the MTL drops the remaining fragments and does not send any more continuation fragments.

Receive Preemption

When FPE is enabled, the MAC receiver passes the incoming packets and differentiates between express packets and preemptable packets. An SMD containing an SMD-E indicates an express packet; an SMD containing an SMD-S indicates the first mPacket of a preemptable packet.

If an mPacket containing an SMD-S is received when the MAC has not completely received the previous preempted packet, MAC sets a CRC error status for the previously received partial packet.

When an SMD-S is detected, the MAC records the frame count indicated by the SMD and then begins sending data on the MRI interface.

The MAC checks the last four bytes of the mPacket. If the last four bytes of the mPacket do not match CRC, it indicates the end of the packet with or without a CRC error (CRC check result). If the last four octets of the mPacket match, it indicates that the packet was preempted.

An SMD containing an SMD-C indicates an mPacket that continues the data for a preempted packet. Upon receiving an SMD value of SMD-C, the MAC checks the following:

1. A preempted packet is in progress
2. The frame count indicated by the SMD matches the frame count of the packet in progress
3. The frag_count value indicates the next fragment count

If any of these checks fail, the mPacket is discarded and MAC sets a CRC error status for the partially received packet.

If all the checks pass, the next fragment count is incremented modulo 4.

When a packet is preempted, the MAC saves the state of the partially received packet (filter check status, time stamp, length fields and so on) and can process any received express packets before the continuation fragment is received.

The MAC receiver sends a dummy status for all of the mPacket fragments that are successfully received. It sends the Rx status with the final fragment. If an error is detected during any of the fragments, the Rx status is sent, and the fragment is marked as final fragment. All subsequent continuation fragments received for this packet are dropped in the MAC.

The MAC communicates the following frame type information to the MTL using a 2-bit preemption control signal on the MRI interface. Refer to the ***MRI Preemption Control Values*** table.

1. Express frame
2. Preemption frame (full or fragment)
3. Continuation fragment (non-final or final)

Table 30-100: MRI Preemption Control Values

Qualifier	Preemption Control Values	Frame Types
SoF	00	Start express
SoF	01	Start preemption
SoF	10	Continuation fragment
SoF	11	Reserved

Table 30-100: MRI Preemption Control Values (Continued)

Qualifier	Preemption Control Values	Frame Types
EoF	00	End of frame
EoF	01	End of fragment

Data Alignment

When a received frame cannot be fragmented on any byte boundary, the MAC retains the unaligned bytes of data in the previous fragment and resends them with the next fragment. See the ***MAC Data Alignment*** figure.

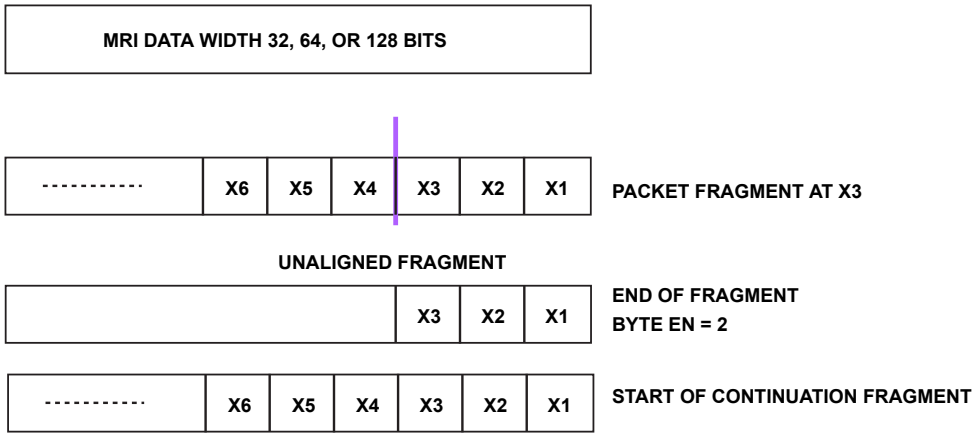


Figure 30-56: MAC Data Alignment

The MTL can choose to ignore the partial data (based on byte enable value) that comes with the end of the fragment and instead use the data width aligned data received in the first beat of the next continuation fragment.

MTL Receive Preemption

The MTL Rx must have at least two receive queues to support FPE function because the preemptable packets and the express packets must be routed to separate queues. The `EMAC_RXQ_CTL0` registers control the destination Rx queue of a received packet. Program these registers such that preemptable traffic and express traffic are not routed to the same receive queue. The queue mapping for tagged packets is based on VLAN user-priority field. So, the priority of preemptable and express packets are mutually exclusive. Packets of a certain priority (traffic class) are either express or preemptable, but not both.

For the preemptable frames, in addition to the Priority Selected in Rx Queue (PSRQ) routing, a programmable Frame Preemption Residue Queue (FPRQ) is supported. This feature routes all other preemption packets received (untagged, SA/ DA or VLAN filter fails forwarded due to RA being set or VTFE being reset). Refer to the ***MTI Preemption Routing*** table.

Table 30-101: MRI Preemption Routing

Preemption Packet Type	Queue Routing
AV tagged packets passing the SA/DA and VLAN filters	PSRQ*
DCB/generic tagged packets passing the SA/DA and VLAN filters	PSRQ**
Tagged packets failing the SA/DA and VLAN filters without setting the receive all (RA = 0) or setting VTFE =1	Dropped
All other packets or when RA = 1	FPRQ

The express frames use queue 0 as a default queue. In case of filter failures, queue 0 must not be used for receiving preemption frames.

When a fragment is received, it is pushed into its destination receive queue. When this packet is preempted, and is followed by an express packet, it is pushed/stored in a different receive queue. The MTL saves the context of the preempted frame, and processes one or more express frames. When the continuation fragment arrives, it restores the saved context and pushes the remainder of the fragmented frame into its receive queue.

MTL Receive Arbitration

On the ARI interface, data is fetched from the MTL receive FIFO based on the arbitration selected in the EMAC_MTL_OPMODE register. Frame-based arbitration can be used only when all the MTL preemption queues operate in store-and-forward mode. Otherwise, there is a loss of bandwidth on the ARI interface because all of the fragments of a preemptable packet are not available. Therefore, express packets received between the fragments are blocked until all of the fragments are received and transferred; this defeats the purpose of express traffic.

When operating in either threshold (cut through) or store-and-forward modes, PBL-based arbitration is recommended over frame-based arbitration. For PBL-based arbitration, the watermark check is always performed. The arbitration/transfer of data is in chunks of PBL size of data (similar to the concepts of fragments). Therefore, the express queue packets are blocked for less time and the ARI interface transfers the data without a loss of efficiency.

Frames Support in Offload Engines

When frame preemption is enabled, all the packets processed by the offload engines (ARP, PTO) and PTP, PAUSE, and PFC packets must be received as express/normal packets. These offload engine logic and functions do not recognize preemption frames.

Verify and Respond mPackets

When FPE function is present, the MAC can receive and detect the verify and respond mPackets, even when FPE is not enabled by software. When the MAC detects valid verify and respond mPackets, it notifies the software by configuring the EMAC_FPE_CTLSTAT.RVER and EMAC_FPE_CTLSTAT.RRSP fields, respectively. Optionally, an interrupt can be generated. These packets have empty (all-zero) data payload; they are dropped inside the MAC and not forwarded to the MRI.

Software can configure the EMAC_FPE_CTLSTAT.SRSP and EMAC_FPE_CTLSTAT.SVER fields to request the MAC to transmit verify and respond mPackets, respectively. Upon successful transmission of these frames, the MAC clears the EMAC_FPE_CTLSTAT.SRSP / EMAC_FPE_CTLSTAT.SVER bit and sets the EMAC_FPE_CTLSTAT.TVER/EMAC_FPE_CTLSTAT.TRSP bits. Optionally, an interrupt can be generated when these events occur.

NOTE: 1. For preemption frames received with a CRC error, ignore the following status fields:
   - Packet length
   - Dribble error indication
   - Runt frame indication
2. When FPE is configured, the EMAC_RQ0_OPMODE.FUP (RXQ(n)) bit should not be set.
3. When FPE is configured, do not disable CRC checking upon receive. Internally, FPE error fragments are identified using CRC checks. For details, see the EMAC_MAC_EXT_CFG.DCRCC field.
4. Express and preemption traffic must be routed to different DMAs.

MMC Counter and Interrupt Registers

The following MMC counters and associated interrupt registers are instantiated/present in the MAC when the frame preemption feature and any one of the existing MMC counters are selected during RTL configuration.

Table 30-102: MMC Counters and Interrupt Registers

Frame Assembly Error Counter	Description	Associated MMC Counter
Frame assembly error counter	A 32-bit counter that provides the number of MAC frames with reassembly errors on the receiver, due to a mismatch in the fragment count value	EMAC_RXPKT_ASERR_CNT
Frame SMD error counter	A 32-bit counter that provides the number of MAC frames rejected due to arriving with an SMD-C when there was no preceding preempted frame	EMAC_RXPKT_SMDERR_CNT
Frame assembly OK counter	A 32-bit counter that provides the number of MAC frames that were successfully reassembled	EMAC_RXPKT_ASOK_CNT
MAC Rx fragment counter	A 32-bit counter that provides the number of additional mPackets received due to preemption.	EMAC_RXFPE_FRGMNT_CNT
MAC Tx fragment counter	A 32-bit counter that provides the number of additional mPackets transmitted due to preemption.	EMAC_TXFPE_FRGMNT_CNT

Table 30-102: MMC Counters and Interrupt Registers (Continued)

Frame Assembly Error Counter	Description	Associated MMC Counter
Hold request counter	A 32-bit counter that maintains the count of the number of hold requests given to the MAC	EMAC_TXHLDRQ_CNT

Additional Registers Associated with MMC Interrupts

Following are the additional registers associated with MMC interrupts for the MMC error counters:

- EMAC_MMC_FPE_TXINT
- EMAC_MMC_FPE_TXIMSK
- EMAC_MMC_FPE_RXINT
- EMAC_MMC_FPE_RXIMSK

Time-Based Scheduling

The time-based scheduling feature is suitable for traffic whose periodicity and rate are predictable. This feature is optional. To improve the quality-of-service of such traffic:

- The transmit DMA fetches the packet from the host memory for transmission at designated time. This method helps the software to set up the transmit descriptors before the packet is ready/available. It reduces the overhead on the software and avoids constant monitoring of the time. It prepares the descriptors for the targeted transmission time of the packet.
- The MAC transmits the packet only at the designated/pre-determined time even when the packets are fetched in advance. This method helps in maintaining a constant transmission rate that can be consumed by the receiver station; therefore, it avoids congestion and excessive buffering in the network.

The time-based scheduling feature supports fetching and launching an Ethernet packet at (or after) a pre-determined time. The time-based scheduling is supported only in the following modes:

- Full duplex mode
- Link speed is 100 Mbps or higher

NOTE:
- Do not enable time-based scheduling for the channel for which TSO feature is enabled.
- In EST enabled configurations, the EMAC_TBS_CTRL.ESTM bit impacts the time-based scheduling feature only when the DUT processes the GCL list.
- Slot number is meaningful only after GCL is active. So, start the GCL list before enabling the TBS and EST features. If GCL is not started, all gates are deemed open. If packets do not make progress, the Tx queue creates back pressure.

TBS Definitions

The following definitions apply to time-based scheduling:

- Launch time - the time beyond which MTL can schedule the packet for transmission
- Fetch time - the time beyond which the Tx DMA can schedule a packet-fetch from the host memory
- Expiry time - the time beyond which the packet is dropped by MTL

Launch Time

The launch time is specified in the enhanced normal descriptor in the DMA configurations. For more details, see Enhanced Descriptor for Time-Based Scheduling. The launch time can also be specified as a control word for each packet. When the launch time is specified in the DMA configuration, it is valid only for the specific frame that is scheduled for transmission. The launch time is reset after the transmission of one frame.

The formats of the launch time include:

- Normal/absolute format
- EST/offset format

In normal/absolute format, the launch time is an absolute time value at which the packet is launched for transmission. The launch time is interpreted in the normal/absolute format when `EMAC_TBS_CTRL.ESTM` = 0.

The launch time is a 32-bit value, where the most-significant 8-bits represent the time in seconds and the rest 24 bits represent the time in 256 ns. The launch time is compared against the IEEE 1588 based system/PTP time (bits [39:8]) and rolls over after 256 seconds.

The maximum value of the lower 32-bits of system time is 999,999,999 decimal (0x3B9AC9FF). It wraps to zero when reaching this value (representing a full second). Therefore, the maximum value of the lower 24 bits of the launch time (after multiplying by 256) must be 0x3B9AC9.

Because the maximum value of launch time is 256 seconds:

- Launch time is greater than current system time when its value is between system time[39:8] and system time[39:8] + 128 s
- Launch time is less than current system time when its value is between system time[39:8] + 128 s and system time[39:8] + 256 s. This computation is modulo 256.

In EST/offset format, the launch time is an offset value relative to the time indicated by the BTR of the GCL list provided in the GCL Slot Number (GSN). The value in the BTR is always updated to the start time of the current loop of the GCL. For each packet, the GSN value and the launch time value are specified in the enhanced transmit descriptor.

The launch time offset is a 32-bit value; the upper 8-bits represent the time in seconds and the rest 24 bits represent the time in 256 ns, which is added to the BTR value corresponding to GSN. The value of the launch time offset must be smaller than the value of the cycle time (specified in the CTR register that is implemented when EST is enabled). If the CTR is greater than or equal to 1s, the maximum value of the lower-24 bits of launch time offset should be 999,999,999 decimal (0x3B9AC9FF).

In EST/offset format, the launch time is a 64-bit value, which is interpreted as:

Launch time = Launch GSN BTR[63:0] + (Launch time offset[31:0] << 8), which is compared with the system time [63:0].

where:

GSN BTR – the base time value at which the launch GSN loop started execution.

GSN – A modulo 16 count of the GCL loop count. The count is incremented for every new GCL loop; installation of new GCL list does not impact the count. The current GCL slot number can be obtained by reading the EMAC_EST_STAT.CGSN field. The maximum value of GSN is 15. Therefore, GSN values between the EMAC_EST_STAT.CGSN field and CGSN+8 represent current or future slots; all other GSN values are interpreted as elapsed slots or past slots. So, for the correct interpretation of time, GSN value must be between EMAC_EST_STAT.CGSN and CGSN+8 for the correct interpretation of time.

Launch Expiry Time

In the normal/absolute format, when EMAC_TBS_CTRL.LEOV = 1, the EMAC_TBS_CTRL.LEOS bit determines the maximum amount of time a frame is eligible for launch, starting from the time the frame becomes eligible for launch.

The launch expiry offset is a 24-bit value defined in 256 ns units, with a maximum possible value of 999,999,999 ns (0x3B9AC9FF).

Launch expiry time = (launch time[39:8] + EMAC_TBS_CTRL.LEOS [32:8]) * 256 ns

The packet with a specific launch time is considered eligible for transmission when the launch time is less than the system time and (if EMAC_TBS_CTRL.LEOV = 1) the system time is less than the launch expiry time.

When the system time is greater than the launch expiry time the frame is categorized as expired and is dropped from the MTL FIFO.

NOTE:
1. For correct interpretation and meaningful operation, the fetch, launch, and launch expiry time should never be set to a value larger than current system time + 128 s; such a value is interpreted as time that has already elapsed.
2. In full duplex mode, the frames dropped from the MTL FIFO have an error summary (bit 15) and Excessive Deferral (bit 3) of the TxStatus set.

In the EST/offset format, when EMAC_TBS_CTRL.LEOV = 1, the launch expiry GSN offset (EMAC_TBS_CTRL.LEGOS) and launch expiry offset (EMAC_TBS_CTRL.LEOS bits determine the maximum amount of time a frame remains eligible for launch, starting from the time the frame becomes eligible for launch.

The launch expiry offset is computed as:

Launch expiry offset = (EMAC_TBS_CTRL.LEGOS: EMAC_TBS_CTRL.LEOS)

The launch expiry offset is a 24-bit value defined in the units of 256 ns, with a maximum possible value of the smaller of 999,999,999 ns or CTR-1 ns. The EMAC_TBS_CTRL.LEGOS field holds the GSN offset (multiples of CTR time). The EMAC_TBS_CTRL.LEOS field holds maximum value of CTR (sub-CTR values) in ns.

The launch expiry GSN is computed as:

Launch expiry GSN = (launch GSN + EMAC_TBS_CTRL.LEGOS + CCMA)

where,

CCMA - the CTR carry due to modulo addition. This value is 1 if ((launch time offset + EMAC_TBS_CTRL.LEOS) << 8) ≥ CTR.

- When CCMA = 1, launch expiry offset = (launch time offset+ EMAC_TBS_CTRL.LEOS) - CTR
- When CCMA = 0, launch expiry offset = (launch time offset + EMAC_TBS_CTRL.LEOS).

Launch expiry time = launch expiry GSN BTR[63:0] + launch expiry time offset

- When EMAC_TBS_CTRL.LEOV = 0, the launch expiry time is not checked.
- When EMAC_TBS_CTRL.LEOV =1, and
  - The system time is greater than the launch expiry time, the frame is dropped from MTL FIFO. The frame is considered as expired.
  - The launch time is smaller than the system time and launch expiry time is greater than system time, the frame is considered eligible for launch.

NOTE:

1. The max value of EMAC_TBS_CTRL.LEGOS is 7. This implies that when EMAC_TBS_CTRL.LEOV = 1, the frame has a maximum lifetime of < 8 GCL loop iterations after it becomes eligible for launch.
2. The slot number of the first GCL list executed each time after EST is enabled, is zero.

Fetch Time

Software can indicate the fetch time for each packet (EMAC_DMA_TBS_CTL0.FTOV =1). When EMAC_DMA_TBS_CTL0.FTOV = 0, the fetch time offset is not valid and the DMA fetches packets without any time constraints. The fetch time accounts for all possible delays in the DMA fetch operation and ensures that the frame is present in the MTL FIFO before the launch time.

In normal/absolute mode, fetch time is derived/calculated by reducing the time specified in the EMAC_DMA_TBS_CTL0.FTOS field from the given launch time. In a normal mode of operation, the fetch launch time is computed as:

Fetch time[39:8] = (launch time[39:8] - EMAC_DMA_TBS_CTL0.FTOS[31:8])

The fetch time is 32 bits. It is compared against system time[39:8] to determine the eligible for fetching the frame:

- The fetch time is defined as greater than system time when the fetch time is in the range of system time[39:8] and system time[39:8] + 128 s. The frame is considered ineligible for fetch.
- The fetch time is defined as smaller than the system time when the fetch time is in the range of system time[39:8] + 128 s and system time[39:8] + 256 s. The frame is considered as eligible for fetch.

This computation is modulo 256 .

In the EST/offset mode, theEMAC_DMA_TBS_CTL0.FGOS field provides the slot number offset to be deducted from the launch GSN. In this case, the EMAC_DMA_TBS_CTL0.FTOS value should be the smaller of 999,999,999 ns or CTR-1 ns.

When (launch time offset ≥ EMAC_DMA_TBS_CTL0.FTOS):

- Fetch time offset = ((launch time offset - EMAC_DMA_TBS_CTL0.FTOS) x 256 ns)
- CBFS (CTR borrow for fetch subtraction) = 0

When (launch time offset < EMAC_DMA_TBS_CTL0.FTOS):

- Fetch time offset = CTR + ((launch time offset - EMAC_DMA_TBS_CTL0.FTOS) x 256 ns)
- CBFS = 1

The fetch GSN is computed as follows:

Fetch GSN = launch GSN - EMAC_DMA_TBS_CTL0.FGOS - CBFS

Fetch time = Fetch GSN BTR[63:0] + Fetch time offset

The frame is marked eligible for DMA fetch when the fetch time is smaller than the system time.

The max value of EMAC_DMA_TBS_CTL0.FGOS is 7. This implies that when EMAC_DMA_TBS_CTL0.FTOV =1, the frame can be fetched at a maximum of <8 GCL loop iterations before it becomes eligible for launch.

DMA Operations Sequence

When EMAC_DMA_TBS_CTL0.FTOV = 1, the following DMA operations sequence occurs:

1. Fetch the first enhanced normal descriptor. (FD is set).
2. If LTV is set and fetch enabled, compute the fetch time based on the launch time. Wait for the system time to be greater than fetch time.
3. Read the frame (data) from the host memory and transfer to MTL FIFO.
4. Close the normal descriptor.
5. Fetch the next normal descriptor (if the previous descriptor was not the last).
6. Repeat steps 4 to 6, until the last descriptor of the frame (LD is set).

After the last descriptor of a frame, the next enhanced normal descriptor should be programmed with a new launch time and with the LTV bit set. Otherwise, the subsequent frames are processed without any time restrictions.

Control Word

Time-based scheduling ensures that a packet is allowed to arbitrate for transmission only when the system time is greater than the launch time.

- Launch time and GSN use the same field that holds the 64-bit time stamp value for One Step Timestamping Correction (OSTC). When the LTV bit is set and OSTC bit is not set (bit 19 of the first control word), the time stamp field is interpreted as the launch time. Launch time uses the fields meant for OSTC time stamp[39:8]; GSN uses the OSTC time stamp[43:40] field.
- Bit 31 of control word 0 is the equivalent of the LTV field. The last two Control D words provide the launch time and GST.
- The last two control D words are the time stamp for OSTC when bit 19 of control word 0 (OSTC) is set.

When the MTL TxFIFO read controller reads the first control word, it checks bit 31 to determine if the frame has a valid launch time. If bit 31 is set, it reads the subsequent control words to get the launch time.

Impact on Transmission Selection Algorithm

The time-based scheduler does not directly influence the transmission selection algorithm. However, it has an indirect influence as it determines whether a queue is eligible to participate in the scheduling.

When a frame of a specific queue has a valid launch time, the time-based scheduler ensures that the frame participates in transmit scheduling (TRC scheduler) only after the launch time has elapsed.

The gating done by time-base scheduler is only for the scheduling (picking frame for transmission). It might not alter any other characteristics and behavior of the frame. For example, the frame waiting for the launch time to lapse continues to gain credits in CBS. The CBS credits are not lost because the frame is marked as available, but not ready. Strict priority or CBS are the recommended transmit scheduling algorithms for the TBS implementation. Scheduling for TBS-enabled queues is more predictable.

The ***Time-Based Scheduling Flow*** figure shows the time-based scheduling flow.

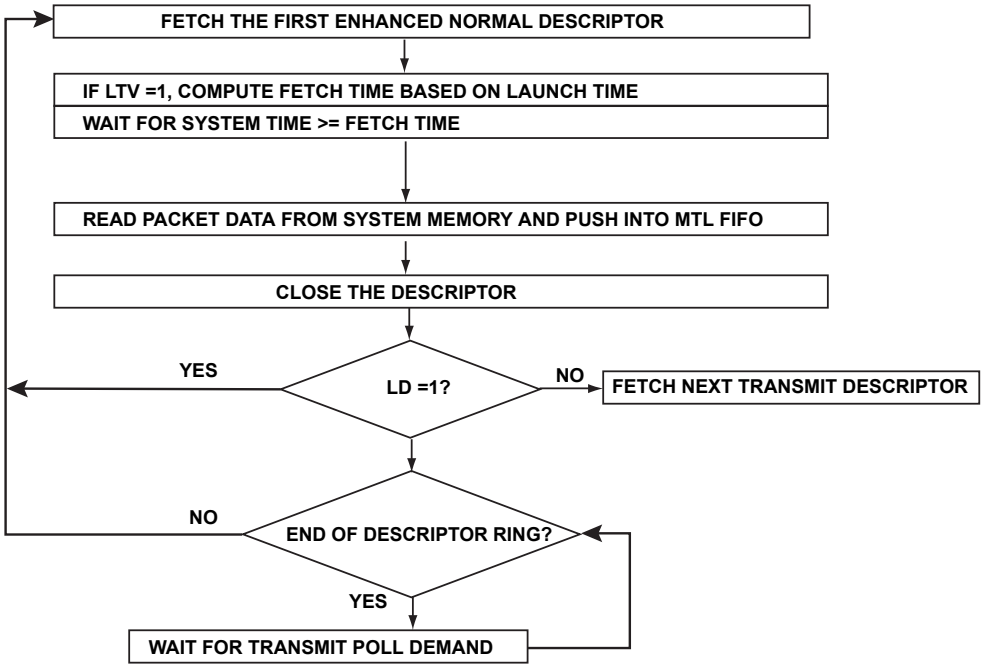


Figure 30-57: Time-Based Scheduling Flow

Flow Control

This section describes the flow control feature of EMAC for transmit and receive paths.

Transmit Flow Control

The transmit flow control involves transmitting pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

Flow Control in Full-Duplex Mode

In full-duplex mode, the EMAC uses one of the following packet types for flow control:

- IEEE 802.3x pause packets
- Priority flow control (PFC) packets

The `EMAC_RXFLOW_CTL.PFCE` bit determines whether PFC packets or IEEE 802.3x pause control packets are used for flow control. If the PFCE bit is set, the MAC sends the PFC packets.

Pause Packet Structure

The *Pause Packet Fields* table describes the fields of a pause packet.

Table 30-103: Pause Packet Fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control Opcode	Contains 0001 for IEEE 802.3x Pause Control packets; 0101 for PFC packets
PT	Contains Pause time specified in the PT field of the MAC_Q#_Tx_Flow_Ctrl register

Pause Packet Control

When the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit is set, the MAC generates and transmits a single pause packet. If the FCB bit is set again after the pause packet transmission is complete, the MAC sends another pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously transmitted pause packet, the application should program the pause time register with appropriate value and then again set the FCB_BPA bit.

Similarly, when the flow control signal from FIFO layer (mti_flowctrl_i) is asserted, the MAC generates and transmits a single pause packet. If the mti_flowctrl_i signal remains asserted at a configurable number of slot times before the pause time runs out, the MAC transmits a second pause packet. This process is repeated if the mti_flowctrl_i signal remains active. If the mti_flowctrl_i signal goes inactive prior to the sampling time, the MAC transmits a pause packet with zero pause time (if the bit is set to 0) to indicate to the remote end that the receive buffer is ready to receive new data packets. For PFC packets, the priority, pause time, and other controls can be specified for a queue in the EMAC_Q[i]_TXFLOW_CTL register. The MAC supports independent flow control for each receive queue. There is one trigger input for each receive queue. Therefore, when multiple triggers come simultaneously, the MAC sends one PFC packet for each trigger. Based on the hardware trigger or software trigger through respective queues, the MAC sends a PFC packet with programmed pause time and priority. If multiple priorities are programmed in same receive queue, multiple priorities are set for the PFC packet with same pause time values. A separate pause timer is available for each receive queue.

Flow Control in Half-Duplex Mode

In half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

The *Transmit MAC Flow Control* table describes the flow control in the transmit path for queue 0 based on the setting of the following bits:

- EMAC_RQ0_OPMODE.EHFC

- EMAC_Q[i]_TXFLOW_CTL.TFE
- EMAC_MAC_CFG.DM

Flow control is similar for all queues.

Table 30-104: Transmit MAC Flow Control

EHFC	TFE	DM	Description
x	0	x	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs back-pressure when bit 0 of the EMAC_Q[i]_TXFLOW_CTL register is set or the sideband signal sbd_flowctrl_i = 1.
1	1	0	The MAC transmitter performs back-pressure when bit 0 of the EMAC_Q[i]_TXFLOW_CTL register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx performs back-pressure when the Rx queue level crosses the threshold set by bits[10:8] of the EMAC_RQ0_OPMODE register.
0	1	1	The MAC transmitter sends the pause packet when bit 0 of the EMAC_Q[i]_TXFLOW_CTL register is set or the sideband signal sbd_flowctrl_i = 1.
1	1	1	The MAC transmitter sends the pause packet when bit 0 of the EMAC_Q[i]_TXFLOW_CTL register is set or the sideband signal sbd_flowctrl_i = 1. In addition, the MAC Tx sends a pause packet when the Rx queue level crosses the threshold set by bits[10:8] of the EMAC_RQ0_OPMODE register.

Enabling Transmit Flow Control

To independently enable transmit flow control for each receive queue, set the EMAC_Q[i]_TXFLOW_CTL.TFE bit.

Triggering Transmit Flow Control

Transmit flow control involves transmitting pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end. The application can request the MAC to send a pause packet or initiate back-pressure by using either of the following methods:

- Software Trigger: In this method, the application sets the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit.
- Hardware Trigger: In this method, the application triggers the flow control by asserting the mti_flowctrl_i signal based on the receive queue threshold.
- Receive Queue Threshold: The flow control operation of the MAC is enabled when the EMAC_RQ0_OPMODE.EHFC bit is set. The flow control signal to the MAC is asserted when the fill level of the receive queue crosses the threshold configured in the EMAC_RQ0_OPMODE.RFA bit field. This flow control signal is de-asserted when the fill-level of the queue is lower than the threshold configured in the EMAC_RQ0_OPMODE.RFD field.

The hardware flow control generated based on the receive queue threshold crossing condition is applicable only when the receive queue size is 4,096 bytes or more.

NOTE: The RFA sets the threshold at which the flow control is triggered and the MAC schedules to send a PAUSE frame to be sent by the local transmitter.

The remote transmitter can continue to send packets until it receives this PAUSE packet. So, the RXQ needs space to take in this data even after the flow-control is triggered, to avoid overflow and loss of packets.

The max-space required that is calculated theoretically can be 2 max-sized packets + a little more when the read from the RXFIFO is not occurring during this whole time.

Theoretically, this makes RFA = 4 or more, which implies that RxQ must have at least a size of 4KB.

Practically, the system works with RFA = 2 (Full – 2KB, with little probability of overflow). When the application has a FIFO size of less than 4K, with RFA = 2, the flow control is triggered with 1000B in the RxQ, which is less than maximum packet size.

The `EMAC_RQ0_OPMODE.RFA` bit field is used for activating the flow control. These bits control the threshold (fill-level of Rx queue) at which the flow control is activated.

For example, if the RxQ size is 2K and the program sets the `EMAC_RQ0_OPMODE.RFA` bit field to 1K; as soon as 1K is received, flow control is activated. If the program sends a packet of 1500 bytes (which is allowed as per spec), flow-control is triggered even before a complete packet is received. This reduces the throughput considerably.

When the RxQ size is 4K, the program can set the RFA to 2K, and the whole packet can be received.

NOTE: Receive pause packets should have a frame size of 64 bytes.

Receive Flow Control

Receive flow control is implemented by the MAC based on the bit value of the respective register, and the destination address and different fields of the received packet.

The *Receive MAC Flow Control* table describes the flow control in the receive path based on the setting of the `EMAC_RXFLOW_CTL.RFE` and `EMAC_MAC_CFG.DM` bits.

Table 30-105: Receive MAC Flow Control

RFE	DM	Description
0	X	The MAC receiver does not detect the received pause packets
1	0	The MAC receiver does not detect the received pause packets but recognizes such packets as control packets.
1	1	The MAC receiver detects or processes the pause packets and responds to such packets by stopping the MAC transmitter.

For EMAC0, the program can enable the PFC packet detection by setting the EMAC_RXFLOW_CTL.PFCE bit. If PFC packet detection is enabled, the MTL transmit queue corresponding to the received priority is blocked when the PFC packet is received. If PFC packet detection is not enabled in configurations with multiple queues and the RFE bit is enabled, the MAC transmitter is blocked when the 802.3x pause packet is received.

The following list describes the receive flow control.

1. The MAC checks the destination address of the received pause packet for either of the following:
    - Multicast destination address: The DA matches the unique multicast address specified for the control packet (48'h0180C2000001).
    - Unicast destination address: The DA matches the content of the MAC address register 0 and the EMAC_RXFLOW_CTL.UP bit. If the UP bit is set, the MAC processes pause packets with a unicast destination address in addition to the unique multicast address.
2. The MAC decodes the following fields of the received packet:
    - Type field: This field is checked for 16'h8808.
    - Opcode field: This field is checked for 16'h0001 (pause packet) or 16'h0101 (PFC packet).
    - Pause Time or Pause Time Vector field: The pause time (for pause packet) is captured to determine the time for which the transmitter needs to be blocked. The Pause Time Vector field (for a PFC packet) is captured to determine the time for which the MTL transmit queue corresponding to the received priority needs to be blocked.
    - Priority Enable Vector field: This field is valid only for PFC packets. It is captured to determine the MTL transmit queue corresponding to the received priority.
3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter does one of the following:
    - For 802.3x Pause packets, the MAC pauses the transmission of any data packet for the duration of the decoded pause time value multiplied by the slot time (64-byte times).
    - For PFC packets, the MAC blocks the transmit queues corresponding to the priority. Based on the priorities assigned to the transmit queues in the EMAC_TXQ_PRTY_MAP0 and EMAC_TXQ_PRTY_MAP1 registers, the MAC asserts the respective bit in the mti_disable_txq_o signal and loads the pause timer corresponding to the priority. The transmit queue corresponding to the priority is blocked until the pause timer expires.
    - In addition to the PFC-based flow control operation, when the PFCE bit is set, all bits of the mti_disable_txq_o signal are asserted while in LPI mode so that the ETS or CBS scheduler can start over after exiting the LPI mode.
4. The MAC transfers the received control packet to the application based on the setting of the EMAC_MACPKT_FILT.PCF bit field.

If subsequent pause or PFC packets are received before the earlier pause time expires, the MAC updates the pause timer with new value.

Loopback Mode

The MAC supports Loopback of transmitted packets to its receiver.

Guidelines for Using Loopback Mode

The following are some guidelines for using the loopback mode:

- Enable loopback only with the full-duplex mode. In half-duplex mode, the carrier sense signal (crs) or collision (col) signal inputs get sampled which may result in issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip (for example, in a FPGA setup), the program should externally generate the transmit and receive clocks and provide these clocks to the MAC.
- Do not loop back big packets. Big packets may get corrupted in the loopback FIFO.

The transmit and receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the `EMAC_TXD[n]` data to the receive path. The asynchronous FIFO is 10-bits (6-bits in 10/100 Mbps mode) wide to accommodate `EMAC_TXD[n]`, TXEN, and TX_ER. The depth of the FIFO is five in 1000 Mbps mode and nine in 10/100 Mbps mode. The FIFO is free running to write on the write clock TX_CLK and read on every read clock RX_CLK.

At the start of each packet read out of the FIFO, the write and read pointers are re-initialized to have an offset of 2 (4 in 10/100 Mbps mode). This avoids overflow or underflow during a packet transfer. This also ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between MII/RMII/RGMII transmit and receive clock frequencies. Therefore, bigger packets should not be looped back because they may become corrupt in this loopback FIFO.

At the end of every received packet, the Receive Protocol Engine module generates received packet status and sends it to the Receive Packet Controller module. The control, missed packet, and filter fail status are added to the Receive status in the Receive Packet Controller module. The MAC does not process ARP packets that are looped back.

Enabling Loopback Mode

To enable this feature, program the `EMAC_MAC_CFG.LM` bit.

Programs can enable loopback for all PHY interfaces. The data is always looped back through an internal asynchronous FIFO on to the internal receive MII/RMII/RGMII interface, irrespective of which PHY interface is selected. The loopback data is also passed through the corresponding transmit PHY interface block, on to the Ethernet line.

NOTE:
- During loopback, the data/packet is reflected on TXD[n] signal.
- Preemption is not supported in loopback mode.

Endian Support

The EMAC DMA supports both little endian and big endian modes. The following table illustrates the order of bytes on 32-bit data bus for little endian and big endian modes, respectively.

Table 30-106: Endian Support

Transfer Size	Address OffsetA[1:0]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
Little-Endian Mode					
Always word transfer	0	MSB	MSB - 1	LSB + 1	LSB
Big Endian Mode					
Always word transfer	0	LSB	LSB + 1	MSB - 1	MSB

The endian mode can be programmed using the `PADS_PCFG0.EMAC0_ENDIANNESS/ PADS_PCFG0.EMAC1_ENDIANNESS` registers. This bit should be set for big-endian mode and cleared for little endian mode. The reset value of this bit is '1' (big endian mode).

EMAC Event Control

The EMAC has registered with the system event controller (SEC) module. Various interrupt sources within the EMAC peripheral are shared through this interrupt line. Refer to the SEC chapter for details on how interrupts work in this product and how to configure them.

EMAC Interrupt Signals

Interrupts from the EMAC are triggered from the EMAC DMA layer MTL, or the EMAC CORE (MAC) layer. The following table shows the EMAC interrupt signals.

Table 30-107: EMAC Interrupt Signals

Interrupt Type	Description
`EMAC_STAT`	Combined interrupt signal includes MAC, MTL, and DMA interrupts.
`EMAC_MAC`	Interrupts from MAC and MTL
`EMAC_DMA[n]_TX`	DMA Transmit Per Channel Interrupts
`EMAC_DMA[n]_RX`	DMA Receive Per Channel Interrupts

Interrupts can be generated because of various events in the EMAC. These events are captured in status registers, and interrupt enables are provided for each source of an interrupt such that the interrupt signal (`EMAC_STAT`) is asserted for an event only when the corresponding interrupt enable is set.

On EMAC, the interrupt status and corresponding enable registers are organized in an hierarchical manner so that it is easier for software to traverse and identify the source of an interrupt event quickly. When `EMAC_STAT` is asserted, the `EMAC_DMA_ISTAT` register is the first level that indicates the major blocks for the interrupt event source.

This register is read-only, and it contains bits corresponding to each DMA channel (transmit and receive pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- EMAC_MAC_ISTAT
- EMAC_MTL_ISTAT
- EMAC_DMA[n]_STAT

Interrupts from the MAC

Interrupts can be generated from the MAC because of various events in the MAC Receiver, Transmitter, or the modules/functions such as RMON counters, EEE and so on. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software. The EMAC_MAC_ISTAT register describes the events that can cause an interrupt from the MAC. The MAC interrupts are disabled by default. Each event can assert the interrupt on the EMAC_STAT or EMAC_MAC signals when the corresponding bit is set in the EMAC_MAC_IEN register.

The interrupt register bits only indicate the block from which the event is reported. The program must read the corresponding status registers and other registers to clear the interrupt. For example, when set high, bit 0 (EMAC_MAC_ISTAT.RGSMIIIS) indicates that the link status on the RGMII, SGMII, or SMII interface has changed. The program must read the EMAC_PHYIF_CTLSTAT register to clear this interrupt event.

NOTE: By default, the MAC interrupt status bits are cleared when the register that contains the source of the interrupt is read. If the EMAC_CSR_SW_CTL.RCWE bit is programmed to 1, the MAC interrupt status bits are cleared when the bit that contains the source of the interrupt is explicitly written to 1.

NOTE: Both the EMAC_STAT and the EMAC_MAC interrupt signals are generated for interrupts from the MAC. The EMAC_MAC interrupt is always generated one cycle ahead of the EMAC_STAT interrupt. If both the EMAC_STAT and EMAC_MAC interrupts are enabled:

- For GIC/ARM, only the EMAC_MAC interrupt service routine is called
- For SEC/SHARC, both the EMAC_STAT and EMAC_MAC interrupt services routines is called

Interrupts from the MTL

The EMAC can generate interrupts because of events in the MAC layer or in the MTL modules. The EMAC_STAT output is the interrupt signal, which is a level signal (asserted until the interrupt source is read and cleared). In the MTL, the interrupts are mainly related to exception events in the TxQ or RxQ in the transmit and receive paths, respectively. The interrupt status is captured and organized in a hierarchical manner to identify the root cause quickly.

The EMAC_MTL_ISTAT register identifies the top-level modules that can cause the interrupt to be asserted. Bits[7:0] identify 8 Queues (transmit or receive). They are read-only bits, and the application should read the corresponding EMAC_Q[n]_INT_CTLSTAT register to identify the exact cause and set the corresponding bits to 1'b1

to clear the interrupt event. The assertion of EMAC_MAC due to these events is enabled by the corresponding enable bits in the EMAC_Q[n]_INT_CTLSTAT register.

Interrupts from DMA

The EMAC_STAT interrupt is a level signal and gets de-asserted only when all the enabled interrupt events are cleared in their respective status registers and correspondingly all the bits in the EMAC_DMA_ISTAT register are cleared.

The EMAC_DMA[n]_STAT register captures all the interrupt events of that transmit DMA and receive DMA channel pair. The EMAC_DMA[n]_IEN register contains the corresponding enable bits for each interrupt event. There are two groups of interrupts in the DMA channel, normal and abnormal. They are indicated by the EMAC_DMA[n]_STAT.NIS and EMAC_DMA[n]_STAT.AIS bits. The normal group is for events that happen during the normal transfer of packets (EMAC_DMA[n]_STAT.TI, EMAC_DMA[n]_STAT.RI, EMAC_DMA[n]_STAT.TBU) while the abnormal interrupt events are for error events. Interrupt events are cleared by writing 1'b1 to the corresponding bit position. When all the enabled interrupt events are cleared (including the NIS and AIS), the interrupt source from the DMA channel is cleared and the corresponding bit in the EMAC_DMA_ISTAT register is also cleared.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. For example, the EMAC_DMA[n]_STAT.RI bit indicates that one or more packets were transferred to the application buffer. The driver must scan all descriptors, from the last recorded position to the first one, owned by the DMA to determine how many packets are received.

An interrupt is generated only once for multiple events. The driver must scan the EMAC_DMA_ISTAT register for the cause of the interrupt and clear the source in the respective status register. The EMAC_STAT is cleared only when all the bits of the EMAC_DMA_ISTAT register are cleared.

Periodic Scheduling of Transmit and Receive Interrupt

To improve the throughput and performance, EMAC supports interrupt timer and transmit descriptor to generate interrupts periodically, instead of every DMA transfer.

It is not preferable to generate interrupts for every packet transferred by DMA (EMAC_DMA[n]_STAT.RI and EMAC_DMA[n]_STAT.TI) for system throughput performance reasons. The EMAC gives the flexibility to schedule the interrupt at regular intervals using two methods:

- Set the interrupt on completion bit in transmit descriptor (TDES2[31]) once for every "required" number of packets to be transmitted.
- Similarly, set the IOC (see RDES3[30]) bit only at some specific intervals of receive descriptors. This way, whenever a received packet transfer to system memory is complete, and any of the descriptors used for that packet transfer has the IOC bit set, only then is the event is generated.

In addition, an interrupt timer (EMAC_DMA[n]_RXINTWDTMR) is given for flexible control and periodic scheduling of receive interrupts. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the receive DMA completes a transfer of a received packet to system memory without asserting the Receive Interrupt because the corresponding interrupt of completion IOC bit (see Table 30-26 Receive Descriptor Fields 3

(RDES3)) is not set. When this timer runs out as per the programmed value, the bit is set, and the interrupt is asserted when the corresponding EMAC_DMA[n]_IEN.RIE bit is enabled. The timer is stopped and cleared before it expires, iwhen the RI bit is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Per Channel Transfer Complete Interrupt

The transmit transfer complete interrupt (TI) and receive transfer complete interrupt (RI) are reflected in theEMAC_DMA[n]_STAT register. The EMAC_DMA[n]_STAT.TI bit is set whenever the transmit DMA channel closes the transmit descriptor in which the IOC (interrupt on completion - TDES2[31]) bit is set. Similarly, the EMAC_DMA[n]_STAT.RI bit is set whenever the receive DMA channel closes the receive descriptor with LD bit set and in any of the descriptors used for transferring that packet, IOC (interrupt enable on completion - RDES3[30]) bit is set.

The common EMAC_STAT output signal is asserted for the transfer complete interrupts only when the corresponding interrupts are enabled in the EMAC_DMA[n]_IEN register.

EMAC also supports the following per channel transfer complete interrupt signals.

- EMAC_DMA[n]_TX (transmit per channel interrupts)
- EMAC_DMA[n]_RX (receive per channel interrupts)

The behavior of the RI/TI/EMAC_DMA[n]_TX/EMAC_DMA[n]_RX changes depending on the settings of the EMAC_DMA_MODE.INTM bit field [17:16]. The following explains the transfer complete interrupt behavior.

Table 30-108: Transfer Complete Interrupt Behavior

S. No.	Interrupt Mode (INTM)	Behavior of EMAC_DMA[n]_TX and EMAC_DMA[n]_RX	Behavior of EMAC_STAT
1	0	A pulse is asserted on these output signals when corresponding TX/RX transfer complete event (for which IOC descriptor bits are enabled) is detected, irrespective of the corresponding interrupt status.	The TI/RI status signals are set whenever the "Transfer complete" event is detected. The bits get cleared whenever the software driver writes '1' to these bits. The EMAC_STAT is asserted whenever the corresponding interrupts are also enabled in the EMAC_DMA[n]_IEN register. The EMAC_DMA[n]_STAT.NIS status bit is asserted for RI/TI events.
2	1	These signals reflect the value of corresponding EMAC_DMA[n]_STAT.TI/ EMAC_DMA[n]_STAT.TI bits when the corresponding interrupt enable is set. Therefore, they are level signals and are cleared by the application by writing 1'b1 to the RI/TI status bits. This signal is not asserted when the corresponding interrupt enable bit is not set.	The EMAC_STAT signal and EMAC_DMA[n]_STAT.NIS status bit are not asserted for RI/TI events.

Table 30-108: Transfer Complete Interrupt Behavior (Continued)

S. No.	Interrupt Mode (INTM)	Behavior of EMAC_DMA[n]_TX and EMAC_DMA[n]_RX	Behavior of EMAC_STAT
3	2	In this mode, RI/TI interrupts are queued. These signals reflect the value of corresponding `EMAC_DMA[n]_STAT.TI`/ `EMAC_DMA[n]_STAT.TI` bits when the corresponding interrupt enable is set. They are level signals and cleared by software by writing 1'b1 to the RI/TI status bits. However, it is set again if another TI/RI event(s) is detected before the TI/RI bits are cleared for the previous event.	The RI/TI status bits are set whenever the Transfer Complete event is detected and gets reset whenever software driver clears those bits by writing 1. However, if another transfer complete event is detected before it is cleared (serviced) by the software, then EMAC automatically set these status bits again. However, the `EMAC_STAT` signal is not generated based on TI/RI. The `EMAC_DMA[n]_STAT.NIS` status bit is not asserted for RI/TI events.

EMAC Triggers

EMAC0 contains:

- 12 trigger requesters – 8 for receive DMA channels (`EMAC_DMA[n]_RX`) and 4 for MCGR DMA request (EMAC0_MCGR[n])
- 20 trigger completers – 8 for receive DMA channels (`EMAC_DMA[n]_RX_STOP`), 8 for transmit DMA channels (`EMAC_DMA[n]_TX`) and 4 for MCGR DMA Ack (EMAC0_MCGR[n]_ACK)

EMAC1 contains:

- 1 trigger requester – for Rx DMA (EMAC1_DMA0_RX)
- 2 trigger completers – 1 for Rx DMA (EMAC1_DMA0_RX_STOP) and 1 for Tx DMA (EMAC1_DMA0_TX)

NOTE:
- Each of the DMA trigger completers at EMAC interface is controller by 2 trigger signals from the TRU – 1 start trigger, 1 stop trigger. When the TRU gives a start trigger, the trigger completer at EMAC interface is asserted. When the TRU gives a stop trigger, the trigger completer at EMAC interface is de-asserted.
- An error condition occurs when the TRU generates a 2nd start trigger that occurs before a stop trigger or a 2nd stop trigger before a start trigger. This error condition does not affect the trigger signal at EMAC. It is just to check the status and is stored as a W1C status bit in ECO_REG9 of the MISC REG block.

EMAC Programming Model

This section provides the programming model and programming procedures for the Ethernet MAC peripheral.

Initializing DMA

Use the following procedure to initialize DMA.

1. Perform a software reset by setting the EMAC_DMA_MODE.SWR bit. This action resets all of the EMAC internal registers and logic.
2. Wait for the completion of the reset process by polling the EMAC_DMA_MODE.SWR bit which is only cleared (automatically) after the reset operation completes.
3. Program the following fields to initialize the EMAC_DMA_SYSBMODE register:
    a. EMAC_DMA_SYSBMODE.AAL
    b. Fixed burst or undefined burst (EMAC_DMA_SYSBMODE.FB)
    c. Maximum SCB outstanding request limit (EMAC_DMA_SYSBMODE.WR_OSR_LMT and EMAC_DMA_SYSBMODE.RD_OSR_LMT for both write and read)
    d. If fixed length value is enabled (EMAC_DMA_SYSBMODE.FB bit is set), select the maximum burst length possible on the SCB bus (EMAC_DMA_SYSBMODE.BLEN16, EMAC_DMA_SYSBMODE.BLEN8 and EMAC_DMA_SYSBMODE.BLEN4 bits).
4. Create a descriptor list for transmit and receive. In addition, ensure that the descriptors are owned by DMA (set bit 31 of descriptor TDES3/RDES3). For more information about descriptors, see DMA Descriptors.
5. Program the transmit and receive ring length registers (EMAC_DMA[n]_TXDSC_RLEN and EMAC_DMA[n]_RXCTL2.RDRL). The ring length programmed must be at least four.
6. Initialize the receive and transmit descriptor list address with the base address of the transmit and receive descriptor (EMAC_DMA[n]_TXDSC_ADDR and EMAC_DMA[n]_RXDSC_ADDR). Also, program transmit and receive tail pointer registers indicating to the DMA about the available descriptors (EMAC_DMA[n]_TXDSC_TLPTR and EMAC_DMA[n]_RXDSC_TLPTR).
7. Program the settings of the following registers for the parameters like maximum burst-length (PBL) initiated by DMA, descriptor skip lengths, OSP in case of TxDMA, RBSZ in case of RxDMA, and so on:
    a. EMAC_DMA[n]_CTL
    b. EMAC_DMA[n]_TXCTL
    c. EMAC_DMA[n]_RXCTL
8. Enable the interrupts by programming the EMAC_DMA[n]_IEN register.
9. Start the receive and transmit DMAs by setting the EMAC_DMA[n]_RXCTL.SR and the EMAC_DMA[n]_TXCTL.ST bits.
10. Repeat steps 4 to 9 for all the transmit DMA and receive DMA channels selected in the hardware.

Initializing the MTL Registers

The Transaction layer (MTL) registers must be initialized to establish the transmit and receive operating modes and commands. Use the following steps to initialize the MTL registers:

1. Program the transmit scheduling (EMAC_MTL_OPMODE.SCHALG) and receive arbitration algorithm (EMAC_MTL_OPMODE.RAA) fields to initialize the MTL operation in case of multiple transmit and receive queues.
2. Program the receive queue to DMA mapping in the EMAC_RXQ_DMA_MAP0 and EMAC_RXQ_DMA_MAP1 registers.
3. Program the following fields to initialize the mode of operation in the EMAC_TQ0_OPMODE register.
    a. Transmit store and forward (EMAC_TQ0_OPMODE.TSF) or transmit threshold control (EMAC_TQ0_OPMODE.TTC) in case of threshold mode.
    b. Transmit queue enable (EMAC_TQ0_OPMODE.TXQEN) to value 2‘b10 to enable transmit queue0.
    c. Transmit queue size (EMAC_TQ0_OPMODE.TQS).
4. Program the following fields to initialize the mode of operation in the EMAC_RQ0_OPMODE register:
    a. Receive store and forward (EMAC_RQ0_OPMODE.RSF) or EMAC_RQ0_OPMODE.RTC in case of threshold mode.
    b. Flow control activation and de-activation thresholds for MTL receive FIFO (EMAC_RQ0_OPMODE.RFA and EMAC_RQ0_OPMODE.RFD).
    c. Error Packet and undersized good packet forwarding enable (EMAC_RQ0_OPMODE.FEP and EMAC_RQ0_OPMODE.FUP).
    d. Receive queue size (EMAC_RQ0_OPMODE.RQS).
5. Repeat previous two steps for all MTL transmit and receive queues selected in the configuration.

Initializing the MAC

The MAC configuration registers establish the operating mode of the MAC. These registers must be initialized before initializing the DMA.

The following MAC initialization operations can be performed after DMA initialization. If the MAC initialization is completed before the DMA is configured, enable the MAC receiver (the last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the receive FIFO and overflow.

1. Provide the required number of MAC address registers:EMAC_ADDR0_HI and EMAC_ADDR0_LO.
2. Program the following fields to set the appropriate filters for the incoming frames in the packet filter register:
    a. Receive all (EMAC_MACPKT_FILT.RA)

b. Promiscuous mode (EMAC_MACPKT_FILT.PR)

c. Hash or perfect filter (EMAC_MACPKT_FILT.HPF)

d. Unicast, multicast, broadcast, and control frames filter settings

3. Program the following fields for proper flow control in the flow control register:

a. Pause time and other pause frame control bits (EMAC_Q[i]_TXFLOW_CTL.PT, EMAC_Q[i]_TXFLOW_CTL.PLT)

b. Transmit flow control bits (EMAC_Q[i]_TXFLOW_CTL.TFE)

c. Flow control busy (EMAC_Q[i]_TXFLOW_CTL.FCB_BPA)

4. Program the EMAC_MAC_IEN register, as required, and if applicable, for the configuration.

5. Program the appropriate fields in the EMAC_MAC_CFG register. For example: Inter-packet gap (EMAC_MAC_CFG.IPG) while transmission and jabber disable (EMAC_MAC_CFG.JD).

6. Set bit 0 (EMAC_MAC_CFG.RE) and 1 (EMAC_MAC_CFG.TE) to start the MAC transmitter and receiver.

7. To support jumbo transmit/receive packets, configure the MAC configuration register as follows:

a. Set the EMAC_MAC_CFG.JE field to 1

b. Set the EMAC_MAC_CFG.JD and EMAC_MAC_CFG.WD fields to 1 to avoid giant packet error reporting

c. Set the EMAC_MAC_CFG.GPSLCE field to 1

d. Set the EMAC_MAC_EXT_CFG.GPSL field to a value > 9026

8. To support transmit/receive packets up to 16K, configure the MAC configuration register as follows:

a. Set the EMAC_MAC_CFG.JD and EMAC_MAC_CFG.WD fields to 1 to avoid giant packet error reporting

b. Set the EMAC_MAC_CFG.GPSLCE field to 1

c. Set the EMAC_MAC_EXT_CFG.GPSL field to 16383

Performing Normal Transmit and Receive Operations

During normal operation of the EMAC, normal and transmit interrupts are read, descriptors polled, the DMA is suspended (when it does not own descriptors), and the values of the current host transmitter or receiver descriptor pointers are read for debugging.

For normal operations, use the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the host (either transmit or receive).

2. Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.

   ADDITIONAL INFORMATION: When the descriptors are not DMA (or no descriptor is available), the DMA goes into the SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and writing the descriptor tail pointer to the transmit/receive tail pointer registers (EMAC_DMA[n]_TXDSC_TLPTR and EMAC_DMA[n]_RXDSC_TLPTR).

3. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process (EMAC_DMA[n]_TXDSC_CUR and EMAC_DMA[n]_RXDSC_CUR).

4. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process (EMAC_DMA[n]_TXBUF_CUR and EMAC_DMA[n]_RXBUF_CUR).

Stopping and Starting Transmission

Use the following procedure to stop and start EMAC transfers.

1. Disable the MAC transmitter and receiver by clearing the EMAC_MAC_CFG.RE and EMAC_MAC_CFG.TE bits.

2. Ensure that both the transmit and receive queues are empty (EMAC_TQ0_DBG.TXQSTS and EMAC_RQ0_DBG.RXQSTS is 0).

3. Disable the receive DMA (whenapplicable), after ensuring that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of EMAC_RQ0_DBG register, EMAC_RQ0_DBG.PRXQ = 0 and EMAC_RQ0_DBG.RXQSTS = 00).

4. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver.

   ADDITIONAL INFORMATION: Do not change the configuration (such as duplex mode, speed, port, or loop back) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active. Similarly, do not change the DMA-related configuration when transmit and receive DMA are active.

Switching to New Descriptor List in Receive DMA

Switching to a new descriptor list is different in the receive DMA compared to the transmit DMA. Switching to a new descriptor list is permitted when the receive DMA is in SUSPEND state, as clarified by the following points:

- Generally, receive DMA prepares the descriptors in advance.
- If the receive DMA goes to the SUSPEND state due to the descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames are lost because of a receive FIFO overflow. Therefore, the software is allowed to create a new descriptor list and program the receive DMA to start using it immediately, without going into the STOP state.

Multi-Channel Multi-Queuing

Transmit

Use the following procedure for a transmit operation.

1. Program the transmit queue size in the EMAC_TQ0_OPMODE.TQS bit field. Based on the value programmed in this bit field, the size of the queue is determined. In the transmit operation, the number of channels is equal to the number of the queues. Due to this reason, the channel-to-queue mapping is fixed.
2. For a queue to be used, enable the queue using the EMAC_TQ0_OPMODE.TXQEN bit. Enable the EMAC_DMA[n]_TXCTL.ST bit and the corresponding EMAC_TQ0_OPMODE.TXQEN bit.
3. Program the scheduling method in the EMAC_MTL_OPMODE.SCHALG bit field.
4. Program the EMAC_TQ0_QTMWGT register for DCB queues as per the selected algorithm. For CBS algorithms in AVB queues, the EMAC_TQ[n]_ETS_CTL, EMAC_TQ[n]_SSCRDT, EMAC_TQ[n]_HICRDT and EMAC_TQ[n]_LOCRDT registers also need to be programmed as required.
5. If DCB is enabled and the PFC function is required, program the EMAC_TXQ_PRTY_MAP0 register to assign a fixed priority to the queue. This priority is used for determining if the corresponding queue should stop transmitting a packet based on the received PFC packet.

Receive

Use the following procedure for the receive operation.

1. Program the receive queue size in the EMAC_RQ0_OPMODE.RQS bit field. Based on the value programmed in this bit field, the size of the queue is determined.
2. Enable the receive queues 0 to 7 in the EMAC_RXQ_CTL0.RXQ0EN to EMAC_RXQ_CTL0.RXQ7EN bit fields for AV or DCB. In DMA configurations, enable the EMAC_DMA[n]_RXCTL.SR bit of statically or dynamically mapped EMAC_DMA[n]_RXCTL register and the corresponding EMAC_RXQ_CTL0.RXQ0EN to EMAC_RXQ_CTL0.RXQ7EN bits.
3. The MAC routes the receive packets to the receive queues based on following packet types:
   - AV PTP packets – based on the programming of the EMAC_RXQ_CTL1.PTPQ bit.
   - AV untagged control packets – based on the programming of the EMAC_RXQ_CTL1.AVCPQ bit.
   - Data Center Bridging (DCB) related Link Layer Discovery Protocol (LLDP) packets. Program the EMAC_RXQ_CTL1.DCBCPQ bits to indicate to the MAC which queue receives the DCB packets.
   - VLAN Tag Priority field in VLAN tagged packets – program the PSRQ7-0 bits in the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers for the routing of tagged packets based on the USP (user priority) field of the received packets to the receive queues 0 to 7.
   - The AV tagged control and data packets are also routed based on PSRQ field of the EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers. The priorities set in PSRQ7-0 should be unique.

4. If multiple receive DMA channels are enabled, the following programming is required for proper arbitration and mapping:
    a. Program the EMAC_MTL_OPMODE.RAA bit field to select the arbitration algorithm to decide which RxQ is read out from the RxFIFO memory.
    b. Program the EMAC_RQ0_CTL register to decide the weights and the packet arbitration for each RxQ.
    c. If static mapping is programmed in the EMAC_RXQ_DMA_MAP0/EMAC_RXQ_DMA_MAP1 registers (the DDMACH bits are reset to 0), the RXQx2DMA and others need to be programmed to select the channel for which each queue is mapped.
    d. Set the EMAC_RXQ_DMA_MAP0 DDMACH bits to select dynamic mapping of packets in each receive queue.
    e. In dynamic channel mapping, the routing of a packet to a specific receive DMA channel is decided by the value of the DCS field in the lowest MAC address register (for example EMAC_ADDR0_HI.DCS).

Recovering from DMA Channel Failure

When the DMA channel issues a bus error, follow these steps to recover from the failure.

Recovering from the Receive DMA Channel Failure

Follow these steps a bus error occurs in the receive DMA channel:

1. Set the EMAC_DMA[n]_RXCTL.RPF bit to 1. This flushes all the packets one after the other. This step is optional. However, setting this bit prevents HOL (head-of-line) blocking in the receive queues when packets sent to the receive DMA are stopped due to bus error.
2. Re-program the specific registers of the DMA channel.
3. Start the DMA channel.

Recovering from the Transmit DMA Channel Failure

Follow these steps if a bus error occurs in the transmit DMA channel:

1. Stop the specific DMA channel, even if it is in an active state.
2. Flush the corresponding MTL queue.
3. Re-program the specific registers of the DMA channel.
4. Start the DMA channel.

NOTE: Reprogramming the DMA channel registers might not always be successful in recovering from a bus error. If EMAC is not fully functional after reprogramming the DMA, as a workaround, issue a soft reset to recover from the bus error.

Disabling the Receive Queue

Follow these steps to disable specific receive queues

1. Disable the receiver by setting the EMAC_MAC_CFG.RE bit to 0.
2. Read the debug register (ensure that the EMAC_MAC_DBG.RPESTS and EMAC_MAC_DBG.RFCFCSTS fields are 0) and the EMAC_RQ0_DBG registers of all the receive queues.
3. When the EMAC_RQ0_DBG registers are read as 0, disable the intended receive queue by programming the corresponding fields in the EMAC_RXQ_CTL0 register.
4. Enable the receiver by setting the EMAC_MAC_CFG.RE bit to 1.

IEEE 1588 Time Stamping (PTP)

The following sections provide programming information for using for IEEE 1588 Time Stamping.

Initialization Guidelines for System Time Generation

Programs can enable the time stamp feature by setting the EMAC_TM_CTL.TSENA bit. However, it is essential that the time stamp counter be initialized after this bit is set. Complete the following steps during EMAC initialization:

1. Mask the time stamp trigger interrupt by clearing the EMAC_MAC_IEN.TSIE bit.
2. Set the EMAC_TM_CTL.TSENA bit 0 to enable time stamping.
3. Program the EMAC_TM_SUBSEC register based on the PTP clock frequency.
4. If the Fine Correction method is required, program the EMAC_TM_ADDEND register and set the EMAC_TM_CTL.TSADDREG bit.
5. Poll the EMAC_TM_CTL register until the EMAC_TM_CTL.TSADDREG bit is cleared.
6. Program the EMAC_TM_CTL.TSCFUPDT bit to select the Fine Update method (if required).
7. Program the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT registers with the appropriate time value.
8. Set the EMAC_TM_CTL.TSINIT bit.

   ADDITIONAL INFORMATION: The time stamp counter starts operation as soon as it is initialized with the value written in the time stamp update registers. If one-step time stamping is enabled:

   a. To enable one-step time stamping, program bit 27 of the TDES3 Context Descriptor.

   b. Program the EMAC_TM_INGASYM_CORR and EMAC_TM_EGASYM_CORR registers to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper time stamping.

If a time stamp operation is disabled by clearing the EMAC_TM_CTL.TSENA bit, repeat all these steps to restart the time stamp operation.

System Time Correction

The following sections provide programming information for system time correction.

Coarse Correction Method

To synchronize or update the system time in one process (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT).
2. Set the EMAC_TM_CTL.TSUPDT bit. The value in the time stamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

Fine Correction Method

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm explained in Fine Correction Method, calculate the rate by which to make the system time increments slower or faster.
2. Update the EMAC_TM_ADDEND register with the new value and set the EMAC_TM_CTL.TSADDREG bit.
3. Wait for the time for the new value of the Addend register to be active. You can do this by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers.
5. Enable the time stamp interrupt using the EMAC_MAC_IEN.TSIE bit.
6. Set bit 4 in the EMAC_TM_CTL register.
7. When this trigger causes an interrupt, read the EMAC_MAC_ISTAT register.
8. Reprogram the EMAC_MAC_ISTAT register with the old value and set the EMAC_TM_CTL.TSADDREG bit again.

PTP Offloading

The following sections provide programming information for using PTP offload.

Automatic Periodic Generation of PTP Sync Messages

Follow these steps to enable automatic periodic generation of PTP sync messages:

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSMSTRENA, and EMAC_TM_CTL.TSEVNTENA bit fields to zero, one, and one respectively, to configure the node as ordinary or boundary requester (one, one, and one for transparent requester).
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable the PTP offload feature and domain number to send in egress PTP Sync message.
3. Program the EMAC_PTO_CTL.ASYNCEN bit to enable periodic generation of PTP Sync messages.
4. Program the 80-bit Source Port Identity in theEMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to send in egress PTP Sync message.
5. Program the EMAC_LOG_MSG_INTVL.LSI bit field to program the periodicity of the PTP Sync messages. For example, value of 1 corresponds to 21 which translates to PTP Sync message every 2 seconds and value of 0xFF (twos complement of -1) corresponds to 2-1 which translates to PTP Sync message every 0.536 seconds.
6. Program the EMAC_MAC_IEN.TSIE bit to enable generation of time stamp interrupt.
7. Wait for a EMAC_STAT or EMAC_MAC interrupt generated when the EMAC_TM_STAT.TXTSSIS bit is set, which indicates that the time stamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers.

Periodic Generation of PTP Pdelay_Req Messages

Follow these steps to enable automatic periodic generation of PTP Pdelay_Req messages.

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSMSTRENA, and EMAC_TM_CTL.TSEVNTENA bit fields to 1, 0, and 1 respectively to configure the node as transparent completer (1, 1, and 1 for transparent requester OR 3, X, and X for Peer-to-Peer Transparent).
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable PTP Offload feature and domain Number to send in egress PTP Pdelay_Req message.
3. Program the EMAC_PTO_CTL.APDREQEN bit to enable periodic generation of PTP Pdelay_Req messages.
4. Program the 80-bit Source Port Identity in theEMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to send in egress PTP Pdelay_Req message.
5. Program the EMAC_LOG_MSG_INTVL.LMPDRI bit field to program the periodicity of the PTP Pdelay_Req messages. For example, value of 1 corresponds to 21 which translates to PTP Pdelay_Req message every 2 seconds and value of 0xFF (twos complement of -1) corresponds to 2-1 which translates to PTP Pdelay_Req message every 0.536 seconds.
6. Program the EMAC_MAC_IEN.TSIE bit to enable generation of Timestamp Interrupt.
7. Wait for EMAC_STAT or EMAC_MAC interrupt generated on setting of the EMAC_TM_STAT.TXTSSIS bit is set, which indicates that the time stamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers.

Response Generation - Ordinary or Boundary Requester Mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary Requester mode (periodic PTP sync messages generated and PTP Delay_Resp message generated in response to PTP Delay_Req message).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSMSTRENA and EMAC_TM_CTL.TSEVNTENA bit fields to zero, one, and one respectively.
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN field to enable the PTP offload feature and domain number to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
4. Program the EMAC_LOG_MSG_INTVL.DRSYNCR and EMAC_LOG_MSG_INTVL.LSI bit fields. The sum of both these fields is updated in the logMinDelayReqInterval field of the PTP Delay_Resp message.

Response Generation - Ordinary or Boundary Completer Mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary completer mode (PTP Delay_Req message generated in response to PTP Sync message).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSMSTRENA and EMAC_TM_CTL.TSEVNTENA bit fields to zero, zero, and one respectively.
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable PTP Offload feature and domain Number to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
4. Program the EMAC_LOG_MSG_INTVL.DRSYNCR bit field to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages.

Response Generation - Transparent Completer Mode

Follow these steps to enable generation of PTP response messages for transparent completer mode (PTP Delay_Req message generated in response to PTP Sync message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSMSTRENA and EMAC_TM_CTL.TSEVNTENA bit fields to one, zero, and one respectively.

2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable the PTP offload feature and domain number to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
4. Program the EMAC_LOG_MSG_INTVL.DRSYNCR bit field and EMAC_LOG_MSG_INTVL.LMPDRI bit fields to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages and periodicity of the PTP Pdelay_Req messages.
5. Program the EMAC_MAC_IEN.TSIE bit to enable generation of the timestamp interrupt.
6. Wait for the EMAC_STAT or EMAC_MAC interrupt that is generated when the EMAC_TM_STAT.TXTSSIS bit is set. This indicates that the timestamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers for egress PTP Pdelay_Req and Pdelay_Resp messages.

Response Generation - Transparent Requester Mode

Follow these steps to enable generation of PTP response messages for transparent requester mode (PTP Delay_Resp message generated in response to PTP Delay_Req message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req or Sync messages generated).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSMSTRENA and EMAC_TM_CTL.TSEVNTENA bit fields to one, one, and 1one respectively.
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable the PTP offload feature and domain number to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
4. Program the EMAC_LOG_MSG_INTVL.DRSYNCR, EMAC_LOG_MSG_INTVL.LSI and EMAC_LOG_MSG_INTVL.LMPDRI bit fields. The sum of EMAC_LOG_MSG_INTVL.DRSYNCR and EMAC_LOG_MSG_INTVL.LSI is updated in logMinDelayReqInterval field of PTP Delay_Resp message and periodicity of the PTP Sync or Pdelay_Req messages.
5. Program the EMAC_MAC_IEN.TSIE bit to enable generation of the timestamp interrupt.
6. Wait for the EMAC_STAT or EMAC_MAC interrupt that is generated when the EMAC_TM_STAT.TXTSSIS bit is set. This indicates that the timestamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers for egress PTP Sync, Pdelay_Req and Pdelay_Resp messages.

Response Generation - Peer-to-Peer Transparent Mode

Follow these steps to enable generation of PTP response messages for Peer-to-Peer Transparent mode (PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated).

1. Program the EMAC_TM_CTL.SNAPTYPSEL, EMAC_TM_CTL.TSMSTRENA and EMAC_TM_CTL.TSEVNTENA bit fields to 3, X, and X respectively.
2. Program the EMAC_PTO_CTL.PTOEN bit and EMAC_PTO_CTL.DN bit field to enable the PTP offload feature and domain number to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
3. Program the 80-bit source port identity in the EMAC_SRCPRT_IDNTY0 through EMAC_SRCPRT_IDNTY2 registers to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
4. Program the EMAC_LOG_MSG_INTVL.LMPDRI bit field to indicate periodicity of the PTP Pdelay_Req messages.
5. Program the EMAC_MAC_IEN.TSIE bit to enable generation of the timestamp interrupt.
6. Wait for the EMAC_STAT or EMAC_MAC interrupt that is generated when the EMAC_TM_STAT.TXTSSIS bit is set. This indicates that the timestamp for PTP Sync message is captured in the EMAC_TXTMSTAT_SEC and EMAC_TXTMSTAT_NSEC registers for egress PTP Pdelay_Req and Pdelay_Resp messages.

Using the Audio Video Feature

The following sections provide procedures to program audio video features.

Initializing the DMA in Audio Video Feature

1. Provide a software reset to reset all internal registers and logic (EMAC_DMA_MODE.SWR bit).
2. Wait for the completion of the reset process. Poll the EMAC_DMA_MODE.SWR bit, which is cleared only after the reset operation is completed.
3. Program the fields to initialize the DMA register by setting the values in the EMAC_DMA_MODE register.
4. Create a proper descriptor list for transmit and receive. In addition, ensure that the DMA owns the transmit and receive descriptors.

   ADDITIONAL INFORMATION: When OSF mode is used, at least two transmit descriptors are required. For more information about descriptors, see the DMA Descriptors section.
5. Ensure that the software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors.

6. Program the transmit and receive ring length registers (EMAC_DMA[n]_TXDSC_RLEN and EMAC_DMA[n]_RXCTL2.RDRL). The programmed ring length must be at least 4.
7. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (EMAC_DMA[n]_TXDSC_ADDR and EMAC_DMA[n]_RXDSC_ADDR). In addition, program the transmit and receive tail pointer registers that indicate to the DMA the available descriptors (EMAC_DMA[n]_TXDSC_TLPTR and EMAC_DMA[n]_RXDSC_TLPTR).
8. Program the following fields to initialize the mode of operation in the EMAC_TQ0_OPMODE register:
    a. Transmit store and forward (EMAC_TQ0_OPMODE.TSF)
    b. Transmit threshold control (EMAC_TQ0_OPMODE.TTC)
    c. Transmit queue enable (EMAC_TQ0_OPMODE.TXQEN) to value 2‘b10 to enable transmit queue0
    d. Transmit queue size (EMAC_TQ0_OPMODE.TQS)
9. Enable the interrupts by programming the EMAC_DMA[n]_IEN register.
10. Repeat steps 4 through 9 for all additional channels of AV feature.
11. Program the EMAC_TQ[n]_ETS_CTL register, idleSlope (EMAC_TQ0_QTMWGT), sendSlope (EMAC_TQ[n]_SSCRDT, hiCredit (EMAC_TQ[n]_HICRDT), and loCredit (EMAC_TQ[n]_LOCRDT) registers of the AVQueues.
12. Start the receive and transmit DMA by setting the EMAC_DMA[n]_TXCTL.ST bit and the EMAC_DMA[n]_RXCTL.SR bit.

Enabling Slot Number Checking

The program enables the ability to report the average bits that are transmitted in a slot. The EMAC_TQ0_ETS_STAT register of the additional AV channels provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting.

1. Enable time stamping by following the steps described in the “Initialization Guidelines for System Time Generation."
2. Program the EMAC_TQ[n]_ETS_CTL.SLC bits [6:4] of a channel with the number of slots over which the average transmitted bits per slot need to be computed.

    ADDITIONAL INFORMATION: The frequency of this interrupt depends on the value programmed in this step. For example, when the value 0 is used in the EMAC_TQ[n]_ETS_CTL.SLC field, the interrupt is generated at every 125 microseconds. When not required, disable this interrupt to stop the interrupt flooding.
3. Enable the EMAC_Q0_INT_CTLSTAT.ABPSIE bit of a channel to generate the average bits per slot interrupt.
4. Read the EMAC_TQ0_ETS_STAT.ABS bits [16:0] of a channel on each interrupt.

ADDITIONAL INFORMATION: The software can read the EMAC_TQ0_ETS_STAT.ABS bits in polling mode even if the EMAC_Q0_INT_CTLSTAT.ABPSIE bit is not enabled. When high, bit 1 (EMAC_Q0_INT_CTLSTAT.ABPSIS) indicates that a new value is updated in the ABS field.

Disabling Flow Control for AV Enabled Queues

Receive Flow

Program the EMAC_RQ0_OPMODE.EHFC (enable hardware flow control) bit of the corresponding receive queue's operation mode register to 0.

Transmit Flow

Program the EMAC_TXQ_PRTY_MAP0.PSTQ0 through EMAC_TXQ_PRTY_MAP1.PSTQ7 bit fields corresponding to the AV enabled transmit queue (using the EMAC_TXQ_PRTY_MAP0 and EMAC_TXQ_PRTY_MAP1 registers) to 0.

Programming Energy Efficient Ethernet

The following sections provide procedures to program energy efficient Ethernet.

Entering and Exiting the Transmit LPI Mode

EEE enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of Physical layers to operate in the Low-Power Idle (LPI) mode. In the transmit path, the software must set the EMAC_LPI_CTLSTAT.LPIEN bit to indicate to the MAC to stop transmission and initiate the LPI protocol. Follow these steps for entering/existing the transmit LPI Mode:

1. Read the PHY register through the MDIO interface, check if the remote end has the EEE capability, and then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop receive clock in LPI mode.)
3. Program the EMAC_LPI_TMRSCTL.LST and EMAC_LPI_TMRSCTL.TWT bit fields.
4. Read the link status of the PHY chip by using the MDIO interface and update the EMAC_LPI_CTLSTAT.PLS bit accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Program the EMAC_ONEUS_TIC_CNT register as per the frequency of SCLK0.
6. Program the EMAC_LPI_ENTR_TMR.LPIET bit field with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set the EMAC_LPI_CTLSTAT.LPIATE and EMAC_LPI_CTLSTAT.LPITXA to enable the auto-entry into LPI and auto-exit of MAC from LPI state.
8. Set the EMAC_LPI_CTLSTAT.LPIEN bit to make the MAC transmitter enter the LPI state.

The MAC enters the LPI mode after completing all scheduled packets and remains IDLE for the time indicated by the EMAC_LPI_ENTR_TMR.LPIET bit field. It sets the EMAC_LPI_ENTR_TMR.LPIET bit [0] after entry to the LPI state.

When a packet is scheduled for transmission (when the transmit DMA comes out of the IDLE state), the MAC transmitter exits the LPI state automatically. It waits for TWT time before setting the EMAC_LPI_CTLSTAT.TLPIEX interrupt status bit and then resumes the packet transmission.

9. The MAC transmitter re-enters the LPI state if it remains IDLE for EMAC_LPI_ENTR_TMR.LPIET time and sets the EMAC_LPI_CTLSTAT.LPIEN bit and the entry-exit cycle continues.
10. Reset the EMAC_LPI_CTLSTAT.TLPIEX bit in case the application wants to over-ride the auto-entry/exit modes and make the MAC transmitter exit the LPI state directly.

To ensure MAC enter the LPI state only after completion of transmission of all the queued frames in the Tx FIFO, set the EMAC_LPI_CTLSTAT.LPITXA bit [19].

Programming Flexible Pulse-Per-Second Output

The following sections provide procedures to program Flexible Pulse-Per-Second Output.

Generating Single Pulse on PPS

To program single pulse on PPS, use the following procedure.

1. Program 11 or 10 (for interrupt) using the EMAC_PPS_CTL.TRGTMODSEL0 bits [6:5]. This instructs the MAC to use the target time registers (EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC) for the start time of PPS signal output.
2. Program the start time value in the target time registers (EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC).
3. Program the width of the PPS signal output in the EMAC_PPS[n]_WID register.
4. Program the EMAC_PPS_CTL.PPSCMD1 bits [3:0] to 0001. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in the target time registers.

Generating Next Pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), the program can cancel the pulse generation by giving the cancel start command (PPSCMD = 0011) before the programmed start time elapses. The behavior of the next pulse can be programmed in advance. To program the next pulse, follow this procedure.

1. Program the start time for the next pulse in the target yime registers. This time should be more than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in the EMAC_PPS[n]_WID register.

3. Program the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits [3:0], PPSCMD to generate a single pulse after the time at which the previous pulse is de-asserted. This instructs the MAC to generate single pulse on the PPS signal output, at the time programmed in target time registers. When this command executes before the previous pulse becomes low, then the new command overwrites the previous command and the EMAC may generate only one extended pulse.

Generating a Pulse Train on PPS

The EMAC_PPS_CTL.TRGTMODSEL0 bits program the target time registers to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

Complete the following steps to program the target time registers to generate only interrupt event.

1. Program the EMAC_PPS_CTL.TRGTMODSEL0 bits = 00 (for interrupt). This instructs the MAC to use the target time registers for target time interrupt.
2. Program a target time value in the target time registers. This instructs the MAC to generate an interrupt when the target time elapses. If the mode select bits [6:5] are changed (for example, to control the PPS), then the interrupt generation is over-written with the new mode and new programmed target time register value.

The TSTARGTERR[n] interrupt is generated when the programmed target time is smaller (that is, time in the past) compared to system time in the EMAC_TM_SEC and EMAC_TM_NSEC registers. Therefore, to avoid unwanted TSTARGTERR[n] interrupt, the correct writing order is as follows:

1. EMAC_PPS[n]_TGTM_NSEC
2. EMAC_PPS[n]_TGTM_SEC
3. EMAC_PPS[n]_INTVL
4. EMAC_PPS[n]_WID
5. EMAC_PPS_CTL.PPSCTRL_PPSCMD and EMAC_PPS_CTL.PPSEN0

Programming TSO

The TCP Segmentation Offload (TSO) engine is used to offload the TCP segmentation functions to the hardware. To program the TSO, set the TSE bit to enable TCP packet segmentation, and program descriptor fields to enable TSO for the current packet. Use the following steps to program TSO.

1. Program the EMAC_DMA[n]_TXCTL.TSE bit to enable TCP packet segmentation in that DMA channel.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must be programmed to enable TSO for the current packet:

a. Enable the TSE in bit 18 of the TDES3 register

b. Program the length of the un-segmented TCP/IP packet payload in bits [17:0] of the TDES3 register and the TCP header in bits [22:19] of TDES3.

c. Program the maximum size of the segment using the `EMAC_DMA[n]_CTL.MSS` bits or MSS in the context descriptor. If the MSS field is programmed in both the DMA channel control register and in the context descriptor, the latest software programmed sequence is considered.

3. The header of the unsegmented TCP/IP packet should be in buffer 1 of the first descriptor and this buffer must not hold any payload bytes. The payload is allocated to buffer 2 and the buffers of the subsequent descriptors.

   ADDITIONAL INFORMATION: If the TSE bit is enabled in the TDES3 register for a non-TCP-IP packet, the result is unpredictable.

Programming UFO

EMAC supports fragmentation of UDP packets into smaller IPv4 fragments. Enable the IP fragmentation by programming the `EMAC_DMA[n]_TXCTL.TSE_MODE` bits [14:13] as follows:

- Set to 2'b01 to enable fragmentation of UDP over IPv4 with checksum
- Set to 2'b10 to enable fragmentation of UDP over IPv4 without checksum

NOTE: In both the cases, if the `EMAC_DMA[n]_TXCTL.TSE` bit [12] is set to 1 for a TCP packet, DMA enables TCP segmentation; fragmentation is not enabled, because fragmentation is supported only for UDP packets.

Software Guidelines

When the software creates Transmit Descriptor of the UDP/IPv4 packet that is to be fragmented, the software must:

1. Set the desired fragment size (MFS) in the MSS field of TDES2 of the Context Descriptor. The MFS must be set only once, unless the MFS needs to be changed (for TCP packet targeted for segmentation). The MFS must be a multiple of 8 bytes for IP fragmentation.
2. Set the TSE bit of TDES3 of normal descriptor.
3. Program the THL = 2 in TDES3 of normal descriptor.
4. Program the correct value in the TPL field of TDES3 of normal descriptor. This value should be equal to length of UDP Header + UDP payload.

NOTE:
- Enable fragmentation only for IPv4; not for IPv6.
- Set the DF flag (do not fragment) in the IPv4 header packet to 0.
- Set the fragmentation offset in the IPv4 header packet to 0.

When the input packet is UDP over IPv6, do not enable IP fragmentation. Set the TSE bit to 0 in the TDES3 of normal descriptor and let the software perform the fragmentation.

Programming Header Payload Split Receive

The Header-Payload Split support can be enabled by

1. Setting the EMAC_DMA[n]_CTL.SPH bit field.
2. Selecting the Header-Payload split option by setting the EMAC_SPLM_OFST_CFG.SPLM bit field.

The DMA can process the header and payload of the received packets separately.

NOTE:
- In case of L3/L4 header-payload split, if the IPv4 header length or IPv6 payload length fields are corrupt, the split might not always happen at the correct boundary.
- When the EMAC_DMA[n]_CTL.SPH bit field is set to 1, buffer 1 of only the first descriptor is used with buffer size indicated by the EMAC_MAC_EXT_CFG.HDSMS bit field.
- Buffer 2 of the first descriptor and subsequent descriptors are used with buffer size indicated by the EMAC_DMA[n]_RXCTL.RBSZ_13_Y/EMAC_DMA[n]_RXCTL.RBSZ_X_0 bit fields.
- When the HL field of the first descriptor (RDES2) is zero, it indicates packet is not split into header and payload.
- When the HL field of the first descriptor (RDES2) is non-zero, it indicates that the packet is split into header and payload.

Programming VLAN Filtering on Receive

Use the following steps to program VLAN filtering on receive:

1. Program the EMAC_VLANTAG_CTL register as follows to select the filtering method.
   a. EMAC_VLANTAG_CTL.ETV – Enable 12-bit VLAN tag comparison or 16-bit VLAN tag comparison.
   b. EMAC_VLANTAG_CTL.VTHM – VLAN tag hash table match enable.
   c. EMAC_VLANTAG_CTL.ERIVLT – Enable inner VLAN tag or outer VLAN tag (to enable the inner or outer VLAN tag filtering, enable double VLAN processing by setting the EMAC_VLANTAG_CTL.EDVLP bit)
   d. EMAC_VLANTAG_CTL.ERSVLM – Enable receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set EMAC_VLANTAG_CTL.ESVL ESVL)
   e. EMAC_VLANTAG_CTL.DOVLTC – Ignores VLAN type for tag match
   f. EMAC_VLANTAG_CTL.VTIM – Enable VLAN tag inverse match instead of the normal VLAN tag matching

2. Program the EMAC_VLANTAG_CTL.ETV bit field to select 12-bit or 16-bit VLAN tag.
3. If hash filtering of VLAN tag is enabled, program the EMAC_VLAN_HASHTBL register.

   ADDITIONAL INFORMATION: When the EMAC_VLANTAG_CTL.ETV bit is reset, upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and used to index the content of the EMAC_VLAN_HASHTBL register. When the ETV bit is set, upper 4 bits of calculated CRC-32 of VLAN tag are used to index the content of the EMAC_VLAN_HASHTBL register. For example, when the EMAC_VLANTAG_CTL.ETV bit is set, a hash value of 4b'1000 selects Bit 8 of the VLAN hash table. When the ETV bit is reset a hash value of 4'b1000 selects bit 7 of the VLAN hash table.

Programming Extended VLAN Filtering and Routing on Receive

For the indirect access of the per VLAN tag registers, follow these steps:

Write

1. Write the required data into the EMAC_VLANTAG_DAT register.
2. Program the EMAC_VLANTAG_CTL.OFS bit field with the required filter register's offset and command type to the EMAC_VLANTAG_CTL.CT bit field. For a write command, set this bit to 0.
3. Write 1 to the EMAC_VLANTAG_CTL.OB bit field and wait until the bit is reset before performing the next write. This is to ensure that the appropriate VLAN tag filter register has been programmed.

Read

1. Program the EMAC_VLANTAG_CTL.OFS bit field with the required register's offset and command type to the EMAC_VLANTAG_CTL.CT bit field. For a read command, set this bit to 1.
2. Write 1 to the EMAC_VLANTAG_CTL.OB bit field and wait until the bit is reset. The appropriate VLAN tag filter register's value is available in the EMAC_VLANTAG_DAT register.

Programming Queue/Channel Based VLAN Inclusion Registers

The following sections provide the procedures for reading and writing the Queue/Channel based VLAN inclusion register.

Writing to the Indirect Registers

To write to the queue/channel based VLAN inclusion register, use the following steps.

1. Set the EMAC_VLAN_INCL.CBTI bit to 1, to enable queue/channel based VLAN tag insertion on all transmitted packets.

   ADDITIONAL INFORMATION: This bit must be set before any indirect access to the queue/channel specific MAC_VLAN_INCL[n] register.
2. Set the EMAC_VLAN_INCL.RDWR bit field to 1, to indicate a write access.

3. Program the VLAN tag and VLAN type to be inserted in packets from a particular queue/channel using the EMAC_VLAN_INCL.VLT and EMAC_VLAN_INCL.CSVL bit fields. The corresponding offset address in the EMAC_VLAN_INCL.ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on) must be set.
4. Set the EMAC_VLAN_INCL.RDWR bit to 1 to indicate write access.

   ADDITIONAL INFORMATION: The write to byte 0 (byte 3 in Big Endian mode) of the EMAC_VLAN_INCL register initiates access to indirect access MAC_VLAN_INCL[n] register.
5. The EMAC_VLAN_INCL.BUSY bit is set by EMAC to indicate the progress of access to indirect access MAC_VLAN_INCL[n] register. On completion of the access, the EMAC_VLAN_INCL.BUSY bit is cleared. The application must not attempt subsequent accesses to the MAC_VLAN_INCL[n] register when this bit is one.
6. Repeat step 3 through 5 to insert VLAN tag and VLAN type in packets from the remaining queues/channels. The application must ensure that the required VLAN tag and VLAN type for all the queues/channels are programmed; otherwise unintended VLAN tag and VLAN type might be inserted.

Reading the Indirect Registers

To read the queue/channel based VLAN inclusion register, complete the following steps:

1. Set the EMAC_VLAN_INCL.CBTI bit to 1, to enable queue/channel based VLAN tag insertion on all the transmitted packets. This bit must be set before indirect access to the queue/channel specific MAC_VLAN_INCL[n] register.
2. Program the read offset address in the EMAC_VLAN_INCL.ADDR ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on). Set the EMAC_VLAN_INCL.RDWR bit to 0, to indicate read access.
3. The EMAC_VLAN_INCL.BUSY field is set by EMAC to indicate the progress of access to the indirect access MAC_VLAN_INCL[n] register. On completion of the access, the BUSY field is cleared. The bits [15:0] and bit [19] of the EMAC_VLAN_INCL register contains the VLAN tag and VLAN type respectively, from the corresponding queue/channel/offset address.
4. Repeat step 2 and step 3 to read VLAN tag and VLAN type from the remaining queues/channels.

Programming EST

Program the gate control values and time intervals in the Software Owned Gate Control List (SWOL) along with the other EST related registers to appropriate values. The following sub-sections provide step by step details for programming the GCL and the other EST related registers.

GCL and GCL Linked Registers

Follow these steps to program the Gate Control List (GCL) and the four other registers implemented per GCL.

The GCL and the four other GCL-linked registers are accessed through indirect addressing using the EMAC_EST_GCL_CTL and EMAC_EST_GCL_DAT registers. The EMAC_EST_STAT.SWOL bit field indicates if GCL0 or GCL1 is owned by software.

1. Write the 32-bit write data to EMAC_EST_GCL_DAT register. Then program the EMAC_EST_GCL_CTL register to write the write address and other control information.

   ADDITIONAL INFORMATION: In the EMAC_EST_GCL_DAT register, write data consists of up to 8 bits (configurable) of gate controls and up to 24 bits (configurable) of time interval. Gate close is indicated by programming a 0 and gate open is indicated by programming a 1. For a 4-TC and 20-bit time interval configuration, the data width is 24 bits and the remaining 8 bits are reserved/read-only. Write the data in the following format: {8'h0, TC3, TC2, TC1, TC0, 20-bit Time Interval} where TCx = 0 or 1.

2. Program the EMAC_EST_GCL_CTL.SRWO bit to 1 (to start a write op) and program the address and R/W fields appropriately.

3. Poll and check for the EMAC_EST_GCL_CTL.SRWO bit to be cleared by hardware to indicate the completion of the previous operation before initiating a new R/W operation using the same indirect addressing mode.

4. Repeat steps 3, 4, 5 until the programming of the GCL is completed.

5. Using the indirect addressing method, program the BTR, CTR, TER and LLR registers. Set the EMAC_EST_GCL_CTL.GCRR bit appropriately. This bit interprets the address field as belonging to these registers (instead of the GCL).

6. After programming of the GCL and the related registers, program the EMAC_EST_CTL register to allow hardware to own and process the GCL.

   ADDITIONAL INFORMATION: When the list length (as indicated in LLR) is 1, the associated time interval should be smaller than the value of the cycle time register. Otherwise, an error is reported (as detailed in the Error handling section) as a single set of gate controls add no value in the TSN context.

   ADDITIONAL INFORMATION: The time unit in all the GCL related registers is seconds and nanoseconds. In cases where internally generated PTP System Time is used, the nano-seconds field must be programmed to use the digital rollover mode (EMAC_TM_CTL.TSCTRLSSR bit field must be set to 1).

EST Registers

After completing the steps described in GCL and GCL Linked Registers, program the EMAC_EST_CTL register.

1. Set the current time offset value (EMAC_EST_CTL.CTOV and time internal left shift fields (EMAC_EST_CTL.TILS appropriately. Also, set the enable EST (EMAC_EST_CTL.EEST) and switch to SWOL (EMAC_EST_CTL.SSWL) bits.

   This enables EMAC to own and process the new GCL and switch to the new GCL at the BTR value. If enabled, EMAC generates an interrupt when the switch to the new list occurs.

   Software must address any other interrupts received during the hardware execution of the GCL

2. Set the EMAC_EST_CTL.SSWL bit field to handoff to the controller.

   The EMAC_EST_CTL.SSWL is reset/cleared by the controller when it successfully switches to the new list. The controller also flips the EMAC_EST_CTL.SSWL bit to indicate the new GCL that the software owns. To

install a new GCL, program the GCL it owns (indicated by SSWL bit) as described in GCL and GCL Linked Registers.

3. Program the EMAC_EST_CTL register as described. Ensure that the new BTR is set to an appropriate value to avoid BTR error that might need software intervention in some cases.

   ADDITIONAL INFORMATION: To avoid transmission overruns, the packet length (frame size) information should always be available. Therefore, in the DMA configurations, program the packet length in the first descriptor of every transmit frame. Similarly, in the MTL configuration, provide the packet length in the control word.

Programming Launch Time in Time-Based Scheduling

The launch time is programmed in the enhanced normal transmit descriptors in DMA configurations and is driven as a control word in MTL configurations as follows.

The OSTC and launch time features are mutually exclusive and should not be used together. If a context descriptor is received with a valid OSTC value immediately before receiving a first normal descriptor with the LTV bit set, then the LTV is ignored.

Programming Media Clock Generation and Recovery

The following sections provide procedures for programming the media clock.

Media Clock Generation

1. Program the appropriate presentation time control (supported generation modes "1001-1011") to the EMAC_PPS_CTL.PPSCTRL_PPSCMD (for 0th instance) and EMAC_PPS_CTL.PPSCMD1 through EMAC_PPS_CTL.PPSCMD3 bits. Program all the EMAC_PPS_CTL.MCGREN0 through EMAC_PPS_CTL.MCGREN3 bits to enable MCGR mode.

   ADDITIONAL INFORMATION: The application must drive the appropriate trigger signal to the corresponding EMAC_PTPAUX_MCG_IN[n] pin, for the instances enabled for media clock generation.

2. Based on the programmed mode, MCGR instance[n] captures the time stamp and programs it in the EMAC_PPS[n]_TGTM_SEC register. Then, the trigger requester event EMAC_MCGR[n] is generated.

   For every request generated by the MCGR instance n, EMAC reads the corresponding EMAC_PPS[n]_TGTM_SEC register and acknowledges the read request by accepting the corresponding completer trigger request.

Media Clock Recovery

1. Enable the Current Presentation Time (CPT) counter by setting the EMAC_TM_CTL.PTGE bit field. In addition to programming the initialization values for the system time, update the EMAC_PRSNTM_UPDT bits with the equivalent presentation time initial value. This is a requirement for the EMAC_TM_CTL.TSINIT bit field to be set.

ADDITIONAL INFORMATION: The increment values used for the system time is also used for the current presentation time. This is because, the increment value is in Sub-seconds and Sub-nanoseconds.

2. Along with programming the update values for the system time, update the EMAC_PRSNTM_UPDT register with the equivalent presentation time update value (32-bit ns) when an update is needed. Finally, set the EMAC_TM_CTL.TSUPDT bit field ensure update is completed.
3. Program the appropriate presentation time control (supported recovery modes "0001 - 0011") to the EMAC_PPS_CTL.PPSCTRL_PPSCMD bit field (for 0th instance) and EMAC_PPS_CTL.PPSCMD1 through EMAC_PPS_CTL.PPSCMD3 bits. Program all the EMAC_PPS_CTL.MCGREN0 through EMAC_PPS_CTL.MCGREN3 bits to enable MCGR mode.
4. For instances enabled for media clock recovery, the corresponding EMAC_MCGR[n] trigger signal is set by the EMAC. For every request generated by EMAC, program the target presentation time into the EMAC_PPS[n]_TGTM_SEC register and acknowledge the request by asserting the corresponding trigger completer signal. Acknowledgment can be given before/after programming the target presentation time.
5. Observe the recovered media clock on the corresponding EMAC_PTPPPS[n] pin.

EMAC Programming Concepts

The following sections provide basic information and guidelines to help with programming the EMAC module.

IEEE 802.3 Ethernet Packet Structure

The ***IEEE 802.3 Frame Structure*** table provides typical frame format of an Ethernet packet. Refer to the IEEE standards for detailed information on Ethernet packets and their format.

Table 30-109: IEEE 802.3 Frame Structure

Parameter	Description	Position in Ethernet Packet	Total Bytes
PREAMBLE	This parameter is a 56-bit (7-byte) pattern of alternating 1 and 0 bits (#10101010), which allows devices on the network to detect a new incoming frame for synchronization.	1	7
SFD	The SFD (#10101011) is a 1-byte pattern designed to break the preamble pattern, and signal the start of the actual frame.	2	1
DA	48-bit destination address. This parameter can be a unicast, multicast, or broadcast address.	3	6
SA	48-bit long source address, typically a unicast, multicast, or broadcast address.	4	6
LT	Typically this field is the length, in terms of the number of bytes, and can be anywhere between 0–1500. When the value is greater than or equal to 0x0600, this field also indicates the type of special payload carried by the frame. Examples include 0x8808 for flow control and 0x0800 for IPv4.	5	2
DATA	Actual application data payload, usually between 0–1500.	6	0–1500

Table 30-109: IEEE 802.3 Frame Structure (Continued)

Parameter	Description	Position in Ethernet Packet	Total Bytes
PAD	This field compensates for data frames that are shorter than 64 bytes long, not including the preamble.	7	0–46
FCS	The frame check sequence is a 32-bit cyclic redundancy check that detects corrupted data within the entire frame. This parameter is generated from a CRC-32 polynomial code (CRC-32-IEEE): G (x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1.	8	4

Frame Size Statistics for Application Software

Table 30-110: Ethernet Frame Size Statistics

Frame size statistics	VLAN-specific change Comments	
Information bytes/Header	4 byte 802.1Q header inserted after source address and before Type/LAN in 802.3 packets = 22 bytes.	6 x 2 + 2 + 4 = 18 bytes (DA+SA+LT+FCS)
Minimum Frame Size (typical)	If DATA is NULL, 42-byte padding makes 64 bytes (42 +22)	64 bytes. If DATA is NULL, 46-byte padding makes 64 bytes (46 +18)
Maximum Frame Size (typical)	1522 bytes	1518 bytes (1500 bytes DATA and 18-bytes header)
Jumbo Frame Size	9022 bytes	Typical industry standard. Ethernet jumbo frame size treated as 9018 bytes.

Software Visualization of Programmable Packet Size

The ***Visualization of Programmable Packet Size*** table provides the byte sizes of packets with various configurations.

Table 30-111: Visualization of Programmable Packet Size

Size in Bytes	Comments
16384	Receive watchdog and transmit jabber disabled, jumbo frames enabled.
10240	Receive watchdog and transmit jabber disabled, jumbo frames disabled.
2048	Receive watchdog and transmit jabber enabled.
1518	Typical max size of Ethernet frame. Receive watchdog and transmit jabber enabled.
64	Typical minimum size of Ethernet frame.
< 64	Runt frames requiring Zero-PAD.

Ethernet Packet Structure in C

The following is an example for Ethernet packet structure in the C language.

```
typedef struct ETHER_PACKET
{
    char  dst_addr[6];              //destination address
    char  src_addr[6];              //source address
    char  length[2];                //length of actual data
    char  data[DATA_SIZE];          //application data
    char  fdlimit[DELIMIT_SIZE];    //32-bit delimit (if manual appending)
    char  fcs[4];                   //crc frame checksum, used by RX buffer.
} ETHER_PACKET;
```

PTP Header Structure in C

The following code is an example of the PTP message format.

```
/* PTP Message Format (Refer to PTP Frame Over IPv4)*/
    typedef struct EMAC_PTP_HEADER
    {
    unsigned char    messageType:4,    //PTP Version 2 message type
    transportSpecific:4;
    unsigned char    versionPTP;       //PTP Version (1 or 2)
    unsigned short    messageLength;
    unsigned char    domainNumber;
    unsigned char    RESERVED1;
    unsigned short    flagField;
    unsigned char    correctionField[8];
    unsigned char    RESERVED2[4];
    unsigned char    sourcePortIdentity[10];
    unsigned short    sequenceid;
    unsigned char    controlField;     //PTP Version 1 message type
    unsigned char    logMessageInterval;
    }EMAC_PTP_HEADER;
```

ADSP-SC59x EMAC Register Descriptions

RGMII, RMII, MII Ethernet Controller (EMAC) contains the following registers.

Table 30-112: ADSP-SC59x EMAC Register List

Name	Description
EMAC_DMA[n]_CTL	DMA Channel n Control Register
EMAC_DMA[n]_RXBUF_CUR	DMA Channel n Current Application Receive Buffer Address Register
EMAC_DMA[n]_RXDSC_CUR	DMA Channel n Current Application Receive Descriptor Register
EMAC_DMA[n]_TXBUF_CUR	DMA Channel n Current Application Transmit Buffer Address Register
EMAC_DMA[n]_TXDSC_CUR	DMA Channel n Current Application Transmit Descriptor Register
EMAC_DMA[n]_IEN	DMA Channel n Interrupt Enable Register

Table 30-112: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_DMA[n]_MISSFRM_CNT	DMA Channel n Missed Frame Counter Register
EMAC_DMA[n]_RXDSC_ADDR	DMA Channel n Receive Descriptor List Address Register
EMAC_DMA[n]_RXDSC_TLPTR	DMA Channel n Receive Descriptor Tail Pointer Register
EMAC_DMA[n]_RXCTL	DMA Channel n Receive Control Register
EMAC_DMA[n]_RXCTL2	DMA Channel n Extended Receive Control Register
EMAC_DMA[n]_RXINTWDTMR	DMA Channel n Receive Interrupt Watchdog Timer Register
EMAC_DMA[n]_SFN_CTLSTAT	DMA Channel n Slot Function Control and Status register
EMAC_DMA[n]_STAT	DMA Channel n Status Register
EMAC_DMA[n]_TXDSC_ADDR	DMA Channel n Transmit Descriptor List Address Register
EMAC_DMA[n]_TXDSC_RLEN	DMA Channel n Transmit Descriptor Ring Length Register
EMAC_DMA[n]_TXDSC_TLPTR	DMA Channel n Transmit Descriptor Tail Pointer Register
EMAC_DMA[n]_TXCTL	DMA Channel n Transmit Control Register
EMAC_DMA_DBG_STAT0	DMA Debug Status 0 Register
EMAC_DMA_DBG_STAT1	DMA Debug Status 1 Register
EMAC_DMA_DBG_STAT2	DMA Debug Status 2 Register
EMAC_DMA_ISTAT	DMA Interrupt Status Register
EMAC_DMA_MODE	DMA Mode Register
EMAC_DMA_SYSBMODE	DMA System Bus Mode Register
EMAC_DMA_TBS_CTL0	DMA TBS Control 0 Register
EMAC_DMA_TBS_CTL1	DMA TBS Control 1 Register
EMAC_DMA_TBS_CTL2	DMA TBS Control 2 Register
EMAC_DMA_TBS_CTL3	DMA TBS Control 3 Register
EMAC_ONEUS_TIC_CNT	LPI One Microsecond Tic Counter Register
EMAC_ARP_ADDR	Address Resolution Protocol (ARP) Register
EMAC_ADDR0_HI	MAC Address 0 High Register
EMAC_ADDR0_LO	MAC Address 0 Low Register
EMAC_ADDR[i]_HI	MAC Address n High Register
EMAC_ADDR[i]_LO	MAC Address n Low Register
EMAC_TM_AUX_CTL	Auxiliary Time Stamp Control Register
EMAC_TM_AUXTS_NSEC	Auxiliary Time Stamp Nanoseconds Register
EMAC_TM_AUXTS_SEC	Auxiliary Time Stamp Seconds Register

Table 30-112: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_CSR_SW_CTL	CSR Software Control Register
EMAC_MAC_CFG	MAC Configuration Register
EMAC_MAC_DBG	Debug Register
EMAC_SPLM_OFST_CFG	MAC Split Mode and Offset Configuration Register
EMAC_MAC_EXT_CFG	MAC Extended Configuration Register
EMAC_FPE_CTLSTAT	Frame Preemption Control and Status Register
EMAC_HW_FTR_PSNT0	Hardware Feature Presence Register 0
EMAC_HW_FTR_PSNT1	Hardware Feature Presence Register 1
EMAC_HW_FTR_PSNT2	Hardware Feature Presence Register 2
EMAC_HW_FTR_PSNT3	Hardware Feature Presence Register 3
EMAC_HASHTBL_REG0	Hash Table Register 0
EMAC_HASHTBL_REG1	Hash Table Register 1
EMAC_HASHTBL_REG2	Hash Table Register 2
EMAC_HASHTBL_REG3	Hash Table Register 3
EMAC_HASHTBL_REG4	Hash Table Register 4
EMAC_HASHTBL_REG5	Hash Table Register 5
EMAC_HASHTBL_REG6	Hash Table Register 6
EMAC_HASHTBL_REG7	Hash Table Register 7
EMAC_INDR_ACC_CTL	Indirect Access Control Register
EMAC_INDR_ACC_DAT	Indirect Access Data Register
EMAC_INVLAN_INCL	Inner VLAN Tag Inclusion or Replacement Register
EMAC_MAC_IEN	MAC Interrupt Enable Register
EMAC_MAC_ISTAT	MAC Interrupt Status Register
EMAC_L3L4_CTL7	Layer 3 and Layer 4 Control Register
EMAC_L3L4_CTL[i]	Layer 3 and Layer 4 Control Register
EMAC_LPI_CTLSTAT	LPI Control and Status Register
EMAC_LPI_ENTR_TMR	Transmit LPI Entry Timer Register
EMAC_LPI_TMRSCTL	LPI Timers Control Register
EMAC_L3_ADDR0_REG7	Layer 3 Address 0 Register 7
EMAC_L3_ADDR0_REG[i]	Layer 3 Address 0 Register [i]
EMAC_L3_ADDR1_REG7	Layer 3 Address 1 Register 7

Table 30-112: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_L3_ADDR1_REG[i]	Layer 3 Address 1 Register [i]
EMAC_L3_ADDR2_REG7	Layer 3 Address 2 Register 7
EMAC_L3_ADDR2_REG[i]	Layer 3 Address 2 Register [i]
EMAC_L3_ADDR3_REG7	Layer 3 Address 3 Register 7
EMAC_L3_ADDR3_REG[i]	Layer 3 Address 3 Register [i]
EMAC_L4_ADDR7	Layer 4 Address Register [i]
EMAC_L4_ADDR[i]	Layer 4 Address Register [i]
EMAC_LOG_MSG_INTVL	PTP Log Message Interval Register
EMAC_MDIO_ADDR	MDIO Address Register
EMAC_MDIO_DATA	MDIO Data Register
EMAC_PHYIF_CTLSTAT	PHY Interface Control and Status Register
EMAC_PPS[n]_INTVL	PPS n Interval Register
EMAC_PPS[n]_TGTM_NSEC	PPS n Target Time Nanoseconds Register
EMAC_PPS[n]_TGTM_SEC	PPS n Target Time Seconds Register
EMAC_PPS[n]_WID	PPS n Width Register
EMAC_PPS_CTL	PPS Control Register
EMAC_PTO_CTL	PTP Time Stamp Offload Control Register
EMAC_MACPKT_FILT	MAC Packet Filter Register
EMAC_PRSNTM_UPDT	Presentation Time Update Register
EMAC_PRSNTM_NSEC	Presentation Time Nanosecond Register
EMAC_Q0_TXFLOW_CTL	Queue 0 Transmit Flow Control Register
EMAC_Q[i]_TXFLOW_CTL	Queue [i] Transmit Flow Control Register
EMAC_RXQ_CTL0	Receive Queue Control 0 Register
EMAC_RXQ_CTL1	Receive Queue Control 1 Register
EMAC_RXQ_CTL2	Receive Queue Control 2 Register
EMAC_RXQ_CTL3	Receive Queue Control 3 Register
EMAC_RXQ_CTL4	Receive Queue Control 4 Register
EMAC_RXFLOW_CTL	Receive Flow Control Register
EMAC_RXTX_STAT	Receive Transmit Status Register
EMAC_SRCPRT_IDNTY0	Source Port Identity 0 Register
EMAC_SRCPRT_IDNTY1	Source Port Identity 1 Register

Table 30-112: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_SRCPRT_IDNTY2	Source Port Identity 2 Register
EMAC_TM_SUBSEC	Subsecond Increment Register
EMAC_TM_HISEC	System Time Higher Word Seconds Register
EMAC_TM_NSEC	System Time Nanoseconds Register
EMAC_TM_NSECUPDT	System Time Nanoseconds Update Register
EMAC_TM_SEC	System Time Seconds Register
EMAC_TM_SECUPDT	System Time Seconds Update Register
EMAC_TM_ADDEND	Time Stamp Addend Register
EMAC_TM_CTL	Time Stamp Control Register
EMAC_TM_EGASYM_CORR	Time Stamp Egress Asymmetry Correction Register
EMAC_TM_EGCORR_NSEC	Time Stamp Egress Correction Nanosecond Register
EMAC_TM_EGCORR_SNSEC	Time Stamp Egress Correction Sub Nanosecond Register
EMAC_TM_EG_LTNCY	Time Stamp Egress Latency Register
EMAC_TM_INGASYM_CORR	Time Stamp Ingress Asymmetry Correction Register
EMAC_TM_INGCORR_NSEC	Time Stamp Ingress Correction Nanosecond Register
EMAC_TM_INGCORR_SNSEC	Time Stamp Ingress Correction Sub Nanosecond Register
EMAC_TM_ING_LTNCY	Time Stamp Ingress Latency Register
EMAC_TM_STAT	Time Stamp Status Register
EMAC_TXQ_PRTY_MAP0	Transmit Queue Priority Mapping 0 Register
EMAC_TXQ_PRTY_MAP1	Transmit Queue Priority Mapping 1 Register
EMAC_TXTMSTAT_NSEC	Transmit Time Stamp Status Nanoseconds Register
EMAC_TXTMSTAT_SEC	Transmit Time Stamp Status Seconds Register
EMAC_VLAN_HASHTBL	VLAN Hash Table Register
EMAC_VLAN_INCL	VLAN Tag Inclusion or Replacement Register
EMAC_VLANTAG_CTL	VLAN Tag Control Register
EMAC_VLANTAG_DAT	VLAN Tag Indirect Access Data Register
EMAC_WDT_TMOUT	Watchdog Timeout Register
EMAC_MMC_CTRL	MMC Control Register
EMAC_MMC_FPE_RXINT	MMC Receive FPE Interrupt Status Register
EMAC_MMC_FPE_RXIMSK	MMC Receive FPE Interrupt Mask Register
EMAC_MMC_FPE_TXINT	MMC Transmit FPE Interrupt Status Register

Table 30-112: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_MMC_FPE_TXIMSK	MMC Transmit FPE Interrupt Mask Register
EMAC_MMC_IPC_RXINT	MMC Receive IPC Interrupt Status Register
EMAC_MMC_IPC_RXIMSK	MMC Receive IPC Interrupt Mask Register
EMAC_RXFPE_FRGMNT_CNT	MMC Receive FPE Fragment Counter Register
EMAC_MMC_RXINT	MMC Receive Interrupt Status Register
EMAC_MMC_RXIMSK	MMC Receive Interrupt Mask Register
EMAC_RXPKT_ASERR_CNT	MMC Receive FPE Packet Assembly Error Counter Register
EMAC_RXPKT_ASOK_CNT	MMC Receive FPE Packet Assembly Okay Counter Register
EMAC_RXPKT_SMDERR_CNT	MMC Receive FPE Packet SMD Error Counter Register
EMAC_TXFPE_FRGMNT_CNT	MMC Transmit FPE Packet Fragment Counter Register
EMAC_TXHLDRQ_CNT	MMC Transmit FPE Hold Request Counter Register
EMAC_MMC_TXINT	MMC Transmit Interrupt Status Register
EMAC_MMC_TXIMSK	MMC Transmit Interrupt Mask Register
EMAC_EST_CTL	EST Control Register
EMAC_EST_EXT_CTL	EST Extended Control Register
EMAC_EST_FRMSZ_CAP	EST Frame Size Capture Register
EMAC_EST_FRMSZ_ERR	EST Frame Size Error Register
EMAC_EST_GCL_CTL	EST Gate Control List Control Register
EMAC_EST_GCL_DAT	EST Gate Control List Data Register
EMAC_EST_IEN	EST Interrupt Enable Register
EMAC_EST_SCH_ERR	EST Scheduling Error Register
EMAC_EST_STAT	EST Status Register
EMAC_FPE_ADV	Frame Preemption Advance Register
EMAC_FPE_CTRLSTS	MTL Frame Preemption Control and Status Register
EMAC_MTL_ISTAT	MTL Interrupt Status Register
EMAC_MTL_OPMODE	MTL Operation Mode Register
EMAC_Q0_INT_CTLSTAT	Queue 0 Interrupt Control and Status Register
EMAC_Q[n]_INT_CTLSTAT	Queue n Interrupt Control and Status Register
EMAC_RQ0_CTL	Receive Queue Control Register
EMAC_RQ0_DBG	Receive Queue 0 Debug Register
EMAC_RQ0_MSPKTOF_CNT	Receive Queue 0 Missed Packet and Overflow Counter Register

Table 30-112: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_RQ0_OPMODE	Receive Queue 0 Operation Mode Register
EMAC_RQ[n]_CTL	Receive Queue n Control Register
EMAC_RQ[n]_DBG	Receive Queue n Debug Register
EMAC_RQ[n]_MSPKTOF_CNT	Receive Queue n Missed Packet and Overflow Counter Register
EMAC_RQ[n]_OPMODE	Receive Queue n Operation Mode Register
EMAC_RXQ_DMA_MAP0	Receive Queue to DMA Channel Mapping Register 0
EMAC_RXQ_DMA_MAP1	Receive Queue to DMA Channel Mapping Register 1
EMAC_TBS_CTRL	MTL Time Based Scheduling Control Register
EMAC_TQ0_DBG	Transmit Queue 0 Debug Register
EMAC_TQ0_ETS_STAT	Transmit Queue 0 ETS Status Register
EMAC_TQ0_OPMODE	Transmit Queue 0 Operation Mode Register
EMAC_TQ0_QTMWGT	Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register
EMAC_TQ0_URFL	Transmit Queue 0 Underflow Packet Counter Register
EMAC_TQ[n]_DBG	Transmit Queue n Debug Register
EMAC_TQ[n]_ETS_CTL	Transmit Queue n ETS Control Register
EMAC_TQ[n]_ETS_STAT	Transmit Queue n ETS Status Register
EMAC_TQ[n]_HICRDT	Transmit Queue n CBS HiCredit Register
EMAC_TQ[n]_LOCRDT	Transmit Queue n CBS LoCredit Register
EMAC_TQ[n]_OPMODE	Transmit Queue n Operation Mode Register
EMAC_TQ[n]_QTMWGT	Transmit Queue n IdleSlopeCredit, Quantum or Weights Register
EMAC_TQ[n]_SSCRDT	Transmit Queue n SendSlopeCredit Register
EMAC_TQ[n]_URFL	Transmit Queue n Underflow Packet Counter Register
EMAC_RXICMP_ERR_OCNT	Receive ICMP Error Octets Counter Register
EMAC_RXICMP_ERR_PCNT	Receive ICMP Error Packets Counter Register
EMAC_RXICMP_OCNT_G	Receive ICMP Good Octets Counter Register
EMAC_RXICMP_PCNT_G	Receive ICMP Good Packets Counter Register
EMAC_RXIPV4_FRGMT_OCNT	Receive IPv4 Fragmented Octets Counter Register
EMAC_RXIPV4_FRGMT_PCNT	Receive IPv4 Fragmented Packets Counter Register
EMAC_RXIPV4_OCNT_G	Receive IPv4 Good Octets Counter Register
EMAC_RXIPV4_PCNT_G	Receive IPv4 Good Packets Counter Register
EMAC_RXIPV4_HDRERR_OCNT	Receive IPv4 Header Error Octets Counter Register

Table 30-112: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_RXIPV4_HDRERR_PCNT	Receive IPv4 Header Error Packets Counter Register
EMAC_RXIPV4_PYLD_OCNT	Receive IPv4 Payload Octets Counter Register
EMAC_RXIPV4_PYLD_PCNT	Receive IPv4 Payload Packets Counter Register
EMAC_RXIPV4_UDPCSD_OCNT	Receive IPv4 UDP Checksum Disable Octets Counter Register
EMAC_RXIPV4_UDPCSD_PCNT	Receive IPv4 UDP Checksum Disable Packets Counter Register
EMAC_RXIPV6_OCNT_G	Receive IPv6 Good Octets Counter Register
EMAC_RXIPV6_PCNT_G	Receive IPv6 Good Packets Counter Register
EMAC_RXIPV6_HDRERR_OCNT	Receive IPv6 Header Error Octets Counter Register
EMAC_RXIPV6_HDRERR_PCNT	Receive IPv6 Header Error Packets Counter Register
EMAC_RXIPV6_PYLD_OCNT	Receive IPv6 Payload Octets Counter Register
EMAC_RXIPV6_PYLD_PCNT	Receive IPv6 Payload Packets Counter Register
EMAC_RXTCP_ERR_OCNT	Receive TCP Error Octets Counter Register
EMAC_RXTCP_ERR_PCNT	Receive TCP Error Packets Counter Register
EMAC_RXTCP_OCNT_G	Receive TCP Good Octets Counter Register
EMAC_RXTCP_PCNT_G	Receive TCP Good Packets Counter Register
EMAC_RXUDP_ERR_OCNT	Receive UDP Error Octets Counter Register
EMAC_RXUDP_ERR_PCNT	Receive UDP Error Packets Counter Register
EMAC_RXUDP_OCNT_G	Receive UDP Good Octets Counter Register
EMAC_RXUDP_PCNT_G	Receive UDP Good Packets Counter Register
EMAC_RX1024TOMAX_GB	Receive 1024 - Maximum Octets Packets Good Bad Counter Register
EMAC_RX128TO255_GB	Receive 128 - 255 Octets Packets Good Bad Counter Register
EMAC_RX256TO511_GB	Receive 256 - 511 Octets Packets Good Bad Counter Register
EMAC_RX512TO1023_GB	Receive 512 - 1023 Octets Packets Good Bad Counter Register
EMAC_RX64_GB	Receive 64 Octets Packets Good Bad Counter Register
EMAC_RX65TO127_GB	Receive 65 - 127 Octets Packets Good Bad Counter Register
EMAC_RX_ALGNERR_PCNT	Receive Alignment Error Packets Counter Register
EMAC_RX_BCERR_PCNT_G	Receive Broadcast Packets Good Counter Register
EMAC_RX_CRCERR_PCNT	Receive CRC Error Packets Counter Register
EMAC_RX_CTL_PCNT_G	Receive Control Packets Good Counter Register
EMAC_RX_FIFOOVFL_PCNT	Receive FIFO Overflow Packets Counter Register
EMAC_RX_JBERR_PCNT	Receive Jabber Error Packets Counter Register

Table 30-112: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_RX_LPITRAN_CNT	Receive LPI Transition Counter Register
EMAC_RX_LPIUSEC_CNT	Receive LPI Microseconds Counter Register
EMAC_RX_LENERR_PCNT	Receive Length Error Packets Counter Register
EMAC_RX_MCST_PCNT_G	Receive Multicast Packets Good Counter Register
EMAC_RX_OCNT_G	Receive Octet Good Counter Register
EMAC_RX_OCNT_GB	Receive Octet Good Bad Counter Register
EMAC_RX_ORTYPE_PCNT	Receive Out of Range Type Packets Counter Register
EMAC_RX_OVRSZ_PCNT	Receive Oversize Packets Good Counter Register
EMAC_RX_PCNT_GB	Receive Packets Good Bad Counter Register
EMAC_RX_PAUSE_PCNT	Receive Pause Packets Counter Register
EMAC_RX_ERR_PCNT	Receive Error Packets Counter Register
EMAC_RX_RNTERR_PCNT	Receive Runt Error Packets Counter Register
EMAC_RX_UNDRSZ_PCNT_G	Receive Undersize Packets Good Counter Register
EMAC_RX_UCST_PCNT_G	Receive Unicast Packets Good Counter Register
EMAC_RX_VLAN_PCNT_GB	Receive VLAN Packets Good Bad Counter Register
EMAC_RX_WDTERR_PCNT	Receive Watchdog Error Packets Counter Register
EMAC_TX1024TOMAX_GB	Transmit 1024-Max Octets Packets Good Bad Counter Register
EMAC_TX128TO255_GB	Transmit 128-255 Octets Packets Good Bad Counter Register
EMAC_TX256TO511_GB	Transmit 256-511 Octets Packets Good Bad Counter Register
EMAC_TX512TO1023_GB	Transmit 512-1023 Octets Packets Good Bad Counter Register
EMAC_TX64_GB	Transmit 64 Octets Packets Good Bad Counter Register
EMAC_TX65TO127_GB	Transmit 65-127 Octets Packets Good Bad Counter Register
EMAC_TX_BCST_PCNT_G	Transmit Broadcast Packets Good Counter Register
EMAC_TX_BCST_PCNT_GB	Transmit Broadcast Packets Good Bad Counter Register
EMAC_TX_CRERR_PCNT	Transmit Carrier Error Packets Counter Register
EMAC_TX_DFRD_PCNT	Transmit Deferred Packets Counter Register
EMAC_TX_EXCOL_PCNT	Transmit Excessive Collision Packets Counter Register
EMAC_TX_EXDFRL_PCNT	Transmit Excessive Deferral Error Counter Register
EMAC_TX_LPITRAN_CNT	Transmit LPI Transition Counter Register
EMAC_TX_LPIUSEC_CNT	Transmit LPI Microseconds Counter Register
EMAC_TX_LTCOL_PCNT	Transmit Late Collision Packets Counter Register

Table 30-112: ADSP-SC59x EMAC Register List (Continued)

Name	Description
EMAC_TX_MCST_PCNT_G	Transmit Multicast Packets Good Counter Register
EMAC_TX_MCST_PCNT_GB	Transmit Multicast Packets Good Bad Counter Register
EMAC_TX_MCOL_PCNT_G	Transmit Multiple Collision Packets Good Counter Register
EMAC_TX_OSIZE_PCNT_G	Transmit Osize Packets Good Counter Register
EMAC_TX_OCNT_G	Transmit Octet Good Counter Register
EMAC_TX_OCNT_GB	Transmit Octet Good Bad Counter Register
EMAC_TX_PCNT_G	Transmit Packet Good Counter Register
EMAC_TX_PCNT_GB	Transmit Packet Good Bad Counter Register
EMAC_TX_PAUSE_PCNT	Transmit Pause Packets Counter Register
EMAC_TX_SNGLCOL_PCNT_G	Transmit Single Collision Packets Good Counter Register
EMAC_TX_URFL_PCNT	Transmit Underflow Error Packets Counter Register
EMAC_TX_UCST_PCNT_GB	Transmit Unicast Packets Good Bad Counter Register
EMAC_TX_VLAN_PCNT_G	Transmit VLAN Packets Good Counter Register

DMA Channel n Control Register

The EMAC_DMA[n]_CTL register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

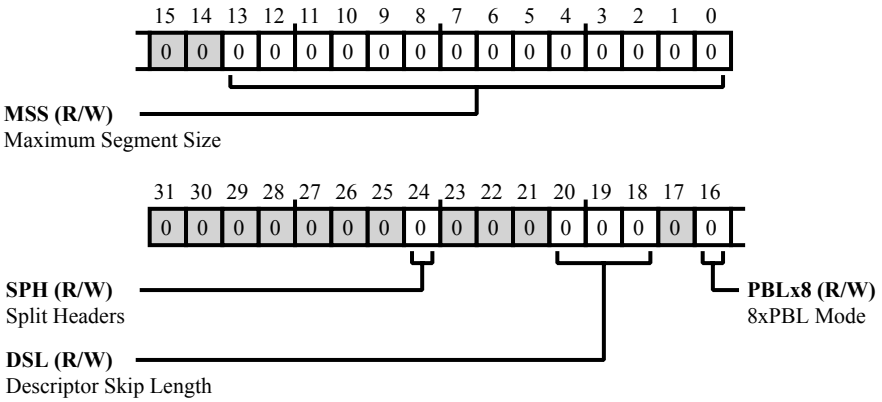


Figure 30-58: EMAC_DMA[n]_CTL Register Diagram

Table 30-113: EMAC_DMA[n]_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	SPH	Split Headers. When the EMAC_DMA[n]_CTL.SPH bit is set (=1), the DMA splits the header and payload in the receive path. The DMA writes the header to Buffer Address1 of RDES0. The DMA writes the payload to the buffer to which the Buffer Address2 is pointing. The software must ensure that the header fits into the receive buffers. If the header length exceeds the receive buffer size, the DMA does not split the header and payload. The EMAC_DMA[n]_CTL.SPH bit is available only when the split header feature is enabled (EMAC_HW_FTR_PSNT1.SPHEN =1).	
		0	Split Headers Feature is Disabled
		1	Split Headers Feature is Enabled
20:18 (R/W)	DSL	Descriptor Skip Length. The EMAC_DMA[n]_CTL.DSL bit specifies the number of 32-bit words to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor. When EMAC_DMA[n]_CTL.DSL = 0, the DMA takes the descriptor table as contiguous.	

Table 30-113: EMAC_DMA[n]_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	PBLX8	8xPBL Mode. When the EMAC_DMA[n]_CTL.PBLX8 bit is set, the PBL value programmed in bits[21:16] in EMAC_DMA[n]_TXCTL and bits[21:16] in EMAC_DMA[n]_RXCTL is multiplied by eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.	
		0	8xPBL Mode is Disabled
		1	8xPBL Mode is Enabled
13:0 (R/W)	MSS	Maximum Segment Size. The EMAC_DMA[n]_CTL.MSS field specifies the maximum segment size to be used while segmenting the packet. The EMAC_DMA[n]_CTL.MSS field is valid only when EMAC_DMA[n]_TXCTL.TSE =1. The value programmed in this field must be more than the configured data width in bytes. Use a EMAC_DMA[n]_CTL.MSS value of 64 bytes or more.	

DMA Channel n Current Application Receive Buffer Address Register

The EMAC_DMA[n]_RXBUF_CUR register points to the current Rx buffer address read by the DMA.

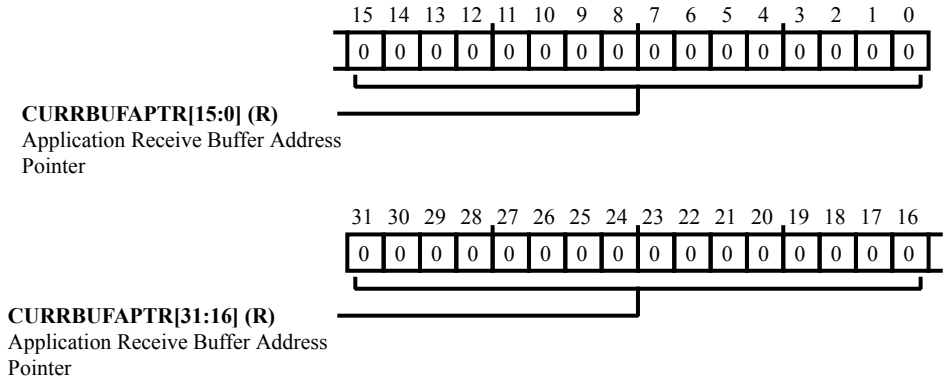


Figure 30-59: EMAC_DMA[n]_RXBUF_CUR Register Diagram

Table 30-114: EMAC_DMA[n]_RXBUF_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CURRBUFAPTR	Application Receive Buffer Address Pointer. The DMA updates the EMAC_DMA[n]_RXBUF_CUR.CURRBUFAPTR field during the Rx operation. This pointer is cleared on reset.

DMA Channel n Current Application Receive Descriptor Register

The EMAC_DMA[n]_RXDSC_CUR register points to the current receive descriptor read by the DMA.

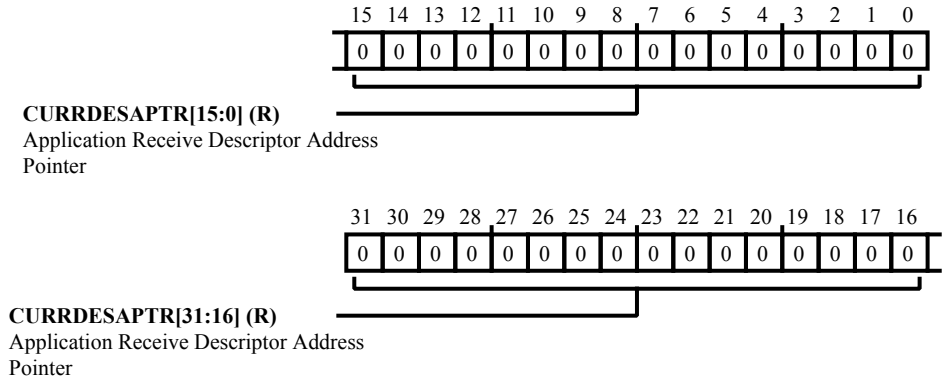


Figure 30-60: EMAC_DMA[n]_RXDSC_CUR Register Diagram

Table 30-115: EMAC_DMA[n]_RXDSC_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CURRDESAPTR	Application Receive Descriptor Address Pointer. The DMA updates the EMAC_DMA[n]_RXDSC_CUR.CURRDESAPTR field during the Rx operation. This pointer is cleared on reset.

DMA Channel n Current Application Transmit Buffer Address Register

The EMAC_DMA[n]_TXBUF_CUR register points to the current Tx buffer address read by the DMA.

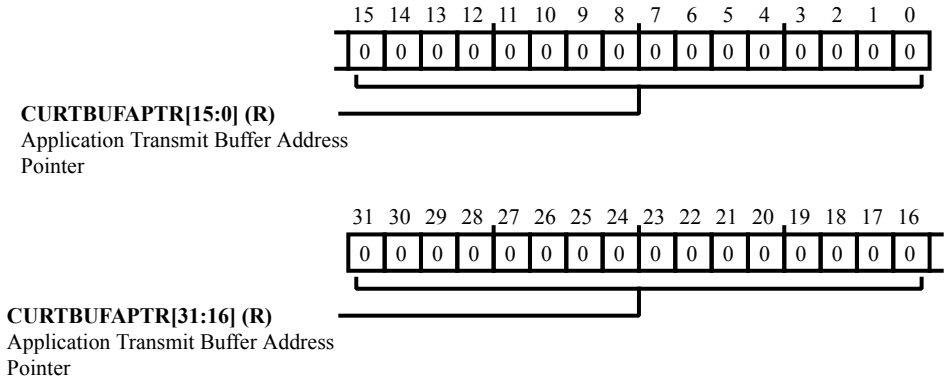


Figure 30-61: EMAC_DMA[n]_TXBUF_CUR Register Diagram

Table 30-116: EMAC_DMA[n]_TXBUF_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CURTBUFAPTR	Application Transmit Buffer Address Pointer. The DMA updates the EMAC_DMA[n]_TXBUF_CUR.CURTBUFAPTR field during the Tx operation. This pointer is cleared on reset.

DMA Channel n Current Application Transmit Descriptor Register

The EMAC_DMA[n]_TXDSC_CUR register points to the current transmit descriptor read by the DMA.

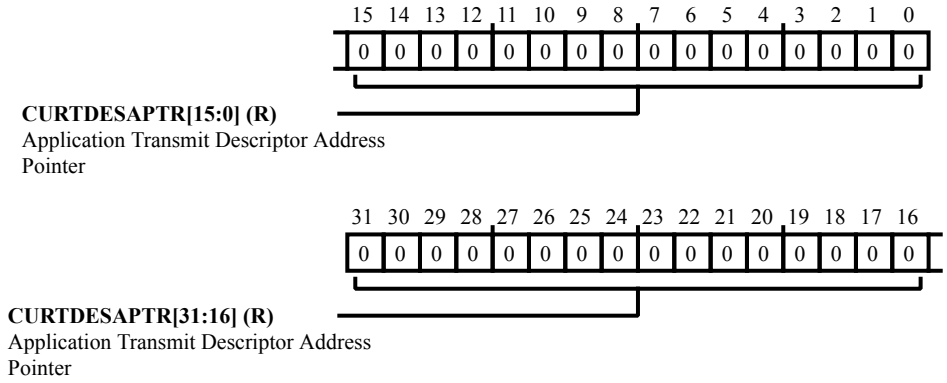


Figure 30-62: EMAC_DMA[n]_TXDSC_CUR Register Diagram

Table 30-117: EMAC_DMA[n]_TXDSC_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CURTDESAPTR	Application Transmit Descriptor Address Pointer. The DMA updates the EMAC_DMA[n]_TXDSC_CUR.CURTDESAPTR field during the Tx operation. This pointer is cleared on reset.

DMA Channel n Interrupt Enable Register

The EMAC_DMA[n]_IEN register enables the interrupts reported by the status register.

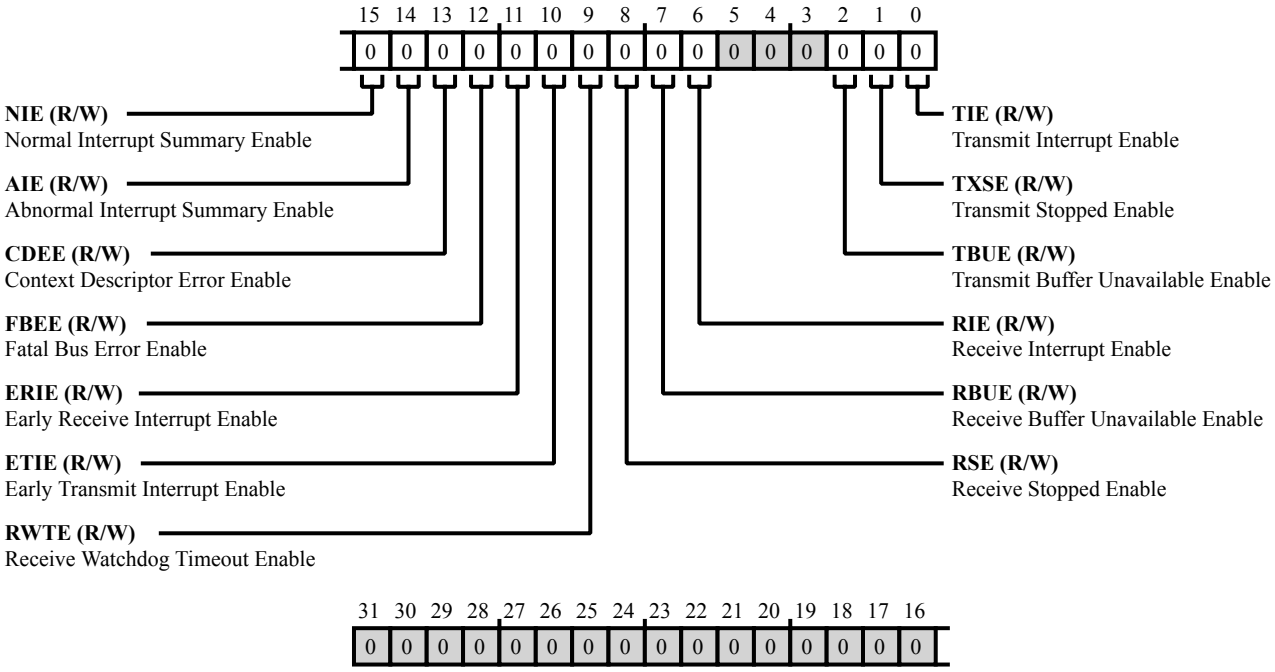


Figure 30-63: EMAC_DMA[n]_IEN Register Diagram

Table 30-118: EMAC_DMA[n]_IEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	NIE	Normal Interrupt Summary Enable. When the EMAC_DMA[n]_IEN.NIE bit is set (=1), the normal interrupt summary is enabled. The EMAC_DMA[n]_IEN.NIE bit enables the following interrupts in the EMAC_DMA[n]_STAT register: - Bit 0: Transmit Interrupt - Bit 2: Transmit Buffer Unavailable - Bit 6: Receive Interrupt - Bit 11: Early Receive Interrupt When EMAC_DMA[n]_IEN.NIE = 0, the normal interrupt summary is disabled.	
		0	Disable
		1	Enable

Table 30-118: EMAC_DMA[n]_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14 (R/W)	AIE	Abnormal Interrupt Summary Enable. When the EMAC_DMA[n]_IEN.AIE bit is set, the abnormal interrupt summary is enabled. The EMAC_DMA[n]_IEN.AIE bit enables the following interrupts in the EMAC_DMA[n]_STAT register: - Bit 1: Transmit Process Stopped - Bit 7: Rx Buffer Unavailable - Bit 8: Receive Process Stopped - Bit 9: Receive Watchdog Timeout - Bit 10: Early Transmit Interrupt - Bit 12: Fatal Bus Error - Bit 13: Context Descriptor Error When EMAC_DMA[n]_IEN.AIE = 0, the abnormal interrupt summary is disabled.	
		0	Disable
		1	Enable
13 (R/W)	CDEE	Context Descriptor Error Enable. When EMAC_DMA[n]_IEN.CDEE =1 and EMAC_DMA[n]_IEN.AIE =1, the descriptor error interrupt is enabled. When EMAC_DMA[n]_IEN.CDEE = 0, the descriptor error interrupt is disabled.	
		0	Disable
		1	Enable
12 (R/W)	FBEE	Fatal Bus Error Enable. When EMAC_DMA[n]_IEN.FBEE =1 and EMAC_DMA[n]_IEN.AIE =1, the fatal bus error interrupt is enabled. When EMAC_DMA[n]_IEN.FBEE = 0, the fatal bus error interrupt is disabled.	
		0	Disable
		1	Enable
11 (R/W)	ERIE	Early Receive Interrupt Enable. When EMAC_DMA[n]_IEN.ERIE =1 and EMAC_DMA[n]_IEN.NIE =1, the early receive interrupt is enabled. When EMAC_DMA[n]_IEN.ERIE = 0, the early receive interrupt is disabled.	
		0	Disable
		1	Enable

Table 30-118: EMAC_DMA[n]_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	ETIE	Early Transmit Interrupt Enable. When EMAC_DMA[n]_IEN.ETIE =1 and EMAC_DMA[n]_IEN.AIE =1, the early transmit interrupt is enabled. When EMAC_DMA[n]_IEN.ETIE = 0, the early transmit interrupt is disabled.	
		0	Disable
		1	Enable
9 (R/W)	RWTE	Receive Watchdog Timeout Enable. When EMAC_DMA[n]_IEN.RWTE =1 and EMAC_DMA[n]_IEN.AIE =1, the receive watchdog timeout interrupt is enabled. When EMAC_DMA[n]_IEN.RWTE = 0, the receive watchdog timeout interrupt is disabled.	
		0	Disable
		1	Enable
8 (R/W)	RSE	Receive Stopped Enable. When EMAC_DMA[n]_IEN.RSE =1 and EMAC_DMA[n]_IEN.AIE =1, the receive stopped interrupt is enabled. When EMAC_DMA[n]_IEN.RSE = 0, the receive stopped interrupt is disabled.	
		0	Disable
		1	Enable
7 (R/W)	RBUE	Receive Buffer Unavailable Enable. When EMAC_DMA[n]_IEN.RBUE =1 and EMAC_DMA[n]_IEN.AIE =1, the receive buffer unavailable interrupt is enabled. When EMAC_DMA[n]_IEN.RBUE = 0, the receive buffer unavailable interrupt is disabled.	
		0	Disable
		1	Enable
6 (R/W)	RIE	Receive Interrupt Enable. When EMAC_DMA[n]_IEN.RIE =1 and EMAC_DMA[n]_IEN.NIE =1, the receive interrupt is enabled. When EMAC_DMA[n]_IEN.RIE = 0, the receive interrupt is disabled.	
		0	Disable
		1	Enable

Table 30-118: EMAC_DMA[n]_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	TBUE	Transmit Buffer Unavailable Enable. When `EMAC_DMA[n]_IEN.TBUE` =1 and EMAC_DMA[n]_IEN.NIE =1, the transmit buffer unavailable interrupt is enabled. When `EMAC_DMA[n]_IEN.TBUE` = 0, the transmit buffer unavailable interrupt is disabled.	
		0	Disable
		1	Enable
1 (R/W)	TXSE	Transmit Stopped Enable. When `EMAC_DMA[n]_IEN.TXSE` =1 and EMAC_DMA[n]_IEN.AIE =1, the transmission stopped interrupt is enabled. When `EMAC_DMA[n]_IEN.TXSE` = 0, the transmission stopped interrupt is disabled.	
		0	Disable
		1	Enable
0 (R/W)	TIE	Transmit Interrupt Enable. When `EMAC_DMA[n]_IEN.TIE` =1 and EMAC_DMA[n]_IEN.NIE =1, the transmit interrupt is enabled. When `EMAC_DMA[n]_IEN.TIE` = 0, the transmit interrupt is disabled.	
		0	Disable
		1	Enable

DMA Channel n Missed Frame Counter Register

The EMAC_DMA[n]_MISSFRM_CNT register indicates the number of frames that the DMA dropped due to a bus error or due to programming the EMAC_DMA[n]_RXCTL.RPF bit.

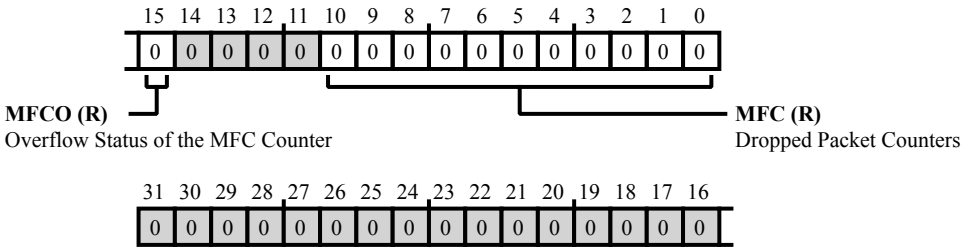


Figure 30-64: EMAC_DMA[n]_MISSFRM_CNT Register Diagram

Table 30-119: EMAC_DMA[n]_MISSFRM_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	MFCO	Overflow Status of the MFC Counter. When EMAC_DMA[n]_MISSFRM_CNT.MFCO bit is set (=1), the MFC counter does not get incremented further. The bit is cleared (=0) when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Miss Frame Counter Overflow Not Occurred
		1	Miss Frame Counter Overflow Occurred
10:0 (R/NW)	MFC	Dropped Packet Counters. The EMAC_DMA[n]_MISSFRM_CNT.MFC field indicates the number of frames that are dropped by the DMA either because of a bus error or because of programing the EMAC_DMA[n]_RXCTL.RPF bit. The counter is cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.	

DMA Channel n Receive Descriptor List Address Register

The EMAC_DMA[n]_RXDSC_ADDR register points the DMA to the start of the receive descriptor list.

The descriptor lists reside in the physical memory space of the application. It must be 32-bit aligned. The DMA internally converts it to bus width aligned address by making the corresponding least significant bits low.

Writing to the EMAC_DMA[n]_RXDSC_ADDR register is permitted only when reception is stopped. When stopped, this register must be written to before the receive start command is given.

Write to this register only when Rx DMA has stopped (EMAC_DMA[n]_RXCTL.SR = 0). When stopped, this register can be written with a new descriptor list address.

When EMAC_DMA[n]_RXCTL.SR = 1, the DMA takes the newly programmed descriptor base address.

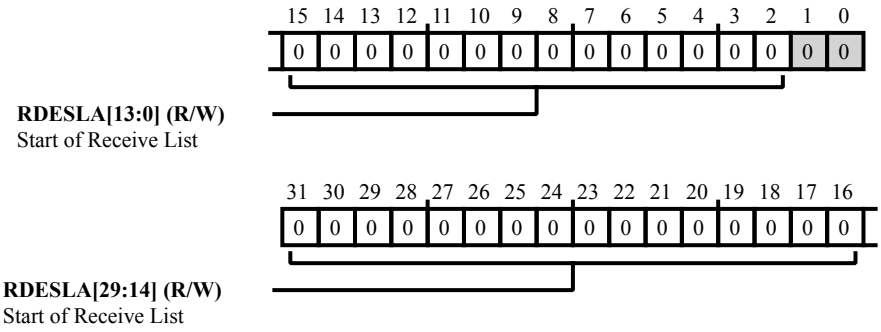


Figure 30-65: EMAC_DMA[n]_RXDSC_ADDR Register Diagram

Table 30-120: EMAC_DMA[n]_RXDSC_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	RDESLA	Start of Receive List. The EMAC_DMA[n]_RXDSC_ADDR.RDESLA field contains the base address of the first descriptor in the Rx descriptor list. The DMA ignores the last two bits (1:0) and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

DMA Channel n Receive Descriptor Tail Pointer Register

The EMAC_DMA[n]_RXDSC_TLPTR register indicates an offset from the base and indicates the location of the last valid descriptor.

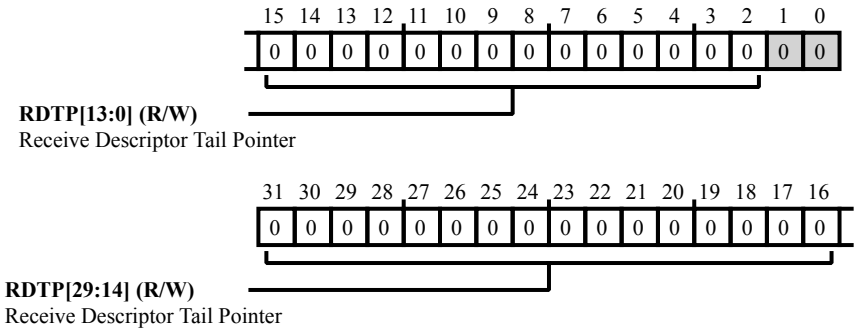


Figure 30-66: EMAC_DMA[n]_RXDSC_TLPTR Register Diagram

Table 30-121: EMAC_DMA[n]_RXDSC_TLPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	RDTP	Receive Descriptor Tail Pointer. The EMAC_DMA[n]_RXDSC_TLPTR.RDTP field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

DMA Channel n Receive Control Register

The EMAC_DMA[n]_RXCTL register controls the Rx features such as PBL, buffer size, and extended status.

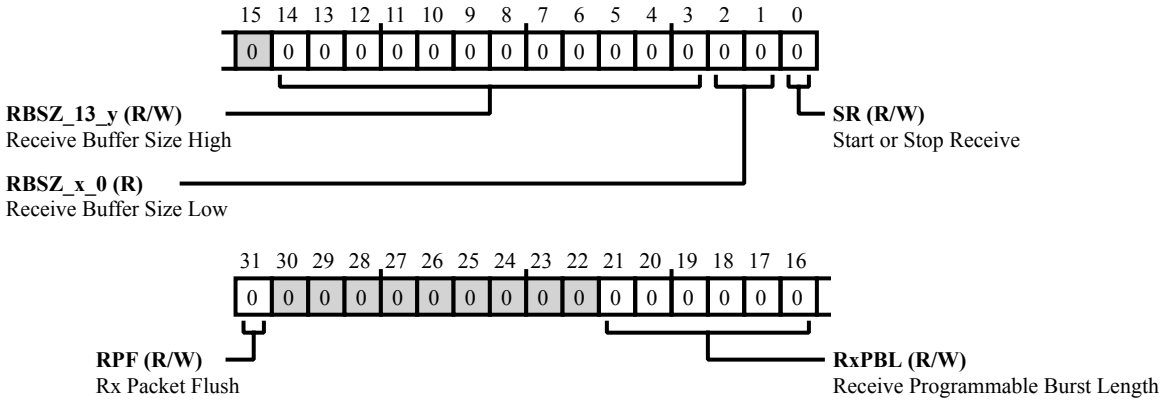


Figure 30-67: EMAC_DMA[n]_RXCTL Register Diagram

Table 30-122: EMAC_DMA[n]_RXCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	RPF	Rx Packet Flush. The EMAC_DMA[n]_RXCTL.RPF bit enables the Rx packet flush feature. When set (= 1), the EMAC module automatically flushes the packet from the Rx queues designated to this DMA Rx channel, when it is stopped. When the EMAC_DMA[n]_RXCTL.RPF bit remains set and the DMA is restarted by the software driver, the packets residing in the Rx queues that were received when this RxDMA was stopped, are flushed out. The packets that are received by the MAC after the RxDMA is restarted are routed to the RxDMA. The flushing is done on the read side of the Rx queue. When the EMAC_DMA[n]_RXCTL.RPF bit is cleared (= 0), the EMAC module does not flush the packet in the Rx queue destined to this RxDMA channel when it is in STOP state. This operation can cause head-of-line blocking in the corresponding Rx queue. Note: The stopping of packet flow from a Rx DMA channel to the application by setting RPF works only when there is one-to-one mapping of Rx queue to Rx DMA channels. In dynamic mapping mode, setting this bit can flush packets from unintended Rx queues which are destined to the stopped Rx DMA channel.	
		0	Rx Packet Flush is Disabled
		1	Rx Packet Flush is Enabled

Table 30-122: EMAC_DMA[n]_RXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
21:16 (R/W)	RXPBL	Receive Programmable Burst Length. The EMAC_DMA[n]_RXCTL.RXPBL field indicates the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. Program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the 8xPBL mode in the EMAC_DMA[n]_CTL register. 2. Set the RxPBL. Note: The maximum value of RxPBL must be less than or equal to half the Rx queue size (EMAC_RQ[n]_OPMODE.RQS) in terms of beats. This is required so that the Rx queue has space to store at least another Rx PBL worth of data while the Rx DMA is transferring a block of data. For example, in the total locations in Rx queue of size 512 bytes is 128, so RxPBL and 8xPBL needs to be programmed to less than or equal to 64.
14:3 (R/W)	RBSZ_13_Y	Receive Buffer Size High. RBSZ[13:0] is split into two fields: higher EMAC_DMA[n]_RXCTL.RBSZ_13_Y and lower RBSZ_x_0. The RBSZ[13:0] field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16K bytes. The buffer size is applicable to payload buffers when split headers are enabled. Note: The buffer size must be a multiple of 4 bytes. This is required even if the value of buffer address pointer is not aligned to data bus width. Hence, the lower RBSZ_x_0 bits are read-only and the value is considered as all-zero. Thus, the EMAC_DMA[n]_RXCTL.RBSZ_13_Y indicates the buffer size in terms of locations (with the width same as bus width).
2:1 (R/NW)	RBSZ_X_0	Receive Buffer Size Low. RBSZ[13:0] is split into two fields RBSZ_13_y and RBSZ_x_0. The EMAC_DMA[n]_RXCTL.RBSZ_X_0 field is the lower field whose width is based on data bus width of the configuration. The EMAC_DMA[n]_RXCTL.RBSZ_X_0 field is 2 bits wide. The EMAC_DMA[n]_RXCTL.RBSZ_X_0 field is read-only (RO).

Table 30-122: EMAC_DMA[n]_RXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	SR	Start or Stop Receive. When the `EMAC_DMA[n]_RXCTL.SR` bit is set (=1), the DMA tries to acquire the descriptor from the receive list and processes the incoming packets. The DMA tries to acquire descriptor from either of the following positions: The current position in the list - This is the address set by the EMAC_DMA[n]_RXDSC_ADDR register. The position at which the Rx process was previously stopped - If the DMA does not own the current descriptor, the reception is suspended and the EMAC_DMA[n]_STAT.RBU bit is set (=1). The start receive command is effective only when the reception is stopped. If the command is issued before configuring the EMAC_DMA[n]_RXDSC_ADDR register, the DMA behavior is unpredictable. When `EMAC_DMA[n]_RXCTL.SR` bit = 0, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the receive list is saved, and it becomes the current position after the Rx process is restarted. The stop receive command is effective only when the Rx process is in the running (waiting for Rx packet) or suspended state.	
		0	Stop Receive
		1	Start Receive

DMA Channel n Extended Receive Control Register

The EMAC_DMA[n]_RXCTL2 register controls the Rx features such as Rx descriptor ring length and alternate Rx buffer size.

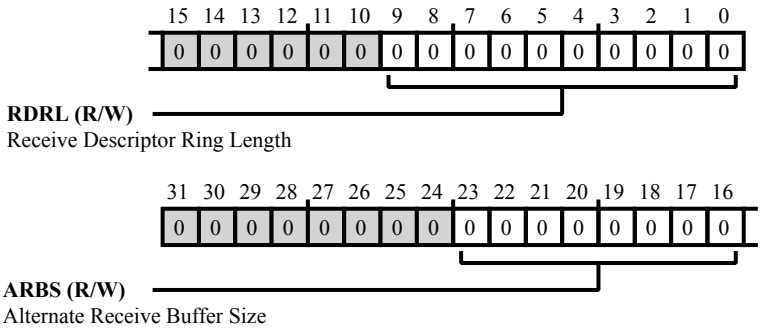


Figure 30-68: EMAC_DMA[n]_RXCTL2 Register Diagram

Table 30-123: EMAC_DMA[n]_RXCTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	ARBS	Alternate Receive Buffer Size. The EMAC_DMA[n]_RXCTL2.ARBS field indicates the size in bytes for buffer 1, when programmed to a non-zero value (when the split header feature is not enabled). When the split header feature is enabled, the EMAC_DMA[n]_RXCTL2.ARBS field indicates the buffer size for header data. The maximum alternate buffer is limited to 1020 bytes for the 32-bit data bus width. When EMAC_DMA[n]_RXCTL2.ARBS= 0, the Rx Buffer1 and Rx Buffer2 sizes are based on EMAC_DMA[n]_RXCTL.RBSZ.
9:0 (R/W)	RDRL	Receive Descriptor Ring Length. The EMAC_DMA[n]_RXCTL2 register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. For example, program any value up to 0x3FF in this field. The EMAC_DMA[n]_RXCTL2.RDRL field is 10 bits wide. When programming 0x3FF, there are 1024 descriptors. For 10 descriptors, program it to a value of 0x9.

DMA Channel n Receive Interrupt Watchdog Timer Register

The EMAC_DMA[n]_RXINTWDTMR register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the EMAC_DMA[n]_STAT.RI bit.

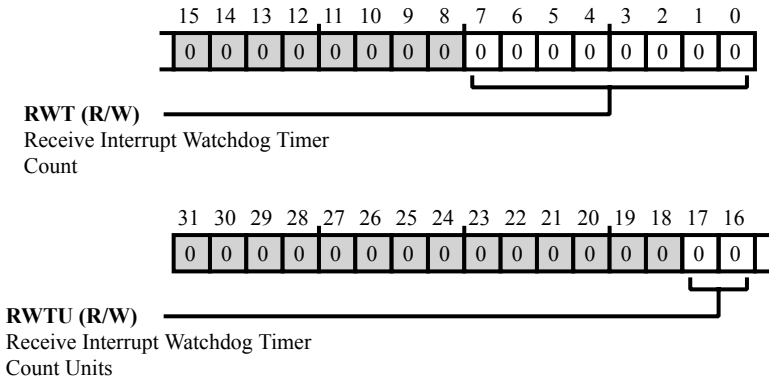


Figure 30-69: EMAC_DMA[n]_RXINTWDTMR Register Diagram

Table 30-124: EMAC_DMA[n]_RXINTWDTMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	RWTU	Receive Interrupt Watchdog Timer Count Units. The EMAC_DMA[n]_RXINTWDTMR.RWTU field indicates the number of SCLK0 cycles corresponding to one unit in the EMAC_DMA[n]_RXINTWDTMR.RWT field. For example, when EMAC_DMA[n]_RXINTWDTMR.RWT=2 and EMAC_DMA[n]_RXINTWDTMR.RWTU = 1, the watchdog timer is configured for 2 * 512 = 1024 SCLK0 cycles.	
		0	256
		1	512
		2	1024
		3	2048

Table 30-124: EMAC_DMA[n]_RXINTWDTMR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	RWT	Receive Interrupt Watchdog Timer Count. The `EMAC_DMA[n]_RXINTWDTMR.RWT` field indicates the number of SCLK0 cycles, multiplied by factor indicated in `EMAC_DMA[n]_RXINTWDTMR.RWTU` field, for which the watchdog timer is set. The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which EMAC_DMA[n]_STAT.RI = 0 register, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set (=1) because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.

DMA Channel n Slot Function Control and Status register

The EMAC_DMA[n]_SFN_CTLSTAT register contains the control bits for slot function and the status for transmit path.

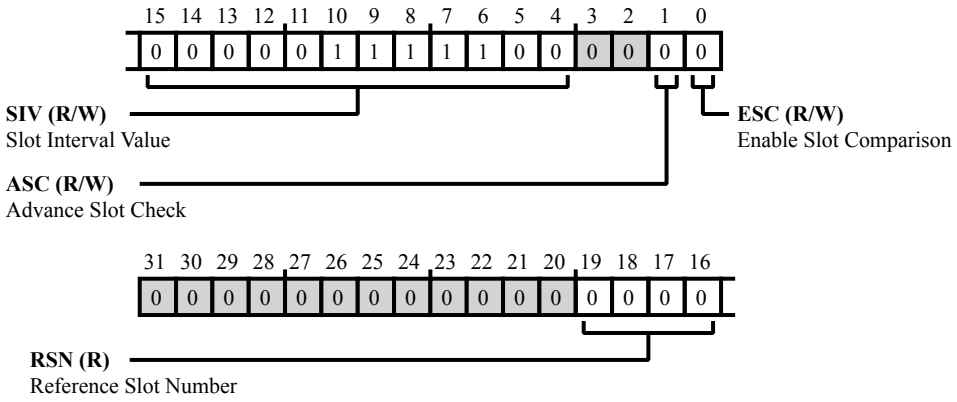


Figure 30-70: EMAC_DMA[n]_SFN_CTLSTAT Register Diagram

Table 30-125: EMAC_DMA[n]_SFN_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16 (R/NW)	RSN	Reference Slot Number. The EMAC_DMA[n]_SFN_CTLSTAT.RSN field gives the current value of the reference slot number in the DMA. It is used for slot comparison.	
15:4 (R/W)	SIV	Slot Interval Value. The EMAC_DMA[n]_SFN_CTLSTAT.SIV field controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 us while the maximum value 4095 specifies the slot interval of 4096us. The default/reset value is 0x07C which corresponds to a slot interval of 125 us.	
1 (R/W)	ASC	Advance Slot Check. When set, the EMAC_DMA[n]_SFN_CTLSTAT.ASC bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the Tx descriptor is: - equal to the reference slot number given in the RSN field or - ahead of the reference slot number by up to two slots. The EMAC_DMA[n]_SFN_CTLSTAT.ASC bit is applicable only when the ESC bit is set.	
		0	Advance Slot Check is Disabled
		1	Advance Slot Check is Enabled

Table 30-125: EMAC_DMA[n]_SFN_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	ESC	Enable Slot Comparison. When set (=1), the `EMAC_DMA[n]_SFN_CTLSTAT.ESC` bit enables the checking of the slot numbers programmed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is: - equal to the reference slot number or - ahead of the reference slot number by one slot. When cleared (=0), the `EMAC_DMA[n]_SFN_CTLSTAT.ESC` bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed. Note: The UFO (UDP Fragmentation over IPv4)/TSO/USO must not be enabled with the TBS/AVB slot number check. The UFO/TSO/USO involves multiple packets/segments/fragments transmission for single packet received from application, and the slot number checks are applicable for the fetching of only the first segment/fragment. As a result, it can be difficult for software to specify a slot number for subsequent packets.	
		0	Slot Comparison is Disabled
		1	Slot Comparison is Enabled

DMA Channel n Status Register

The software driver (application) reads the EMAC_DMA[n]_STAT register during interrupt service routine or polling to determine the status of the DMA.

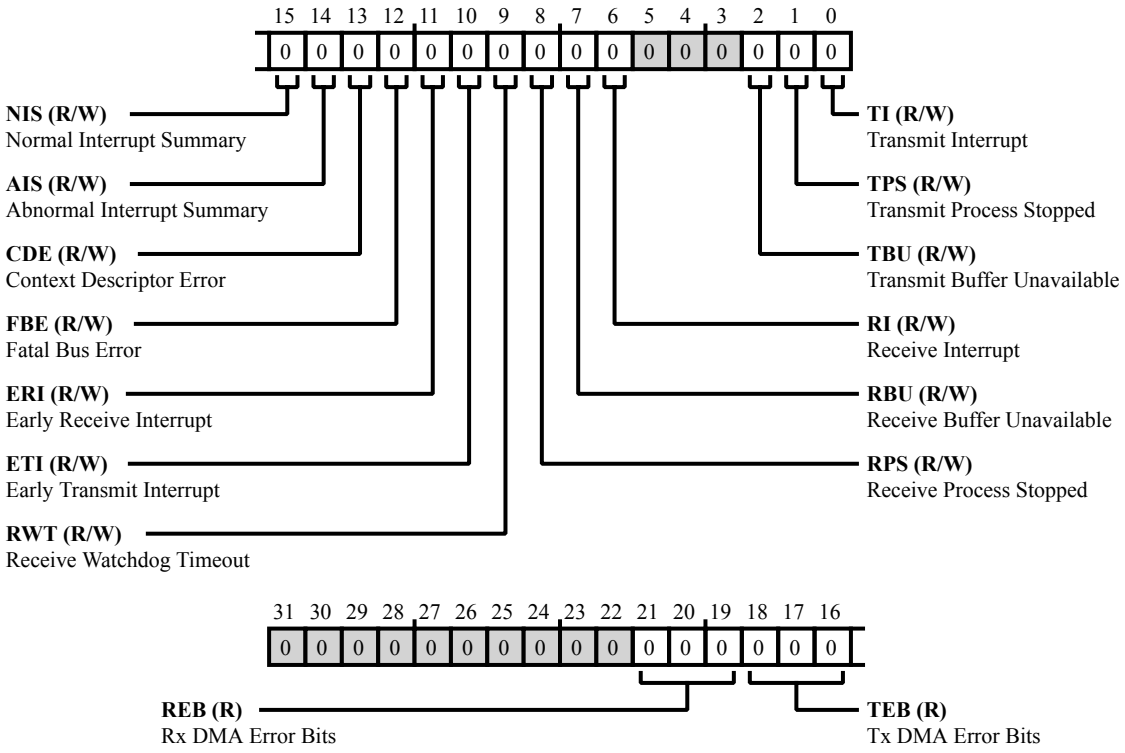


Figure 30-71: EMAC_DMA[n]_STAT Register Diagram

Table 30-126: EMAC_DMA[n]_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:19 (R/NW)	REB	Rx DMA Error Bits. The EMAC_DMA[n]_STAT.REB field indicates the type of error that caused a bus error. For example, an error response on the SPB or SCB interface. Bit 21 -- 1'b1: Error during data transfer by Rx DMA -- 1'b0: No Error during data transfer by Rx DMA Bit 20 -- 1'b1: Error during descriptor access -- 1'b0: Error during data buffer access Bit 19 -- 1'b1: Error during read transfer -- 1'b0: Error during write transfer The EMAC_DMA[n]_STAT.REB field is valid only when the FBE bit is set. It does not generate an interrupt.
18:16 (R/NW)	TEB	Tx DMA Error Bits. The EMAC_DMA[n]_STAT.TEB field indicates the type of error that caused a bus error. For example, error response on the SPB or SCB interface. Bit 18 -- 1'b1: Error during data transfer by Tx DMA -- 1'b0: No Error during data transfer by Tx DMA Bit 17 -- 1'b1: Error during descriptor access -- 1'b0: Error during data buffer access Bit 16 -- 1'b1: Error during read transfer -- 1'b0: Error during write transfer The EMAC_DMA[n]_STAT.TEB field is valid only when the FBE bit is set. It does not generate an interrupt.

Table 30-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	NIS	Normal Interrupt Summary. The EMAC_DMA[n]_STAT.NIS bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the EMAC_DMA[n]_IEN register: Bit 0: Transmit Interrupt Bit 2: Transmit Buffer Unavailable Bit 6: Receive Interrupt Bit 11: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in EMAC_DMA[n]_IEN register) affect the Normal Interrupt Summary bit. This is a sticky bit; the EMAC_DMA[n]_STAT.NIS bit must be cleared (by writing 1) each time a corresponding bit that causes NIS to be set is cleared. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Normal Interrupt Summary Status Not Detected
		1	Normal Interrupt Summary Status Detected
14 (R/W)	AIS	Abnormal Interrupt Summary. The EMAC_DMA[n]_STAT.AIS bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the EMAC_DMA[n]_IEN register: Bit 1: Transmit Process Stopped Bit 7: Receive Buffer Unavailable Bit 8: Receive Process Stopped Bit 10: Early Transmit Interrupt Bit 12: Fatal Bus Error Bit 13: Context Descriptor Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit. Clear the EMAC_DMA[n]_STAT.AIS bit (by writing 1 to the EMAC_DMA[n]_STAT.AIS bit) each time a corresponding bit, which causes AIS to be set, is cleared. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Abnormal Interrupt Summary Status Not Detected
		1	Abnormal Interrupt Summary Status Detected

Table 30-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	CDE	Context Descriptor Error. The EMAC_DMA[n]_STAT.CDE bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one's descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Context Descriptor Error Status Not Detected
		1	Context Descriptor Error Status Detected
12 (R/W)	FBE	Fatal Bus Error. The EMAC_DMA[n]_STAT.FBE bit indicates that a bus error occurred (as described in the EB field). When the EMAC_DMA[n]_STAT.FBE bit is set, the corresponding DMA channel engine disables all bus accesses. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Fatal Bus Error Status Not Detected
		1	Fatal Bus Error Status Detected
11 (R/W)	ERI	Early Receive Interrupt. The EMAC_DMA[n]_STAT.ERI bit when set indicates that the RxDMA has completed the transfer of packet data to the memory. In configurations supporting ERIC, ERIC = 0: The EMAC_DMA[n]_STAT.ERI bit is set only after the Rx DMA has completely filled up a receive buffer with packet data. ERIC = 1: The EMAC_DMA[n]_STAT.ERI bit is set after every burst transfer of data from the Rx DMA to the buffer. The setting of RI bit automatically clears the EMAC_DMA[n]_STAT.ERI bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Early Receive Interrupt Status Not Detected
		1	Early Receive Interrupt Status Detected

Table 30-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	ETI	Early Transmit Interrupt. The EMAC_DMA[n]_STAT.ETI bit when set indicates that the TxDMA has completed the transfer of packet data to the MTL TXFIFO memory. In configurations supporting ETIC: ETIC = 0: The EMAC_DMA[n]_STAT.ETI bit is set only after the Tx DMA has transferred a complete packet to MTL. ETIC = 1: The EMAC_DMA[n]_STAT.ETI bit is set after completion of (partial) packet data transfer from buffers in the transmit descriptor in which IOC = 1. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Early Transmit Interrupt Status Not Detected
		1	Early Transmit Interrupt Status Detected
9 (R/W)	RWT	Receive Watchdog Timeout. The EMAC_DMA[n]_STAT.RWT bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1) is received.	
		0	Receive Watchdog Timeout Status Not Detected
		1	Receive Watchdog Timeout Status Detected
8 (R/W)	RPS	Receive Process Stopped. The EMAC_DMA[n]_STAT.RPS bit is asserted when the Rx process enters the STOP state. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Receive Process Stopped Status Not Detected
		1	Receive Process Stopped Status Detected

Table 30-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RBU	Receive Buffer Unavailable. The EMAC_DMA[n]_STAT.RBU bit indicates that the application owns the next descriptor in the receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the EMAC_DMA[n]_RXDSC_TLPTR register of a channel. The EMAC_DMA[n]_STAT.RBU bit is set only when the DMA owns the previous Rx descriptor. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Receive Buffer Unavailable Status Not Detected
		1	Receive Buffer Unavailable Status Detected
6 (R/W)	RI	Receive Interrupt. The EMAC_DMA[n]_STAT.RI bit indicates that the packet reception is complete. When packet reception is complete, bit 31 of RDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. The reception remains in the running state. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Receive Interrupt Status Not Detected
		1	Receive Interrupt Status Detected
2 (R/W)	TBU	Transmit Buffer Unavailable. The EMAC_DMA[n]_STAT.TBU bit indicates that the application owns the next descriptor in the transmit list, and the DMA cannot acquire it. Transmission is suspended. The EMAC_DMA_DBG_STAT0.TPS0 field indicates the transmit process state transitions. To resume processing the transmit descriptors, the application must: 1. Change the ownership of the descriptor by setting bit 31 of TDES3. 2. Issue a Transmit Poll Demand command. 3. For ring mode, the application should advance the EMAC_DMA[n]_TXDSC_TLPTR register of a channel. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Transmit Buffer Unavailable Status Not Detected
		1	Transmit Buffer Unavailable Status Detected

Table 30-126: EMAC_DMA[n]_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	TPS	Transmit Process Stopped. The `EMAC_DMA[n]_STAT.TPS` bit is set when the transmission is stopped. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Transmit Process Stopped Status Not Detected
		1	Transmit Process Stopped Status Detected
0 (R/W)	TI	Transmit Interrupt. The `EMAC_DMA[n]_STAT.TI` bit indicates that the packet transmission is complete. When transmission is complete, bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Transmit Interrupt Status Not Detected
		1	Transmit Interrupt Status Detected

DMA Channel n Transmit Descriptor List Address Register

The EMAC_DMA[n]_TXDSC_ADDR register points the DMA to the start of transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be 32-bit aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

Write to the EMAC_DMA[n]_TXDSC_ADDR register only when the Tx DMA has stopped (EMAC_DMA[n]_TXCTL.ST = 0). When stopped, this register can be written with a new descriptor list address. When EMAC_DMA[n]_TXCTL.ST = 1, the DMA takes the newly-programmed descriptor base address. If this register is not changed when EMAC_DMA[n]_TXCTL.ST = 0, the DMA takes the descriptor address where it was stopped earlier.

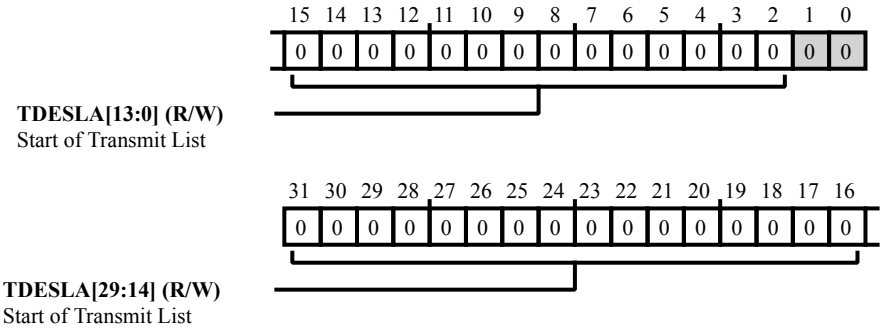


Figure 30-72: EMAC_DMA[n]_TXDSC_ADDR Register Diagram

Table 30-127: EMAC_DMA[n]_TXDSC_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	TDESLA	Start of Transmit List. The EMAC_DMA[n]_TXDSC_ADDR.TDESLA field contains the base address of the first descriptor in the transmit descriptor list. The DMA ignores the LSB bits (1:0) and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

DMA Channel n Transmit Descriptor Ring Length Register

The EMAC_DMA[n]_TXDSC_RLEN register contains the length of the transmit descriptor ring.

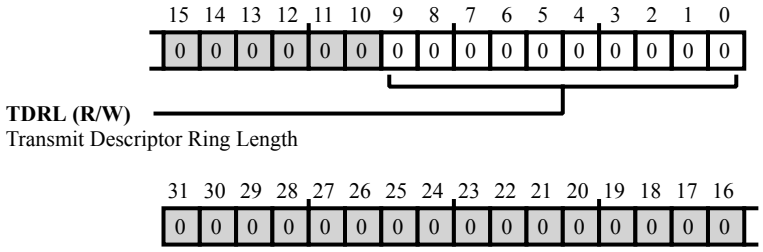


Figure 30-73: EMAC_DMA[n]_TXDSC_RLEN Register Diagram

Table 30-128: EMAC_DMA[n]_TXDSC_RLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/W)	TDRL	Transmit Descriptor Ring Length. The EMAC_DMA[n]_TXDSC_RLEN.TDRL field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. A minimum ring descriptor length of 4 is recommended. For example, program any value up to 0x3FF in this field. The EMAC_DMA[n]_TXDSC_RLEN.TDRL field is 10 bits wide. To program 0x3FF, 1024 descriptors are available. For 10 descriptors, program it to a value of 0x9.

DMA Channel n Transmit Descriptor Tail Pointer Register

The EMAC_DMA[n]_TXDSC_TLPTR register points to an offset from the base and indicates the location of the last valid descriptor.

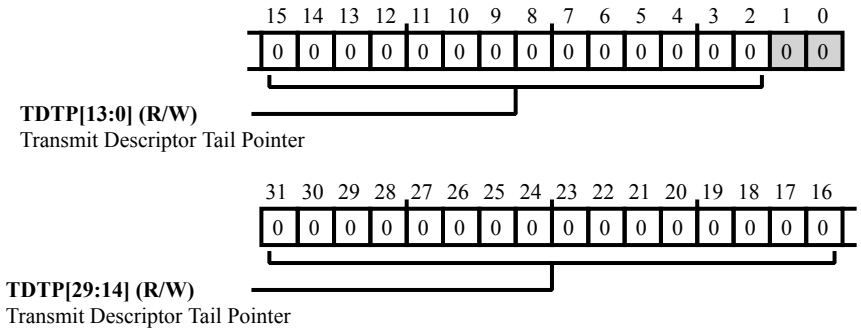


Figure 30-74: EMAC_DMA[n]_TXDSC_TLPTR Register Diagram

Table 30-129: EMAC_DMA[n]_TXDSC_TLPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:2 (R/W)	TDTP	Transmit Descriptor Tail Pointer. The EMAC_DMA[n]_TXDSC_TLPTR.TDTP field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

DMA Channel n Transmit Control Register

The EMAC_DMA[n]_TXCTL register controls the Tx features such as PBL, TCP segmentation, and Tx channel weights.

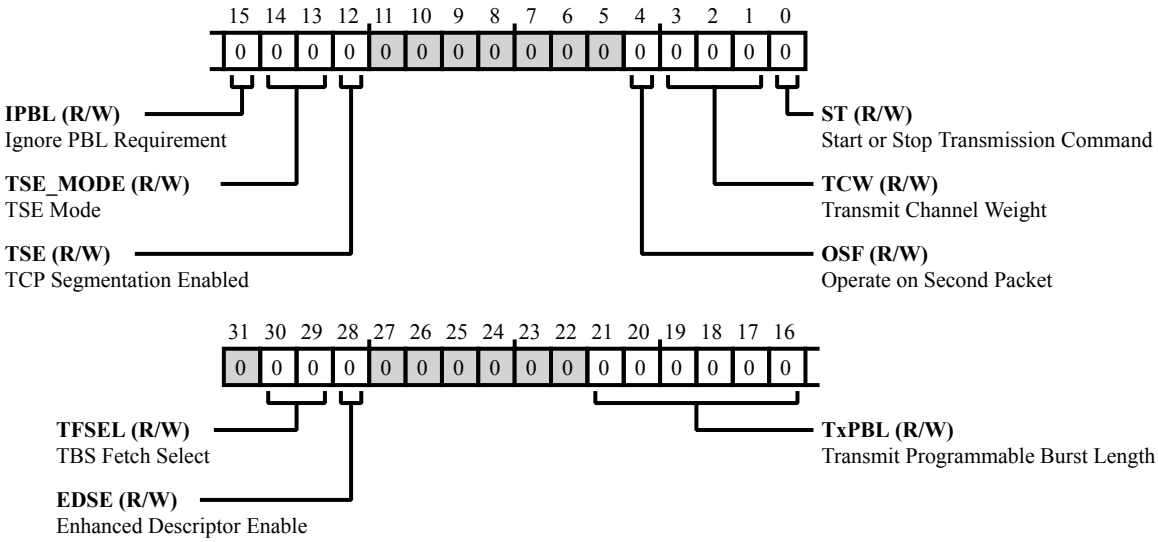


Figure 30-75: EMAC_DMA[n]_TXCTL Register Diagram

Table 30-130: EMAC_DMA[n]_TXCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:29 (R/W)	TFSEL	TBS Fetch Select. The EMAC_DMA[n]_TXCTL.TFSEL field select the EMAC_DMA_TBS_CTRL0 - EMAC_DMA_TBS_CTRL3 register fields (FTOS, FGSN, FTOV) for the channel.	
		0	EMAC_DMA_TBS_CTL0
		1	EMAC_DMA_TBS_CTL1
		2	EMAC_DMA_TBS_CTL2
		3	EMAC_DMA_TBS_CTL3
28 (R/W)	EDSE	Enhanced Descriptor Enable. When EMAC_DMA[n]_TXCTL.EDSE = 1, the corresponding channel uses enhanced descriptors that are 32 bytes for both normal and context descriptors. When cleared (=0), the corresponding channel uses the descriptors that are 16 bytes.	
		0	Enhanced Descriptor is Disabled
		1	Enhanced Descriptor is Enabled

Table 30-130: EMAC_DMA[n]_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:16 (R/W)	TXPBL	Transmit Programmable Burst Length. The `EMAC_DMA[n]_TXCTL.TXPBL` field indicates the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. Program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the 8xPBL mode in EMAC_DMA[n]_CTL register. 2. Set the TxPBL. Note: The maximum value of TxPBL must be less than or equal to half the Tx queue size (EMAC_TQ[n]_OPMODE.TQS) in terms of beats. This is required so that the Tx queue has space to store at least another Tx PBL worth of data while the MTL Tx queue controller is transferring data to MAC. For example, in 64-bit data width configurations the total locations in Tx queue of size 512 bytes is 64, TxPBL and 8xPBL must be programmed to less than or equal to 32.	
15 (R/W)	IPBL	Ignore PBL Requirement. When `EMAC_DMA[n]_TXCTL.IPBL` = 1, the DMA does not check for PBL number of locations in the MTL before initiating a transfer. If space is not available, the MTL can use handshaking to slow the DMA. Note: The `EMAC_DMA[n]_TXCTL.IPBL` bit/mode must not be used when multiple transmit DMA channels are enabled as it might block other transmit and receive DMA channels from accessing the read data channel of SCB bus until space is available in transmit queue for the current transfer.	
		0	Ignore PBL Requirement is Disabled
		1	Ignore PBL Requirement is Enabled
14:13 (R/W)	TSE_MODE	TSE Mode. The `EMAC_DMA[n]_TXCTL.TSE_MODE` field indicates the TSE mode.	
		0	TSO/USO. Segmentation functionality is enabled.
		1	UFO with Checksum. Enables the UDP fragmentation functionality with checksum for all the UDP packets.
		2	UFO Without Checksum. Enables the UDP fragmentation functionality without Checksum for all the UDP packets.
		3	Reserved

Table 30-130: EMAC_DMA[n]_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	TSE	TCP Segmentation Enabled. When EMAC_DMA[n]_TXCTL.TSE = 1, the DMA performs the TCP segmentation or UDP segmentation/fragmentation for packets in this channel. The TCP segmentation or UDP packet's segmentation/fragmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When EMAC_DMA[n]_TXCTL.TSE = 1, the TxPBL value must be greater than 4.	
		0	TCP Segmentation is Disabled
		1	TCP Segmentation is Enabled
4 (R/W)	OSF	Operate on Second Packet. When EMAC_DMA[n]_TXCTL.OSF = 1, the DMA processes the second packet of the transmit data, even before the status for the first packet is obtained.	
		0	Operate on Second Packet Disabled
		1	Operate on Second Packet Enabled
3:1 (R/W)	TCW	Transmit Channel Weight. The EMAC_DMA[n]_TXCTL.TCW field indicates the weight assigned to the corresponding transmit channel. When reset is complete, EMAC_DMA[n]_TXCTL.TCW = 0 for all channels by default, resulting in equal weights to all channels.	

Table 30-130: EMAC_DMA[n]_TXCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	ST	Start or Stop Transmission Command. When the EMAC_DMA[n]_TXCTL.ST bit is set, transmission is placed in the running state. The DMA checks the transmit list at the current position for a packet to be transmitted. The DMA tries to acquire descriptor from either of the following positions: The current position in the list - This is the base address of the transmit list configured by the EMAC_DMA[n]_TXDSC_ADDR register. The position at which the transmission was previously stopped - If the DMA does not own the current descriptor, the transmission enters the suspended state and the EMAC_DMA[n]_STAT.TBU =1. The start transmission command is effective only when the transmission is stopped. If the command is issued before configuring the EMAC_DMA[n]_TXDSC_ADDR register, the DMA behavior is unpredictable. When EMAC_DMA[n]_TXCTL.ST = 0, the transmission process is placed in the stopped state after completing the transmission of the current packet. The Next Descriptor position in the transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, program EMAC_DMA[n]_TXDSC_ADDR register with a new value when EMAC_DMA[n]_TXCTL.ST = 0. The new value is considered when EMAC_DMA[n]_TXCTL.ST =1 again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the suspended state.	
		0	Stop Transmission Command
		1	Start Transmission Command

DMA Debug Status 0 Register

The EMAC_DMA_DBG_STAT0 register gives the receive and transmit process status for DMA channels 0-2 for debugging purpose.

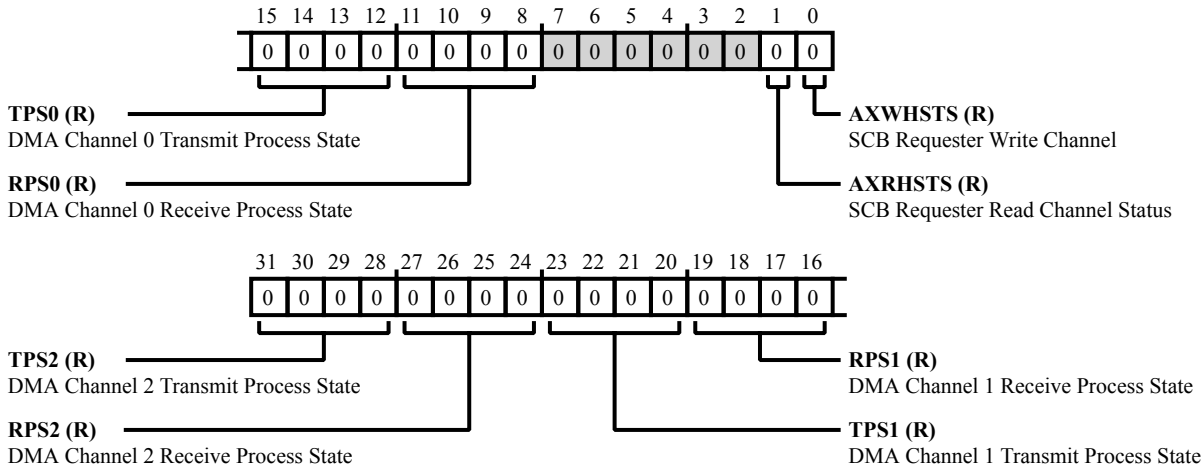


Figure 30-76: EMAC_DMA_DBG_STAT0 Register Diagram

Table 30-131: EMAC_DMA_DBG_STAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	TPS2	DMA Channel 2 Transmit Process State. The EMAC_DMA_DBG_STAT0.TPS2 field indicates the Tx DMA FSM state for channel 2. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT0.TPS2 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
27:24 (R/NW)	RPS2	DMA Channel 2 Receive Process State. The EMAC_DMA_DBG_STAT0.RPS2 field indicates the Rx DMA FSM state for channel 2. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT0.RPS2 field does not generate an interrupt.	

Table 30-131: EMAC_DMA_DBG_STAT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
23:20 (R/NW)	TPS1	DMA Channel 1 Transmit Process State. The `EMAC_DMA_DBG_STAT0.TPS1` field indicates the Tx DMA FSM state for channel 1. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT0.TPS1` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
19:16 (R/NW)	RPS1	DMA Channel 1 Receive Process State. The `EMAC_DMA_DBG_STAT0.RPS1` field indicates the Rx DMA FSM state for channel 1. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT0.RPS1` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)

Table 30-131: EMAC_DMA_DBG_STAT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
15:12 (R/NW)	TPS0	DMA Channel 0 Transmit Process State. The `EMAC_DMA_DBG_STAT0.TPS0` field indicates the Tx DMA FSM state for channel 0. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT0.TPS0` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
11:8 (R/NW)	RPS0	DMA Channel 0 Receive Process State. The `EMAC_DMA_DBG_STAT0.RPS0` field indicates the Rx DMA FSM state for channel 0. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT0.RPS0` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)

Table 30-131: EMAC_DMA_DBG_STAT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	AXRHSTS	SCB Requester Read Channel Status. When high, the `EMAC_DMA_DBG_STAT0.AXRHSTS` bit indicates that the read channel of the SCB requester is active, and it is transferring the data.	
		0	Status Not Detected
		1	Status Detected
0 (R/NW)	AXWHSTS	SCB Requester Write Channel. When high, the `EMAC_DMA_DBG_STAT0.AXWHSTS` bit indicates that the write channel of the SCB requester is active, and it is transferring data.	
		0	Status Not Detected
		1	Status Detected

DMA Debug Status 1 Register

The EMAC_DMA_DBG_STAT1 register gives the receive and transmit process status for DMA channels 3-6.

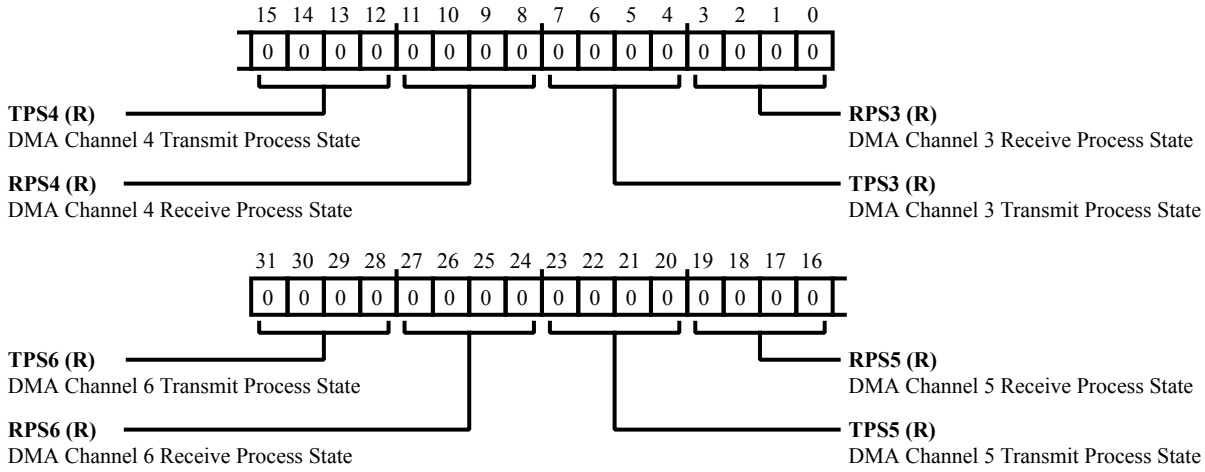


Figure 30-77: EMAC_DMA_DBG_STAT1 Register Diagram

Table 30-132: EMAC_DMA_DBG_STAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:28 (R/NW)	TPS6	DMA Channel 6 Transmit Process State. The EMAC_DMA_DBG_STAT1.TPS6 field indicates the Tx DMA FSM state for channel 6. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT1.TPS6 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
27:24 (R/NW)	RPS6	DMA Channel 6 Receive Process State. The EMAC_DMA_DBG_STAT1.RPS6 field indicates the Rx DMA FSM state for channel 6. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT1.RPS6 field does not generate an interrupt.	

Table 30-132: EMAC_DMA_DBG_STAT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
23:20 (R/NW)	TPS5	DMA Channel 5 Transmit Process State. The `EMAC_DMA_DBG_STAT1.TPS5` field indicates the Tx DMA FSM state for channel 5. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.TPS5` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
19:16 (R/NW)	RPS5	DMA Channel 5 Receive Process State. The `EMAC_DMA_DBG_STAT1.RPS5` field indicates the Rx DMA FSM state for channel 5. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.RPS5` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)

Table 30-132: EMAC_DMA_DBG_STAT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
15:12 (R/NW)	TPS4	DMA Channel 4 Transmit Process State. The `EMAC_DMA_DBG_STAT1.TPS4` field indicates the Tx DMA FSM state for channel 4. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.TPS4` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
11:8 (R/NW)	RPS4	DMA Channel 4 Receive Process State. The `EMAC_DMA_DBG_STAT1.RPS4` field indicates the Rx DMA FSM state for channel 4. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.RPS4` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)
7:4 (R/NW)	TPS3	DMA Channel 3 Transmit Process State. The `EMAC_DMA_DBG_STAT1.TPS3` field indicates the Tx DMA FSM state for channel 3. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.TPS3` field does not generate an interrupt.	

Table 30-132: EMAC_DMA_DBG_STAT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
3:0 (R/NW)	RPS3	DMA Channel 3 Receive Process State. The `EMAC_DMA_DBG_STAT1.RPS3` field indicates the Rx DMA FSM state for channel 3. The MSB of this field always returns 0. The `EMAC_DMA_DBG_STAT1.RPS3` field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)

DMA Debug Status 2 Register

The EMAC_DMA_DBG_STAT2 register gives the receive and transmit process status for DMA channel 7.

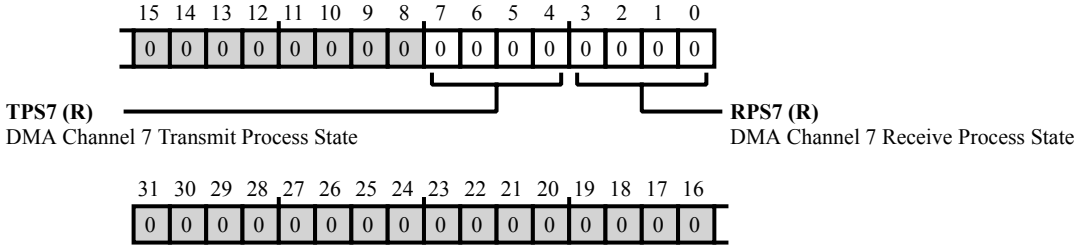


Figure 30-78: EMAC_DMA_DBG_STAT2 Register Diagram

Table 30-133: EMAC_DMA_DBG_STAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/NW)	TPS7	DMA Channel 7 Transmit Process State. The EMAC_DMA_DBG_STAT2.TPS7 field indicates the Tx DMA FSM state for channel 7. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT2.TPS7 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Transmit Command Issued)
		1	Running (Fetching Tx Transfer Descriptor)
		2	Running (Waiting for Status)
		3	Running (Reading Data from System Memory Buffer and Queuing It to the Tx Buffer (Tx FIFO))
		4	Time Stamp Write State
		5	Reserved
		6	Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
		7	Running (Closing Tx Descriptor)
3:0 (R/NW)	RPS7	DMA Channel 7 Receive Process State. The EMAC_DMA_DBG_STAT2.RPS7 field indicates the Rx DMA FSM state for channel 7. The MSB of this field always returns 0. The EMAC_DMA_DBG_STAT2.RPS7 field does not generate an interrupt.	
		0	Stopped (Reset or Stop Receive Command Issued)
		1	Running (Fetching Rx Transfer Descriptor)
		2	Reserved
		3	Running (Waiting for Rx Packet)
		4	Suspended (Rx Descriptor Unavailable)

Table 30-133: EMAC_DMA_DBG_STAT2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		5	Running (Closing the Rx Descriptor)
		6	Time Stamp Write State
		7	Running (Transferring the Received Packet Data from the Rx Buffer to the System Memory)

DMA Interrupt Status Register

The application reads the EMAC_DMA_ISTAT register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

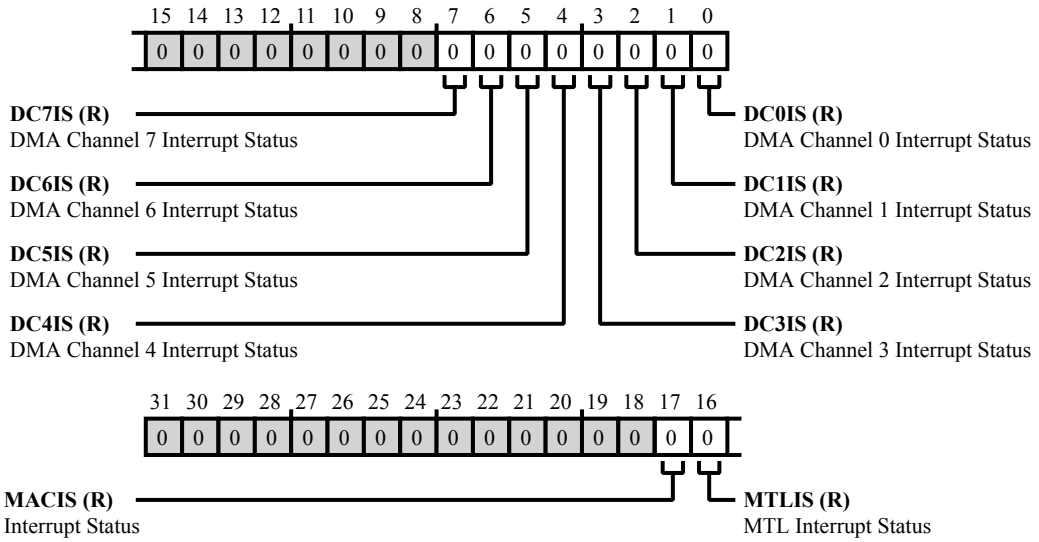


Figure 30-79: EMAC_DMA_ISTAT Register Diagram

Table 30-134: EMAC_DMA_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/NW)	MACIS	Interrupt Status. The EMAC_DMA_ISTAT.MACIS bit indicates an interrupt event in the MAC. To clear the EMAC_DMA_ISTAT.MACIS bit (=0), the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.	
		0	MAC Interrupt Status Not Detected
		1	MAC Interrupt Status Detected
16 (R/NW)	MTLIS	MTL Interrupt Status. The EMAC_DMA_ISTAT.MTLIS bit indicates an interrupt event in the MTL. To clear the EMAC_DMA_ISTAT.MTLIS bit (=0), the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.	
		0	MTL Interrupt Status Not Detected
		1	MTL Interrupt Status Detected

Table 30-134: EMAC_DMA_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	DC7IS	DMA Channel 7 Interrupt Status. The EMAC_DMA_ISTAT.DC7IS bit indicates an interrupt event in DMA channel 7. To clear the EMAC_DMA_ISTAT.DC7IS bit (=0), the software must read the corresponding register in DMA channel 7 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 7 Interrupt Status Not Detected
		1	DMA Channel 7 Interrupt Status Detected
6 (R/NW)	DC6IS	DMA Channel 6 Interrupt Status. The EMAC_DMA_ISTAT.DC6IS bit indicates an interrupt event in DMA channel 6. To clear the EMAC_DMA_ISTAT.DC6IS bit (=0), the software must read the corresponding register in DMA channel 6 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 6 Interrupt Status Not Detected
		1	DMA Channel 6 Interrupt Status Detected
5 (R/NW)	DC5IS	DMA Channel 5 Interrupt Status. The EMAC_DMA_ISTAT.DC5IS bit indicates an interrupt event in DMA channel 5. To clear the EMAC_DMA_ISTAT.DC5IS bit (=0), the software must read the corresponding register in DMA channel 5 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 5 Interrupt Status Not Detected
		1	DMA Channel 5 Interrupt Status Detected
4 (R/NW)	DC4IS	DMA Channel 4 Interrupt Status. The EMAC_DMA_ISTAT.DC4IS bit indicates an interrupt event in DMA channel 4. To clear the EMAC_DMA_ISTAT.DC4IS bit (=0), the software must read the corresponding register in DMA channel 4 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 4 Interrupt Status Not Detected
		1	DMA Channel 4 Interrupt Status Detected
3 (R/NW)	DC3IS	DMA Channel 3 Interrupt Status. The EMAC_DMA_ISTAT.DC3IS bit indicates an interrupt event in DMA channel 3. To clear the EMAC_DMA_ISTAT.DC3IS bit (=0), the software must read the corresponding register in DMA channel 3 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 3 Interrupt Status Not Detected
		1	DMA Channel 3 Interrupt Status Detected

Table 30-134: EMAC_DMA_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	DC2IS	DMA Channel 2 Interrupt Status. The EMAC_DMA_ISTAT.DC2IS bit indicates an interrupt event in DMA channel 2. To clear the EMAC_DMA_ISTAT.DC2IS bit (=0), the software must read the corresponding register in DMA channel 2 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 2 Interrupt Status Not Detected
		1	DMA Channel 2 Interrupt Status Detected
1 (R/NW)	DC1IS	DMA Channel 1 Interrupt Status. The EMAC_DMA_ISTAT.DC1IS bit indicates an interrupt event in DMA channel 1. To clear the EMAC_DMA_ISTAT.DC1IS bit (=0), the software must read the corresponding register in DMA channel 1 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 1 Interrupt Status Not Detected
		1	DMA Channel 1 Interrupt Status Detected
0 (R/NW)	DC0IS	DMA Channel 0 Interrupt Status. The EMAC_DMA_ISTAT.DC0IS bit indicates an interrupt event in DMA channel 0. To clear the EMAC_DMA_ISTAT.DC0IS bit (=0), the software must read the corresponding register in DMA channel 0 to get the exact cause of the interrupt and clear its source.	
		0	DMA Channel 0 Interrupt Status Not Detected
		1	DMA Channel 0 Interrupt Status Detected

DMA Mode Register

The EMAC_DMA_MODE register establishes the bus operating modes for the DMA.

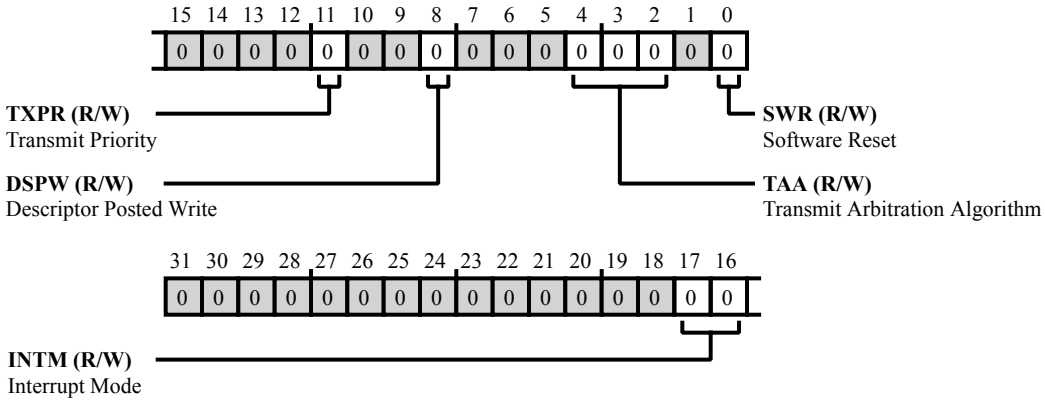


Figure 30-80: EMAC_DMA_MODE Register Diagram

Table 30-135: EMAC_DMA_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	INTM	Interrupt Mode. The `EMAC_DMA_MODE.INTM` field defines the interrupt mode of the EMAC module. The behavior of the following outputs change depending on the following settings: - EMAC_DMA[n]_TX (Transmit Per Channel Interrupt) - EMAC_DMA[n]_RX (Receive Per Channel Interrupt) - EMAC_STAT (Common Interrupt) It also changes the behavior of the EMAC_DMA[n]_STAT.RI and EMAC_DMA[n]_STAT.TI bits. - 00: EMAC_DMA[n]_TX/RX are pulse signals for each TX/RX packet transfer completion events (irrespective of whether corresponding interrupts are enabled) for which IOC bits are enabled in descriptor. EMAC_STAT is also asserted when corresponding interrupts are enabled and cleared only when software clears the corresponding RI/TI status bits. - 01: EMAC_DMA[n]_TX/RX are level signals asserted on TX/RX packet transfer completion event when corresponding interrupts are enabled and deasserted when the software clears the corresponding RI/TI status bits. The EMAC_STAT is not asserted for these TX/RX packet transfer completion events. - 10: EMAC_DMA[n]_TX/RX are level signals asserted on TX/RX packet transfer completion event when corresponding interrupts are enabled and deasserted when the software clears the corresponding RI/TI status bits. However, the signal is asserted again if the same event occurred again before it was cleared. The EMAC_STAT is not asserted for these TX/RX packet transfer completion events. - 11: Reserved For more details, see Transfer Complete Interrupt Behavior.	
		0	See Above Description
		1	See Above Description
		2	See Above Description
		3	Reserved
11 (R/W)	TXPR	Transmit Priority. When set, the `EMAC_DMA_MODE.TXPR` bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.	
		0	Transmit Priority is Disabled
		1	Transmit Priority is Enabled

Table 30-135: EMAC_DMA_MODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	DSPW	Descriptor Posted Write. When the EMAC_DMA_MODE.DSPW bit is cleared (=0), it indicates that the descriptor writes are always non-posted. When set (=1), the EMAC_DMA_MODE.DSPW bit indicates that the descriptor is non-posted only when the IOC (Interrupt on Completion) is set in last descriptor. Otherwise, the descriptor writes are always posted.	
		0	Disable
		1	Enable
4:2 (R/W)	TAA	Transmit Arbitration Algorithm. The EMAC_DMA_MODE.TAA field is used to select the arbitration algorithm for the transmit side when multiple Tx DMAs are selected.	
		0	Fixed Priority (Channel 0 Has the Lowest Priority and the Last Channel Has the Highest Priority)
		1	Weighted Strict Priority (WSP)
		2	Weighted Round-Robin (WRR)
		3	Reserved
0 (R/W)	SWR	Software Reset. When the EMAC_DMA_MODE.SWR bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. The EMAC_DMA_MODE.SWR bit is automatically cleared after the reset operation is complete in all the EMAC clock domains. Before reprogramming any the EMAC register, a value of zero should be read in the EMAC_DMA_MODE.SWR bit. The EMAC_DMA_MODE.SWR bit must be read at least 4 SCLK0 clock cycles after it is written to 1. Note: The reset operation is complete only when all resets in all active clock domains are deasserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Software Reset is Disabled
		1	Software Reset is Enabled

DMA System Bus Mode Register

The EMAC_DMA_SYSBMODE register controls the behavior of the SPB or SCB requester. It mainly controls burst splitting and number of outstanding requests.

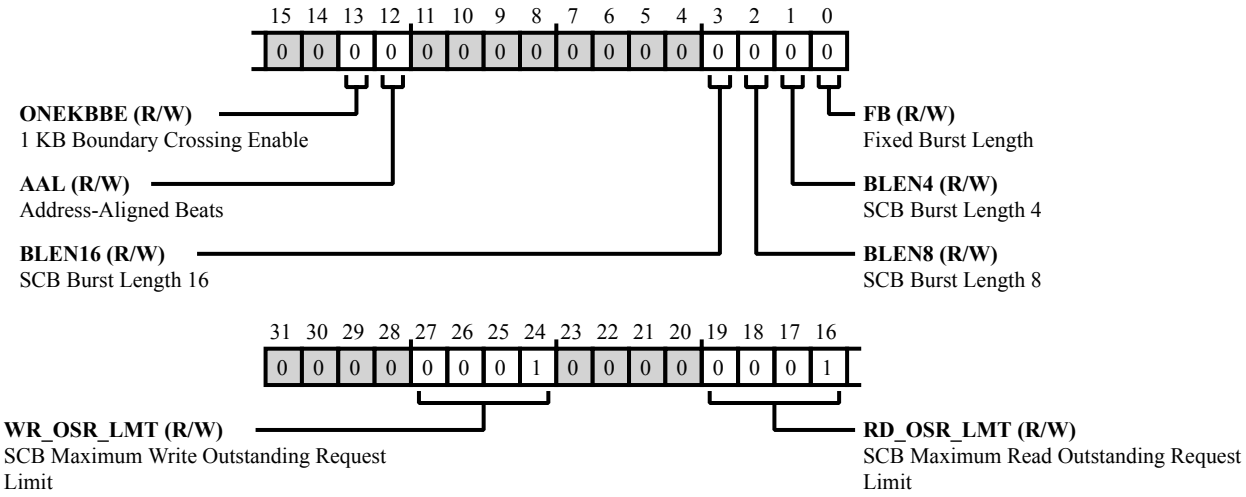


Figure 30-81: EMAC_DMA_SYSBMODE Register Diagram

Table 30-136: EMAC_DMA_SYSBMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27:24 (R/W)	WR_OSR_LMT	SCB Maximum Write Outstanding Request Limit. This value limits the maximum outstanding request on the SCB write interface. Maximum outstanding requests = EMAC_DMA_SYSBMODE.WR_OSR_LMT + 1	
19:16 (R/W)	RD_OSR_LMT	SCB Maximum Read Outstanding Request Limit. This value limits the maximum outstanding request on the SCB read interface. Maximum outstanding requests = EMAC_DMA_SYSBMODE.RD_OSR_LMT + 1.	
13 (R/W)	ONEKBBE	1 KB Boundary Crossing Enable. The EMAC_DMA_SYSBMODE.ONEKBBE bit indicates whether the burst transfers performed by the SCB requester do not cross a 1 KB or (the default) 4 KB boundary.	
		0	Disable
		1	Enable

Table 30-136: EMAC_DMA_SYSBMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	AAL	Address-Aligned Beats. When the EMAC_DMA_SYSBMODE.AAL = 1, the SCB or SPB requester performs address-aligned burst transfers on read and write channels. When the EMAC_DMA_SYSBMODE.AAL = 0, the SCB or SPB requester performs burst transfers on read and write channels without aligning to address boundaries.	
		0	Address-Aligned Beats is Disabled
		1	Address-Aligned Beats is Enabled
3 (R/W)	BLEN16	SCB Burst Length 16. When EMAC_DMA_SYSBMODE.BLEN16 = 1 or EMAC_DMA_SYSBMODE.FB = 1, the SCB requester can select a burst length of 16 on the SCB interface. When EMAC_DMA_SYSBMODE.FB = 0, writing to the EMAC_DMA_SYSBMODE.BLEN16 bit has no effect.	
		0	Disable
		1	Enable
2 (R/W)	BLEN8	SCB Burst Length 8. When EMAC_DMA_SYSBMODE.BLEN8 = 1 or EMAC_DMA_SYSBMODE.FB = 1, the SCB requester can select a burst length of 8 on the SCB interface. When EMAC_DMA_SYSBMODE.FB = 0, writing to the EMAC_DMA_SYSBMODE.BLEN8 bit has no effect.	
		0	Disable
		1	Enable
1 (R/W)	BLEN4	SCB Burst Length 4. When EMAC_DMA_SYSBMODE.BLEN4 = 1 or EMAC_DMA_SYSBMODE.FB = 1, the SCB requester can select a burst length of 4 on the SCB interface. When EMAC_DMA_SYSBMODE.FB = 0, writing to the EMAC_DMA_SYSBMODE.BLEN4 bit has no effect.	
		0	Disable
		1	Enable

Table 30-136: EMAC_DMA_SYSBMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	FB	Fixed Burst Length. The EMAC_DMA_SYSBMODE.FB bit enables a fixed burst length. When EMAC_DMA_SYSBMODE.FB= 1, the SCB requester initiates burst lengths as indicated by the BLEN16, BLEN8, or BLEN4 field -- Burst transfers of length 1. When EMAC_DMA_SYSBMODE.FB = 0, the SCB requester initiates burst transfers that are equal to or less than the maximum allowed burst length programmed in bits[7:4].	
		0	Fixed Burst Length is Disabled
		1	Fixed Burst Length is Enabled

DMA TBS Control 0 Register

The EMAC_DMA_TBS_CTL0 register is used to control the Time Based Scheduling (TBS) attributes.

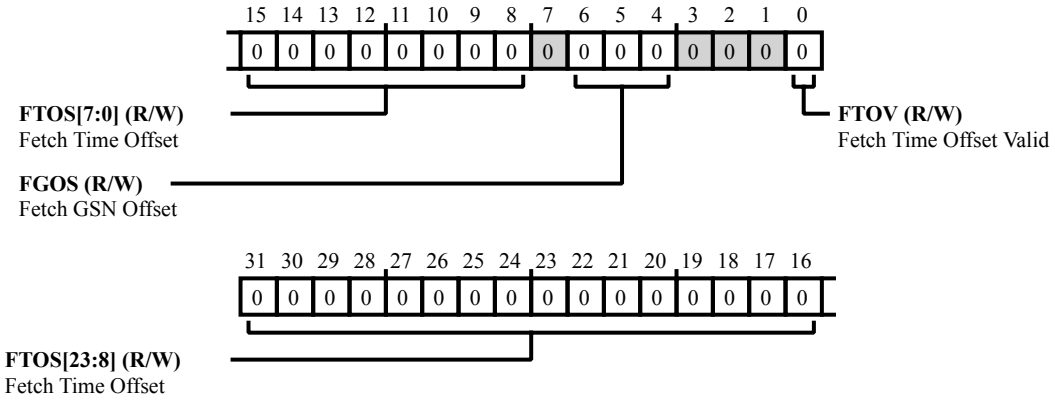


Figure 30-82: EMAC_DMA_TBS_CTL0 Register Diagram

Table 30-137: EMAC_DMA_TBS_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	FTOS	Fetch Time Offset. The EMAC_DMA_TBS_CTL0.FTOS field indicates the value in units of 256 nanoseconds, that has to be deducted from the launch time to compute the fetch Time. Max value: 999,999,999 ns, additionally should be smaller than CTR-1 value when ESTM mode is enabled since this value is a modulo CTR value.	
6:4 (R/W)	FGOS	Fetch GSN Offset. The EMAC_DMA_TBS_CTL0.FGOS field indicates the number of GSN slots that must be deducted from the Launch GSN to compute the Fetch GSN. Value valid only when EMAC_DMA_TBS_CTL0.FTOV is set.	
0 (R/W)	FTOV	Fetch Time Offset Valid. When set (=1), the EMAC_DMA_TBS_CTL0.FTOV bit indicates that the EMAC_DMA_TBS_CTL0.FTOS field is valid. When cleared (=0), the EMAC_DMA_TBS_CTL0.FTOV bit indicates that the Fetch Offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.	
		0	Fetch Time Offset is Invalid
		1	Fetch Time Offset is Valid

DMA TBS Control 1 Register

The EMAC_DMA_TBS_CTL1 register is used to control the Time Based Scheduling (TBS) attributes.

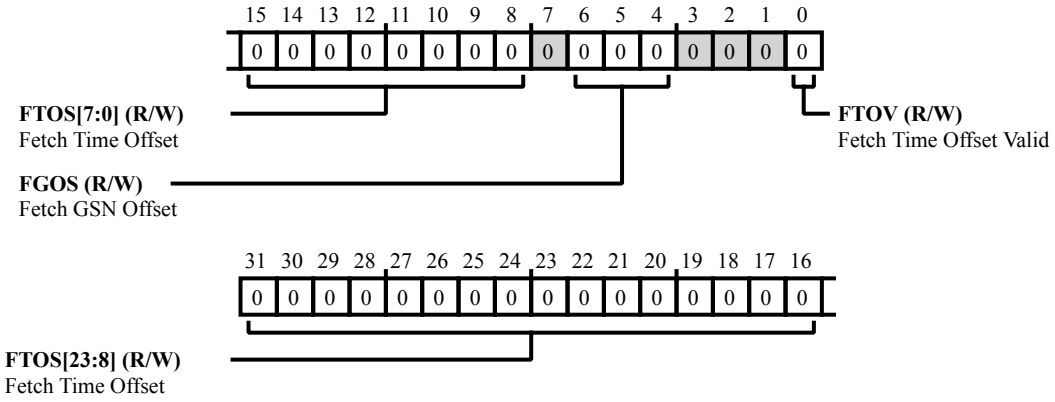


Figure 30-83: EMAC_DMA_TBS_CTL1 Register Diagram

Table 30-138: EMAC_DMA_TBS_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	FTOS	Fetch Time Offset. The value in units of 256 nanoseconds, that has to be deducted from the Launch time to compute the Fetch Time. Max value: 999,999,999 ns, additionally should be smaller than CTR-1 value when ESTM mode is enabled since this value is a modulo CTR value.	
6:4 (R/W)	FGOS	Fetch GSN Offset. The number GSN slots that must be deducted from the Launch GSN to compute the Fetch GSN. Value valid only when EMAC_DMA_TBS_CTL1.FTOV= 1.	
0 (R/W)	FTOV	Fetch Time Offset Valid. When set (=1), the EMAC_DMA_TBS_CTL1.FTOV bit indicates that the EMAC_DMA_TBS_CTL1.FTOS field is valid. When not set, indicates the Fetch Offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.	
		0	Fetch Time Offset is Invalid
		1	Fetch Time Offset is Valid

DMA TBS Control 2 Register

The EMAC_DMA_TBS_CTL2 register is used to control the Time Based Scheduling (TBS) attributes.

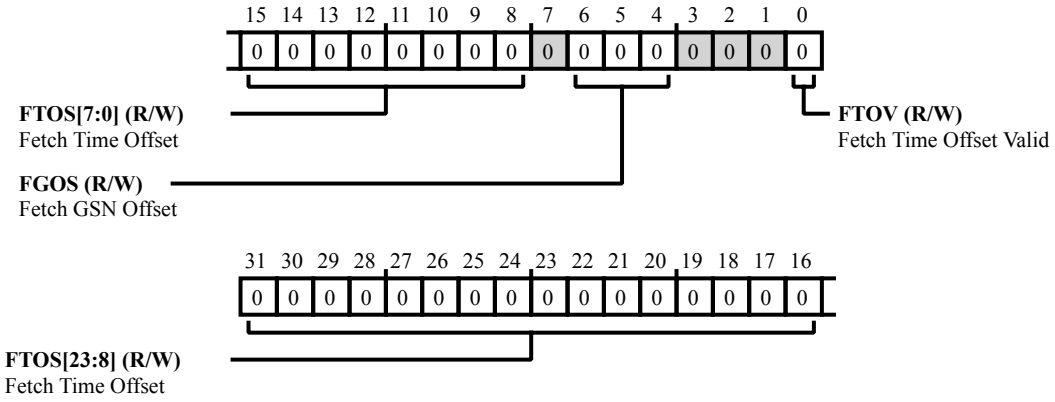


Figure 30-84: EMAC_DMA_TBS_CTL2 Register Diagram

Table 30-139: EMAC_DMA_TBS_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	FTOS	Fetch Time Offset. The value in units of 256 nanoseconds, that has to be deducted from the Launch time to compute the Fetch Time. Max value: 999,999,999 ns, additionally should be smaller than CTR-1 value when ESTM mode is enabled since this value is a modulo CTR value.	
6:4 (R/W)	FGOS	Fetch GSN Offset. The EMAC_DMA_TBS_CTL2.FGOS field indicates the number GSN slots that must be deducted from the Launch GSN to compute the Fetch GSN. Value valid only when EMAC_DMA_TBS_CTL2.FTOV= 1.	
0 (R/W)	FTOV	Fetch Time Offset Valid. When set (=1), the EMAC_DMA_TBS_CTL2.FTOV bit indicates that the EMAC_DMA_TBS_CTL2.FTOS field is valid. When not set, indicates the Fetch Offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.	
		0	Fetch Time Offset is Invalid
		1	Fetch Time Offset is Valid

DMA TBS Control 3 Register

The EMAC_DMA_TBS_CTL3 register is used to control the TBS attributes.

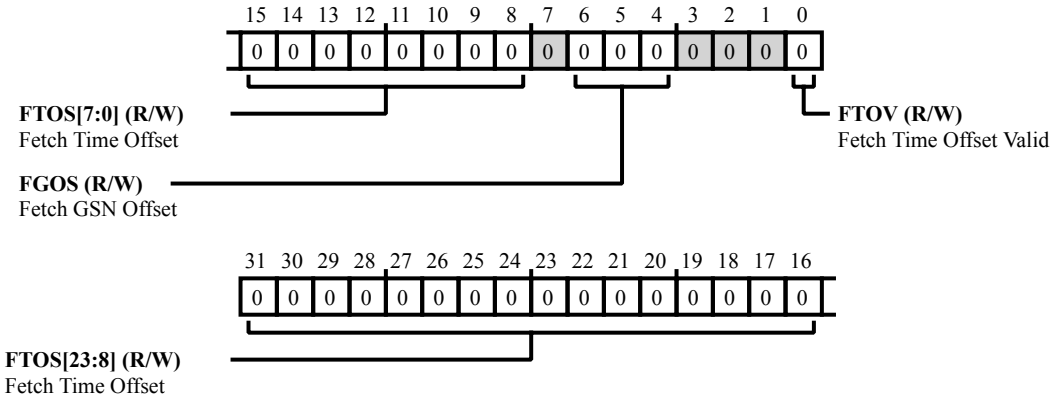


Figure 30-85: EMAC_DMA_TBS_CTL3 Register Diagram

Table 30-140: EMAC_DMA_TBS_CTL3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	FTOS	Fetch Time Offset. The value in units of 256 nanoseconds, that has to be deducted from the Launch time to compute the Fetch Time. Max value: 999,999,999 ns, additionally should be smaller than CTR-1 value when ESTM mode is enabled since this value is a modulo CTR value.	
6:4 (R/W)	FGOS	Fetch GSN Offset. The EMAC_DMA_TBS_CTL3.FGOS field indicates the number of GSN slots that must be deducted from the Launch GSN to compute the Fetch GSN. Value valid only when EMAC_DMA_TBS_CTL3.FTOV is set.	
0 (R/W)	FTOV	Fetch Time Offset Valid. When set (=1), the EMAC_DMA_TBS_CTL3.FTOV bit indicates that the EMAC_DMA_TBS_CTL3.FTOS field is valid. When cleared (=0), the EMAC_DMA_TBS_CTL3.FTOV bit indicates that the Fetch Offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.	
		0	Fetch Time Offset is Invalid
		1	Fetch Time Offset is Valid

LPI One Microsecond Tic Counter Register

The EMAC_ONEUS_TIC_CNT register controls the generation of the reference time (1 microsecond tic) for all of the LPI timers. This timer has to be programmed by the software initially.

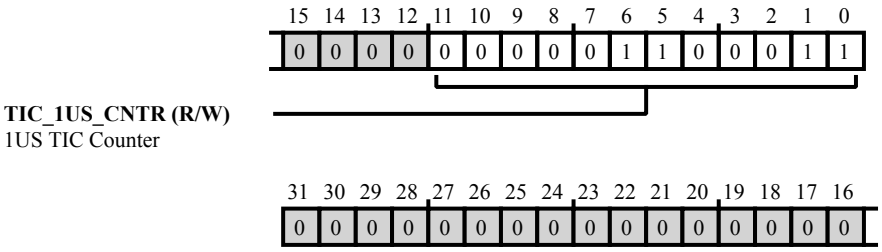


Figure 30-86: EMAC_ONEUS_TIC_CNT Register Diagram

Table 30-141: EMAC_ONEUS_TIC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	TIC_1US_CNTR	1US TIC Counter. The application must program the EMAC_ONEUS_TIC_CNT.TIC_1US_CNTR field so that the number of clock cycles of SCLK0 is 1us. (Subtract 1 from the value before programming). For example, if the SCLK0 is 100MHz, this field must be programmed to value 100 - 1 = 99 (which is 0x63). This is required to generate the 1 us events that are used to update some of the EEE-related counters.

Address Resolution Protocol (ARP) Register

The EMAC_ARP_ADDR register contains the IPv4 destination address of the MAC.

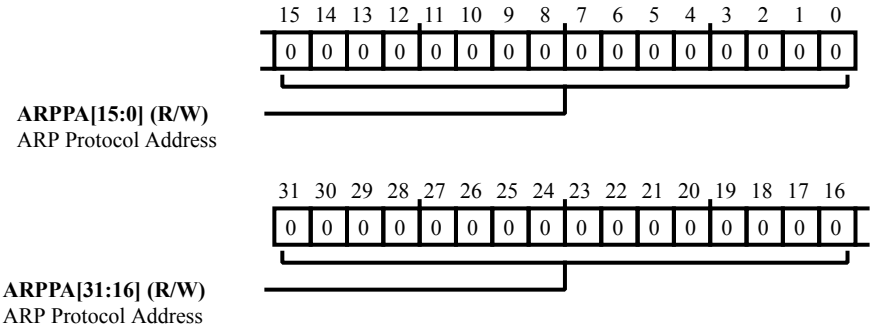


Figure 30-87: EMAC_ARP_ADDR Register Diagram

Table 30-142: EMAC_ARP_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ARPPA	ARP Protocol Address. The EMAC_ARP_ADDR.ARPPA field contains the IPv4 destination address of the MAC. This address is used for perfect match with the protocol address of the target field in the received ARP packet.

MAC Address 0 High Register

The EMAC_ADDR0_HI register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the (G)MII interface corresponds to the LS byte (Bits [7:0]) of the EMAC_ADDR0_LO register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the (G)MII as the destination address, then the MAC Address 0 Register [47:0] is compared with 0x665544332211.

The MAC address registers are configured to be double-synchronized to the (G)MII clock domains. Hence, the synchronization is triggered only when bits[31:24] (in little endian mode) or bits[7:0] (in big endian mode) of the EMAC_ADDR0_LO register are written. For proper synchronization updates, the consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

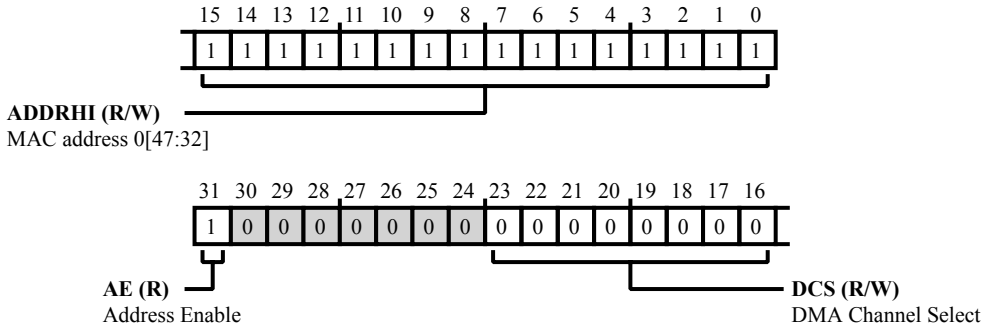


Figure 30-88: EMAC_ADDR0_HI Register Diagram

Table 30-143: EMAC_ADDR0_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	AE	Address Enable. The EMAC_ADDR0_HI.AE bit is always set (=1).	
		0	Invalid
		1	The bit is always set
23:16 (R/W)	DCS	DMA Channel Select. When EMAC_MAC_EXT_CFG.PDC = 0, the EMAC_ADDR0_HI.DCS field contains the binary representation of the DMA channel number to which an Rx packet whose DA matches the MAC Address 0 content is routed. When EMAC_MAC_EXT_CFG.PDC = 1, the EMAC_ADDR0_HI.DCS field contains the one-hot representation of one or more DMA channel numbers to which an Rx packet whose DA matches the MAC Address 0 content is routed.	
15:0 (R/W)	ADDRHI	MAC address 0[47:32]. The EMAC_ADDR0_HI.ADDRHI field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.	

MAC Address 0 Low Register

The EMAC_ADDR0_LO register holds the lower 32 bits of the 6-byte first MAC address of the station.

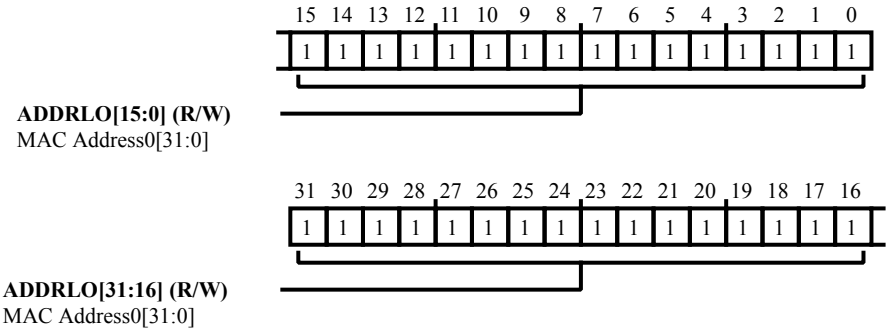


Figure 30-89: EMAC_ADDR0_LO Register Diagram

Table 30-144: EMAC_ADDR0_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRLO	MAC Address0[31:0]. The EMAC_ADDR0_LO.ADDRLO field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address n High Register

The EMAC_ADDR[i]_HI register holds the upper 16 bits of the second 6-byte MAC address of the station.

If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the EMAC_ADDR0_LO register are written. For proper synchronization updates, the consecutive writes to this EMAC_ADDR0_LO register should be performed after at least four clock cycles in the destination clock domain.

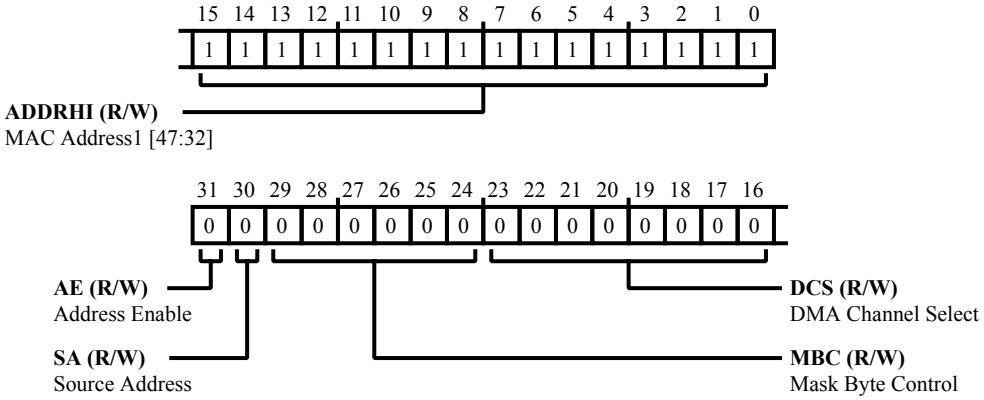


Figure 30-90: EMAC_ADDR[i]_HI Register Diagram

Table 30-145: EMAC_ADDR[i]_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	AE	Address Enable. When the EMAC_ADDR[i]_HI.AE bit is set (=1), the address filter module uses the second MAC address for perfect filtering. When EMAC_ADDR[i]_HI.AE = 0, the address filter module ignores the address for filtering.	
		0	Address is Ignored
		1	Address is Enabled
30 (R/W)	SA	Source Address. When the EMAC_ADDR[i]_HI.SA bit is set (=1), MAC address 1 [47:0] is used to compare with the SA fields of the received packet. When EMAC_ADDR[i]_HI.SA = 0, MAC address 1 [47:0] is used to compare with the DA fields of the received packet.	
		0	Compare with Destination Address
		1	Compare with Source Address

Table 30-145: EMAC_ADDR[i]_HI Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	MBC	Mask Byte Control. These bits are mask control bits for comparing each of the MAC address bytes. When set (=1), the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC address 1 registers. Each bit controls the masking of the bytes as follows: Bit 29: EMAC_ADDR[i]_HI[15:8] Bit 28: EMAC_ADDR[i]_HI[7:0] Bit 27: EMAC_ADDR[i]_HI[31:24] .. Bit 24: EMAC_ADDR[i]_HI[7:0] Filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.
23:16 (R/W)	DCS	DMA Channel Select. When EMAC_MAC_EXT_CFG.PDC = 0, the EMAC_ADDR[i]_HI.DCS field contains the binary representation of the DMA channel number to which an Rx packet whose DA matches the MAC Address(n) content is routed. When EMAC_MAC_EXT_CFG.PDC = 1, the EMAC_ADDR[i]_HI.DCS field contains the one-hot representation of one or more DMA channel numbers to which an Rx packet whose DA matches the MAC address(n) content is routed.
15:0 (R/W)	ADDRHI	MAC Address1 [47:32]. The EMAC_ADDR[i]_HI.ADDRHI field contains the upper 16 bits[47:32] of the second 6-byte MAC address.

MAC Address n Low Register

The EMAC_ADDR[i]_LO register holds the lower 32 bits of the second 6-byte MAC address of the station.

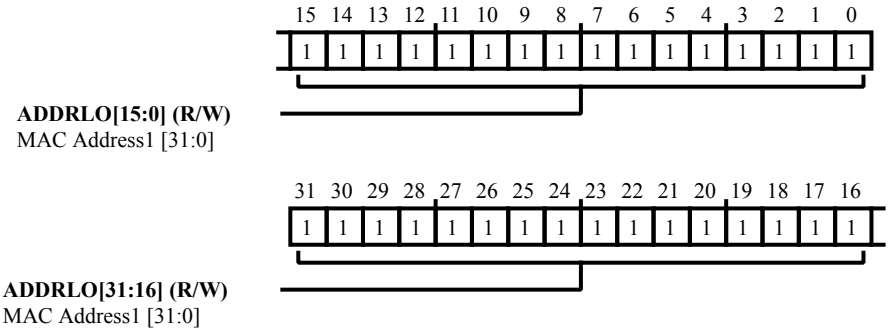


Figure 30-91: EMAC_ADDR[i]_LO Register Diagram

Table 30-146: EMAC_ADDR[i]_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDRLO	MAC Address1 [31:0]. The EMAC_ADDR[i]_LO.ADDRLO field contains the lower 32 bits of second 6-byte MAC address. The content of this field is undefined until loaded by the application after the initialization process.

Auxiliary Time Stamp Control Register

The EMAC_TM_AUX_CTL register controls the auxiliary time stamp snapshot.

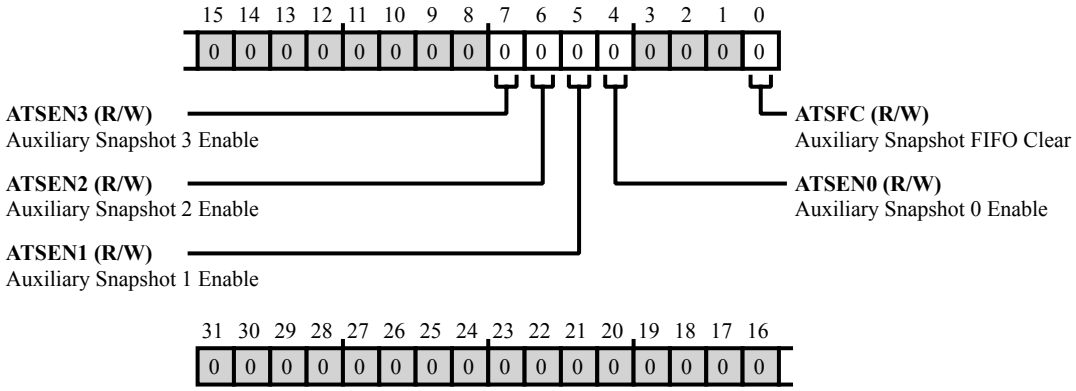


Figure 30-92: EMAC_TM_AUX_CTL Register Diagram

Table 30-147: EMAC_TM_AUX_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	ATSEN3	Auxiliary Snapshot 3 Enable. The EMAC_TM_AUX_CTL.ATSEN3 bit controls the capturing of auxiliary snapshot trigger 3. When the EMAC_TM_AUX_CTL.ATSEN3 bit is set, the auxiliary snapshot of the event on ETH0_PTPAUX_MCG_IN3 input is enabled. When EMAC_TM_AUX_CTL.ATSEN3 = 0, the events on this input are ignored.	
		0	Disable
		1	Enable
6 (R/W)	ATSEN2	Auxiliary Snapshot 2 Enable. The EMAC_TM_AUX_CTL.ATSEN2 bit controls the capturing of auxiliary snapshot trigger 2. When the EMAC_TM_AUX_CTL.ATSEN2 bit is set, the auxiliary snapshot of the event on ETH0_PTPAUX_MCG_IN2 input is enabled. When EMAC_TM_AUX_CTL.ATSEN2 = 0, the events on this input are ignored.	
		0	Disable
		1	Enable
5 (R/W)	ATSEN1	Auxiliary Snapshot 1 Enable. The EMAC_TM_AUX_CTL.ATSEN1 bit controls the capturing of auxiliary snapshot trigger 1. When the EMAC_TM_AUX_CTL.ATSEN1 bit is set, the auxiliary snapshot of the event on ETH0_PTPAUX_MCG_IN1 input is enabled. When EMAC_TM_AUX_CTL.ATSEN1 = 0, the events on this input are ignored.	
		0	Disable
		1	Enable

Table 30-147: EMAC_TM_AUX_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	ATSEN0	Auxiliary Snapshot 0 Enable. The EMAC_TM_AUX_CTL.ATSEN0 bit controls the capturing of auxiliary snapshot trigger 0. When the EMAC_TM_AUX_CTL.ATSEN0 bit is set, the auxiliary snapshot of the event on ETH0_PTPAUX_MCG_IN0 input is enabled. When EMAC_TM_AUX_CTL.ATSEN0 = 0, the events on this input are ignored.	
		0	Disable
		1	Enable
0 (R/W)	ATSFC	Auxiliary Snapshot FIFO Clear. When set, the EMAC_TM_AUX_CTL.ATSFC bit resets the pointers of the Auxiliary Snapshot FIFO. The EMAC_TM_AUX_CTL.ATSFC bit is cleared when the pointers are reset and the FIFO is empty. When the EMAC_TM_AUX_CTL.ATSFC bit is high, the auxiliary snapshots are stored in the FIFO. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Disable
		1	Enable

Auxiliary Time Stamp Nanoseconds Register

The EMAC_TM_AUXTS_NSEC register, along with EMAC_TM_AUXTS_SEC register, gives the 64-bit time stamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4, 8, or 16 as selected while configuring the IP.

Multiple snapshots can be stored in this FIFO. Bits[29:25] in EMAC_TM_STAT register indicate the fill-level of the FIFO. The top of the FIFO is removed only when the last byte of EMAC_TM_AUXTS_SEC register is read. In the Little Endian mode, this means when bits[31:24] are read and in Big Endian mode, bits[7:0] are read.

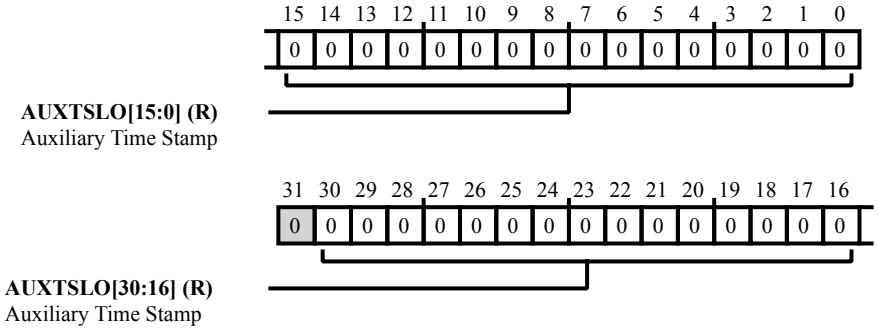


Figure 30-93: EMAC_TM_AUXTS_NSEC Register Diagram

Table 30-148: EMAC_TM_AUXTS_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30:0 (R/NW)	AUXTSLO	Auxiliary Time Stamp. The EMAC_TM_AUXTS_NSEC.AUXTSLO field contains the lower 31 bits (nanoseconds field) of the auxiliary time stamp.

Auxiliary Time Stamp Seconds Register

The EMAC_TM_AUXTS_SEC register contains the lower 32 bits of the seconds field of the auxiliary time stamp register.

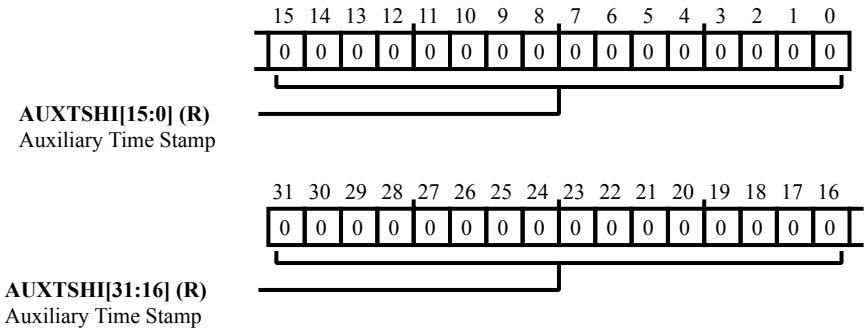


Figure 30-94: EMAC_TM_AUXTS_SEC Register Diagram

Table 30-149: EMAC_TM_AUXTS_SEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	AUXTSHI	Auxiliary Time Stamp. Contains the lower 32 bits of the seconds field of the auxiliary time stamp.

CSR Software Control Register

The EMAC_CSR_SW_CTL register contains software programmable controls for changing the Control and Status Registers (CSR) access response and status bits clearing.

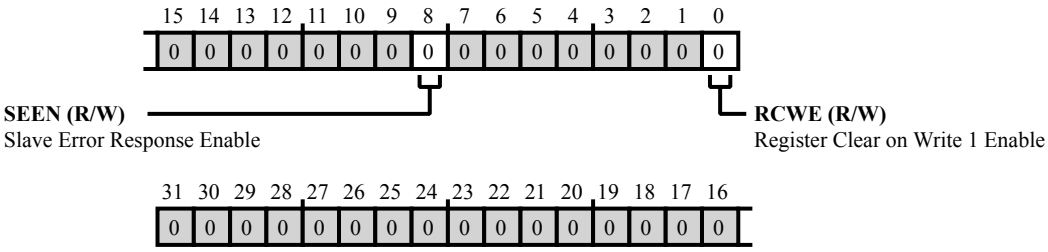


Figure 30-95: EMAC_CSR_SW_CTL Register Diagram

Table 30-150: EMAC_CSR_SW_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	SEEN	Slave Error Response Enable. When the EMAC_CSR_SW_CTL.SEEN bit is set, the MAC responds with a target error for accesses to reserved registers in CSR space. When EMAC_CSR_SW_CTL.SEEN = 0, the MAC responds with an okay response to any register accessed from CSR space.	
		0	Disable
		1	Enable
0 (R/W)	RCWE	Register Clear on Write 1 Enable. When the EMAC_CSR_SW_CTL.RCWE bit is set, the access mode of some register fields changes to (W1C) write one to clear; the application must set that respective bit to 1 to clear it. When EMAC_CSR_SW_CTL.RCWE = 0, the access mode of these register fields remain as Clear on read.	
		0	Disable
		1	Enable

MAC Configuration Register

The EMAC_MAC_CFG register establishes the operating mode of the MAC.

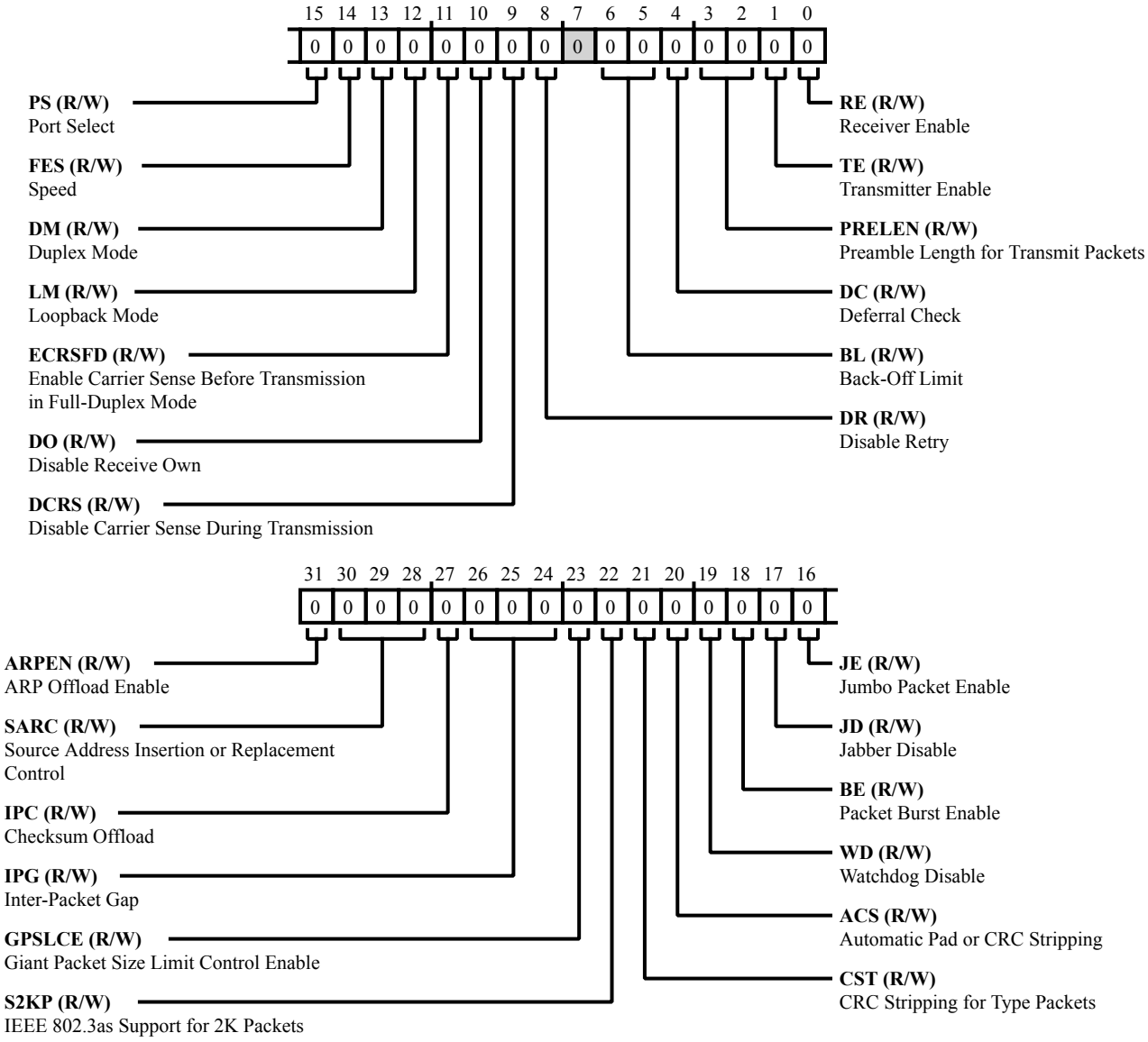


Figure 30-96: EMAC_MAC_CFG Register Diagram

Table 30-151: EMAC_MAC_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ARPEN	ARP Offload Enable. When the EMAC_MAC_CFG.ARPEN bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It forwards the ARP packet to the application and also indicate the events in the RxStatus. When EMAC_MAC_CFG.ARPEN = 0, the MAC receiver does not recognize any ARP packet and indicates them as type frame in the RxStatus. The EMAC_MAC_CFG.ARPEN bit is available only when enable IPv4 ARP offload is selected.	
		0	ARP Offload is Disabled
		1	ARP Offload is Enabled
30:28 (R/W)	SARC	Source Address Insertion or Replacement Control. The EMAC_MAC_CFG.SARC field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of bits[29:28]: 2'b0x: - Reserved 2'b10: - If Bit 30 = 0, the MAC inserts the content of the MAC Address 0 registers in the SA field of all transmitted packets. - If Bit 30 = 1 and the Enable MAC Address Register 1 option is selected while configuring the IP, the MAC inserts the content of the MAC Address 1 registers in the SA field of all transmitted packets. 2'b11: - If Bit 30 = 0, the MAC replaces the content of the MAC Address 0 registers in the SA field of all transmitted packets. - If Bit 30 = 1 and the MAC Address Register 1 is enabled, the MAC replaces the content of the MAC Address 1 registers in the SA field of all transmitted packets. Note: Changes to this field take effect only on the start of a packet. A write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.	
		0	Reserved
		2	Contents of MAC Address 0 Inserted in SA Field
		3	Contents of MAC Address 0 Replaces SA Field
		6	Contents of MAC Address 1 Inserted in SA Field
		7	Contents of MAC Address 1 Replaces SA Field

Table 30-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	IPC	Checksum Offload. When set, the EMAC_MAC_CFG.IPC bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When EMAC_MAC_CFG.IPC = 0, the COE function in the receiver is disabled. The layer 3 and layer 4 packet filter and enable split header features automatically select the IPC full checksum offload engine on the receive side. When any of these features are enabled, the EMAC_MAC_CFG.IPC bit must be set.	
		0	IP Header/payload Checksum Checking is Disabled
		1	IP Header/payload Checksum Checking is Enabled
26:24 (R/W)	IPG	Inter-Packet Gap. The EMAC_MAC_CFG.IPG field control the minimum IPG between packets during transmission. This range of minimum IPG is valid in full-duplex mode. In the half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered. When a JAM pattern is being transmitted because of back pressure activation, the MAC does not consider the minimum IPG. This function (IPG less than 96 bit times) is valid only when EMAC_MAC_EXT_CFG.EIPGEN = 0. When EMAC_MAC_EXT_CFG.EIPGEN = 1, the minimum IPG (greater than 96 bit times) is controlled as per the description given in the EMAC_MAC_EXT_CFG.EIPG field register.	
		0	96 Bit Times IPG
		1	88 Bit Times IPG
		2	80 Bit Times IPG
		3	72 Bit Times IPG
		4	64 Bit Times IPG
		5	56 Bit Times IPG
		6	48 Bit Times IPG
		7	40 Bit Times IPG

Table 30-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	GPSLCE	Giant Packet Size Limit Control Enable. When the EMAC_MAC_CFG.GPSLCE bit is set, the MAC considers the value in EMAC_MAC_EXT_CFG.GPSL to declare a received packet as giant packet. The EMAC_MAC_CFG.GPSLCE field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit. When EMAC_MAC_CFG.GPSLCE = 0, the MAC considers a received packet as giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet). The watchdog timeout limit, jumbo packet enable and 2K packet enable have higher precedence over the EMAC_MAC_CFG.GPSLCE bit, that is the MAC considers a received packet as giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with jumbo packet enabled and greater than 2,000 bytes with 2K packet enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.	
		0	Giant Packet Size Limit Control is Disabled
		1	Giant Packet Size Limit Control is Enabled
22 (R/W)	S2KP	IEEE 802.3as Support for 2K Packets. When the EMAC_MAC_CFG.S2KP bit is set (=1), the MAC considers all packets with up to 2,000 bytes length as normal packets. When EMAC_MAC_CFG.JE = 0, the MAC considers all received packets of size more than 2K bytes as giant packets. When EMAC_MAC_CFG.S2KP = 0 and EMAC_MAC_CFG.JE = 0, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. Note: When EMAC_MAC_CFG.JE = 1, setting the EMAC_MAC_CFG.S2KP bit has no effect on the giant packet status.	
		0	Support Upto 2K Packet is Disabled
		1	Support Upto 2K Packet is Enabled
21 (R/W)	CST	CRC Stripping for Type Packets. When the EMAC_MAC_CFG.CST bit is set (=1), the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application.	
		0	CRC Stripping for Type Packets is Disabled
		1	CRC Stripping for Type Packets is Enabled

Table 30-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	ACS	Automatic Pad or CRC Stripping. When the EMAC_MAC_CFG.ACS bit is set, the MAC strips the pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When EMAC_MAC_CFG.ACS = 0, the MAC passes all incoming packets to the application, without any modification.	
		0	Automatic Pad or CRC Stripping is Disabled
		1	Automatic Pad or CRC Stripping is Enabled
19 (R/W)	WD	Watchdog Disable. When the EMAC_MAC_CFG.WD bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes. When EMAC_MAC_CFG.WD = 0, the MAC does not allow more than 2,048 bytes (10,240 if EMAC_MAC_CFG.JE=1) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.	
		0	Watchdog is Enabled
		1	Watchdog is Disabled
18 (R/W)	BE	Packet Burst Enable. When the EMAC_MAC_CFG.BE bit is set, the MAC allows packet bursting during transmission in the RGMII half-duplex mode.	
		0	Packet Burst is Disabled
		1	Packet Burst is Enabled
17 (R/W)	JD	Jabber Disable. When the EMAC_MAC_CFG.JD bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes. When EMAC_MAC_CFG.JD = 0, if the application sends more than 2,048 bytes of data (10,240 if EMAC_MAC_CFG.JE is set (=1)) during transmission, the MAC does not send rest of the bytes in that packet.	
		0	Jabber is Enabled
		1	Jabber is Disabled

Table 30-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	JE	Jumbo Packet Enable. When the EMAC_MAC_CFG.JE bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.	
		0	Jumbo Packet is Disabled
		1	Jumbo Packet is Enabled
15 (R/W)	PS	Port Select. The EMAC_MAC_CFG.PS bit selects the Ethernet line speed. The EMAC_MAC_CFG.PS bit, along with bit 14, selects the exact line speed. In the 10/100 Mbps only (always 1) or 1000 Mbps only (always 0) configurations, the EMAC_MAC_CFG.PS bit is read-only (RO) with appropriate value. In the default 10/100/1000 Mbps configurations, the EMAC_MAC_CFG.PS bit is read-write (R/W).	
		0	1000 Mbps Operation
		1	10 or 100 Mbps Operation
14 (R/W)	FES	Speed. The EMAC_MAC_CFG.FES bit selects the speed mode.	
		0	10 Mbps when PS = 1; 1 Gbps when PS = 0
		1	100 Mbps when PS = 1; reserved when PS = 0
13 (R/W)	DM	Duplex Mode. When the EMAC_MAC_CFG.DM bit is set, the MAC operates in the full-duplex mode in which it can transmit and receive simultaneously. The EMAC_MAC_CFG.DM bit is RO with default value of 1'b1 in the full-duplex-only configurations.	
		0	Half-duplex Mode
		1	Full-duplex Mode
12 (R/W)	LM	Loopback Mode. When the EMAC_MAC_CFG.LM bit is set, the MAC operates in the loopback mode at GMII or MII. The (G)MII Rx clock input (clk_rx_i) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.	
		0	Loopback is Disabled
		1	Loopback is Enabled

Table 30-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	ECRSFD	Enable Carrier Sense Before Transmission in Full-Duplex Mode. When the EMAC_MAC_CFG.ECRSFD bit is set, the MAC transmitter checks the CRS signal before packet transmission in the full-duplex mode. The MAC starts the transmission only when the CRS signal is low. When EMAC_MAC_CFG.ECRSFD = 0, the MAC transmitter ignores the status of the CRS signal.	
		0	ECRSFD is Disabled
		1	ECRSFD is Enabled
10 (R/W)	DO	Disable Receive Own. When the EMAC_MAC_CFG.DO bit is set, the MAC disables the reception of packets when the gmii_txen_o is asserted in the half-duplex mode. When EMAC_MAC_CFG.DO = 0, the MAC receives all packets given by the PHY. The EMAC_MAC_CFG.DO bit is not applicable in the full-duplex mode.	
		0	Enable Receive Own
		1	Disable Receive Own
9 (R/W)	DCRS	Disable Carrier Sense During Transmission. When the EMAC_MAC_CFG.DCRS bit is set, the MAC transmitter ignores the (G)MII CRS signal during packet transmission in the half-duplex mode. As a result, no errors are generated because of loss of carrier or no carrier during transmission. When EMAC_MAC_CFG.DCRS = 0, the MAC transmitter generates errors because of carrier sense. The MAC can even abort the transmission.	
		0	Enable Carrier Sense During Transmission
		1	Disable Carrier Sense During Transmission
8 (R/W)	DR	Disable Retry. When the EMAC_MAC_CFG.DR bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current packet transmission and reports a packet abort error with excessive collision error in the Tx packet status. When EMAC_MAC_CFG.DR = 0, the MAC retries based on the settings of the BL field. The EMAC_MAC_CFG.DR bit is applicable only in the half-duplex mode.	
		0	Enable Retry
		1	Disable Retry

Table 30-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:5 (R/W)	BL	Back-Off Limit. The EMAC_MAC_CFG.BL field determines the random integer number (r) of slot time delays (4,096 bit times for 1000/2500 Mbps; 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. n = retransmission attempt. The random integer r takes the value in the range 0 <= r < 2^k The EMAC_MAC_CFG.BL bit is applicable only in the half-duplex mode.	
		0	k is Minimum of Retransmission Attempt and 10
		1	k is Minimum of Retransmission Attempt and 8
		2	k is Minimum of Retransmission Attempt and 4
		3	k is Minimum of Retransmission Attempt and 1
4 (R/W)	DC	Deferral Check. When the EMAC_MAC_CFG.DC bit is set, the deferral check function is enabled in the MAC. The MAC issues a packet abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode. If the MAC is configured for 1000 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on GMII or MII. The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset (=0), and it is restarted. When EMAC_MAC_CFG.DC = 0, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. The EMAC_MAC_CFG.DC bit is applicable only in the half-duplex mode.	
		0	Deferral Check Function is Disabled
		1	Deferral Check Function is Enabled

Table 30-151: EMAC_MAC_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	PRELEN	Preamble Length for Transmit Packets. The EMAC_MAC_CFG.PRELEN field control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the full-duplex mode.	
		0	7 Bytes of Preamble
		1	5 Bytes of Preamble
		2	3 Bytes of Preamble
		3	Reserved
1 (R/W)	TE	Transmitter Enable. When the EMAC_MAC_CFG.TE bit is set, the Tx state machine of the MAC is enabled for transmission on the GMII or MII interface. When EMAC_MAC_CFG.TE = 0, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.	
		0	Transmitter is Disabled
		1	Transmitter is Enabled
0 (R/W)	RE	Receiver Enable. When the EMAC_MAC_CFG.RE bit is set, the Rx state machine of the MAC is enabled for receiving packets from the GMII or MII interface. When the EMAC_MAC_CFG.RE = 0, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the GMII or MII interface.	
		0	Receiver is Disabled
		1	Receiver is Enabled

Debug Register

The EMAC_MAC_DBG register provides the debug status of various MAC blocks.

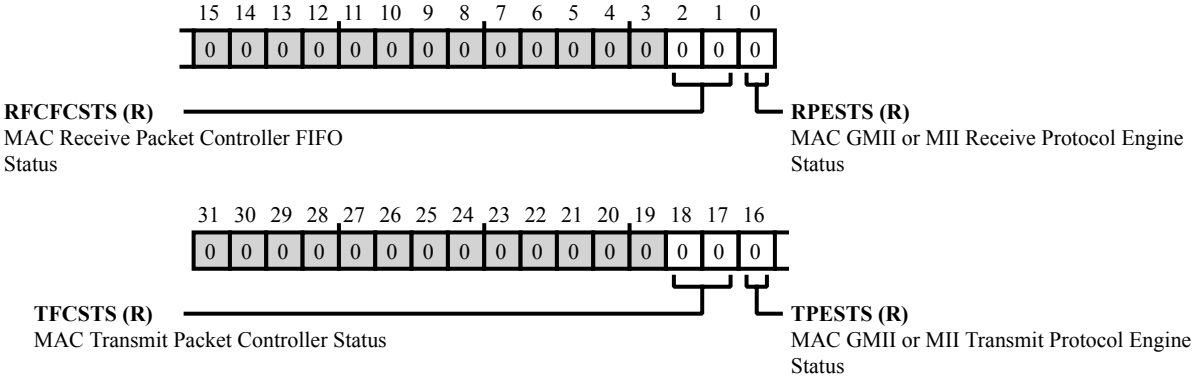


Figure 30-97: EMAC_MAC_DBG Register Diagram

Table 30-152: EMAC_MAC_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18:17 (R/NW)	TFCSTS	MAC Transmit Packet Controller Status. The EMAC_MAC_DBG.TFCSTS field indicates the state of the MAC Transmit Packet Controller module.	
		0	Idle State
		1	Waiting for One of the Following: Status of the Previous Packet OR IPG or Back off Period to Be Over
		2	Generating and Transmitting a Pause Control Packet (in Full-Duplex Mode)
		3	Transferring Input Packet for Transmission
16 (R/NW)	TPESTS	MAC GMII or MII Transmit Protocol Engine Status. When the EMAC_MAC_DBG.TPESTS bit is set, it indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.	
		0	MAC GMII or MII Transmit Protocol Engine Status Not Detected
		1	MAC GMII or MII Transmit Protocol Engine Status Detected
2:1 (R/NW)	RFCFCSTS	MAC Receive Packet Controller FIFO Status. The EMAC_MAC_DBG.RFCFCSTS field indicates the active state of the small FIFO read and write controllers of the MAC Receive Packet Controller module.	

Table 30-152: EMAC_MAC_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	RPESTS	MAC GMII or MII Receive Protocol Engine Status. When the `EMAC_MAC_DBG.RPESTS` bit is set, it indicates that the MAC GMII or MII receive protocol engine is actively receiving data, and it is not in the Idle state.	
		0	MAC GMII or MII Receive Protocol Engine Status Not Detected
		1	MAC GMII or MII Receive Protocol Engine Status Detected

MAC Split Mode and Offset Configuration Register

The EMAC_SPLM_OFST_CFG register contains the split mode control field and offset field for the split header feature.

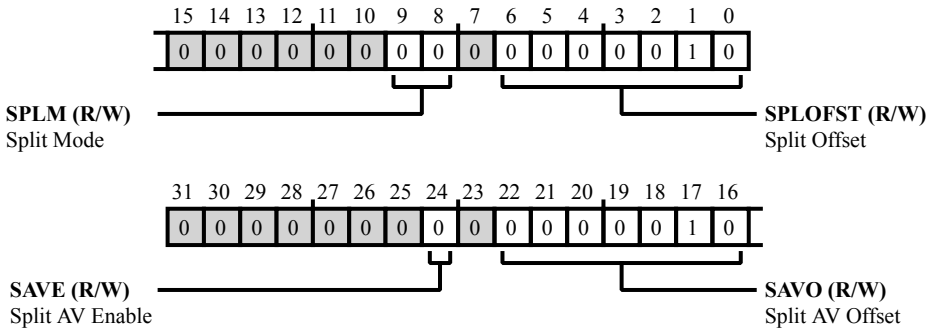


Figure 30-98: EMAC_SPLM_OFST_CFG Register Diagram

Table 30-153: EMAC_SPLM_OFST_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	SAVE	Split AV Enable. When EMAC_SPLM_OFST_CFG.SAVE = 1, and the received packet is an AV Type packet, the header is split at SAVO bytes from the beginning of Length/Type field of the packet, for L2 Split. When EMAC_SPLM_OFST_CFG.SAVE = 0, header is split at SPLOFST bytes from the beginning of Length/Type field of the frame, for L2 Split.	
22:16 (R/W)	SAVO	Split AV Offset. When SAVE bit is set (=1), and the received packet is an AV Type packet, the EMAC_SPLM_OFST_CFG.SAVO field indicates the value of the offset from the beginning of Length/Type field at which header should be split, when appropriate SPLM is selected. The reset value of this field is 2 bytes, indicating a split at L2 header. Value is in terms of bytes.	
9:8 (R/W)	SPLM	Split Mode. The EMAC_SPLM_OFST_CFG.SPLM field indicates the mode of splitting the incoming Rx packets.	
		0	Split at L3/L4 Header
		1	Split at L2 Header with an Offset. Always Split at SPLOFST Bytes from the Beginning of Length/Type Field of the Frame
		2	Combination Mode: Split Similar to SPLM=00 for IP Packets That are Untagged or Tagged and VLAN Stripped
		3	Reserved

Table 30-153: EMAC_SPLM_OFST_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	SPLOFST	Split Offset. The `EMAC_SPLM_OFST_CFG.SPLOFST` field indicates the value of offset from the beginning of Length/Type field at which header split should take place when the appropriate SPLM is selected. The reset value of this field is 2 bytes indicating a split at L2 header. Value is in terms of bytes.

MAC Extended Configuration Register

The EMAC_MAC_EXT_CFG register establishes the operating mode of the MAC.

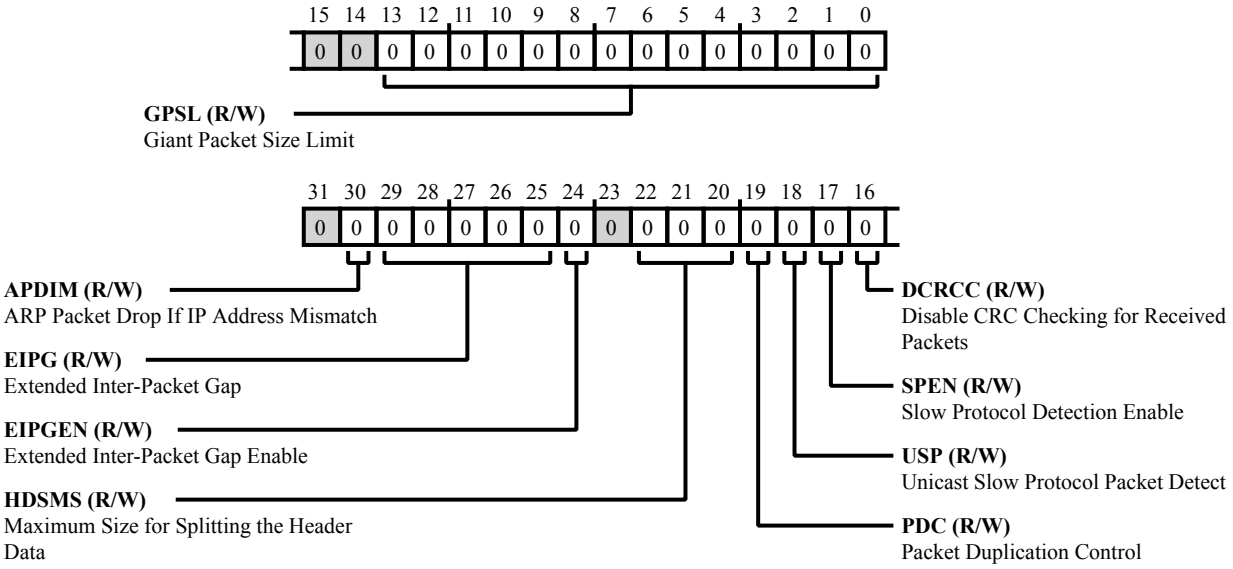


Figure 30-99: EMAC_MAC_EXT_CFG Register Diagram

Table 30-154: EMAC_MAC_EXT_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	APDIM	ARP Packet Drop If IP Address Mismatch. When the EMAC_MAC_EXT_CFG.APDIM bit is set, packet for which target protocol address does not match IPv4 address is dropped in the MTL layer. When EMAC_MAC_EXT_CFG.APDIM = 0, when target protocol address does not match, packet is forwarded to MTL maintaining backward compatibility	
		0	Mux Select to Drop the Arp Packet If Target Protocol Address Mismatches IPv4 Address Disabled
		1	Mux Select to Drop the Arp Packet If Target Protocol Address Mismatches IPv4 Address Enabled

Table 30-154: EMAC_MAC_EXT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29:25 (R/W)	EIPG	Extended Inter-Packet Gap. The value in the EMAC_MAC_EXT_CFG.EIPG field is applicable when EMAC_MAC_EXT_CFG.EIPGEN = 1. The EMAC_MAC_EXT_CFG.EIPG field (as most significant bits), along with IPG, gives the minimum IPG greater than 96 bit times in steps of 8 bit times: {EIPG, IPG} 8'h00 - 104 bit times 8'h01 - 112 bit times 8'h02 - 120 bit times ---------- 8'hFF - 2144 bit times	
24 (R/W)	EIPGEN	Extended Inter-Packet Gap Enable. When the EMAC_MAC_EXT_CFG.EIPGEN bit is set, the MAC interprets EMAC_MAC_CFG.IPG and EMAC_MAC_CFG.EIPG together as minimum IPG greater than 96 bit times in steps of 8 bit times. When EMAC_MAC_EXT_CFG.EIPGEN = 0, the MAC ignores EMAC_MAC_CFG.EIPG and interprets EMAC_MAC_CFG.IPGas minimum IPG less than or equal to 96 bit times in steps of 8 bit times. Note: Enable the extended inter-packet gap feature only when operating in full-duplex mode. There might be undesirable effects on back-pressure function and frame transmission if it is enabled in half-duplex mode.	
		0	Extended Inter-Packet Gap is Disabled
		1	Extended Inter-Packet Gap is Enabled
22:20 (R/W)	HDSMS	Maximum Size for Splitting the Header Data. The EMAC_MAC_EXT_CFG.HDSMS field indicates the maximum header size allowed for splitting the header data in the received packet.	
		0	Maximum Size for Splitting the Header Data is 64 Bytes
		1	Maximum Size for Splitting the Header Data is 128 Bytes
		2	Maximum Size for Splitting the Header Data is 256 Bytes
		3	Maximum Size for Splitting the Header Data is 512 Bytes
		4	Maximum Size for Splitting the Header Data is 1024 Bytes
		5	Reserved

Table 30-154: EMAC_MAC_EXT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	PDC	Packet Duplication Control. When the `EMAC_MAC_EXT_CFG.PDC` bit is set, the received packet with multicast/broadcast destination address is routed to multiple receive DMA channels. The receive DMA channels is identified by the DCS field of EMAC_ADDR[n]_HI register corresponding to the MAC Address register that matches the multicast/broadcast destination address in the received packet. The DCS field is interpreted to be a one-hot value, each bit corresponding to the receive DMA channel. When `EMAC_MAC_EXT_CFG.PDC` = 0, the received packet is routed to single receive DMA channel. The receive DMA channel is identified by EMAC_ADDR[n]_HI.DCS corresponding to the MAC Address register that matches the Destination address in the received packet. The DCS field is interpreted as a binary value.	
		0	Packet Duplication Control is Disabled
		1	Packet Duplication Control is Enabled
18 (R/W)	USP	Unicast Slow Protocol Packet Detect. When the `EMAC_MAC_EXT_CFG.USP` bit is set, the MAC detects the slow protocol packets with unicast address of the station specified in the EMAC_ADDR0_HI and EMAC_ADDR0_LO registers. The MAC also detects the slow protocol packets with the slow protocols multicast address (01-80-C2-00-00-02). When `EMAC_MAC_EXT_CFG.USP` = 0, the MAC detects only slow protocol packets with the slow protocol multicast address specified in the IEEE 802.3-2015, Section 5.	
		0	Unicast Slow Protocol Packet Detection is Disabled
		1	Unicast Slow Protocol Packet Detection is Enabled
17 (R/W)	SPEN	Slow Protocol Detection Enable. When the `EMAC_MAC_EXT_CFG.SPEN` bit is set, MAC processes the slow protocol packets (Ether Type 0x8809) and provides the slow protocol subtype and code fields in Rx status. When `EMAC_MAC_EXT_CFG.SPEN` = 0, the MAC forwards all error-free slow protocol packets to the application. The MAC considers such packets as normal type packets.	
		0	Slow Protocol Detection is Disabled
		1	Slow Protocol Detection is Enabled

Table 30-154: EMAC_MAC_EXT_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	DCRCC	Disable CRC Checking for Received Packets. When the EMAC_MAC_EXT_CFG.DCRCC bit is set, the MAC receiver does not check the CRC field in the received packets. When EMAC_MAC_EXT_CFG.DCRCC = 0, the MAC receiver always checks the CRC field in the received packets.	
		0	CRC Checking is Enabled
		1	CRC Checking is Disabled
13:0 (R/W)	GPSL	Giant Packet Size Limit. If the received packet size is greater than the value programmed in the EMAC_MAC_EXT_CFG.GPSL field in units of bytes, the MAC declares the received packet as giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes. For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. When the enable double VLAN processing option is selected, the MAC adds 8 bytes to the programmed value for double VLAN tagged packets. The value in this field is applicable when EMAC_MAC_CFG.GPSLCE = 1.	

Frame Preemption Control and Status Register

The EMAC_FPE_CTLSTAT register controls the operation of frame preemption.

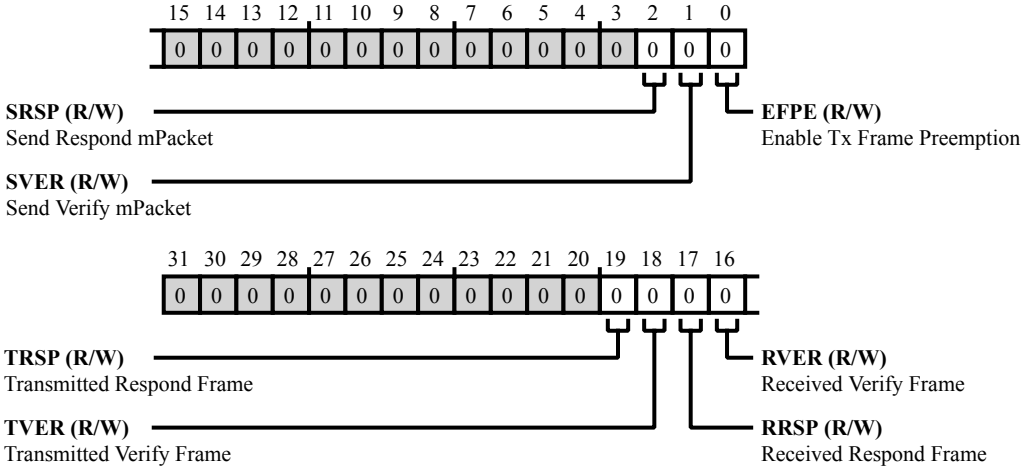


Figure 30-100: EMAC_FPE_CTLSTAT Register Diagram

Table 30-155: EMAC_FPE_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	TRSP	Transmitted Respond Frame. The EMAC_FPE_CTLSTAT.TRSP bit is set when a Respond mPacket is transmitted (triggered by EMAC_FPE_CTLSTAT.SRSP = 1). An interrupt can be generated for this event if EMAC_MAC_IEN.FPEIE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Not Transmitted Respond Frame
		1	Transmitted Respond Frame
18 (R/W)	TVER	Transmitted Verify Frame. The EMAC_FPE_CTLSTAT.TVER bit is set when a Verify mPacket is transmitted (triggered by EMAC_FPE_CTLSTAT.SVER = 1). An interrupt can be generated for this event if EMAC_MAC_IEN.FPEIE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Not Transmitted Verify Frame
		1	Transmitted Verify Frame

Table 30-155: EMAC_FPE_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	RRSP	Received Respond Frame. The `EMAC_FPE_CTLSTAT.RRSP` bit is set when a Respond mPacket is received. An interrupt can be generated for this event if EMAC_MAC_IEN.FPEIE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Not Received Respond Frame
		1	Received Respond Frame
16 (R/W)	RVER	Received Verify Frame. The `EMAC_FPE_CTLSTAT.RVER` bit is set when a Verify mPacket is received. An interrupt can be generated for this event if EMAC_MAC_IEN.FPEIE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Not Received Verify Frame
		1	Received Verify Frame
2 (R/W)	SRSP	Send Respond mPacket. When set (=1), the `EMAC_FPE_CTLSTAT.SRSP` bit indicates to hardware to send a Respond mPacket. Reset by hardware after sending the Respond mPacket. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Send Respond MPacket is Disabled
		1	Send Respond MPacket is Enabled
1 (R/W)	SVER	Send Verify mPacket. When set (=1), the `EMAC_FPE_CTLSTAT.SVER` bit indicates to hardware to send a verify mPacket. Reset by hardware after sending the Verify mPacket. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Send Verify MPacket is Disabled
		1	Send Verify MPacket is Enabled
0 (R/W)	EFPE	Enable Tx Frame Preemption. When set (=1), the `EMAC_FPE_CTLSTAT.EFPE` bit enables Tx Frame Preemption.	
		0	Tx Frame Preemption is Disabled
		1	Tx Frame Preemption is Enabled

Hardware Feature Presence Register 0

The EMAC_HW_FTR_PSNT0 register indicates the presence of first set of the optional features or functions of the EMAC module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

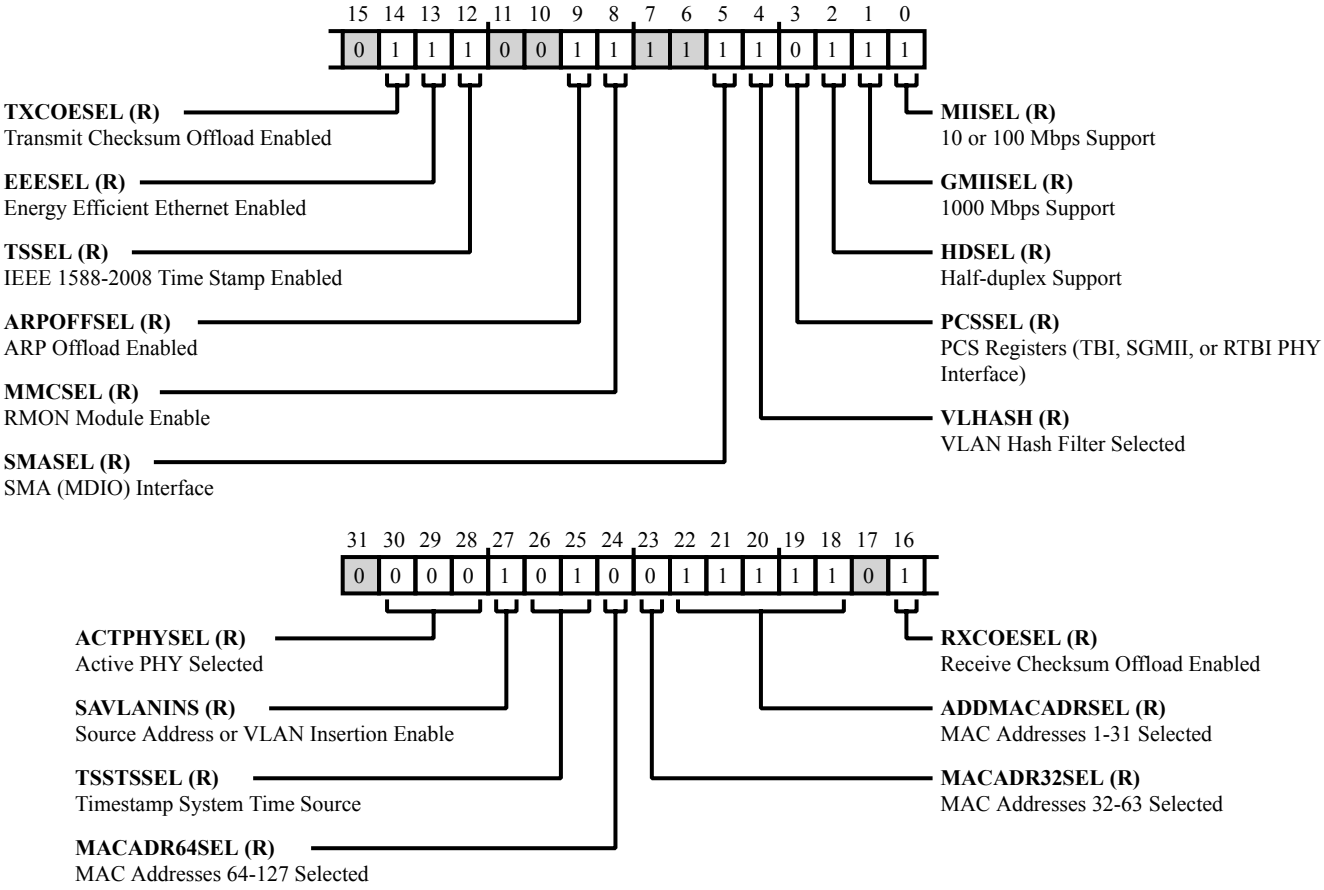


Figure 30-101: EMAC_HW_FTR_PSNT0 Register Diagram

Table 30-156: EMAC_HW_FTR_PSNT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:28 (R/NW)	ACTPHYSEL	Active PHY Selected. When there are multiple PHY interfaces in the configuration, the EMAC_HW_FTR_PSNT0.ACTPHYSEL field indicates the sampled value of PADS_PCFG0.EMACPHYISEL during reset deassertion.	
		0	GMII or MII
		1	RGMII
		2	SGMII

Table 30-156: EMAC_HW_FTR_PSNT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	TBI
		4	RMII
		5	RTBI
		6	SMII
		7	RevMII
27 (R/NW)	SAVLANINS	Source Address or VLAN Insertion Enable. The EMAC_HW_FTR_PSNT0.SAVLANINS bit is set (=1) when the Enable SA and VLAN Insertion on Tx option is selected.	
		0	Source Address or VLAN Insertion Enable Option is Not Selected
		1	Source Address or VLAN Insertion Enable Option is Selected
26:25 (R/NW)	TSSTSSEL	Timestamp System Time Source. The EMAC_HW_FTR_PSNT0.TSSTSSEL bit indicates the source of the time stamp system time. The EMAC_HW_FTR_PSNT0.TSSTSSEL bit is set (=1) when the Enable IEEE 1588 time stamp Support option is selected.	
		0	Internal
		1	External
		2	Both
		3	Reserved
24 (R/NW)	MACADR64SEL	MAC Addresses 64-127 Selected. The EMAC_HW_FTR_PSNT0.MACADR64SEL bit is set (=1) when the Enable Additional 64 MAC Address Registers (64-127) option is selected.	
		0	MAC Addresses 64-127 Select Option is Not Selected
		1	MAC Addresses 64-127 Select Option is Selected
23 (R/NW)	MACADR32SEL	MAC Addresses 32-63 Selected. The EMAC_HW_FTR_PSNT0.MACADR32SEL bit is set (=1) when the Enable Additional 32 MAC Address Registers (32-63) option is selected.	
		0	MAC Addresses 32-63 Select Option is Not Selected
		1	MAC Addresses 32-63 Select Option is Selected
22:18 (R/NW)	ADDMACADRSEL	MAC Addresses 1-31 Selected. The EMAC_HW_FTR_PSNT0.ADDMACADRSEL bit is set (=1) when the non-zero value is selected for Enable Additional 1-31 MAC Address Registers option.	

Table 30-156: EMAC_HW_FTR_PSNT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	RXCOESEL	Receive Checksum Offload Enabled. The EMAC_HW_FTR_PSNT0.RXCOESEL bit is set (=1) when the Enable Receive TCP/IP Checksum Check option is selected.	
		0	Receive Checksum Offload Enable Option is Not Selected
		1	Receive Checksum Offload Enable Option is Selected
14 (R/NW)	TXCOESEL	Transmit Checksum Offload Enabled. The EMAC_HW_FTR_PSNT0.TXCOESEL bit is set (=1) when the Enable Transmit TCP/IP Checksum Insertion option is selected.	
		0	Transmit Checksum Offload Enable Option is Not Selected
		1	Transmit Checksum Offload Enable Option is Selected
13 (R/NW)	EEESEL	Energy Efficient Ethernet Enabled. The EMAC_HW_FTR_PSNT0.EEESEL bit is set (=1) when the Enable Energy Efficient Ethernet (EEE) option is selected.	
		0	Energy Efficient Ethernet Enable Option is Not Selected
		1	Energy Efficient Ethernet Enable Option is Selected
12 (R/NW)	TSSEL	IEEE 1588-2008 Time Stamp Enabled. The EMAC_HW_FTR_PSNT0.TSSEL bit is set (=1) when the Enable IEEE 1588 Time Stamp Support Option is selected.	
		0	Option is Not Selected
		1	Option is Selected
9 (R/NW)	ARPOFFSEL	ARP Offload Enabled. The EMAC_HW_FTR_PSNT0.ARPOFFSEL bit is set (=1) when the Enable IPv4 ARP Offload option is selected.	
		0	ARP Offload Enable Option is Not Selected
		1	ARP Offload Enable Option is Selected
8 (R/NW)	MMCSEL	RMON Module Enable. The EMAC_HW_FTR_PSNT0.MMCSEL bit is set (=1) when the Enable MAC Management Counters (MMC) option is selected.	
		0	RMON Module Enable Option is Not Selected
		1	RMON Module Enable Option is Selected

Table 30-156: EMAC_HW_FTR_PSNT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	SMASEL	SMA (MDIO) Interface. The `EMAC_HW_FTR_PSNT0.SMASEL` bit is set (=1) when the Enable Station Management (MDIO Interface) option is selected.	
		0	SMA (MDIO) Interface Not Selected
		1	SMA (MDIO) Interface Selected
4 (R/NW)	VLHASH	VLAN Hash Filter Selected. The `EMAC_HW_FTR_PSNT0.VLHASH` bit is set (=1) when the Enable VLAN Hash Table Based Filtering option is selected.	
		0	VLAN Hash Filter Not Selected
		1	VLAN Hash Filter Selected
3 (R/NW)	PCSSEL	PCS Registers (TBI, SGMII, or RTBI PHY Interface). The `EMAC_HW_FTR_PSNT0.PCSSEL` bit is set (=1) when the TBI, SGMII, or RTBI PHY interface option is selected.	
		0	No PCS Registers (TBI, SGMII, or RTBI PHY Interface)
		1	PCS Registers (TBI, SGMII, or RTBI PHY Interface)
2 (R/NW)	HDSEL	Half-duplex Support. The `EMAC_HW_FTR_PSNT0.HDSEL` bit is set (=1) when the half-duplex mode is selected,	
		0	No Half-duplex Support
		1	Half-duplex Support
1 (R/NW)	GMIISEL	1000 Mbps Support. The `EMAC_HW_FTR_PSNT0.GMIISEL` bit is set (=1) when 1000 Mbps is selected as the mode of operation.	
		0	No 1000 Mbps Support
		1	1000 Mbps Support
0 (R/NW)	MIISEL	10 or 100 Mbps Support. The `EMAC_HW_FTR_PSNT0.MIISEL` bit is set (=1) when 10/100 Mbps is selected as the mode of operation.	
		0	No 10 or 100 Mbps Support
		1	10 or 100 Mbps Support

Hardware Feature Presence Register 1

The EMAC_HW_FTR_PSNT1 register indicates the presence of second set of the optional features or functions of the EMAC module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

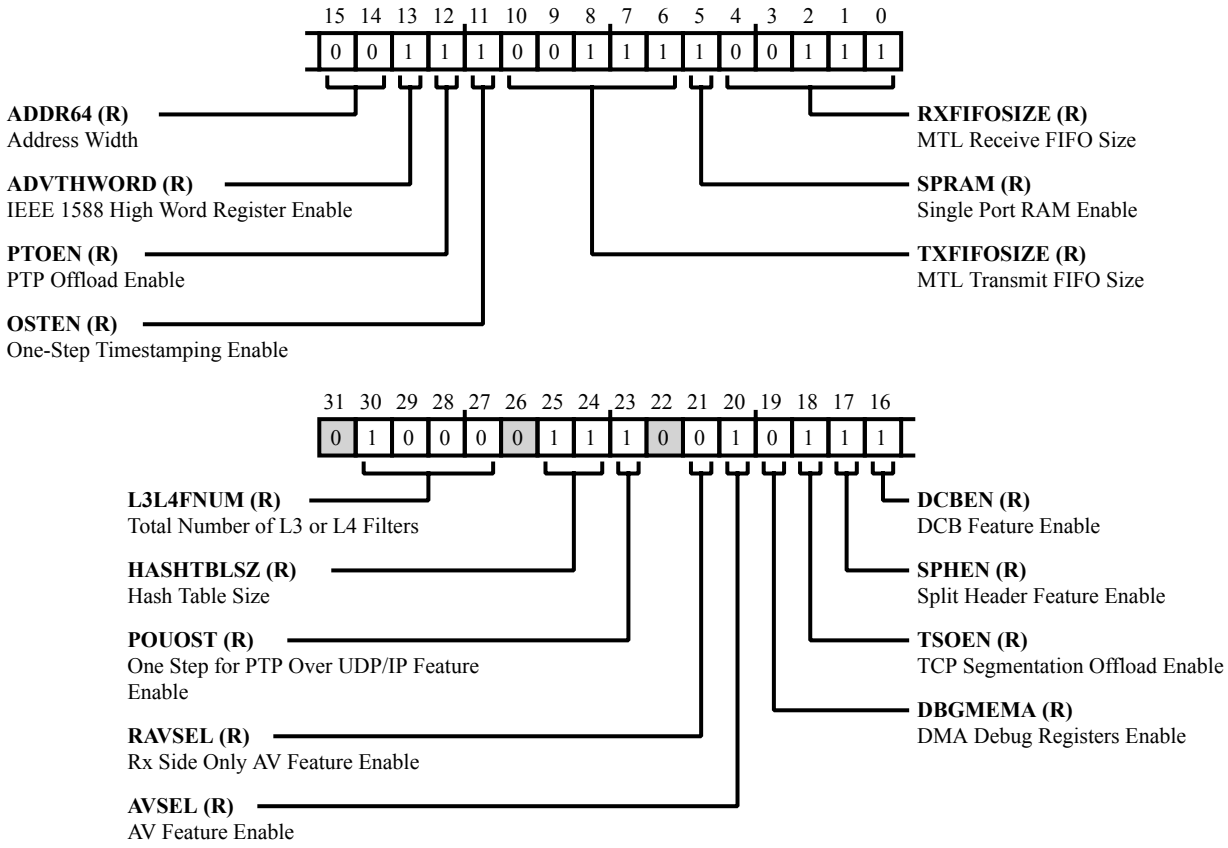


Figure 30-102: EMAC_HW_FTR_PSNT1 Register Diagram

Table 30-157: EMAC_HW_FTR_PSNT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:27 (R/NW)	L3L4FNUM	Total Number of L3 or L4 Filters. The EMAC_HW_FTR_PSNT1.L3L4FNUM field indicates the total number of L3 or L4 filters.	
		0	No L3 or L4 Filter
		1	1 L3 or L4 Filter
		2	2 L3 or L4 Filters
		3	3 L3 or L4 Filters

Table 30-157: EMAC_HW_FTR_PSNT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4	4 L3 or L4 Filters
		5	5 L3 or L4 Filters
		6	6 L3 or L4 Filters
		7	7 L3 or L4 Filters
		8	8 L3 or L4 Filters
25:24 (R/NW)	HASHTBLSZ	Hash Table Size. The `EMAC_HW_FTR_PSNT1.HASHTBLSZ` field indicates the size of the hash table.	
		0	No Hash Table
		1	64
		2	128
		3	256
23 (R/NW)	POUOST	One Step for PTP Over UDP/IP Feature Enable. The `EMAC_HW_FTR_PSNT1.POUOST` bit is set (=1) when the Enable One Step Time Stamp for PTP over UDP/IP feature is selected.	
		0	One Step for PTP Over UDP/IP Feature is Not Selected
		1	One Step for PTP Over UDP/IP Feature is Selected
21 (R/NW)	RAVSEL	Rx Side Only AV Feature Enable. The `EMAC_HW_FTR_PSNT1.RAVSEL` bit is set (=1) when the Enable Audio Video Bridging option on Rx Side Only is selected.	
		0	Rx Side Only AV Feature is Not Selected
		1	Rx Side Only AV Feature is Selected
20 (R/NW)	AVSEL	AV Feature Enable. The `EMAC_HW_FTR_PSNT1.AVSEL` bit is set (=1) when the Enable Audio Video Bridging option is selected.	
		0	AV Feature is Not Selected
		1	AV Feature is Selected
19 (R/NW)	DBGMEMA	DMA Debug Registers Enable. The `EMAC_HW_FTR_PSNT1.DBGMEMA` bit is set (=1) when the Debug Mode Enable option is selected	
		0	DMA Debug Registers Option is Not Selected
		1	DMA Debug Registers Option is Selected

Table 30-157: EMAC_HW_FTR_PSNT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/NW)	TSOEN	TCP Segmentation Offload Enable. The `EMAC_HW_FTR_PSNT1.TSOEN` bit is set (=1) when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected	
		0	TCP Segmentation Offload Feature is Not Selected
		1	TCP Segmentation Offload Feature is Selected
17 (R/NW)	SPHEN	Split Header Feature Enable. The `EMAC_HW_FTR_PSNT1.SPHEN` bit is set (=1) when the Enable Split Header Structure option is selected	
		0	Split Header Feature is Not Selected
		1	Split Header Feature is Selected
16 (R/NW)	DCBEN	DCB Feature Enable. The `EMAC_HW_FTR_PSNT1.DCBEN` bit is set (=1) when the Enable Data Center Bridging option is selected	
		0	DCB Feature is Not Selected
		1	DCB Feature is Selected
15:14 (R/NW)	ADDR64	Address Width. The `EMAC_HW_FTR_PSNT1.ADDR64` field indicates the configured address width:	
		0	32
		1	40
		2	48
		3	Reserved
13 (R/NW)	ADVTHWORD	IEEE 1588 High Word Register Enable. The `EMAC_HW_FTR_PSNT1.ADVTHWORD` bit is set (=1) when the Add IEEE 1588 Higher Word Register option is selected	
		0	IEEE 1588 High Word Register Option is Not Selected
		1	IEEE 1588 High Word Register Option is Selected
12 (R/NW)	PTOEN	PTP Offload Enable. The `EMAC_HW_FTR_PSNT1.PTOEN` bit is set (=1) when the Enable PTP time stamp Offload Feature is selected.	
		0	PTP Offload Feature is Not Selected
		1	PTP Offload Feature is Selected

Table 30-157: EMAC_HW_FTR_PSNT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	OSTEN	One-Step Timestamping Enable. The `EMAC_HW_FTR_PSNT1.OSTEN` bit is set (=1) when the Enable One-Step time stamp Feature is selected.	
		0	Feature is Not Selected
		1	Feature is Selected
10:6 (R/NW)	TXFIFOSIZE	MTL Transmit FIFO Size. The `EMAC_HW_FTR_PSNT1.TXFIFOSIZE` field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:	
		0	128 Bytes
		1	256 Bytes
		2	512 Bytes
		3	1024 Bytes
		4	2048 Bytes
		5	4096 Bytes
		6	8192 Bytes
		7	16384 Bytes
		8	32 KB
		9	64 KB
		10	128 KB
		11	Reserved
5 (R/NW)	SPRAM	Single Port RAM Enable. The `EMAC_HW_FTR_PSNT1.SPRAM` bit is set (=1) when the Use single port RAM Feature is selected.	
		0	Single Port RAM Feature is Not Selected
		1	Single Port RAM Feature is Selected
4:0 (R/NW)	RXFIFOSIZE	MTL Receive FIFO Size. The `EMAC_HW_FTR_PSNT1.RXFIFOSIZE` field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7:	
		0	128 Bytes
		1	256 Bytes
		2	512 Bytes

Table 30-157: EMAC_HW_FTR_PSNT1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	1024 Bytes
		4	2048 Bytes
		5	4096 Bytes
		6	8192 Bytes
		7	16384 Bytes
		8	32 KB
		9	64 KB
		10	128 KB
		11	256 KB
		12	Reserved

Hardware Feature Presence Register 2

The EMAC_HW_FTR_PSNT2 register indicates the presence of third set of the optional features or functions of the EMAC module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

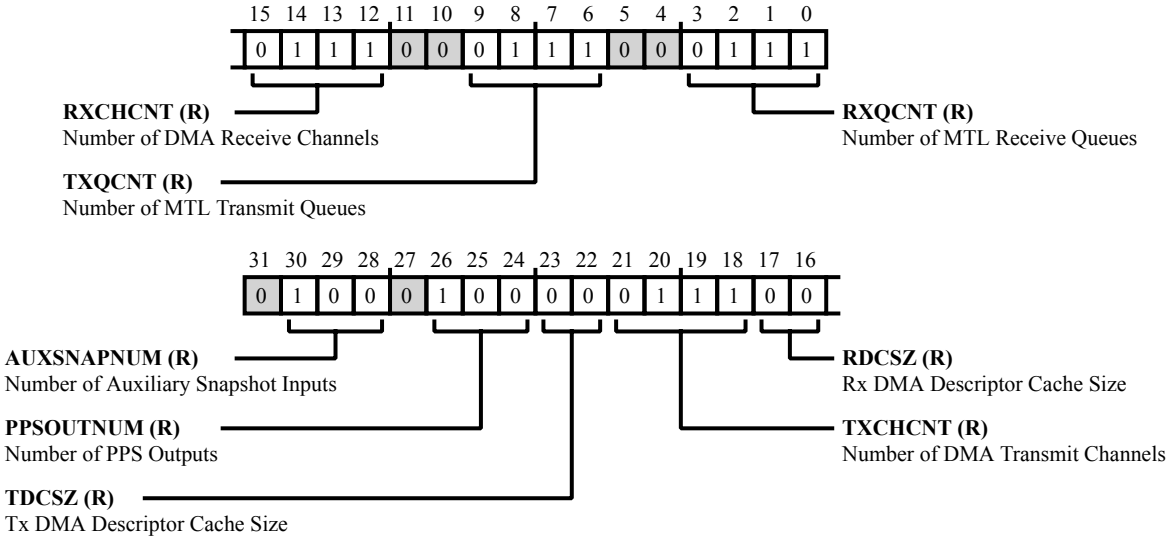


Figure 30-103: EMAC_HW_FTR_PSNT2 Register Diagram

Table 30-158: EMAC_HW_FTR_PSNT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30:28 (R/NW)	AUXSNAPNUM	Number of Auxiliary Snapshot Inputs. The EMAC_HW_FTR_PSNT2.AUXSNAPNUM field indicates the number of auxiliary snapshot inputs.	
		0	No Auxiliary Input
		1	1 Auxiliary Input
		2	2 Auxiliary Input
		3	3 Auxiliary Input
		4	4 Auxiliary Input
		5	Reserved
26:24 (R/NW)	PPSOUTNUM	Number of PPS Outputs. The EMAC_HW_FTR_PSNT2.PPSOUTNUM field indicates the number of PPS outputs.	
		0	No PPS Output
		1	1 PPS Output

Table 30-158: EMAC_HW_FTR_PSNT2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	2 PPS Output
		3	3 PPS Output
		4	4 PPS Output
		5	Reserved
23:22 (R/NW)	TDCSZ	Tx DMA Descriptor Cache Size. The `EMAC_HW_FTR_PSNT2.TDCSZ` field indicates the size of the Tx DMA descriptor cache in terms of 16 byte descriptors. 00 - Cache not configured 01 - Four 16 bytes descriptor 10 - Eight 16 bytes descriptor 11 - Sixteen 16 bytes descriptor	
21:18 (R/NW)	TXCHCNT	Number of DMA Transmit Channels. The `EMAC_HW_FTR_PSNT2.TXCHCNT` field indicates the number of DMA transmit channels.	
		0	1 MTL Tx Channel
		1	2 MTL Tx Channels
		2	3 MTL Tx Channels
		3	4 MTL Tx Channels
		4	5 MTL Tx Channels
		5	6 MTL Tx Channels
		6	7 MTL Tx Channels
		7	8 MTL Tx Channels
17:16 (R/NW)	RDCSZ	Rx DMA Descriptor Cache Size. The `EMAC_HW_FTR_PSNT2.RDCSZ` field indicates the size of the Rx DMA descriptor cache in terms of 16 byte descriptors. 00 - Cache not configured 01 - Four 16 bytes descriptor 10 - Eight 16 bytes descriptor 11 - Sixteen 16 bytes descriptor	
15:12 (R/NW)	RXCHCNT	Number of DMA Receive Channels. The `EMAC_HW_FTR_PSNT2.RXCHCNT` field indicates the number of DMA receive channels.	
		0	1 MTL Rx Channel

Table 30-158: EMAC_HW_FTR_PSNT2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		1	2 MTL Rx Channels
		2	3 MTL Rx Channels
		3	4 MTL Rx Channels
		4	5 MTL Rx Channels
		5	6 MTL Rx Channels
		6	7 MTL Rx Channels
		7	8 MTL Rx Channels
9:6 (R/NW)	TXQCNT	Number of MTL Transmit Queues. The `EMAC_HW_FTR_PSNT2.TXQCNT` field indicates the number of MTL transmit queues.	
		0	1 MTL Tx Queue
		1	2 MTL Tx Queues
		2	3 MTL Tx Queues
		3	4 MTL Tx Queues
		4	5 MTL Tx Queues
		5	6 MTL Tx Queues
		6	7 MTL Tx Queues
		7	8 MTL Tx Queues
3:0 (R/NW)	RXQCNT	Number of MTL Receive Queues. The `EMAC_HW_FTR_PSNT2.RXQCNT` field indicates the number of MTL receive queues.	
		0	1 MTL Rx Queue
		1	2 MTL Rx Queues
		2	3 MTL Rx Queues
		3	4 MTL Rx Queues
		4	5 MTL Rx Queues
		5	6 MTL Rx Queues
		6	7 MTL Rx Queues
		7	8 MTL Rx Queues

Hardware Feature Presence Register 3

The EMAC_HW_FTR_PSNT3 register indicates the presence of fourth set the optional features or functions of the EMAC module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

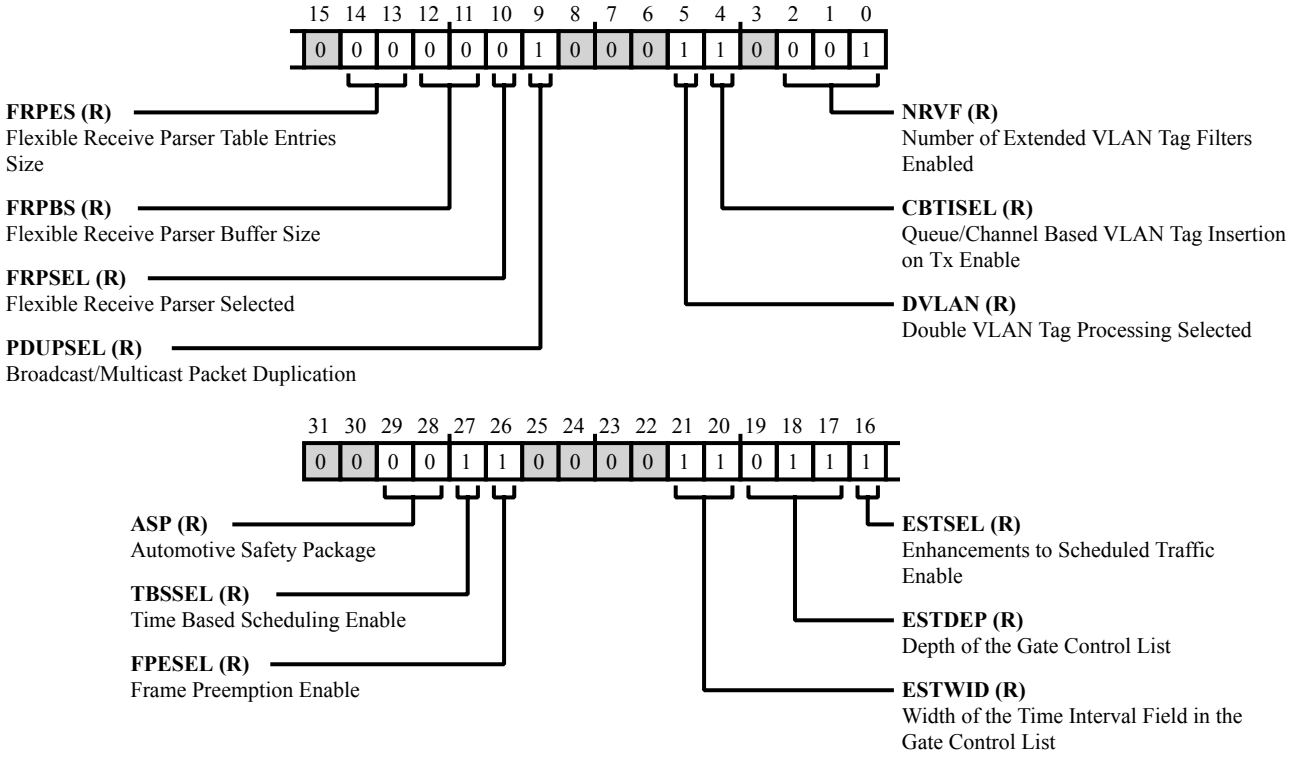


Figure 30-104: EMAC_HW_FTR_PSNT3 Register Diagram

Table 30-159: EMAC_HW_FTR_PSNT3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:28 (R/NW)	ASP	Automotive Safety Package. The EMAC_HW_FTR_PSNT3.ASP field indicates the different safety features.	
		0	No Safety Features Selected
		1	Only "ECC Protection for External Memory" Feature is Selected
		2	All the Automotive Safety Features are Selected Without the "Parity Port Enable for External Interface" Feature
		3	All the Automotive Safety Features are Selected with the "Parity Port Enable for External Interface" Feature

Table 30-159: EMAC_HW_FTR_PSNT3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	TBSSEL	Time Based Scheduling Enable. The `EMAC_HW_FTR_PSNT3.TBSSEL` bit is set (=1) when the Time Based Scheduling feature is selected.	
		0	Time Based Scheduling Enable Feature is Not Selected
		1	Time Based Scheduling Enable Feature is Selected
26 (R/NW)	FPESEL	Frame Preemption Enable. The `EMAC_HW_FTR_PSNT3.FPESEL` bit is set (=1) when the enable frame preemption feature is selected.	
		0	Frame Preemption Enable Feature is Not Selected
		1	Frame Preemption Enable Feature is Selected
21:20 (R/NW)	ESTWID	Width of the Time Interval Field in the Gate Control List. The `EMAC_HW_FTR_PSNT3.ESTWID` field indicates the width of the configured time interval field.	
		0	Width Not Configured
		1	16
		2	20
		3	24
19:17 (R/NW)	ESTDEP	Depth of the Gate Control List. The `EMAC_HW_FTR_PSNT3.ESTDEP` field indicates the depth of the GCL expressed as Log2(DWC_EQOS_EST_DEP)-5	
		0	No Depth Configured
		1	64
		2	128
		3	256
		4	512
		5	1024
		6	Reserved

Table 30-159: EMAC_HW_FTR_PSNT3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/NW)	ESTSEL	Enhancements to Scheduled Traffic Enable. The `EMAC_HW_FTR_PSNT3.ESTSEL` bit is set (=1) when the Enable Enhancements to Scheduling Traffic feature is selected.	
		0	Enable Enhancements to Scheduling Traffic Feature is Not Selected
		1	Enable Enhancements to Scheduling Traffic Feature is Selected
14:13 (R/NW)	FRPES	Flexible Receive Parser Table Entries Size. The `EMAC_HW_FTR_PSNT3.FRPES` field indicates the maximum number of parser entries supported the flexible receive parser.	
		0	64 Entries
		1	128 Entries
		2	256 Entries
		3	Reserved
12:11 (R/NW)	FRPBS	Flexible Receive Parser Buffer Size. The `EMAC_HW_FTR_PSNT3.FRPBS` field indicates the supported max number of bytes of the packet data to be parsed by the flexible receive parser.	
		0	64 Bytes
		1	128 Bytes
		2	256 Bytes
		3	Reserved
10 (R/NW)	FRPSEL	Flexible Receive Parser Selected. The `EMAC_HW_FTR_PSNT3.FRPSEL` bit is set (=1) when the enable flexible programmable receive parser option is selected.	
		0	Flexible Receive Parser Feature is Not Selected
		1	Flexible Receive Parser Feature is Selected
9 (R/NW)	PDUPSEL	Broadcast/Multicast Packet Duplication. The `EMAC_HW_FTR_PSNT3.PDUPSEL` bit is set (=1) when the broadcast/multicast packet duplication feature is selected.	
		0	Broadcast/Multicast Packet Duplication Feature is Not Selected
		1	Broadcast/Multicast Packet Duplication Feature is Selected

Table 30-159: EMAC_HW_FTR_PSNT3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	DVLAN	Double VLAN Tag Processing Selected. The `EMAC_HW_FTR_PSNT3.DVLAN` bit is set (=1) when the enable double VLAN processing feature is selected.	
		0	Double VLAN Option is Not Selected
		1	Double VLAN Option is Selected
4 (R/NW)	CBTISEL	Queue/Channel Based VLAN Tag Insertion on Tx Enable. The `EMAC_HW_FTR_PSNT3.CBTISEL` bit is set (=1) when the enable queue/channel-based VLAN tag insertion on Tx feature is selected.	
		0	Enable Queue/Channel Based VLAN Tag Insertion on Tx Feature is Not Selected
		1	Enable Queue/Channel Based VLAN Tag Insertion on Tx Feature is Selected
2:0 (R/NW)	NRVF	Number of Extended VLAN Tag Filters Enabled. The `EMAC_HW_FTR_PSNT3.NRVF` field indicates the number of extended VLAN tag filters selected.	
		0	No Extended Rx VLAN Filters
		1	4 Extended Rx VLAN Filters
		2	8 Extended Rx VLAN Filters
		3	16 Extended Rx VLAN Filters
		4	24 Extended Rx VLAN Filters
		5	32 Extended Rx VLAN Filters
		6	Reserved

Hash Table Register 0

The EMAC_HASHTBL_REG0 register contains the first 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values.

The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

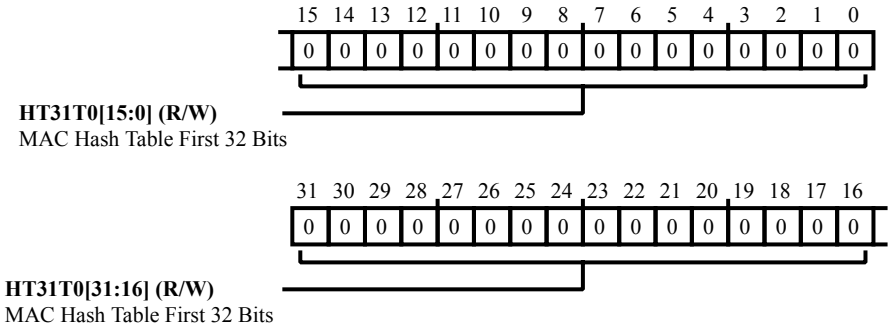


Figure 30-105: EMAC_HASHTBL_REG0 Register Diagram

Table 30-160: EMAC_HASHTBL_REG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT31T0	MAC Hash Table First 32 Bits. The EMAC_HASHTBL_REG0.HT31T0 field contains the first 32 bits [31:0] of the hash table.

Hash Table Register 1

The EMAC_HASHTBL_REG1 register contains the second 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values. The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

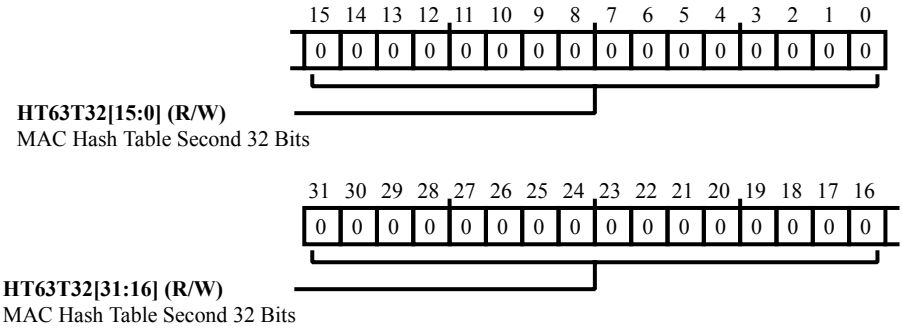


Figure 30-106: EMAC_HASHTBL_REG1 Register Diagram

Table 30-161: EMAC_HASHTBL_REG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT63T32	MAC Hash Table Second 32 Bits. The EMAC_HASHTBL_REG1.HT63T32 field contains the second 32 bits [63:32] of the hash table.

Hash Table Register 2

The EMAC_HASHTBL_REG2 register contains the third 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values. The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

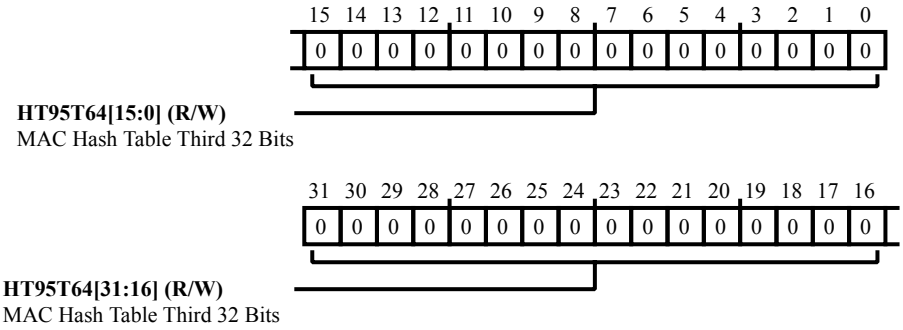


Figure 30-107: EMAC_HASHTBL_REG2 Register Diagram

Table 30-162: EMAC_HASHTBL_REG2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT95T64	MAC Hash Table Third 32 Bits. The EMAC_HASHTBL_REG2.HT95T64 field contains the third 32 bits [95:64] of the hash table.

Hash Table Register 3

The EMAC_HASHTBL_REG3 register contains the fourth 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values. The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

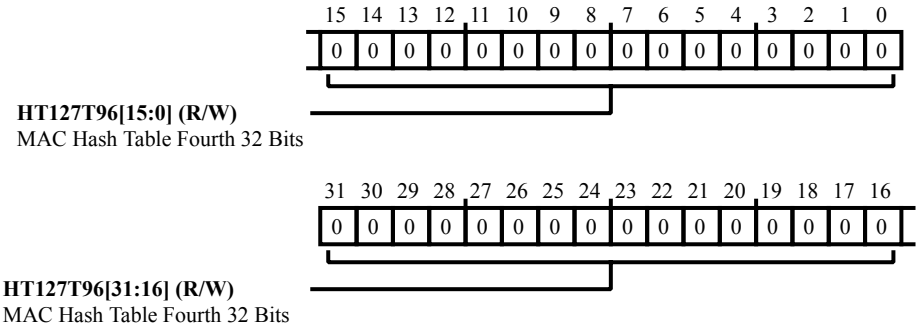


Figure 30-108: EMAC_HASHTBL_REG3 Register Diagram

Table 30-163: EMAC_HASHTBL_REG3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT127T96	MAC Hash Table Fourth 32 Bits. The EMAC_HASHTBL_REG3.HT127T96 field contains the fourth 32 bits [127:96] of the hash table.

Hash Table Register 4

The EMAC_HASHTBL_REG4 register contains the fifth 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values.

The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

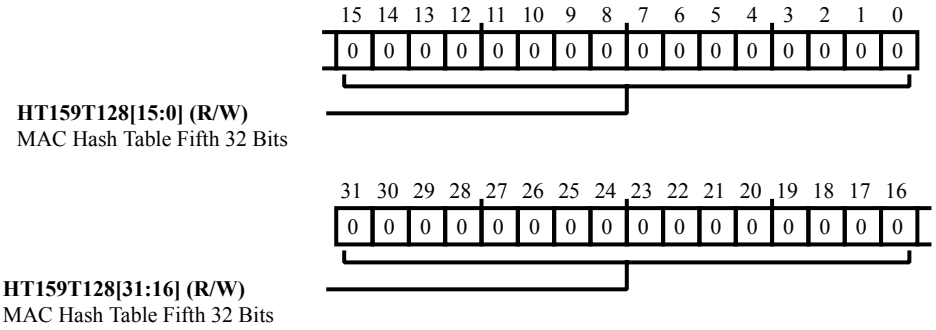


Figure 30-109: EMAC_HASHTBL_REG4 Register Diagram

Table 30-164: EMAC_HASHTBL_REG4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT159T128	MAC Hash Table Fifth 32 Bits. The EMAC_HASHTBL_REG4.HT159T128 field contains the fifth 32 bits [159:128] of the hash table.

Hash Table Register 5

The EMAC_HASHTBL_REG5 register contains the sixth 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values.

The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

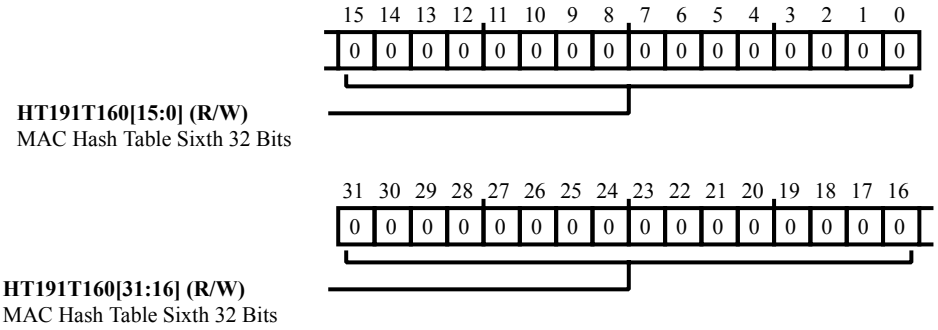


Figure 30-110: EMAC_HASHTBL_REG5 Register Diagram

Table 30-165: EMAC_HASHTBL_REG5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT191T160	MAC Hash Table Sixth 32 Bits. The EMAC_HASHTBL_REG5.HT191T160 field contains the sixth 32 bits [191:160] of the hash table.

Hash Table Register 6

The EMAC_HASHTBL_REG6 register contains the seventh 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values.

The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

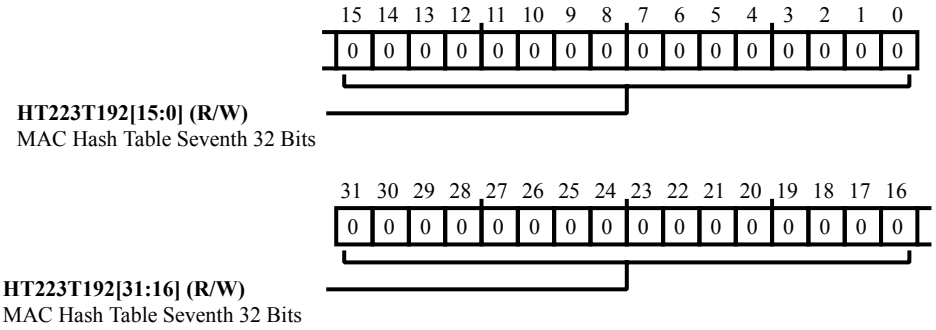


Figure 30-111: EMAC_HASHTBL_REG6 Register Diagram

Table 30-166: EMAC_HASHTBL_REG6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT223T192	MAC Hash Table Seventh 32 Bits. The EMAC_HASHTBL_REG6.HT223T192 field contains the seventh 32 bits [223:192] of the hash table.

Hash Table Register 7

The EMAC_HASHTBL_REG7 register contains the eighth 32 bits of the hash table.

The hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of 64-bit hash (eight in 256-bit hash) bits of the CRC register are used to index the content of the hash table. The most significant bits determines the register to be used (Hash Table Register n), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects bit 0 of the Hash Table Register 1, a value of 8b'10111111 (in 256-bit hash) selects bit 31 of the Hash Table Register 5.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper 6 (or 7 or 8) bits from the value obtained in step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If EMAC_MACPKT_FILT.PM = 1, all multicast packets are accepted regardless of the multicast hash values. The hash table register is configured to be double-synchronized to the (G)MII clock domain; the synchronization is triggered only when bits[31:24] (in little-endian mode) or bits[7:0] (in big-endian mode) of the Hash Table Register n registers are written.

Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.

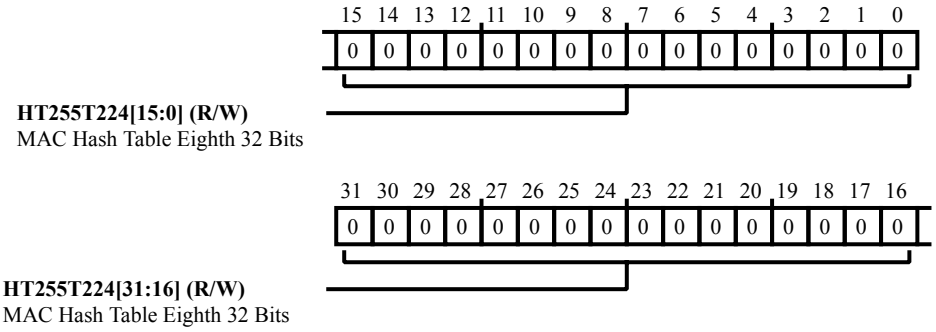


Figure 30-112: EMAC_HASHTBL_REG7 Register Diagram

Table 30-167: EMAC_HASHTBL_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	HT255T224	MAC Hash Table Eighth 32 Bits. The EMAC_HASHTBL_REG7.HT255T224 field contains the eighth 32 bits [255:224] of the hash table.

Indirect Access Control Register

The EMAC_INDR_ACC_CTL register provides the indirect access control and status for MAC_<MSEl>_<AOFF> registers.

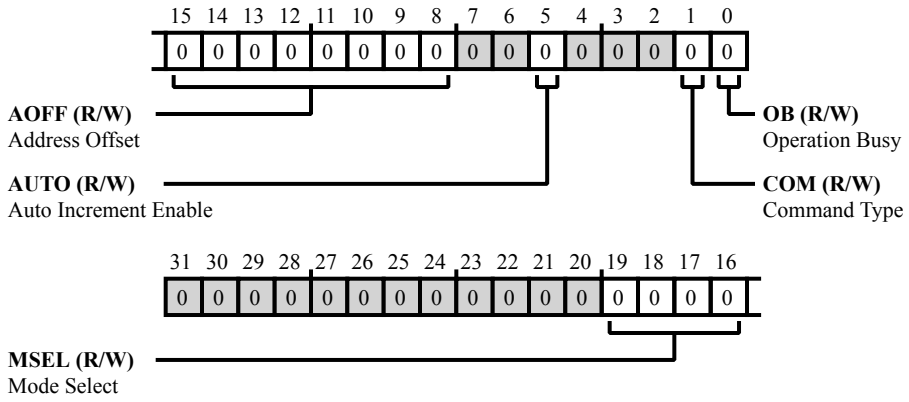


Figure 30-113: EMAC_INDR_ACC_CTL Register Diagram

Table 30-168: EMAC_INDR_ACC_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:16 (R/W)	MSEL	Mode Select. The EMAC_INDR_ACC_CTL.MSEL field is used in indirect access of MAC_<MSEL>_<AOFF>. The EMAC_INDR_ACC_CTL.MSEL field must be set along with initiation of read/write to MAC_<MSEL>_<AOFF> and should not be changed until the OB is cleared. Values: - 0000:Typ_RXQ_ (Type based RXQ mapping) - 0001-1111: Reserved
15:8 (R/W)	AOFF	Address Offset. The EMAC_INDR_ACC_CTL.AOFF field is used in indirect access of MAC_<MSEL>_<AOFF>. The EMAC_INDR_ACC_CTL.AOFF field must be configured along with initiation of read/write to MAC_<MSEL>_<AOFF> and should not be changed until the OB is cleared. Values: - 00000000: IndReg0(Indirect register 0) - 00000001: IndReg1(Indirect register 1) - ... - 00000111: IndReg7(Indirect register 7) - 00001000-11111111: Reserved

Table 30-168: EMAC_INDR_ACC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	AUTO	Auto Increment Enable. The EMAC_INDR_ACC_CTL.AUTO bit enables the address offset auto increment feature. When set (=1), AOFF is incremented by 1. Software should ensure not to cause a wrap condition. Byte wise read/write is not supported when auto increment is enabled. When the EMAC_INDR_ACC_CTL.AUTO bit is cleared (=0), AOFF is not incremented automatically. Software should program the correct Address Offset for each access.	
1 (R/W)	COM	Command Type. The EMAC_INDR_ACC_CTL.COM bit indicates the register access type. - 1: Indicates a read operation. - 0: Indicates a write operation.	
		0	Write Operation
		1	Read Operation
0 (R/W)	OB	Operation Busy. The EMAC_INDR_ACC_CTL.OB bit is set along with a read or write command for initiating the indirect access to MAC_<MSEL>_<AOFF> register. The EMAC_INDR_ACC_CTL.OB bit is cleared (=0) when the read or write command to MAC_<MSEL>_<AOFF> register is complete. The next indirect register access can be initiated only after EMAC_INDR_ACC_CTL.OB = 0. During a write operation, the bit is cleared (=0) only after the data has been written into MAC_<MSEL>_<AOFF> register. During a read operation, the data should be read from the EMAC_INDR_ACC_DAT register only after EMAC_INDR_ACC_CTL.OB = 0.	

Indirect Access Data Register

The EMAC_INDR_ACC_DAT register holds the read/write data for indirect access of MAC_<MSEL_<AOFF> registers.

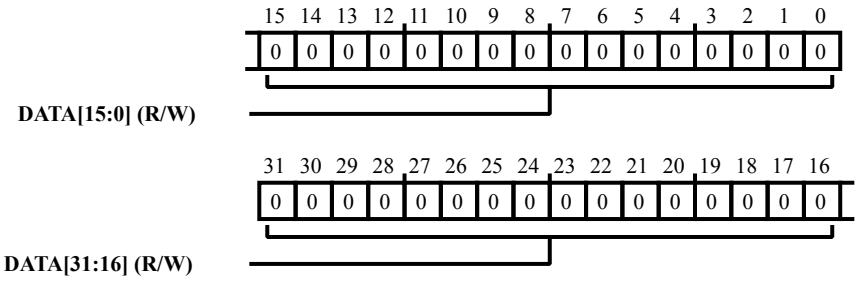


Figure 30-114: EMAC_INDR_ACC_DAT Register Diagram

Table 30-169: EMAC_INDR_ACC_DAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	DATA	. The EMAC_INDR_ACC_DAT.DATA field contains data to read/write for indirect address access associated with EMAC_INDR_ACC_CTL register.

Inner VLAN Tag Inclusion or Replacement Register

The EMAC_INVLAN_INCL register contains the inner VLAN tag to be inserted or replaced in the transmit packet. It also contains the inner VLAN tag insertion controls.

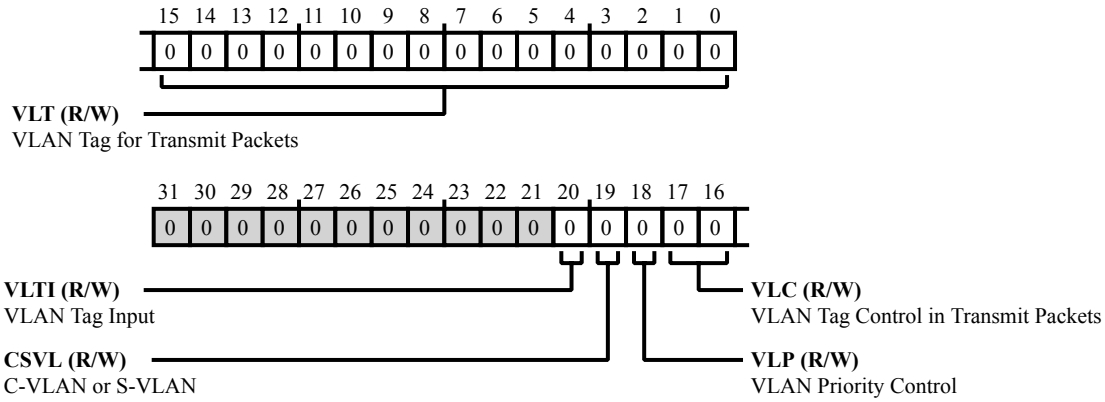


Figure 30-115: EMAC_INVLAN_INCL Register Diagram

Table 30-170: EMAC_INVLAN_INCL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	VLTI	VLAN Tag Input. When the EMAC_INVLAN_INCL.VLTI bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.	
		0	VLAN Tag Input is Disabled
		1	VLAN Tag Input is Enabled
19 (R/W)	CSVL	C-VLAN or S-VLAN. When the EMAC_INVLAN_INCL.CSVL bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 17th and 18th bytes of transmitted packets. When EMAC_INVLAN_INCL.CSVL = 0, C-VLAN type (0x8100) is inserted or replaced in the 17th and 18th bytes of transmitted packets.	
		0	C-VLAN Type (0x8100) is Inserted
		1	S-VLAN Type (0x88A8) is Inserted
18 (R/W)	VLP	VLAN Priority Control. When the EMAC_INVLAN_INCL.VLP bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When EMAC_INVLAN_INCL.VLP = 0, the TDES2.VTIR descriptor control is used and the VLC field is ignored.	
		0	VLAN Priority Control is Disabled
		1	VLAN Priority Control is Enabled

Table 30-170: EMAC_INVLAN_INCL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	VLC	VLAN Tag Control in Transmit Packets. The EMAC_INVLAN_INCL.VLC field indicates the VLAN tag control in transmit packets. 2'b00: No VLAN tag deletion, insertion, or replacement 2'b01: VLAN tag deletion. The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags. 2'b10: VLAN tag insertion. The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 2'b11: VLAN tag replacement. The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8). Note: Changes to this field take effect only on the start of a packet. If this register field is written to when a packet is being transmitted, only the subsequent packet can use the updated value; the current packet does not use the updated value.	
		0	No VLAN Tag Deletion, Insertion, or Replacement
		1	VLAN Tag Deletion
		2	VLAN Tag Insertion
		3	VLAN Tag Replacement
15:0 (R/W)	VLT	VLAN Tag for Transmit Packets. The EMAC_INVLAN_INCL.VLT field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and bits[11:0] are the VID field in the VLAN tag. The following list describes the bits of this field: - Bits[15:13]: User Priority - Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) - Bits[11:0]: VLAN Identifier (VID) field of VLAN tag	

MAC Interrupt Enable Register

The EMAC_MAC_IEN register contains the masks for generating the interrupts.

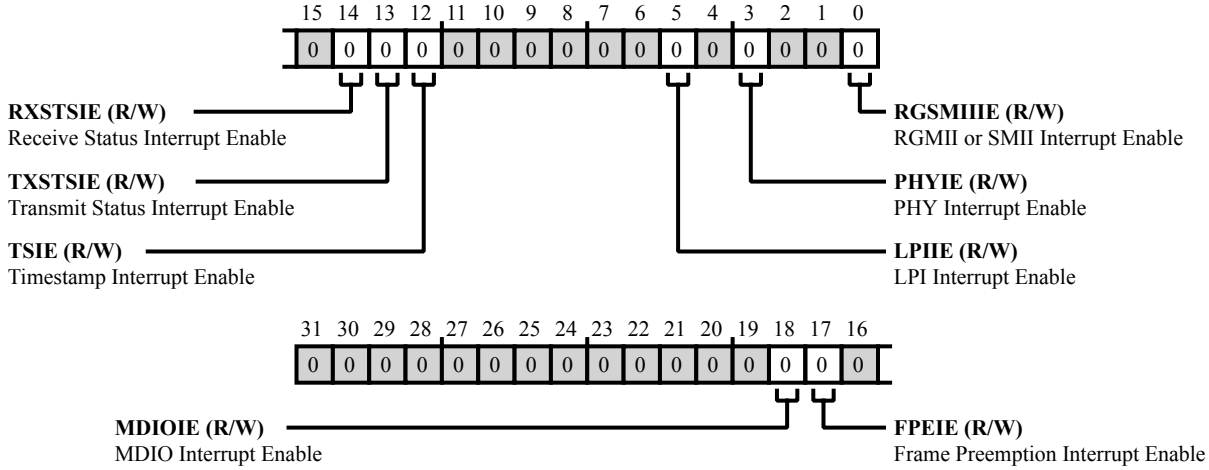


Figure 30-116: EMAC_MAC_IEN Register Diagram

Table 30-171: EMAC_MAC_IEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	MDIOIE	MDIO Interrupt Enable. When the EMAC_MAC_IEN.MDIOIE bit is set, it enables the assertion of the interrupt when EMAC_MAC_ISTAT.MDIOIS = 1.	
		0	MDIO Interrupt is Disabled
		1	MDIO Interrupt is Enabled
17 (R/W)	FPEIE	Frame Preemption Interrupt Enable. When the EMAC_MAC_IEN.FPEIE bit is set, it enables the assertion of the interrupt when EMAC_MAC_ISTAT.FPEIS = 1.	
		0	Frame Preemption Interrupt is Disabled
		1	Frame Preemption Interrupt is Enabled
14 (R/W)	RXSTSIE	Receive Status Interrupt Enable. When the EMAC_MAC_IEN.RXSTSIE bit is set, it enables the assertion of the interrupt signal because of the configuration of the EMAC_MAC_ISTAT.RXSTSIS bit.	
		0	Receive Status Interrupt is Disabled
		1	Receive Status Interrupt is Enabled

Table 30-171: EMAC_MAC_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	TXSTSIE	Transmit Status Interrupt Enable. When the `EMAC_MAC_IEN.TXSTSIE` bit is set, it enables the assertion of the interrupt signal because of the configuration of the EMAC_MAC_ISTAT.TXSTSIS bit.	
		0	Disable
		1	Enable
12 (R/W)	TSIE	Timestamp Interrupt Enable. When the `EMAC_MAC_IEN.TSIE` bit is set, it enables the assertion of the interrupt signal because of the configuration of the EMAC_MAC_ISTAT.TSIS bit.	
		0	Disable
		1	Enable
5 (R/W)	LPIIE	LPI Interrupt Enable. When the `EMAC_MAC_IEN.LPIIE` bit is set, it enables the assertion of the interrupt signal because of EMAC_MAC_ISTAT.LPIIS.	
		0	LPI Interrupt is Disabled
		1	LPI Interrupt is Enabled
3 (R/W)	PHYIE	PHY Interrupt Enable. When the `EMAC_MAC_IEN.PHYIE` bit is set, it enables the assertion of the interrupt signal because of the EMAC_MAC_ISTAT.PHYIS bit.	
		0	PHY Interrupt is Disabled
		1	PHY Interrupt is Enabled
0 (R/W)	RGSMIIIE	RGMII or SMII Interrupt Enable. When the `EMAC_MAC_IEN.RGSMIIIE` bit is set, it enables the assertion of the interrupt signal because of the configuration of the EMAC_MAC_ISTAT.RGSMIIIS bit.	
		0	RGMII or SMII Interrupt is Disabled
		1	RGMII or SMII Interrupt is Enabled

MAC Interrupt Status Register

The EMAC_MAC_ISTAT register contains the status of interrupts.

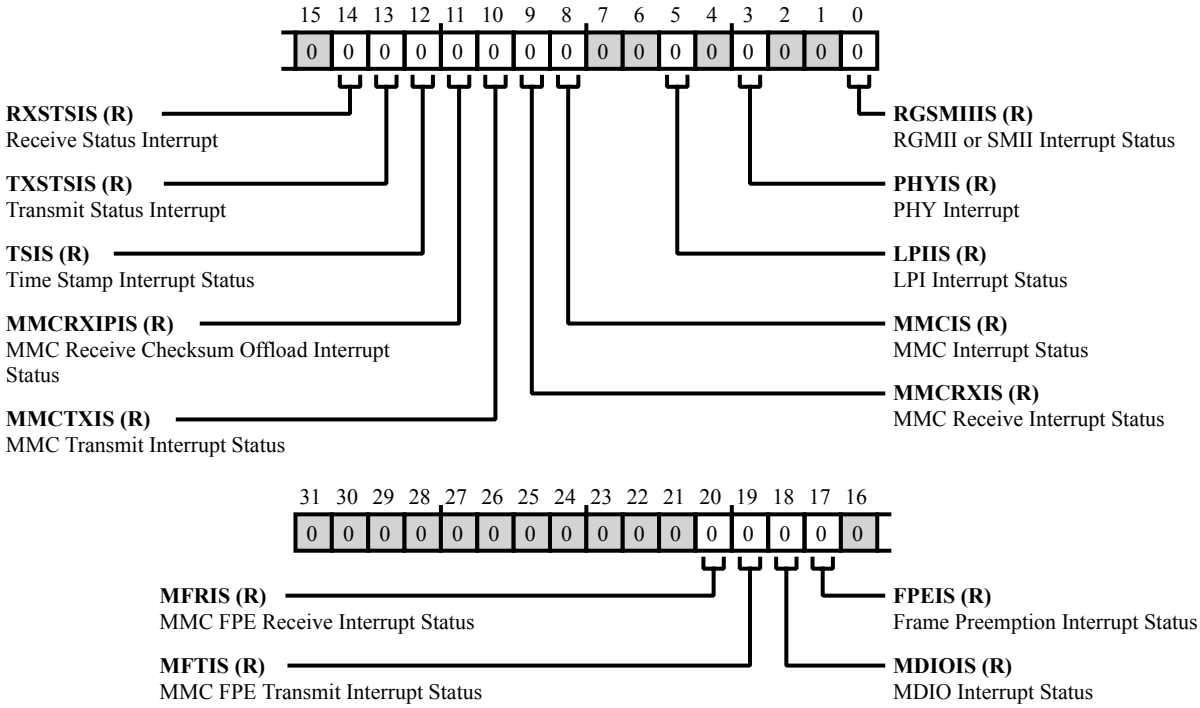


Figure 30-117: EMAC_MAC_ISTAT Register Diagram

Table 30-172: EMAC_MAC_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/NW)	MFRIS	MMC FPE Receive Interrupt Status. The EMAC_MAC_ISTAT.MFRIS bit is set (=1) when an interrupt is generated in the EMAC_MMC_FPE_RXINT register. The EMAC_MAC_ISTAT.MFRIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MFRIS bit is valid only when MMC and FPE support are enabled.	
		0	MMC FPE Receive Interrupt Status Not Active
		1	MMC FPE Receive Interrupt Status Active

Table 30-172: EMAC_MAC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	MFTIS	MMC FPE Transmit Interrupt Status. The EMAC_MAC_ISTAT.MFTIS bit is set (=1) when an interrupt is generated in the EMAC_MMC_FPE_TXINT register. The EMAC_MAC_ISTAT.MFTIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MFTIS bit is valid only when MMC and FPE support are enabled.	
		0	MMC FPE Transmit Interrupt Status Not Active
		1	MMC FPE Transmit Interrupt Status Active
18 (R/NW)	MDIOIS	MDIO Interrupt Status. The EMAC_MAC_ISTAT.MDIOIS bit indicates an interrupt event after the completion of MDIO operation. To reset the EMAC_MAC_ISTAT.MDIOIS bit, the application has to read the EMAC_MAC_ISTAT.MDIOIS bit/write 1 to the EMAC_MAC_ISTAT.MDIOIS bit when EMAC_CSR_SW_CTL.RCWE = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	MDIO Interrupt Status Not Active
		1	MDIO Interrupt Status Active
17 (R/NW)	FPEIS	Frame Preemption Interrupt Status. The EMAC_MAC_ISTAT.FPEIS bit indicates an interrupt event during the operation of frame preemption (Bits[19:16] of EMAC_FPE_CTLSTAT register are set). To reset the EMAC_MAC_ISTAT.FPEIS bit, the application must clear the event in EMAC_FPE_CTLSTAT that caused the Interrupt.	
		0	Frame Preemption Interrupt Status Not Active
		1	Frame Preemption Interrupt Status Active
14 (R/NW)	RXSTSIS	Receive Status Interrupt. The EMAC_MAC_ISTAT.RXSTSIS bit indicates the status of received packets. The EMAC_MAC_ISTAT.RXSTSIS bit is set when the EMAC_RXTX_STAT.RWT = 1. The EMAC_MAC_ISTAT.RXSTSIS bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1) in the EMAC_RXTX_STAT register.	
		0	Receive Interrupt Status Not Active
		1	Receive Interrupt Status Active

Table 30-172: EMAC_MAC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/NW)	TXSTSIS	Transmit Status Interrupt. The EMAC_MAC_ISTAT.TXSTSIS bit indicates the status of transmitted packets. The EMAC_MAC_ISTAT.TXSTSIS bit is set when any of the following bits is set in the EMAC_RXTX_STAT register: Excessive Collision (EXCOL) Late Collision (LCOL) Excessive Deferral (EXDEF) Loss of Carrier (LCARR) No Carrier (NCARR) Jabber Timeout (TJT) The EMAC_MAC_ISTAT.TXSTSIS bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 (when EMAC_CSR_SW_CTL.RCWE = 1) in the EMAC_RXTX_STAT register).	
		0	Status Not Active
		1	Status Active
12 (R/NW)	TSIS	Time Stamp Interrupt Status. If the time stamp feature is enabled, the EMAC_MAC_ISTAT.TSIS bit is set when any of the following conditions are true: The system time value is equal to or exceeds the value specified in the Target Time High and Low registers. There is an overflow in the seconds register. The Target Time Error occurred, that is, programmed target time already elapsed. If the Auxiliary Snapshot feature is enabled, the EMAC_MAC_ISTAT.TSIS bit is set when the auxiliary snapshot trigger is asserted. In configurations other than EQOS_CORE, when drop transmit status is enabled in MTL, the EMAC_MAC_ISTAT.TSIS bit is set when the captured transmit time stamp is updated in the EMAC_TXTMSTAT_NSEC and EMAC_TXTMSTAT_SEC registers. When PTP offload feature is enabled, the EMAC_MAC_ISTAT.TSIS bit is set when the captured transmit time stamp is updated in the EMAC_TXTMSTAT_NSEC and EMAC_TXTMSTAT_SEC registers, for PTO generated Delay Request and Pdelay request packets. The EMAC_MAC_ISTAT.TSIS bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 (when EMAC_CSR_SW_CTL.RCWE = 1) in the EMAC_TM_STAT register.	
		0	Status Not Active
		1	Status Active

Table 30-172: EMAC_MAC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	MMCRXIPIS	MMC Receive Checksum Offload Interrupt Status. The EMAC_MAC_ISTAT.MMCRXIPIS bit is set (=1) when an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. The EMAC_MAC_ISTAT.MMCRXIPIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MMCRXIPIS bit is valid only when the Enable MAC Management Counters (MMC) and Enable Receive TCP/IP Checksum Check options are selected.	
		0	MMC Receive Checksum Offload Interrupt Status Not Active
		1	MMC Receive Checksum Offload Interrupt Status Active
10 (R/NW)	MMCTXIS	MMC Transmit Interrupt Status. The EMAC_MAC_ISTAT.MMCTXIS bit is set (=1) when an interrupt is generated in the EMAC_MMC_TXINT register. The EMAC_MAC_ISTAT.MMCTXIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MMCTXIS bit is valid only when the Enable MAC Management Counters (MMC) option is selected.	
		0	MMC Transmit Interrupt Status Not Active
		1	MMC Transmit Interrupt Status Active
9 (R/NW)	MMCRXIS	MMC Receive Interrupt Status. The EMAC_MAC_ISTAT.MMCRXIS bit is set (=1) when an interrupt is generated in the EMAC_MMC_RXINT register. The EMAC_MAC_ISTAT.MMCRXIS bit is cleared when all bits in this interrupt register are cleared. The EMAC_MAC_ISTAT.MMCRXIS bit is valid only when the Enable MAC Management Counters (MMC) option is selected.	
		0	MMC Receive Interrupt Status Not Active
		1	MMC Receive Interrupt Status Active
8 (R/NW)	MMCIS	MMC Interrupt Status. The EMAC_MAC_ISTAT.MMCIS bit is set (=1) when Bit 11, Bit 10, or Bit 9 is set (=1). The EMAC_MAC_ISTAT.MMCIS bit is cleared only when all these bits are low. The EMAC_MAC_ISTAT.MMCIS bit is valid only when the Enable MAC Management Counters (MMC) option is selected.	
		0	MMC Interrupt Status Not Active
		1	MMC Interrupt Status Active

Table 30-172: EMAC_MAC_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	LPIIS	LPI Interrupt Status. When the Energy Efficient Ethernet feature is enabled, the EMAC_MAC_ISTAT.LPIIS bit is set for any LPI state entry or exit in the MAC transmitter or receiver. The EMAC_MAC_ISTAT.LPIIS bit is cleared when the corresponding interrupt source bit of the EMAC_LPI_CTLSTAT register is read (or corresponding interrupt source bit of EMAC_LPI_CTLSTAT register is written to 1 (when EMAC_CSR_SW_CTL.RCWE = 1).	
		0	LPI Interrupt Status Not Active
		1	LPI Interrupt Status Active
3 (R/NW)	PHYIS	PHY Interrupt. The EMAC_MAC_ISTAT.PHYIS bit is set when rising edge is detected on the phy_intr_i input. The EMAC_MAC_ISTAT.PHYIS bit is cleared when this register is read (or the EMAC_MAC_ISTAT.PHYIS bit is written to 1 (when EMAC_CSR_SW_CTL.RCWE = 1)).	
		0	PHY Interrupt Not Detected
		1	PHY Interrupt Detected
0 (R/NW)	RGSMIIIS	RGMII or SMII Interrupt Status. The EMAC_MAC_ISTAT.RGSMIIIS bit is set because of any change in value of the link status of RGMII or SMII interface (EMAC_PHYIF_CTLSTAT.LNKSTS). The EMAC_MAC_ISTAT.RGSMIIIS bit is cleared when the EMAC_PHYIF_CTLSTAT register is read (or EMAC_PHYIF_CTLSTAT.LNKSTS is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). The EMAC_MAC_ISTAT.RGSMIIIS bit is valid only when the optional RGMII or SMII PHY interface is selected.	
		0	RGMII or SMII Interrupt Status is Not Active
		1	RGMII or SMII Interrupt Status is Active

Layer 3 and Layer 4 Control Register

The EMAC_L3L4_CTL7 register controls the operations of filter 0 of layer 3 and layer 4.

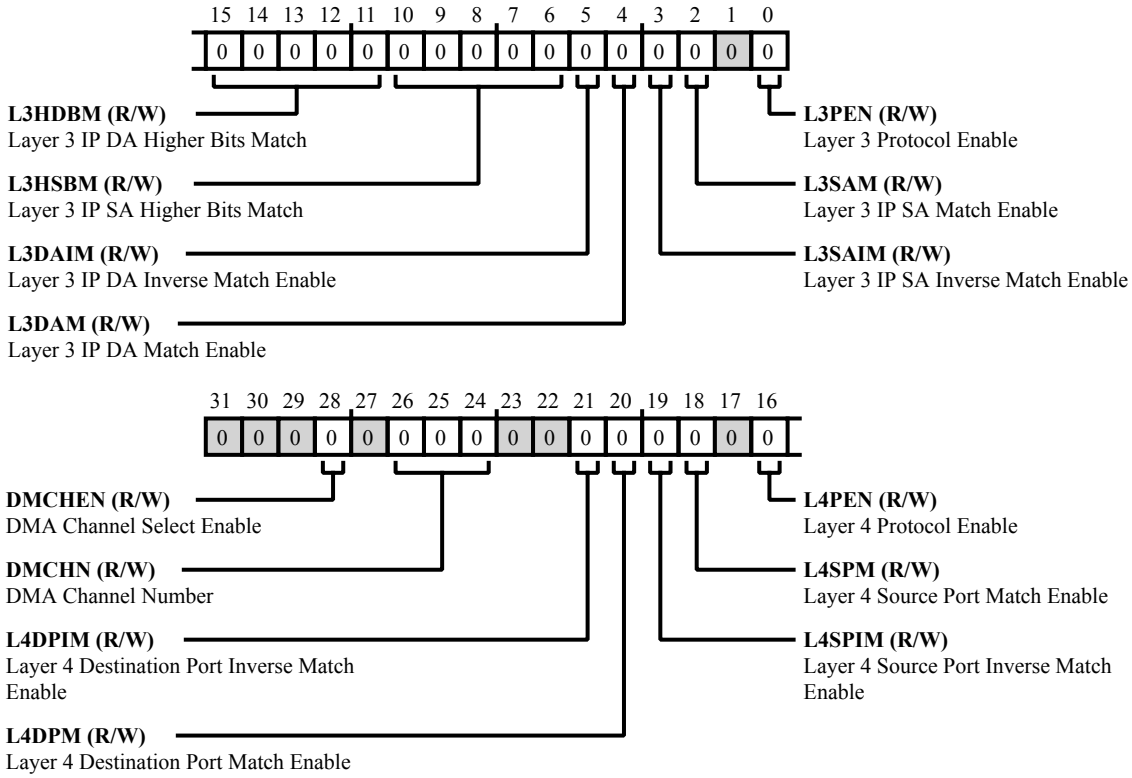


Figure 30-118: EMAC_L3L4_CTL7 Register Diagram

Table 30-173: EMAC_L3L4_CTL7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	DMCHEN	DMA Channel Select Enable. When set, the EMAC_L3L4_CTL7.DMCHEN bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When EMAC_L3L4_CTL7.DMCHEN = 0, the DMA channel is not decided by this filter.	
		0	DMA Channel Select is Disabled
		1	DMA Channel Select is Enabled
26:24 (R/W)	DMCHN	DMA Channel Number. When DMCHEN is set (=1), the EMAC_L3L4_CTL7.DMCHN field selects the DMA channel number to which the packet passed by this filter is routed. The width of the EMAC_L3L4_CTL7.DMCHN field depends on the number of the DMA channels present in the configuration.	

Table 30-173: EMAC_L3L4_CTL7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	L4DPIM	Layer 4 Destination Port Inverse Match Enable. When the EMAC_L3L4_CTL7.L4DPIM bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When EMAC_L3L4_CTL7.L4DPIM = 0, the Layer 4 Destination Port number field is enabled for perfect matching. The EMAC_L3L4_CTL7.L4DPIM bit is valid and applicable only when the EMAC_L3L4_CTL[n].L4DPM= 1.	
		0	Layer 4 Destination Port Inverse Match is Disabled
		1	Layer 4 Destination Port Inverse Match is Enabled
20 (R/W)	L4DPM	Layer 4 Destination Port Match Enable. When the EMAC_L3L4_CTL7.L4DPM bit is set, the Layer 4 Destination Port number field is enabled for matching. When EMAC_L3L4_CTL7.L4DPM = 0, the MAC ignores the Layer 4 Destination Port number field for matching.	
		0	Layer 4 Destination Port Match is Disabled
		1	Layer 4 Destination Port Match is Enabled
19 (R/W)	L4SPIM	Layer 4 Source Port Inverse Match Enable. When the EMAC_L3L4_CTL7.L4SPIM bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When EMAC_L3L4_CTL7.L4SPIM = 0, the Layer 4 Source Port number field is enabled for perfect matching. The EMAC_L3L4_CTL7.L4SPIM bit is valid and applicable only when the EMAC_L3L4_CTL[n].L4SPM= 1.	
		0	Layer 4 Source Port Inverse Match is Disabled
		1	Layer 4 Source Port Inverse Match is Enabled
18 (R/W)	L4SPM	Layer 4 Source Port Match Enable. When the EMAC_L3L4_CTL7.L4SPM bit is set, the Layer 4 Source Port number field is enabled for matching. When EMAC_L3L4_CTL7.L4SPM = 0, the MAC ignores the Layer 4 Source Port number field for matching.	
		0	Layer 4 Source Port Match is Disabled
		1	Layer 4 Source Port Match is Enabled

Table 30-173: EMAC_L3L4_CTL7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	L4PEN	Layer 4 Protocol Enable. When the `EMAC_L3L4_CTL7.L4PEN` bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When `EMAC_L3L4_CTL7.L4PEN` = 0, the Source and Destination Port number fields of TCP packets are used for matching. The layer 4 matching is done only when the EMAC_L3L4_CTL[n].L4SPM or EMAC_L3L4_CTL[n].L4DPM= 1.	
		0	Layer 4 Protocol is Disabled
		1	Layer 4 Protocol is Enabled
15:11 (R/W)	L3HDBM	Layer 3 IP DA Higher Bits Match. For IPv4 packets, the `EMAC_L3L4_CTL7.L3HDBM` field contains the number of higher bits of IP Destination Address that are matched in the IPv4 packets. The following list describes the values of this field: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 31: All bits except MSb are masked For IPv6 packets, the bits[12:11] of this field correspond to bits[6:5] of L3HSBM which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM[1:0] and L3HSBM bits: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 127: All bits except MSb are masked The `EMAC_L3L4_CTL7.L3HDBM` field is valid and applicable only when the L3DAM or L3SAM bit is set.	

Table 30-173: EMAC_L3L4_CTL7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10:6 (R/W)	L3HSBM	Layer 3 IP SA Higher Bits Match. For IPv4 packets, the EMAC_L3L4_CTL7.L3HSBM field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 31: All bits except MSb are masked For IPv6 packets, the EMAC_L3L4_CTL7.L3HSBM field contains bits[4:0] of L3HSBM. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. The EMAC_L3L4_CTL7.L3HSBM field is valid and applicable only when the EMAC_L3L4_CTL[n].L3DAM or L3SAM bit is set (=1).	
5 (R/W)	L3DAIM	Layer 3 IP DA Inverse Match Enable. When the EMAC_L3L4_CTL7.L3DAIM bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When EMAC_L3L4_CTL7.L3DAIM = 0, the Layer 3 IP Destination Address field is enabled for perfect matching. The EMAC_L3L4_CTL7.L3DAIM bit is valid and applicable only when the EMAC_L3L4_CTL[n].L3DAM = 1.	
		0	Layer 3 IP DA Inverse Match is Disabled
		1	Layer 3 IP DA Inverse Match is Enabled
4 (R/W)	L3DAM	Layer 3 IP DA Match Enable. When the EMAC_L3L4_CTL7.L3DAM bit is set, the Layer 3 IP Destination Address field is enabled for matching. When EMAC_L3L4_CTL7.L3DAM = 0, the MAC ignores the Layer 3 IP Destination Address field for matching. Note: When EMAC_L3L4_CTL[n].L3PEN = 1, set either the EMAC_L3L4_CTL7.L3DAM bit or the L3SAM bit because either IPv6 DA or SA can be checked for filtering.	
		0	Layer 3 IP DA Match is Disabled
		1	Layer 3 IP DA Match is Enabled

Table 30-173: EMAC_L3L4_CTL7 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	L3SAIM	Layer 3 IP SA Inverse Match Enable. When the EMAC_L3L4_CTL7.L3SAIM bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When the EMAC_L3L4_CTL7.L3SAIM bit reset, the Layer 3 IP Source Address field is enabled for perfect matching. The EMAC_L3L4_CTL7.L3SAIM bit is valid and applicable only when EMAC_L3L4_CTL[n].L3DAM = 1.	
		0	Layer 3 IP SA Inverse Match is Disabled
		1	Layer 3 IP SA Inverse Match is Enabled
2 (R/W)	L3SAM	Layer 3 IP SA Match Enable. When the EMAC_L3L4_CTL7.L3SAM bit is set, the Layer 3 IP Source Address field is enabled for matching. When EMAC_L3L4_CTL7.L3SAM = 0, the MAC ignores the Layer 3 IP Source Address field for matching. Note: When the EMAC_L3L4_CTL[n].L3PEN bit is set, set either the EMAC_L3L4_CTL7.L3SAM bit or the EMAC_L3L4_CTL[n].L3DAM bit because either IPv6 SA or DA can be checked for filtering.	
		0	Layer 3 IP SA Match is Disabled
		1	Layer 3 IP SA Match is Enabled
0 (R/W)	L3PEN	Layer 3 Protocol Enable. When the EMAC_L3L4_CTL7.L3PEN bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When EMAC_L3L4_CTL7.L3PEN = 0, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets. The layer 3 matching is done only when the EMAC_L3L4_CTL[n].L3SAM or EMAC_L3L4_CTL[n].L3DAM bit is set (=1).	
		0	Layer 3 Protocol is Disabled
		1	Layer 3 Protocol is Enabled

Layer 3 and Layer 4 Control Register

The EMAC_L3L4_CTL[i] register controls the operations of filter 0 of layer 3 and layer 4.

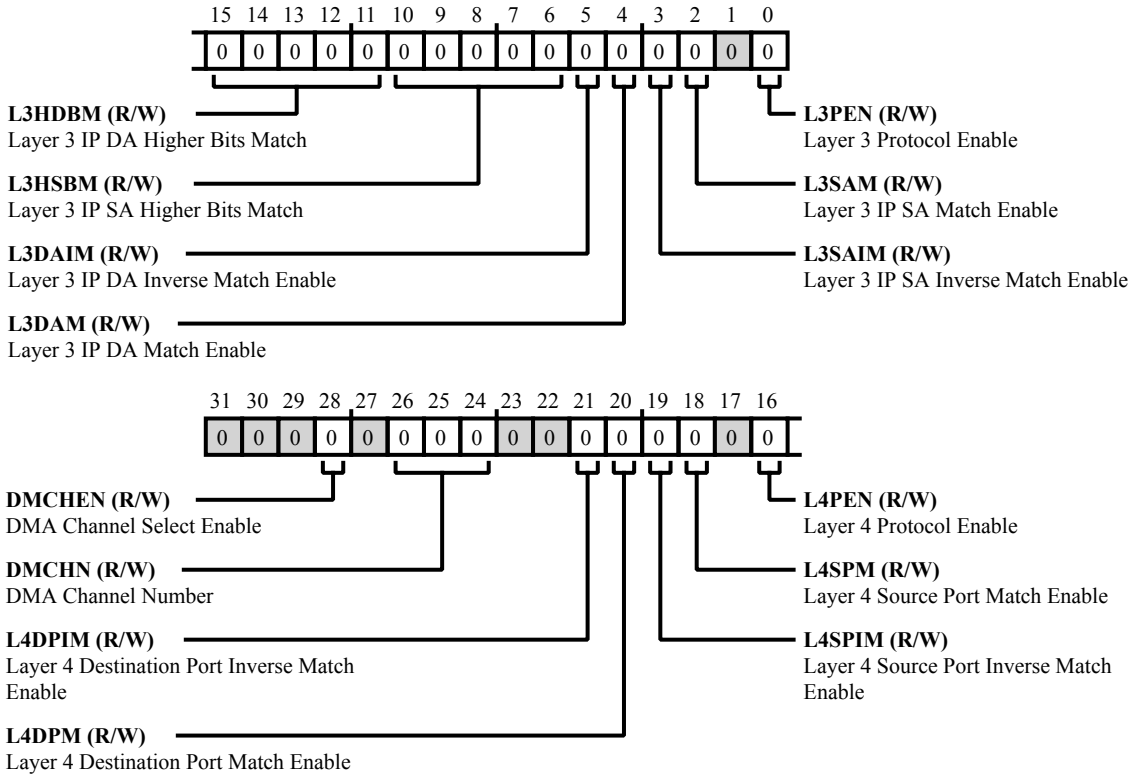


Figure 30-119: EMAC_L3L4_CTL[i] Register Diagram

Table 30-174: EMAC_L3L4_CTL[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	DMCHEN	DMA Channel Select Enable. When set, the EMAC_L3L4_CTL[i].DMCHEN bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When EMAC_L3L4_CTL[i].DMCHEN = 0, the DMA channel is not decided by this filter.	
		0	DMA Channel Select is Disabled
		1	DMA Channel Select is Enabled
26:24 (R/W)	DMCHN	DMA Channel Number. When DMCHEN is set (=1), the EMAC_L3L4_CTL[i].DMCHN field selects the DMA channel number to which the packet passed by this filter is routed. The width of the EMAC_L3L4_CTL[i].DMCHN field depends on the number of the DMA channels present in the configuration.	

Table 30-174: EMAC_L3L4_CTL[i] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	L4DPIM	Layer 4 Destination Port Inverse Match Enable. When the `EMAC_L3L4_CTL[i].L4DPIM` bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When `EMAC_L3L4_CTL[i].L4DPIM` = 0, the Layer 4 Destination Port number field is enabled for perfect matching. The `EMAC_L3L4_CTL[i].L4DPIM` bit is valid and applicable only when the EMAC_L3L4_CTL[n].L4DPM= 1.	
		0	Layer 4 Destination Port Inverse Match is Disabled
		1	Layer 4 Destination Port Inverse Match is Enabled
20 (R/W)	L4DPM	Layer 4 Destination Port Match Enable. When the `EMAC_L3L4_CTL[i].L4DPM` bit is set, the Layer 4 Destination Port number field is enabled for matching. When `EMAC_L3L4_CTL[i].L4DPM` = 0, the MAC ignores the Layer 4 Destination Port number field for matching.	
		0	Layer 4 Destination Port Match is Disabled
		1	Layer 4 Destination Port Match is Enabled
19 (R/W)	L4SPIM	Layer 4 Source Port Inverse Match Enable. When the `EMAC_L3L4_CTL[i].L4SPIM` bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When `EMAC_L3L4_CTL[i].L4SPIM` = 0, the Layer 4 Source Port number field is enabled for perfect matching. The `EMAC_L3L4_CTL[i].L4SPIM` bit is valid and applicable only when the EMAC_L3L4_CTL[n].L4SPM= 1.	
		0	Layer 4 Source Port Inverse Match is Disabled
		1	Layer 4 Source Port Inverse Match is Enabled
18 (R/W)	L4SPM	Layer 4 Source Port Match Enable. When the `EMAC_L3L4_CTL[i].L4SPM` bit is set, the Layer 4 Source Port number field is enabled for matching. When `EMAC_L3L4_CTL[i].L4SPM` = 0, the MAC ignores the Layer 4 Source Port number field for matching.	
		0	Layer 4 Source Port Match is Disabled
		1	Layer 4 Source Port Match is Enabled

Table 30-174: EMAC_L3L4_CTL[i] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	L4PEN	Layer 4 Protocol Enable. When the `EMAC_L3L4_CTL[i].L4PEN` bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When `EMAC_L3L4_CTL[i].L4PEN` = 0, the Source and Destination Port number fields of TCP packets are used for matching. The layer 4 matching is done only when the EMAC_L3L4_CTL[n].L4SPM or EMAC_L3L4_CTL[n].L4DPM= 1.	
		0	Layer 4 Protocol is Disabled
		1	Layer 4 Protocol is Enabled
15:11 (R/W)	L3HDBM	Layer 3 IP DA Higher Bits Match. For IPv4 packets, the `EMAC_L3L4_CTL[i].L3HDBM` field contains the number of higher bits of IP Destination Address that are matched in the IPv4 packets. The following list describes the values of this field: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 31: All bits except MSb are masked For IPv6 packets, the bits[12:11] of this field correspond to bits[6:5] of L3HSBM which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM[1:0] and L3HSBM bits: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 127: All bits except MSb are masked The `EMAC_L3L4_CTL[i].L3HDBM` field is valid and applicable only when the L3DAM or L3SAM bit is set.	

Table 30-174: EMAC_L3L4_CTL[i] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10:6 (R/W)	L3HSBM	Layer 3 IP SA Higher Bits Match. For IPv4 packets, the `EMAC_L3L4_CTL[i].L3HSBM` field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field: - 0: No bits are masked - 1: LSb[0] is masked - 2: Two LSbs [1:0] are masked - .. - 31: All bits except MSb are masked For IPv6 packets, the `EMAC_L3L4_CTL[i].L3HSBM` field contains bits[4:0] of L3HSBM. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. The `EMAC_L3L4_CTL[i].L3HSBM` field is valid and applicable only when the EMAC_L3L4_CTL[n].L3DAM or L3SAM bit is set (=1).	
5 (R/W)	L3DAIM	Layer 3 IP DA Inverse Match Enable. When the `EMAC_L3L4_CTL[i].L3DAIM` bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When `EMAC_L3L4_CTL[i].L3DAIM` = 0, the Layer 3 IP Destination Address field is enabled for perfect matching. The `EMAC_L3L4_CTL[i].L3DAIM` bit is valid and applicable only when the EMAC_L3L4_CTL[n].L3DAM = 1.	
		0	Layer 3 IP DA Inverse Match is Disabled
		1	Layer 3 IP DA Inverse Match is Enabled
4 (R/W)	L3DAM	Layer 3 IP DA Match Enable. When the `EMAC_L3L4_CTL[i].L3DAM` bit is set, the Layer 3 IP Destination Address field is enabled for matching. When `EMAC_L3L4_CTL[i].L3DAM` = 0, the MAC ignores the Layer 3 IP Destination Address field for matching. Note: When EMAC_L3L4_CTL[n].L3PEN = 1, set either the `EMAC_L3L4_CTL[i].L3DAM` bit or the L3SAM bit because either IPv6 DA or SA can be checked for filtering.	
		0	Layer 3 IP DA Match is Disabled
		1	Layer 3 IP DA Match is Enabled

Table 30-174: EMAC_L3L4_CTL[i] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	L3SAIM	Layer 3 IP SA Inverse Match Enable. When the `EMAC_L3L4_CTL[i].L3SAIM` bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When the `EMAC_L3L4_CTL[i].L3SAIM` bit reset, the Layer 3 IP Source Address field is enabled for perfect matching. The `EMAC_L3L4_CTL[i].L3SAIM` bit is valid and applicable only when EMAC_L3L4_CTL[n].L3DAM = 1.	
		0	Layer 3 IP SA Inverse Match is Disabled
		1	Layer 3 IP SA Inverse Match is Enabled
2 (R/W)	L3SAM	Layer 3 IP SA Match Enable. When the `EMAC_L3L4_CTL[i].L3SAM` bit is set, the Layer 3 IP Source Address field is enabled for matching. When `EMAC_L3L4_CTL[i].L3SAM` = 0, the MAC ignores the Layer 3 IP Source Address field for matching. Note: When the EMAC_L3L4_CTL[n].L3PEN bit is set, set either the `EMAC_L3L4_CTL[i].L3SAM` bit or the EMAC_L3L4_CTL[n].L3DAM bit because either IPv6 SA or DA can be checked for filtering.	
		0	Layer 3 IP SA Match is Disabled
		1	Layer 3 IP SA Match is Enabled
0 (R/W)	L3PEN	Layer 3 Protocol Enable. When the `EMAC_L3L4_CTL[i].L3PEN` bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When `EMAC_L3L4_CTL[i].L3PEN` = 0, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets. The layer 3 matching is done only when the EMAC_L3L4_CTL[n].L3SAM or EMAC_L3L4_CTL[n].L3DAM bit is set (=1).	
		0	Layer 3 Protocol is Disabled
		1	Layer 3 Protocol is Enabled

LPI Control and Status Register

The EMAC_LPI_CTLSTAT register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

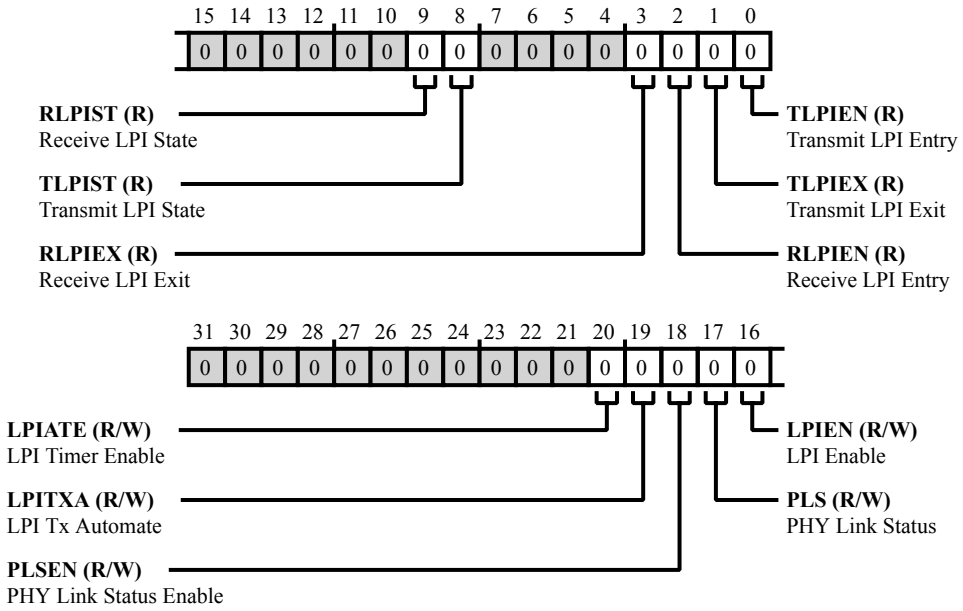


Figure 30-120: EMAC_LPI_CTLSTAT Register Diagram

Table 30-175: EMAC_LPI_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	LPIATE	LPI Timer Enable. The EMAC_LPI_CTLSTAT.LPIATE bit controls the automatic entry of the MAC transmitter into and exit out of the LPI state. When LPIATE, LPITXA and LPIEN bits are set, the MAC transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the EMAC_LPI_ENTR_TMR register. After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again. When EMAC_LPI_CTLSTAT.LPIATE = 0, the LPI auto timer is disabled and MAC transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions.	
		0	LPI Timer is Disabled
		1	LPI Timer is Enabled

Table 30-175: EMAC_LPI_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	LPITXA	LPI Tx Automate. The EMAC_LPI_CTLSTAT.LPITXA bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the transmit side. If the EMAC_LPI_CTLSTAT.LPITXA and LPIEN bits are set (=1), the MAC enters the LPI mode only after all outstanding packets (in the IP) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO flush is set in EMAC_TQ0_OPMODE.FTQ, when the MAC is in the LPI mode, it exits the LPI mode. When the EMAC_LPI_CTLSTAT.LPITXA bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.	
		0	LPI Tx Automate is Disabled
		1	LPI Tx Automate is Enabled
18 (R/W)	PLSEN	PHY Link Status Enable. The EMAC_LPI_CTLSTAT.PLSEN bit enables the link status received on the RGMII receive paths to be used for activating the LPI LS TIMER. When the EMAC_LPI_CTLSTAT.PLSEN bit is set, the MAC uses the link-status bits of the EMAC_PHYIF_CTLSTAT register and the PLS bit for the LPI LS timer trigger. When EMAC_LPI_CTLSTAT.PLSEN = 0, the MAC ignores the link-status bits of the EMAC_PHYIF_CTLSTAT register and takes only the PLS bit.	
		0	PHY Link Status is Disabled
		1	PHY Link Status is Enabled
17 (R/W)	PLS	PHY Link Status. The EMAC_LPI_CTLSTAT.PLS bit indicates the link status of the PHY. The MAC transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER. When the EMAC_LPI_CTLSTAT.PLS bit is set, the link is considered to be okay (UP) and when EMAC_LPI_CTLSTAT.PLS = 0, the link is considered to be down.	
		0	Link is Down
		1	Link is Okay (UP)

Table 30-175: EMAC_LPI_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	LPIEN	LPI Enable. When the EMAC_LPI_CTLSTAT.LPIEN bit is set, it instructs the MAC transmitter to enter the LPI state. When EMAC_LPI_CTLSTAT.LPIEN = 0, it instructs the MAC to exit the LPI state and resume normal transmission. The EMAC_LPI_CTLSTAT.LPIEN bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.	
		0	LPI State is Disabled
		1	LPI State is Enabled
9 (R/NW)	RLPIST	Receive LPI State. When the EMAC_LPI_CTLSTAT.RLPIST bit is set, it indicates that the MAC is receiving the LPI pattern on the GMII or MII interface.	
		0	Receive LPI State Not Detected
		1	Receive LPI State Detected
8 (R/NW)	TLPIST	Transmit LPI State. When the EMAC_LPI_CTLSTAT.TLPIST bit is set, it indicates that the MAC is transmitting the LPI pattern on the GMII or MII interface.	
		0	Transmit LPI State Not Detected
		1	Transmit LPI State Detected
3 (R/NW)	RLPIEX	Receive LPI Exit. When the EMAC_LPI_CTLSTAT.RLPIEX bit is set, it indicates that the MAC receiver has stopped receiving the LPI pattern on the GMII or MII interface, exited the LPI state, and resumed the normal reception. The EMAC_LPI_CTLSTAT.RLPIEX bit is cleared by a read into this register (or the EMAC_LPI_CTLSTAT.RLPIEX bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Note: The EMAC_LPI_CTLSTAT.RLPIEX bit might not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of SCLK0.	
		0	Receive LPI Exit Not Detected
		1	Receive LPI Exit Detected

Table 30-175: EMAC_LPI_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	RLPIEN	Receive LPI Entry. When the EMAC_LPI_CTLSTAT.RLPIEN bit is set, it indicates that the MAC receiver has received an LPI pattern and entered the LPI state. The EMAC_LPI_CTLSTAT.RLPIEN bit is cleared by a read into this register (or the EMAC_LPI_CTLSTAT.RLPIEN bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Note: The EMAC_LPI_CTLSTAT.RLPIEN bit might not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of SCLK0.	
		0	Receive LPI Entry Not Detected
		1	Receive LPI Entry Detected
1 (R/NW)	TLPIEX	Transmit LPI Exit. When the EMAC_LPI_CTLSTAT.TLPIEX bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW timer has expired. The EMAC_LPI_CTLSTAT.TLPIEX bit is cleared by a read into this register (or the EMAC_LPI_CTLSTAT.TLPIEX bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1).	
		0	Transmit LPI Exit Not Detected
		1	Transmit LPI Exit Detected
0 (R/NW)	TLPIEN	Transmit LPI Entry. When the EMAC_LPI_CTLSTAT.TLPIEN bit is set, it indicates that the MAC transmitter has entered the LPI state because of the setting of the LPIEN bit. The EMAC_LPI_CTLSTAT.TLPIEN bit is cleared by a read into this register (or the EMAC_LPI_CTLSTAT.TLPIEN bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1).	
		0	Transmit LPI Entry Not Detected
		1	Transmit LPI Entry Detected

Transmit LPI Entry Timer Register

The EMAC_LPI_ENTR_TMR register controls the Tx LPI entry timer. This counter is enabled only when EMAC_LPI_CTLSTAT.LPITE = 1.

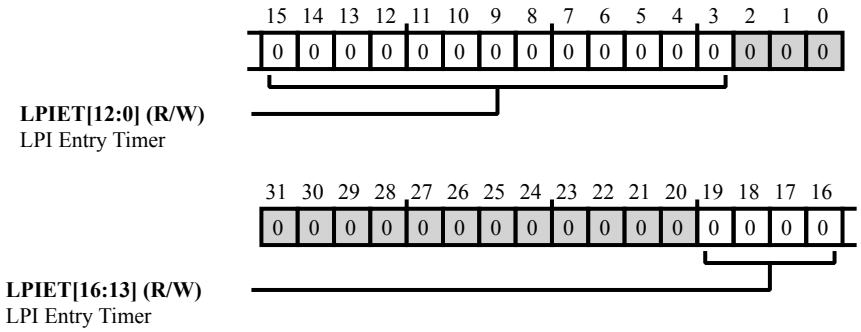


Figure 30-121: EMAC_LPI_ENTR_TMR Register Diagram

Table 30-176: EMAC_LPI_ENTR_TMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:3 (R/W)	LPIET	LPI Entry Timer. The EMAC_LPI_ENTR_TMR.LPIET field specifies the time in microseconds the MAC waits to enter LPI mode, after it has transmitted all the frames. The EMAC_LPI_ENTR_TMR.LPIET field is valid and used only when LPITE and LPITXA are set (=1). Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 microseconds.

LPI Timers Control Register

The EMAC_LPI_TMRSCTL register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

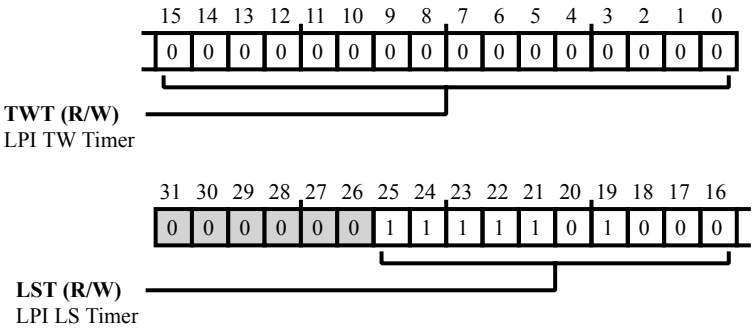


Figure 30-122: EMAC_LPI_TMRSCTL Register Diagram

Table 30-177: EMAC_LPI_TMRSCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:16 (R/W)	LST	LPI LS Timer. The EMAC_LPI_TMRSCTL.LST field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.
15:0 (R/W)	TWT	LPI TW Timer. The EMAC_LPI_TMRSCTL.TWT field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

Layer 3 Address 0 Register 7

For IPv4 packets, the EMAC_L3_ADDR0_REG7 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains bits[31:0] of the 128-bit IP Source Address or Destination Address field.

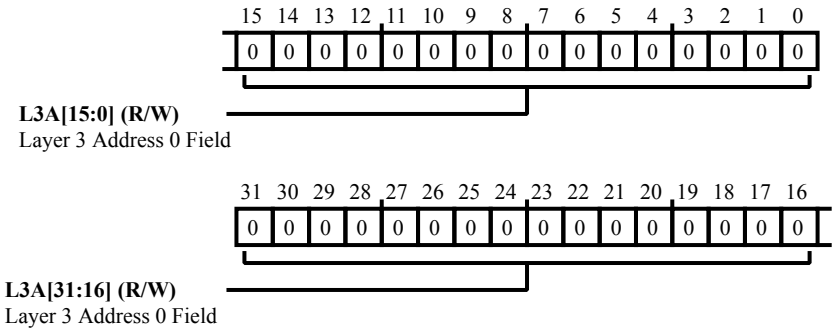


Figure 30-123: EMAC_L3_ADDR0_REG7 Register Diagram

Table 30-178: EMAC_L3_ADDR0_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A	Layer 3 Address 0 Field. When EMAC_L3L4_CTL[n].L3PEN and EMAC_L3L4_CTL[n].L3SAM = 1, the EMAC_L3_ADDR0_REG7.L3A field contains the value to be matched with bits[31:0] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN and EMAC_L3L4_CTL[n].L3DAM = 1, the EMAC_L3_ADDR0_REG7.L3A field contains the value to be matched with bits[31:0] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM = 1, the EMAC_L3_ADDR0_REG7.L3A field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer 3 Address 0 Register [i]

For IPv4 packets, the `EMAC_L3_ADDR0_REG[i]` register contains the 32-bit IP Source Address field. For IPv6 packets, it contains bits[31:0] of the 128-bit IP Source Address or Destination Address field.

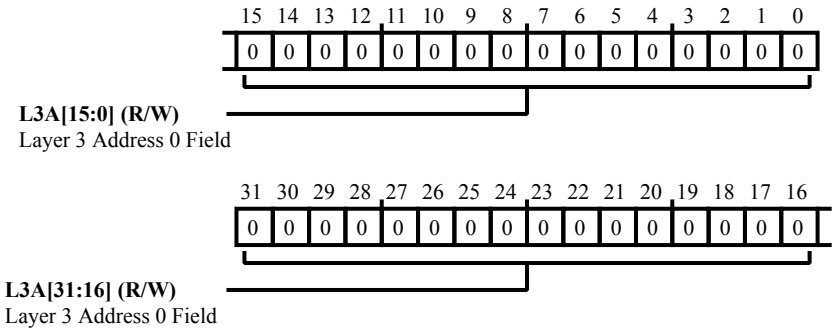


Figure 30-124: EMAC_L3_ADDR0_REG[i] Register Diagram

Table 30-179: EMAC_L3_ADDR0_REG[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A	Layer 3 Address 0 Field. When EMAC_L3L4_CTL[n].L3PEN and EMAC_L3L4_CTL[n].L3SAM = 1, the `EMAC_L3_ADDR0_REG[i].L3A` field contains the value to be matched with bits[31:0] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN and EMAC_L3L4_CTL[n].L3DAM = 1, the `EMAC_L3_ADDR0_REG[i].L3A` field contains the value to be matched with bits[31:0] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM = 1, the `EMAC_L3_ADDR0_REG[i].L3A` field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer 3 Address 1 Register 7

For IPv4 packets, the EMAC_L3_ADDR1_REG7 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains bits[63:32] of the 128-bit IP Source Address or Destination Address field.

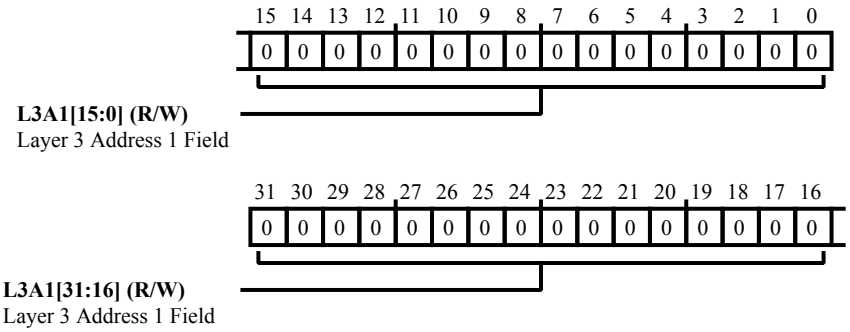


Figure 30-125: EMAC_L3_ADDR1_REG7 Register Diagram

Table 30-180: EMAC_L3_ADDR1_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A1	Layer 3 Address 1 Field. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR1_REG7.L3A1 field contains the value to be matched with bits[63:32] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, the EMAC_L3_ADDR1_REG7.L3A1 field contains the value to be matched with bits[63:32] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR1_REG7.L3A1 field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer 3 Address 1 Register [i]

For IPv4 packets, the `EMAC_L3_ADDR1_REG[i]` register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains bits[63:32] of the 128-bit IP Source Address or Destination Address field.

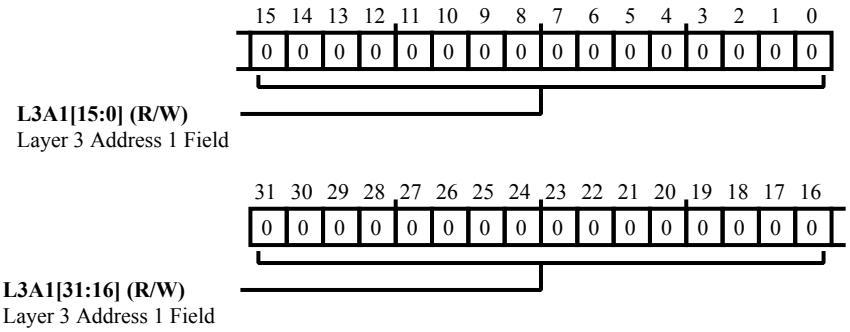


Figure 30-126: EMAC_L3_ADDR1_REG[i] Register Diagram

Table 30-181: EMAC_L3_ADDR1_REG[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A1	Layer 3 Address 1 Field. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the `EMAC_L3_ADDR1_REG[i].L3A1` field contains the value to be matched with bits[63:32] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, the `EMAC_L3_ADDR1_REG[i].L3A1` field contains the value to be matched with bits[63:32] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM =1, the `EMAC_L3_ADDR1_REG[i].L3A1` field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer 3 Address 2 Register 7

The EMAC_L3_ADDR2_REG7 register is reserved for IPv4 packets. For IPv6 packets, the EMAC_L3_ADDR2_REG7 register contains bits[95:64] of 128-bit IP Source Address or Destination Address field.

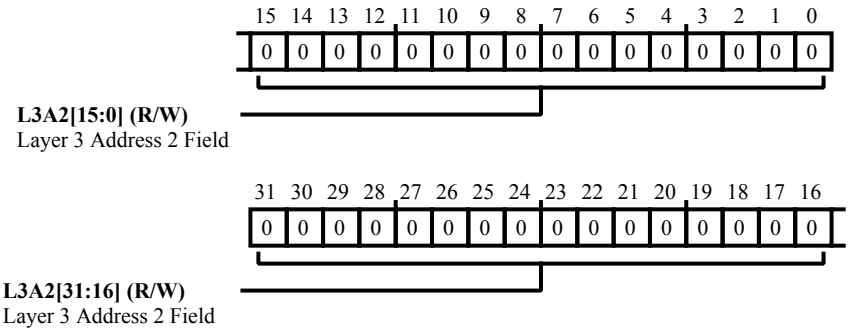


Figure 30-127: EMAC_L3_ADDR2_REG7 Register Diagram

Table 30-182: EMAC_L3_ADDR2_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A2	Layer 3 Address 2 Field. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR2_REG7.L3A2 field contains the value to be matched with bits[95:64] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, the EMAC_L3_ADDR2_REG7.L3A2 field contains the value to be matched with bits[95:64] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0, the EMAC_L3_ADDR2_REG7.L3A2 field is not used.

Layer 3 Address 2 Register [i]

The EMAC_L3_ADDR2_REG[i] register is reserved for IPv4 packets. For IPv6 packets, the EMAC_L3_ADDR2_REG[i] register contains bits[95:64] of 128-bit IP Source Address or Destination Address field.

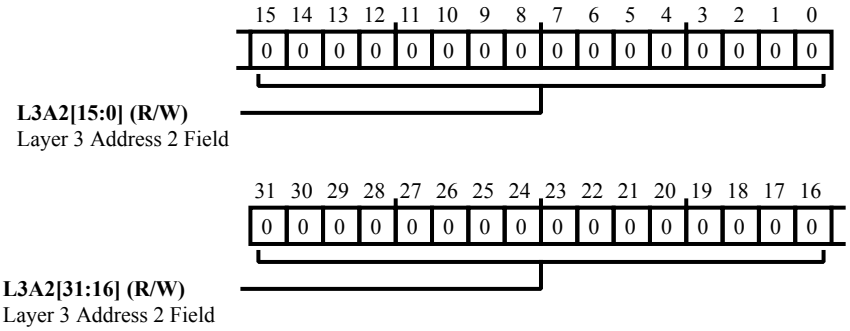


Figure 30-128: EMAC_L3_ADDR2_REG[i] Register Diagram

Table 30-183: EMAC_L3_ADDR2_REG[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A2	Layer 3 Address 2 Field. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR2_REG[i].L3A2 field contains the value to be matched with bits[95:64] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, the EMAC_L3_ADDR2_REG[i].L3A2 field contains the value to be matched with bits[95:64] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0, the EMAC_L3_ADDR2_REG[i].L3A2 field is not used.

Layer 3 Address 3 Register 7

The EMAC_L3_ADDR3_REG7 register is reserved for IPv4 packets. For IPv6 packets, the EMAC_L3_ADDR3_REG7 register contains bits[127:96] of 128-bit IP Source Address or Destination Address field.

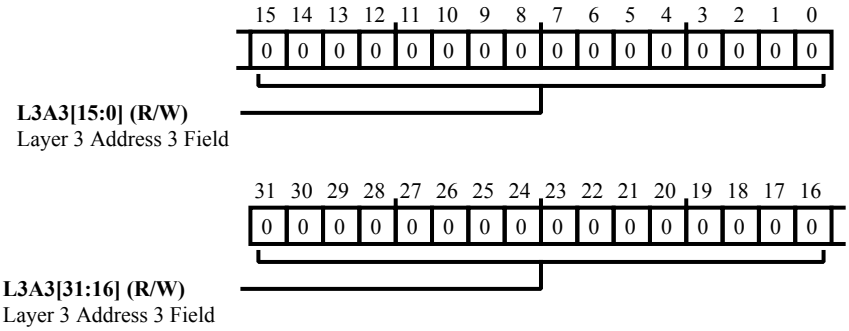


Figure 30-129: EMAC_L3_ADDR3_REG7 Register Diagram

Table 30-184: EMAC_L3_ADDR3_REG7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A3	Layer 3 Address 3 Field. When MAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR3_REG7.L3A3 field contains the value to be matched with bits[127:96] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, this field contains the value to be matched with bits[127:96] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM =1, this field is not used.

Layer 3 Address 3 Register [i]

The EMAC_L3_ADDR3_REG[i] register is reserved for IPv4 packets. For IPv6 packets, the EMAC_L3_ADDR3_REG[i] register contains bits[127:96] of 128-bit IP Source Address or Destination Address field.

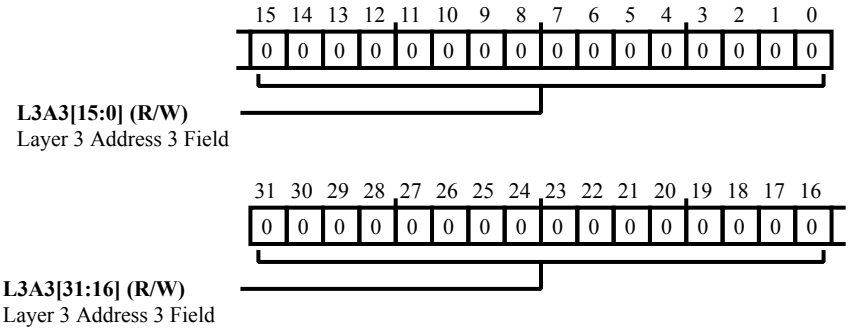


Figure 30-130: EMAC_L3_ADDR3_REG[i] Register Diagram

Table 30-185: EMAC_L3_ADDR3_REG[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	L3A3	Layer 3 Address 3 Field. When MAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3SAM =1, the EMAC_L3_ADDR3_REG[i].L3A3 field contains the value to be matched with bits[127:96] of the IP Source Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 1 and EMAC_L3L4_CTL[n].L3DAM =1, this field contains the value to be matched with bits[127:96] of the IP Destination Address field in the IPv6 packets. When EMAC_L3L4_CTL[n].L3PEN = 0 and EMAC_L3L4_CTL[n].L3SAM =1, this field is not used.

Layer 4 Address Register [i]

The EMAC_L4_ADDR7 register is double synchronized to the Rx clock domain. Thus, for proper synchronization updates, the program should perform consecutive writes to the same layer 3/4 address registers only after at least four clock cycles delay of the (G) MII clock domain.

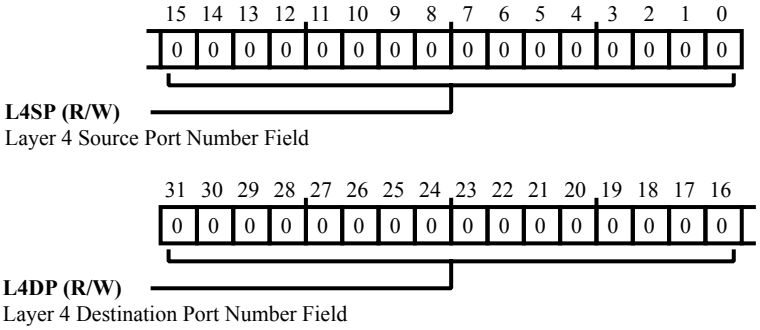


Figure 30-131: EMAC_L4_ADDR7 Register Diagram

Table 30-186: EMAC_L4_ADDR7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	L4DP	Layer 4 Destination Port Number Field. When EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4DPM =1, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets. When the EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4DPM =1, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.
15:0 (R/W)	L4SP	Layer 4 Source Port Number Field. When EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4SPM =1, the EMAC_L4_ADDR7.L4SP field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets. When the EMAC_L3L4_CTL[n].L4PEN = 1 and EMAC_L3L4_CTL[n].L4SPM =1, the EMAC_L4_ADDR7.L4SP field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer 4 Address Register [i]

The `EMAC_L4_ADDR[i]` register is double synchronized to the Rx clock domain. Thus, for proper synchronization updates, the program should perform consecutive writes to the same layer 3/4 address registers only after at least four clock cycles delay of the (G) MII clock domain.

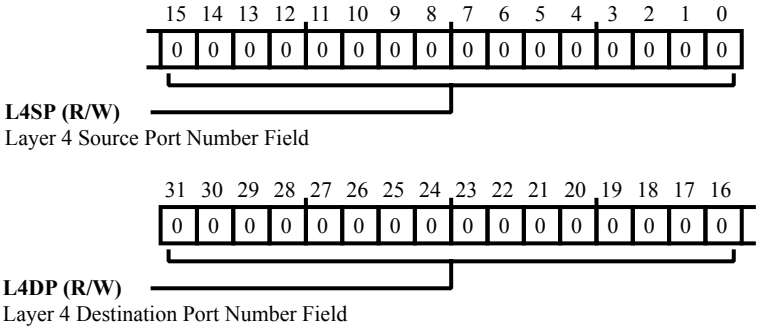


Figure 30-132: EMAC_L4_ADDR[i] Register Diagram

Table 30-187: EMAC_L4_ADDR[i] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	L4DP	Layer 4 Destination Port Number Field. When EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4DPM =1, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets. When the EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4DPM =1, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.
15:0 (R/W)	L4SP	Layer 4 Source Port Number Field. When EMAC_L3L4_CTL[n].L4PEN = 0 and EMAC_L3L4_CTL[n].L4SPM =1, the `EMAC_L4_ADDR[i].L4SP` field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets. When the EMAC_L3L4_CTL[n].L4PEN = 1 and EMAC_L3L4_CTL[n].L4SPM =1, the `EMAC_L4_ADDR[i].L4SP` field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

PTP Log Message Interval Register

The EMAC_LOG_MSG_INTVL register contains the periodic intervals for automatic PTP packet generation.

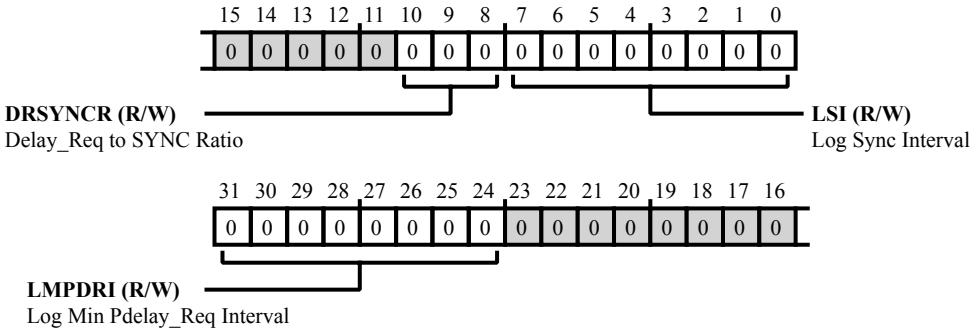


Figure 30-133: EMAC_LOG_MSG_INTVL Register Diagram

Table 30-188: EMAC_LOG_MSG_INTVL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	LMPDRI	Log Min Pdelay_Req Interval. The EMAC_LOG_MSG_INTVL.LMPDRI field indicates logMinPdelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15.Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.	
10:8 (R/W)	DRSYNCR	Delay_Req to SYNC Ratio. In completer mode, the EMAC_LOG_MSG_INTVL.DRSYNCR field is used for controlling the frequency of Delay_Req messages transmitted. The requester sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the completer. The EMAC module receiver processes this value from the received DelayResp messages and updates this field accordingly. In completer mode, the host must not write/update this register unless it has to override the received value. In requester mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message. Access restriction applies. Updated based on the event. Write 1 to set. Write 0 to clear.	
		0	DelayReq Generated for Every Received SYNC
		1	DelayReq Generated Every Alternate Reception of SYNC
		2	For Every 4 SYNC Messages
		3	For Every 8 SYNC Messages
		4	For Every 16 SYNC Messages
		5	For Every 32 SYNC Messages

Table 30-188: EMAC_LOG_MSG_INTVL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	Reserved
7:0 (R/W)	LSI	Log Sync Interval. The `EMAC_LOG_MSG_INTVL.LSI` field indicates the periodicity of the automatically generated SYNC message when the PTP node is the requester. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.	

MDIO Address Register

The EMAC_MDIO_ADDR register controls the management cycles to external PHY through a management interface.

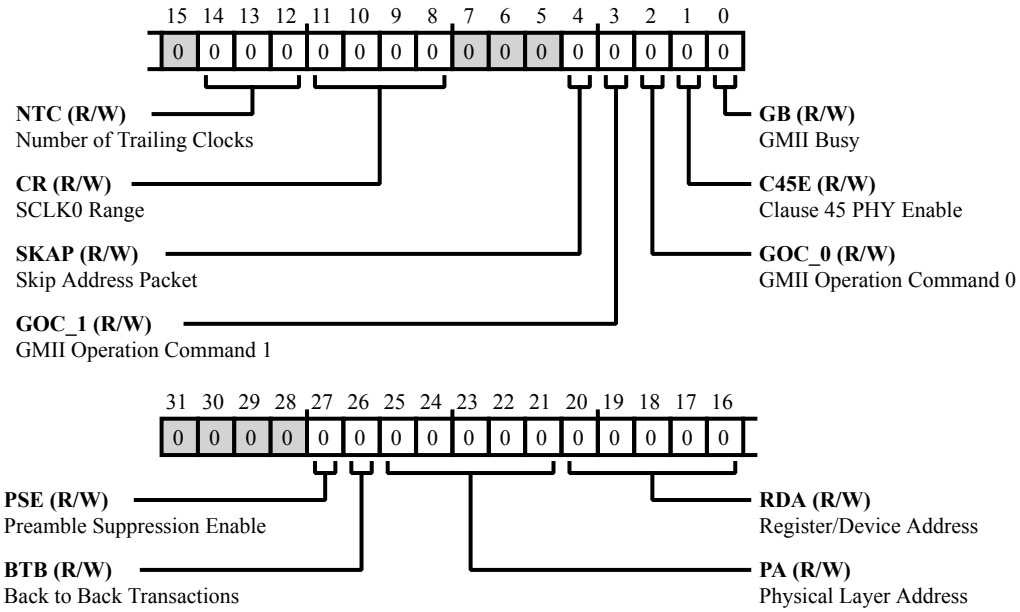


Figure 30-134: EMAC_MDIO_ADDR Register Diagram

Table 30-189: EMAC_MDIO_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	PSE	Preamble Suppression Enable. When the EMAC_MDIO_ADDR.PSE bit is set, the SMA suppresses the 32-bit preamble and transmits MDIO frames with only 1 preamble bit. When the EMAC_MDIO_ADDR.PSE bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.	
		0	Preamble Suppression Disabled
		1	Preamble Suppression Enabled

Table 30-189: EMAC_MDIO_ADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	BTB	Back to Back Transactions. When the EMAC_MDIO_ADDR.BTB bit is set and the NTC has value greater than 0, the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame. When EMAC_MDIO_ADDR.BTB = 0, then the read/write command completion (GB is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame. The EMAC_MDIO_ADDR.BTB bit must not be set when NTC=0.	
		0	Back to Back Transactions Disabled
		1	Back to Back Transactions Enabled
25:21 (R/W)	PA	Physical Layer Address. The EMAC_MDIO_ADDR.PA field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing. The EMAC_MDIO_ADDR.PA field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.	
20:16 (R/W)	RDA	Register/Device Address. The EMAC_MDIO_ADDR.RDA field selects the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.	
14:12 (R/W)	NTC	Number of Trailing Clocks. The EMAC_MDIO_ADDR.NTC field controls the number of trailing clock cycles generated on ETHx_MDC (MDC) after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 3'h3 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.	

Table 30-189: EMAC_MDIO_ADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:8 (R/W)	CR	SCLK0 Range. The EMAC_MDIO_ADDR.CR field indicates the frequency of the MDC clock according to the SCLK0 frequency used: 0000: SCLK0 = 60-100 MHz; MDC clock = SCLK0/42 0001: SCLK0 = 100-150 MHz; MDC clock = SCLK0/62 0010: SCLK0 = 20-35 MHz; MDC clock = SCLK0/16 0011: SCLK0 = 35-60 MHz; MDC clock = SCLK0/26 0100: SCLK0 = 150-250 MHz; MDC clock = SCLK0/102 0101: Reserved 0110: Reserved 0111: Reserved The suggested range of SCLK0 clock frequency applicable for each value (when bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range. When bit 11 is set, a higher frequency of the MDC clock can be achieved than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when SCLK0 is of 100 MHz frequency and program these bits as 1010; the resultant MDC clock is 12.5 MHz which is beyond the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks: 1000: SCLK0 /4 1001: SCLK0 /6 1010: SCLK0 /8 1011: SCLK0 /10 1100: SCLK0 /12 1101: SCLK0 /14 1110: SCLK0 /16 1111: SCLK0 /18	
4 (R/W)	SKAP	Skip Address Packet. When the EMAC_MDIO_ADDR.SKAP bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. The EMAC_MDIO_ADDR.SKAP bit is valid only when EMAC_MDIO_ADDR.C45E = 1.	
		0	Skip Address Packet is Disabled
		1	Skip Address Packet is Enabled

Table 30-189: EMAC_MDIO_ADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	GOC_1	GMII Operation Command 1. The EMAC_MDIO_ADDR.GOC_1 bit is higher bit of the operation command to the PHY, GOC_1 and GOC_O is encoded as follows: 00: Reserved 01: Write 10: Post read increment address for clause 45 PHY 11: Read When clause 22 PHY is enabled, only write and read commands are valid.	
		0	GMII Operation Command 1 is Disabled
		1	GMII Operation Command 1 is Enabled
2 (R/W)	GOC_0	GMII Operation Command 0. EMAC_MDIO_ADDR.GOC_0 is the lower bit of the operation command to the PHY. When in SMA mode (MDIO requester) the EMAC_MDIO_ADDR.GOC_0 bit along with GOC_1 determines the operation to be performed to the PHY.	
		0	GMII Operation Command 0 is Disabled
		1	GMII Operation Command 0 is Enabled
1 (R/W)	C45E	Clause 45 PHY Enable. When the EMAC_MDIO_ADDR.C45E bit is set, Clause 45 capable PHY is connected to MDIO. When EMAC_MDIO_ADDR.C45E = 0, Clause 22 capable PHY is connected to MDIO.	
		0	Clause 45 PHY is Disabled
		1	Clause 45 PHY is Enabled

Table 30-189: EMAC_MDIO_ADDR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	GB	GMII Busy. The application sets the EMAC_MDIO_ADDR.GB bit to instruct the SMA to initiate a read or write access to the MDIO completer. The MAC clears the EMAC_MDIO_ADDR.GB bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in EMAC_MDIO_ADDR and EMAC_MDIO_DATA registers as long as this bit is set. For write transfers, the application must first write 16-bit data in the GDl field (and also RA field when EMAC_MDIO_ADDR.C45E = 1) in EMAC_MDIO_DATA register before setting the EMAC_MDIO_ADDR.GB bit. When EMAC_MDIO_ADDR.C45E = 1, it should also write to EMAC_MDIO_DATA.RA before initiating a read transfer. When a read transfer is completed (GB=0), the data read from the PHY register is valid in EMAC_MDIO_DATA.GD. Note: Even if the addressed PHY is not present, there is no change in the functionality of the EMAC_MDIO_ADDR.GB bit. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	GMII Busy is Disabled
		1	GMII Busy is Enabled

MDIO Data Register

The EMAC_MDIO_DATA register stores the write data to be written to the PHY register located at the address specified in EMAC_MDIO_ADDR. The EMAC_MDIO_DATA register also stores the read data from the PHY register located at the address specified by MDIO Address register.

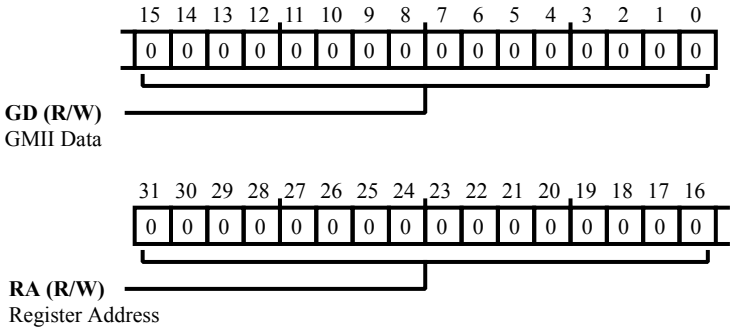


Figure 30-135: EMAC_MDIO_DATA Register Diagram

Table 30-190: EMAC_MDIO_DATA Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RA	Register Address. The EMAC_MDIO_DATA.RA field is valid only when EMAC_MDIO_ADDR.C45E = 1. It contains the register address in the PHY for the intended MDIO frame.
15:0 (R/W)	GD	GMII Data. The EMAC_MDIO_DATA.GD field contains the 16-bit data value read from the PHY after a management read operation or the 16-bit data value to be written to the PHY before a management write operation.

PHY Interface Control and Status Register

The EMAC_PHYIF_CTLSTAT register indicates the status signals received by the RGMII interface (selected at reset) from the PHY. The EMAC_PHYIF_CTLSTAT register is optional.

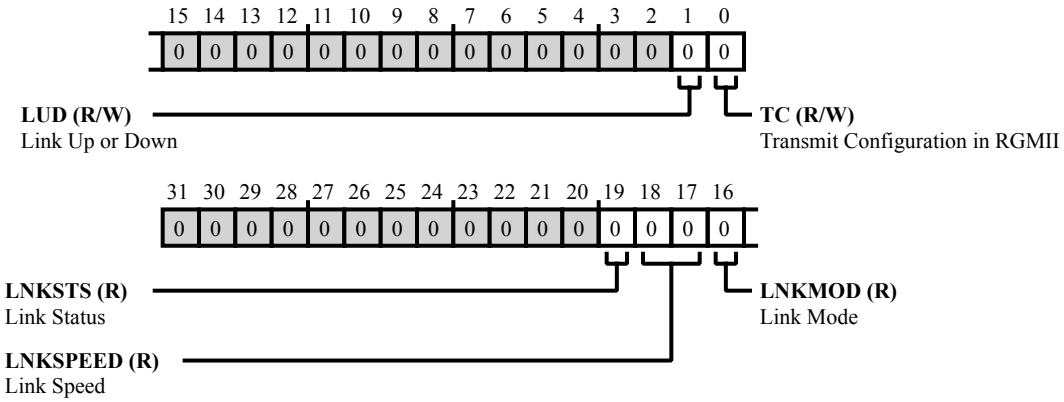


Figure 30-136: EMAC_PHYIF_CTLSTAT Register Diagram

Table 30-191: EMAC_PHYIF_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	LNKSTS	Link Status. The EMAC_PHYIF_CTLSTAT.LNKSTS bit indicates whether the link is up (1'b1) or down (1'b0).	
		0	Link Down
		1	Link up
18:17 (R/NW)	LNKSPEED	Link Speed. The EMAC_PHYIF_CTLSTAT.LNKSPEED bit indicates the current speed of the link.	
		0	2.5 MHz
		1	25 MHz
		2	125 MHz
		3	Reserved
16 (R/NW)	LNKMOD	Link Mode. The EMAC_PHYIF_CTLSTAT.LNKMOD bit indicates the current mode of operation of the link.	
		0	Half-duplex Mode
		1	Full-duplex Mode

Table 30-191: EMAC_PHYIF_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	LUD	Link Up or Down. The `EMAC_PHYIF_CTLSTAT.LUD` bit indicates whether the link is up or down during transmission of configuration in the RGMII, SGMII, or SMII interface.	
		0	Link Down
		1	Link Up
0 (R/W)	TC	Transmit Configuration in RGMII. When set, the `EMAC_PHYIF_CTLSTAT.TC` bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in the RGMII port. When `EMAC_PHYIF_CTLSTAT.TC` = 0, no such information is driven to the PHY.	
		0	Disable Transmit Configuration in RGMII
		1	Enable Transmit Configuration in RGMII

PPS n Interval Register

The EMAC_PPS[n]_INTVL register contains the number of units of sub-second increment value between the rising edges of ETH0_PTPPPSn signal output.

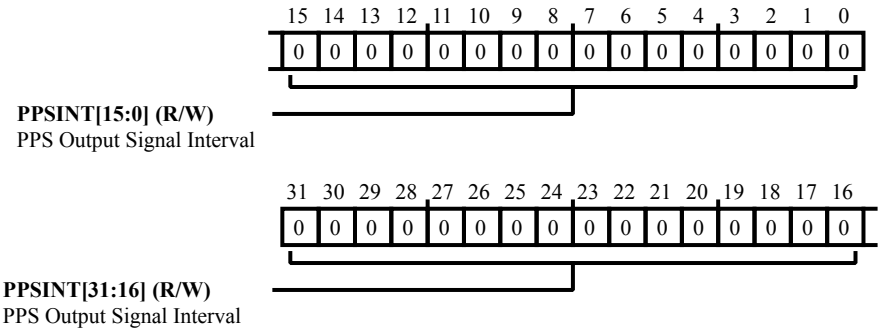


Figure 30-137: EMAC_PPS[n]_INTVL Register Diagram

Table 30-192: EMAC_PPS[n]_INTVL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PPSINT	PPS Output Signal Interval. The EMAC_PPS[n]_INTVL.PPSINT field stores the interval between the rising edges of the ETH0_PTPPPSn signal output. The interval is stored in terms of number of units of sub-second increment value. Program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of the ETH0_PTPPPSn signal output is 100 ns (that is, 5 units of sub-second increment value), program value 4 (5-1) in this register.

PPS n Target Time Nanoseconds Register

PPS0 Target Time Nanoseconds register.

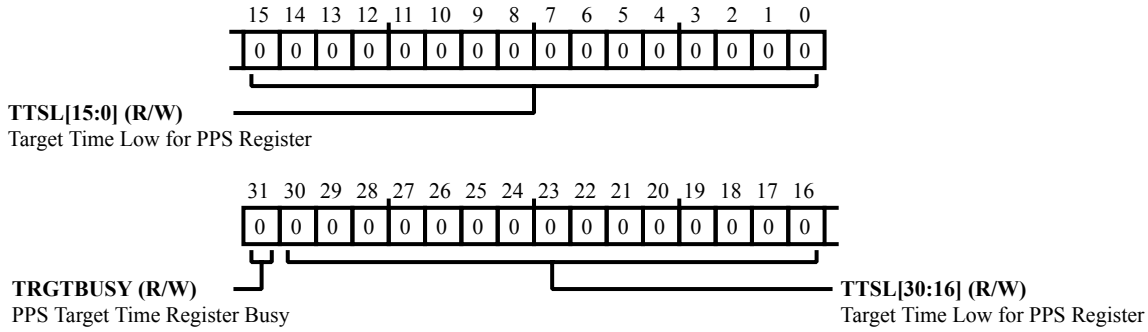


Figure 30-138: EMAC_PPS[n]_TGTM_NSEC Register Diagram

Table 30-193: EMAC_PPS[n]_TGTM_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	TRGTBUSY	PPS Target Time Register Busy. The MAC sets the `EMAC_PPS[n]_TGTM_NSEC.TRGTBUSY` bit when the EMAC_PPS_CTL.PPSCMD0 field is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time registers to the PTP clock domain. The MAC clears the `EMAC_PPS[n]_TGTM_NSEC.TRGTBUSY` bit after synchronizing the Target Time Registers to the PTP clock domain The application must not update the Target Time Registers when the `EMAC_PPS[n]_TGTM_NSEC.TRGTBUSY` bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.	
		0	PPS Target Time Register Busy Status is Not Detected
		1	PPS Target Time Register Busy is Detected
30:0 (R/W)	TTSL	Target Time Low for PPS Register. The `EMAC_PPS[n]_TGTM_NSEC` field stores the time in (signed) nanoseconds. When the value of the time stamp matches the value in both target time stamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in EMAC_PPS_CTL. When EMAC_TM_CTL.TSCTRLSSR = 0, this value should be (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of sub-second increment value. When EMAC_TM_CTL.TSCTRLSSR = 1, this value should not exceed 0x3B9A_C9FF. The actual start or stop time of the PPS signal output might have an error margin up to one unit of sub-second increment value. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	

PPS n Target Time Seconds Register

The EMAC_PPS[n]_TGTM_SEC register, along with the EMAC_PPS[n]_TGTM_NSEC register, is used to schedule an interrupt event (bit 1 of EMAC_TM_STAT) when the system time exceeds the value programmed in these registers.

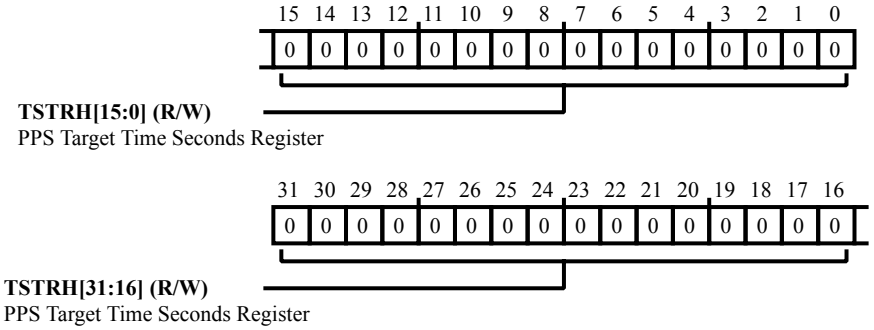


Figure 30-139: EMAC_PPS[n]_TGTM_SEC Register Diagram

Table 30-194: EMAC_PPS[n]_TGTM_SEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSTRH	PPS Target Time Seconds Register. The EMAC_PPS[n]_TGTM_SEC.TSTRH field stores the time in seconds. When the time stamp value matches or exceeds both target time stamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the target time mode selected for the corresponding PPS output in the EMAC_PPS_CTL register. If EMAC_TM_CTL.PTGE = 1 with the presentation time control set in recovery mode, then these bits indicate the TPT being programmed by the application and in generation mode it indicates the CPT generated at the sampled trigger.

PPS n Width Register

The EMAC_PPS[n]_WID register contains the number of units of sub-second increment value between the rising and corresponding falling edges of ETH0_PTPPPSn signal output.

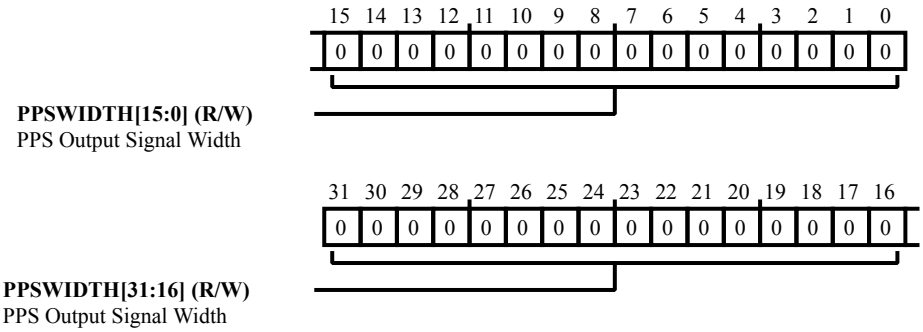


Figure 30-140: EMAC_PPS[n]_WID Register Diagram

Table 30-195: EMAC_PPS[n]_WID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	PPSWIDTH	PPS Output Signal Width. The EMAC_PPS[n]_WID.PPSWIDTH field store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of sub-second increment value. Program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of sub-second increment value), program value 3 (4-1) in this register. Note: The value programmed in this register must be lesser than the value programmed in the EMAC_PPS[n]_INTVL register.

PPS Control Register

Bits[30:24] of the EMAC_PPS_CTL register are valid only when four flexible PPS outputs are selected. Bits[22:16] are valid only when three or more flexible PPS outputs are selected. Bits[14:8] are valid only when two or more flexible PPS outputs are selected. Bits[6:4] are valid only when flexible PPS feature is selected.

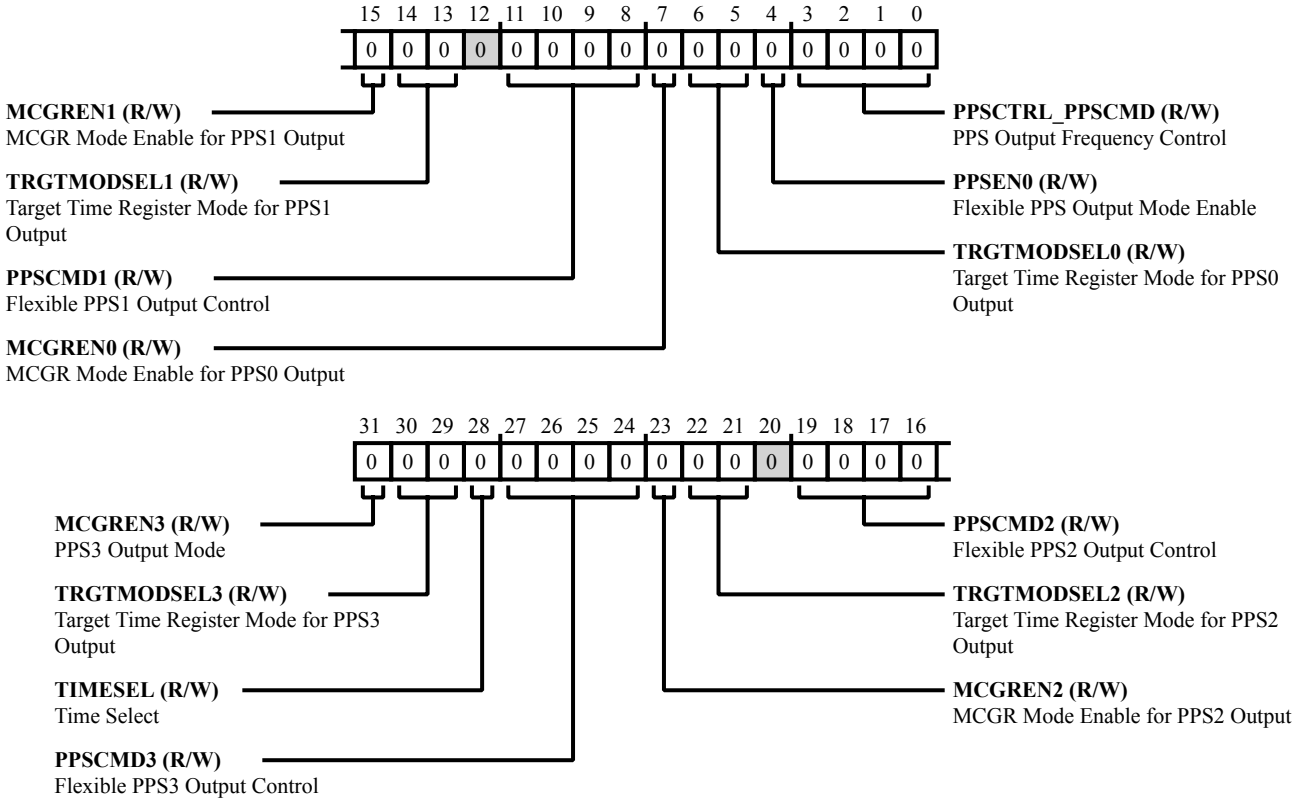


Figure 30-141: EMAC_PPS_CTL Register Diagram

Table 30-196: EMAC_PPS_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	MCGREN3	PPS3 Output Mode. The EMAC_PPS_CTL.MCGREN3 bit indicates whether the 3rd PPS instance operates in PPS or MCGR mode.	
		0	PPS Mode
		1	MCGR Mode

Table 30-196: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
30:29 (R/W)	TRGTMODSEL3	Target Time Register Mode for PPS3 Output. The EMAC_PPS_CTL.TRGTMODSEL3 field indicates the target time registers (EMAC_PPS[3]_TGTM_SEC and EMAC_PPS[3]_TGTM_NSEC) mode for PPS3 output signal.	
		0	Generate the Interrupt Only. Target time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may me observed on the corresponding ETH0_PTPPPS output port.
		1	Enable MCGR Interrupt. The interrupt status bit is indicated by EMAC_TM_STAT.TSTARGT3.
		2	Generate Interrupt and Start/Stop the PPS Output Signal
		3	Start/ Stop the PPS Output Signal Generation. No Interrupt is Asserted
28 (R/W)	TIMESEL	Time Select. When the EMAC_PPS_CTL.TIMESEL bit is set, 64 bit PTP time is used to capture time at MCGR trigger[0] input. When EMAC_PPS_CTL.TIMESEL = 0, presentation time is used to capture time at trigger input, maintaining backward compatibility.	
27:24 (R/W)	PPSCMD3	Flexible PPS3 Output Control. The EMAC_PPS_CTL.PPSCMD3 field controls the flexible PPS3 output (ETH0_PTPPS3) signal. The EMAC_PPS_CTL.PPSCMD3 field is similar to the PPSCMD0[2:0] field. When EMAC_PPS_CTL.MCGREN3= 1, the EMAC_PPS_CTL.PPSCMD3 field indicated by the four bits [27:24] are taken as Presentation Time Control bits for media clock generation and recovery for comparator instance 3. The EMAC_PPS_CTL.PPSCMD3 field is similar to the PPSCMD0 Presentation Time Control bits. When EMAC_PPS_CTL.MCGREN3= 0, only 3 bits from [26:24] are used as PPSCMD3 and the fourth bit must be cleared (=0). Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
23 (R/W)	MCGREN2	MCGR Mode Enable for PPS2 Output. The EMAC_PPS_CTL.MCGREN2 field enables the 2nd PPS instance to operate in PPS or MCGR mode.	
		0	PPS Mode
		1	MCGR Mode

Table 30-196: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
22:21 (R/W)	TRGTMODSEL2	Target Time Register Mode for PPS2 Output. The EMAC_PPS_CTL.TRGTMODSEL2 field indicates the target time registers (EMAC_PPS[2]_TGTM_SEC and EMAC_PPS[2]_TGTM_NSEC) mode for PPS2 output signal.	
		0	Generate the Interrupt Event. Target time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may me observed on the corresponding ETH0_PTPPPS output port.
		1	Enable MCGR Interrupt. The interrupt status bit is indicated by TSTARGT3.
		2	Generate Interrupt and Start/Stop the PPS Output Signal
		3	Start/ Stop the PPS Output Signal Generation. No Interrupt is Asserted
19:16 (R/W)	PPSCMD2	Flexible PPS2 Output Control. The EMAC_PPS_CTL.PPSCMD2 field controls the flexible PPS2 output (ETH0_PTPPS2) signal. The EMAC_PPS_CTL.PPSCMD2 field is similar to the PPSCMD0 field. When EMAC_PPS_CTL.MCGREN2= 1, the EMAC_PPS_CTL.PPSCMD2 field indicated by the four bits, [19:16] are taken as Presentation Time Control bits for media clock generation and recovery for comparator instance 2. The EMAC_PPS_CTL.PPSCMD2 field is similar to the PPSCMD0 Presentation Time Control bits. When EMAC_PPS_CTL.MCGREN2= 0, only 3 bits, [18:16] are used as PPSCMD2 and the fourth bit must be configured as 0. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
15 (R/W)	MCGREN1	MCGR Mode Enable for PPS1 Output. The EMAC_PPS_CTL.MCGREN1 field enables the 1st PPS instance to operate in PPS or MCGR mode.	
		0	PPS Mode
		1	MCGR Mode

Table 30-196: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
14:13 (R/W)	TRGTMODSEL1	Target Time Register Mode for PPS1 Output. The EMAC_PPS_CTL.TRGTMODSEL1 field indicates the target time registers (EMAC_PPS[1]_TGTM_SEC and EMAC_PPS[1]_TGTM_NSEC) mode for PPS1 output signal.	
		0	Generate the Interrupt Event. Target time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may me observed on the corresponding ETH0_PTPPPS output port.
		1	Enable MCGR Interrupt. The interrupt status bit is indicated by TSTARGT3.
		2	Generate Interrupt and Start/Stop the PPS Output Signal
		3	Start/ Stop the PPS Output Signal Generation. No Interrupt is Asserted
11:8 (R/W)	PPSCMD1	Flexible PPS1 Output Control. The EMAC_PPS_CTL.PPSCMD1 field controls the flexible PPS1 output (ETH0_PTPPS1) signal. The EMAC_PPS_CTL.PPSCMD1 field is similar to the PPSCMD0 field. When EMAC_PPS_CTL.MCGREN1= 1, PPSCMD1 indicated by the four bits [11:8] are taken as Presentation Time Control bits for media clock generation and recovery for comparator instance 1. The EMAC_PPS_CTL.PPSCMD1 field is similar to the PPSCMD0 Presentation Time Control bits. When EMAC_PPS_CTL.MCGREN1= 0, only 3 bits, [10:8] are used as PPSCMD1 and the fourth bit must be cleared (=0). Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
7 (R/W)	MCGREN0	MCGR Mode Enable for PPS0 Output. The EMAC_PPS_CTL.MCGREN0 field enables the 0th PPS instance to operate in PPS or MCGR mode. When set, it operates in MCGR mode and when cleared, it operates in PPS mode.	
		0	PPS Mode
		1	MCGR Mode

Table 30-196: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:5 (R/W)	TRGTMODSEL0	Target Time Register Mode for PPS0 Output. The `EMAC_PPS_CTL.TRGTMODSEL0` field indicates the target time registers (EMAC_PPS[0]_TGTM_SEC and EMAC_PPS[0]_TGTM_NSEC) mode for PPS0 output signal.	
		0	Generate the Interrupt Event. The target time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may be observed on the corresponding ETH0_PTPPPS output port.
		1	Enable MCGR Interrupt. The status bit is indicated by TSTARGT.
		2	Generate the Interrupt Event and PPS Output Signal
		3	Start/ Stop the PPS Output Signal Generation. No Interrupt is Asserted
4 (R/W)	PPSEN0	Flexible PPS Output Mode Enable. When the `EMAC_PPS_CTL.PPSEN0` bit is set (=1), bits[3:0] function as PPSCMD. When the `EMAC_PPS_CTL.PPSEN0` bit is cleared (=0), bits[3:0] function as PPSCTRL (fixed PPS mode).	
		0	Flexible PPS Output Mode is Disabled
		1	Flexible PPS Output Mode is Enabled

Table 30-196: EMAC_PPS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	PPSCTRL_PPSCMD	PPS Output Frequency Control. The EMAC_PPS_CTL.PPSCTRL_PPSCMD field controls the frequency of the ETH0_PTPPPS0 output (PTP clock) signal. The default value is 0000, and the PPS output is 1 pulse (of width PTP clock) every second. For other values of EMAC_PPS_CTL.PPSCTRL_PPSCMD, the PPS output becomes a generated clock of following frequencies: - 0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz. - 0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz. - 0011: The binary rollover is 8 Hz; digital rollover is 4 Hz. - 0100: The binary rollover is 16 Hz; digital rollover is 8 Hz. ... - 1111: The binary rollover is 32.768 KHz; digital rollover is 16.384 KHz. See the Fixed Pulse-Per-Second Output topic for details. When EMAC_PPS_CTL.PPSCTRL_PPSCMD = 0011, the PPS (4 Hz) is a sequence of: Three clocks of 50 percent duty cycle and 268 ms period. A fourth clock of 195 ms period (134 ms low and 61 ms high). This behavior is due to the non-linear toggling of bits in the digital rollover mode in the EMAC_TM_NSEC register. For Flexible PPS Output (ETH0_PTPPPS0) control, programming the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, the bits are cleared automatically. Software should ensure that these bits are programmed only when they are 'all-zero'. When EMAC_PPS_CTL.MCGREN0 = 1, the EMAC_PPS_CTL.PPSCTRL_PPSCMD bits are treated as presentation time control bits: - 0000: MCGR operation is not carried out. If configured to this value in the middle of clock recovery or generation, all processing inputs are flushed. - 0001: Capture the presentation time at the rising edge of the EMAC0_MCG_PST trigger request into the EMAC_PPS[0]_TGTM_SEC register. - 0010: Capture the presentation time at the falling edge of the EMAC0_MCG_PST trigger request into the EMAC_PPS[0]_TGTM_SEC register. - 0011: Capture the presentation time at both edges of the EMAC0_MCG_PST trigger request into the EMAC_PPS[0]_TGTM_SEC register. - 0100-1000: Reserved - 1001: Toggle output on compare - 1010: Pulse output low on compare for one PTP clock cycle. - 1011: Pulse output high on compare for one PTP clock cycle. - 1100-1111: Reserved

PTP Time Stamp Offload Control Register

The EMAC_PTO_CTL register controls the PTP offload engine operation.

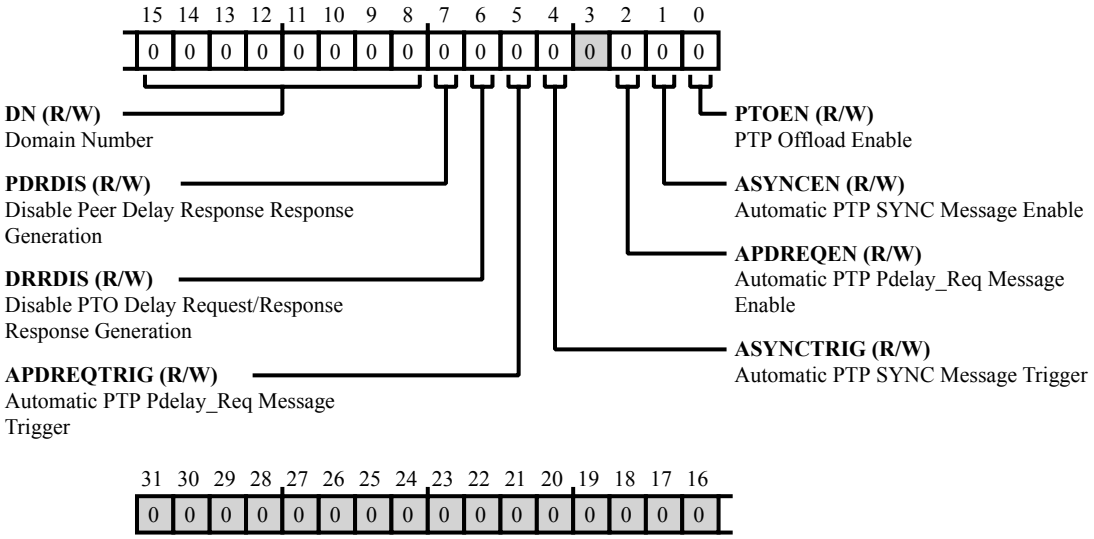


Figure 30-142: EMAC_PTO_CTL Register Diagram

Table 30-197: EMAC_PTO_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15:8 (R/W)	DN	Domain Number. The EMAC_PTO_CTL.DN field indicates the domain number in which the PTP node is operating.	
7 (R/W)	PDRDIS	Disable Peer Delay Response Response Generation. When the EMAC_PTO_CTL.PDRDIS bit is set, the peer delay response (Pdelay_Resp) response is not be generated for received peer delay request (Pdelay_Req) request packet, as required by the programmed mode. Note: Setting the EMAC_PTO_CTL.PDRDIS bit to 1 affects the normal PTP Offload operation and the time synchronization. So, the EMAC_PTO_CTL.PDRDIS bit must be set only if there is problem with Pdelay_Resp generation in hardware and/or Pdelay_Resp generation is handled by software.	
		0	Peer Delay Response Response Generation is Enabled
		1	Peer Delay Response Response Generation is Disabled

Table 30-197: EMAC_PTO_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	DRRDIS	Disable PTO Delay Request/Response Response Generation. When the EMAC_PTO_CTL.DRRDIS bit is set, the delay request and delay response is not generated for received SYNC and delay request packet respectively, as required by the programmed mode.	
		0	PTO Delay Request/Response Response Generation is Enabled
		1	PTO Delay Request/Response Response Generation is Disabled
5 (R/W)	APDREQTRIG	Automatic PTP Pdelay_Req Message Trigger. When the EMAC_PTO_CTL.APDREQTRIG bit is set, one PTP Pdelay_Req message is transmitted. The EMAC_PTO_CTL.APDREQTRIG bit is automatically cleared after the PTP Pdelay_Req message is transmitted. The application should set the EMAC_PTO_CTL.APDREQEN bit for this operation. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Automatic PTP Pdelay_Req Message Trigger is Disabled
		1	Automatic PTP Pdelay_Req Message Trigger is Enabled
4 (R/W)	ASYNCTRIG	Automatic PTP SYNC Message Trigger. When the EMAC_PTO_CTL.ASYNCTRIG bit is set, one PTP SYNC message is transmitted. The EMAC_PTO_CTL.ASYNCTRIG bit is automatically cleared after the PTP SYNC message is transmitted. The application should set the EMAC_PTO_CTL.ASYNCEN bit for this operation. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Automatic PTP SYNC Message Trigger is Disabled
		1	Automatic PTP SYNC Message Trigger is Enabled
2 (R/W)	APDREQEN	Automatic PTP Pdelay_Req Message Enable. When the EMAC_PTO_CTL.APDREQEN bit is set, PTP Pdelay_Req message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in peer-to-peer transparent mode.	
		0	Automatic PTP Pdelay_Req Message is Disabled
		1	Automatic PTP Pdelay_Req Message is Enabled
1 (R/W)	ASYNCEN	Automatic PTP SYNC Message Enable. When the EMAC_PTO_CTL.ASYNCEN bit is set, PTP SYNC message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in clock requester mode.	
		0	Automatic PTP SYNC Message is Disabled
		1	Automatic PTP SYNC Message is Enabled

Table 30-197: EMAC_PTO_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	PTOEN	PTP Offload Enable. When the `EMAC_PTO_CTL.PTOEN` bit is set, the PTP offload feature is enabled.	
		0	PTP Offload Feature is Disabled
		1	PTP Offload Feature is Enabled

MAC Packet Filter Register

The EMAC_MACPKT_FILT register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

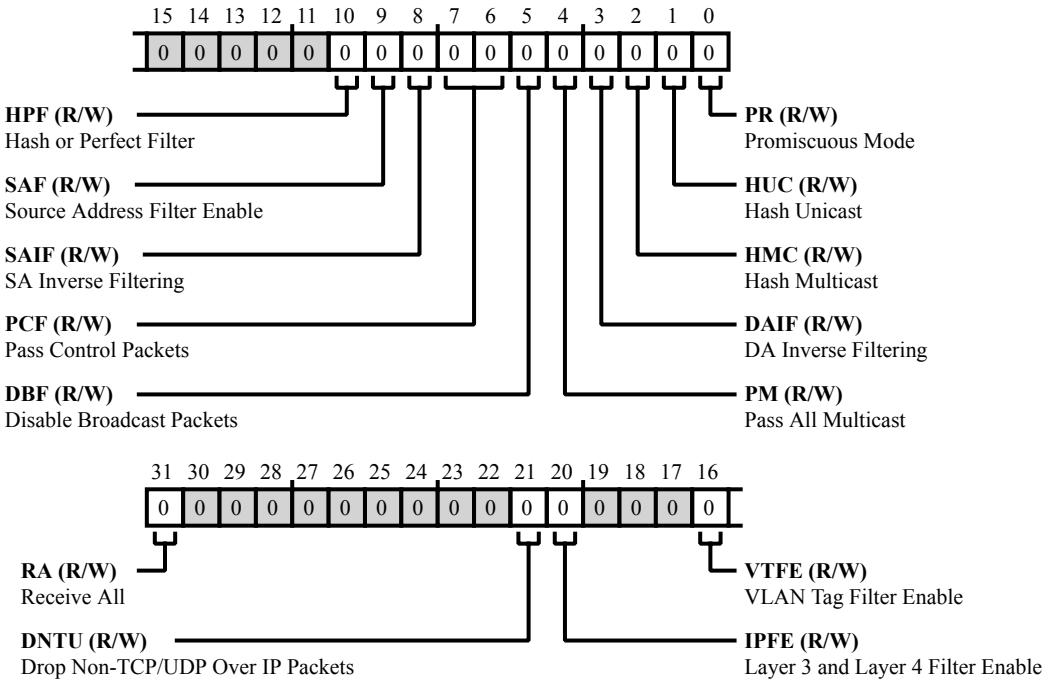


Figure 30-143: EMAC_MACPKT_FILT Register Diagram

Table 30-198: EMAC_MACPKT_FILT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	RA	Receive All. When the EMAC_MACPKT_FILT.RA bit is set, the MAC receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx status word. When EMAC_MACPKT_FILT.RA = 0, the receiver module passes only those packets to the application that pass the SA or DA address filter.	
		0	Receive All is Disabled
		1	Receive All is Enabled

Table 30-198: EMAC_MACPKT_FILT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	DNTU	Drop Non-TCP/UDP Over IP Packets. When the EMAC_MACPKT_FILT.DNTU bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the layer 4 filter. When EMAC_MACPKT_FILT.DNTU = 0, the MAC forwards all non-TCP or UDP over IP packets.	
		0	Forward Non-TCP/UDP Over IP Packets
		1	Drop Non-TCP/UDP Over IP Packets
20 (R/W)	IPFE	Layer 3 and Layer 4 Filter Enable. When the EMAC_MACPKT_FILT.IPFE bit is set, the MAC drops packets that do not match the enabled layer 3 and layer 4 filters. If layer 3 or layer 4 filters are not enabled for matching, the EMAC_MACPKT_FILT.IPFE bit does not have any effect. When EMAC_MACPKT_FILT.IPFE = 0, the MAC forwards all packets irrespective of the match status of the layer 3 and layer 4 fields.	
		0	Layer 3 and Layer 4 Filters are Disabled
		1	Layer 3 and Layer 4 Filters are Enabled
16 (R/W)	VTFE	VLAN Tag Filter Enable. When the EMAC_MACPKT_FILT.VTFE bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN tag. When EMAC_MACPKT_FILT.VTFE = 0, the MAC forwards all packets irrespective of the match status of the VLAN tag.	
		0	VLAN Tag Filter is Disabled
		1	VLAN Tag Filter is Enabled
10 (R/W)	HPF	Hash or Perfect Filter. When the EMAC_MACPKT_FILT.HPF bit is set, the address filter passes a packet if it matches either the perfect filtering or hash filtering as configured by the EMAC_MACPKT_FILT.HUC or EMAC_MACPKT_FILT.HMC bit. When EMAC_MACPKT_FILT.HPF = 0 and the EMAC_MACPKT_FILT.HUC or EMAC_MACPKT_FILT.HMC bit is set, the packet is passed only if it matches the hash filter.	
		0	Hash or Perfect Filter is Disabled
		1	Hash or Perfect Filter is Enabled

Table 30-198: EMAC_MACPKT_FILT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	SAF	Source Address Filter Enable. When the EMAC_MACPKT_FILT.SAF bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet. When EMAC_MACPKT_FILT.SAF = 0, the MAC forwards the received packet to the application with updated SAF bit of the Rx status depending on the SA address comparison. Note: According to the IEEE specification, bit 47 of the SA is reserved. However, in the EMAC module, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.	
		0	SA Filtering is Disabled
		1	SA Filtering is Enabled
8 (R/W)	SAIF	SA Inverse Filtering. When the EMAC_MACPKT_FILT.SAIF bit is set, the address check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA address filter. When EMAC_MACPKT_FILT.SAIF = 0, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA address filter.	
		0	SA Inverse Filtering is Disabled
		1	SA Inverse Filtering is Enabled
7:6 (R/W)	PCF	Pass Control Packets. The EMAC_MACPKT_FILT.PCF field control the forwarding of all control packets (including unicast and multicast pause packets).	
		0	MAC filters all control packets from reaching the application
		1	MAC forwards all control packets except pause packets to the application even if they fail the address filter
		2	MAC forwards all control packets to the application even if they fail the address filter
		3	MAC forwards the control packets that pass the address filter

Table 30-198: EMAC_MACPKT_FILT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	DBF	Disable Broadcast Packets. When the EMAC_MACPKT_FILT.DBF bit is set, the AFM module blocks all the incoming broadcast packets. In addition, it overrides all other filter settings. When EMAC_MACPKT_FILT.DBF = 0, the AFM module passes all received broadcast packets.	
		0	Enable Broadcast Packets
		1	Disable Broadcast Packets
4 (R/W)	PM	Pass All Multicast. When the EMAC_MACPKT_FILT.PM bit is set, it indicates that all the received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When EMAC_MACPKT_FILT.PM = 0, filtering of multicast packet depends on the EMAC_MACPKT_FILT.HMC bit.	
		0	Pass All Multicast is Disabled
		1	Pass All Multicast is Enabled
3 (R/W)	DAIF	DA Inverse Filtering. When the EMAC_MACPKT_FILT.DAIF bit is set, the address check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When EMAC_MACPKT_FILT.DAIF = 0, normal filtering of packets is performed.	
		0	DA Inverse Filtering is Disabled
		1	DA Inverse Filtering is Enabled
2 (R/W)	HMC	Hash Multicast. When the EMAC_MACPKT_FILT.HMC bit is set, the MAC performs the destination address filtering of received multicast packets according to the hash table. When EMAC_MACPKT_FILT.HMC = 0, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.	
		0	Hash Multicast is Disabled
		1	Hash Multicast is Enabled

Table 30-198: EMAC_MACPKT_FILT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	HUC	Hash Unicast. When the EMAC_MACPKT_FILT.HUC bit is set, the MAC performs the destination address filtering of unicast packets according to the hash table. When EMAC_MACPKT_FILT.HUC = 0, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.	
		0	Hash Unicast is Disabled
		1	Hash Unicast is Enabled
0 (R/W)	PR	Promiscuous Mode. When the EMAC_MACPKT_FILT.PR bit is set, the address filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA filter fails status bits of the Rx status word are always cleared when EMAC_MACPKT_FILT.PR is set.	
		0	Promiscuous Mode is Disabled
		1	Promiscuous Mode is Enabled

Presentation Time Update Register

The EMAC_PRSNTM_UPDT register holds the 32-bit value of MAC 1722 Presentation Time in ns, that should be added to the Current Presentation Time Counter value. Init happens when EMAC_TM_CTL.TSINIT = 1, and update happens when EMAC_TM_CTL.TSUPDT = 1.

The EMAC_PRSNTM_UPDT register exists when DWC_EQOS_FLEXI_PPS_OUT_EN is configured.

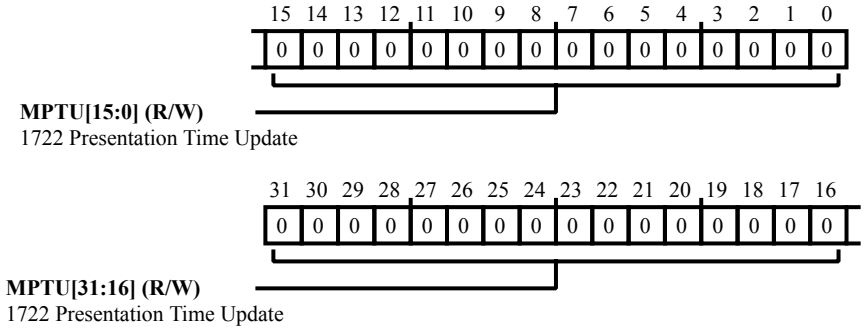


Figure 30-144: EMAC_PRSNTM_UPDT Register Diagram

Table 30-199: EMAC_PRSNTM_UPDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MPTU	1722 Presentation Time Update. The EMAC_PRSNTM_UPDT.MPTU field holds the init value or the update value for the presentation time. When used for update, this field holds the 32-bit value in ns, that should be added to the Current Presentation Time Counter value. When EMAC_TM_NSECUPDT.ADDSUB = 1, this value is directly used for subtraction.

Presentation Time Nanosecond Register

The EMAC_PRSNTM_NSEC register contains the 32-bit binary rollover equivalent time of the PTP system time in ns.

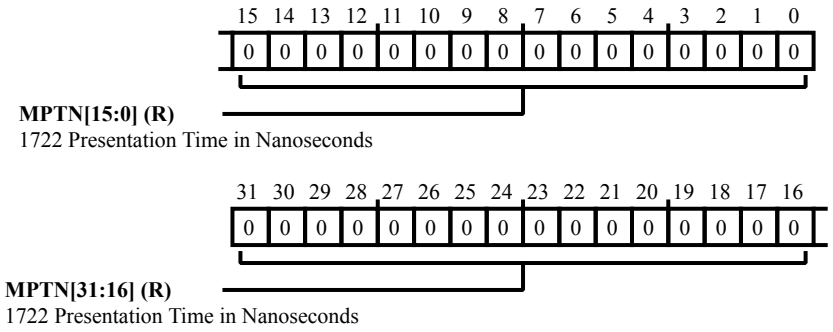


Figure 30-145: EMAC_PRSNTM_NSEC Register Diagram

Table 30-200: EMAC_PRSNTM_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	MPTN	1722 Presentation Time in Nanoseconds. The EMAC_PRSNTM_NSEC.MPTN field indicates the value of the 32-bit binary rollover equivalent time of the PTP system time in nanoseconds.

Queue 0 Transmit Flow Control Register

The EMAC_Q0_TXFLOW_CTL register controls the generation and reception of the Control (Pause Command) packets by the flow control module of the MAC. A write to a register with the Busy bit set (=1) triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

When the PFCE bit in the MAC_Rx_Flow_Ctrl register is enabled, this register controls the generation of Priority Flow Control (PFC) frames with priorities mapped according to PSRQ0 in the MAC_RxQ_Ctrl2 register.

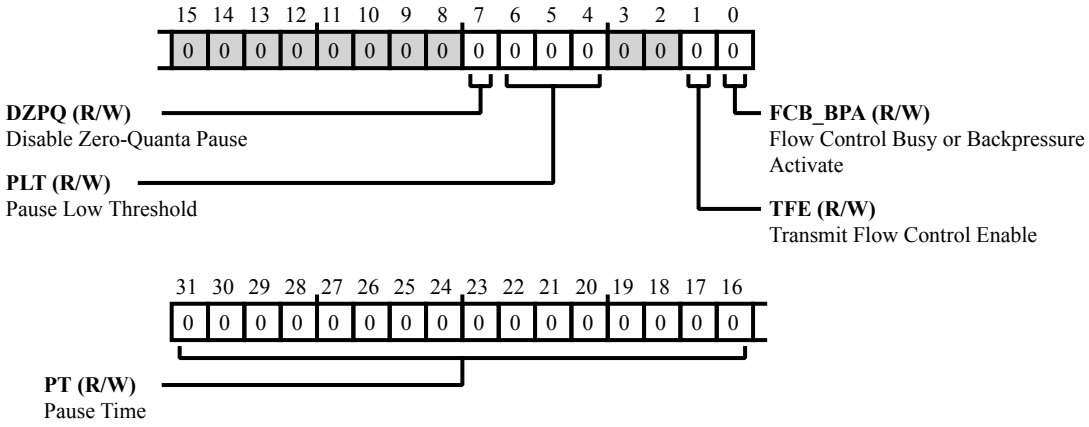


Figure 30-146: EMAC_Q0_TXFLOW_CTL Register Diagram

Table 30-201: EMAC_Q0_TXFLOW_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	PT	Pause Time. The EMAC_Q0_TXFLOW_CTL.PT field holds the value to be used in the Pause Time field in the Tx control packet. If the Pause Time bits are configured to be double-synchronized to the (G)MII clock domain, consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.

Table 30-201: EMAC_Q0_TXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	DZPQ	Disable Zero-Quanta Pause. When the `EMAC_Q0_TXFLOW_CTL.DZPQ` bit is set, it disables the automatic generation of the zero-quanta Pause packets on deassertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i or mti_flowctrl_i). When `EMAC_Q0_TXFLOW_CTL.DZPQ` = 0, normal operation with automatic zero-quanta Pause packet generation is enabled.	
		0	Zero-Quanta Pause Packet Generation is Enabled
		1	Zero-Quanta Pause Packet Generation is Disabled
6:4 (R/W)	PLT	Pause Low Threshold. The `EMAC_Q0_TXFLOW_CTL.PLT` field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of the Pause packet. The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first Pause packet is transmitted. The following list provides the threshold values for different values. The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.	
		0	Pause Time Minus 4 Slot Times
		1	Pause Time Minus 28 Slot Times
		2	Pause Time Minus 36 Slot Times
		3	Pause Time Minus 144 Slot Times
		4	Pause Time Minus 256 Slot Times
		5	Pause Time Minus 512 Slot Times
		6	Reserved

Table 30-201: EMAC_Q0_TXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	TFE	Transmit Flow Control Enable. In the full-duplex mode, when the EMAC_Q0_TXFLOW_CTL.TFE bit is set, the MAC enables the flow control operation to Tx Pause packets. When EMAC_Q0_TXFLOW_CTL.TFE = 0, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets. In the half-duplex mode, when the EMAC_Q0_TXFLOW_CTL.TFE bit is set, the MAC enables the back pressure operation. When EMAC_Q0_TXFLOW_CTL.TFE = 0, the back pressure feature is disabled.	
		0	Transmit Flow Control is Disabled
		1	Transmit Flow Control is Enabled
0 (R/W)	FCB_BPA	Flow Control Busy or Backpressure Activate. The EMAC_Q0_TXFLOW_CTL.FCB_BPA bit initiates a Pause packet in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set. In full-duplex mode, the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit should be read as 1'b0 before writing to this register. To initiate a Pause packet, the application must set the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit to 1'b1. During Control packet transfer, the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit to 1'b0. Do not write to this register until the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit is cleared. When the EMAC_Q0_TXFLOW_CTL.FCB_BPA bit is set (and TFE bit is set) in the half-duplex mode, the MAC asserts the back pressure. During back pressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically OR'ed with the mti_flowctrl_i input signal for the back pressure function. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Flow Control Busy or Backpressure Activate is Disabled
		1	Flow Control Busy or Backpressure Activate is Enabled

Queue [i] Transmit Flow Control Register

The EMAC_Q[i]_TXFLOW_CTL register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A write to a register with the Busy bit set (= 1) triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register. (EMAC_RXFLOW_CTL.PFCE = 1.)

The register controls the generation of PFC control packets of priorities mapped per the PSRQn field in the EMAC_RXQ_CTL2.EMAC_RXQ_CTL3 registers.

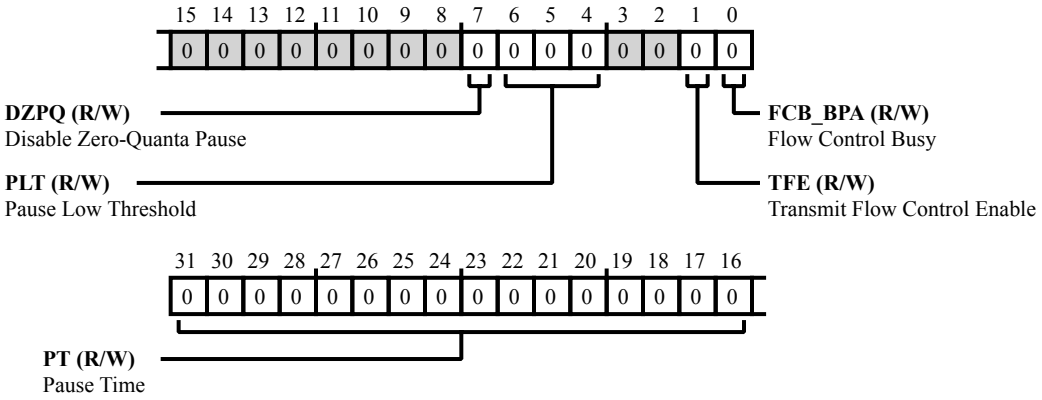


Figure 30-147: EMAC_Q[i]_TXFLOW_CTL Register Diagram

Table 30-202: EMAC_Q[i]_TXFLOW_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16 (R/W)	PT	Pause Time. The EMAC_Q[i]_TXFLOW_CTL.PT field holds the value to be used in the Pause Time field in the Tx control packet. The bits are configured to be double-synchronized to the (G)MII clock domain. Hence, consecutive writes to this register should be performed only after at least four clock cycles in the (G)MII clock domain.	
7 (R/W)	DZPQ	Disable Zero-Quanta Pause. When the EMAC_Q[i]_TXFLOW_CTL.DZPQ bit is set, it disables the automatic generation of the zero-quanta Pause packets on deassertion of the flow-control signal from the FIFO layer. When EMAC_Q[i]_TXFLOW_CTL.DZPQ = 0, normal operation with automatic zero-quanta Pause packet generation is enabled.	
		0	Zero-Quanta Pause Packet Generation is Enabled
		1	Zero-Quanta Pause Packet Generation is Disabled

Table 30-202: EMAC_Q[i]_TXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:4 (R/W)	PLT	Pause Low Threshold. The `EMAC_Q[i]_TXFLOW_CTL.PLT` field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i is checked for automatic retransmission of the Pause packet. The threshold values must be less than the Pause Time (PT) configured in bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first pause packet is transmitted. The following list provides the threshold values for different values. The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.	
		0	PT - 4 Slot Times
		1	PT - 28 Slot Times
		2	PT - 36 Slot Times
		3	PT - 144 Slot Times
		4	PT - 256 Slot Times
		5	PT - 512 Slot Times
		6	Reserved
1 (R/W)	TFE	Transmit Flow Control Enable. When the `EMAC_Q[i]_TXFLOW_CTL.TFE` bit is set in full-duplex mode, the MAC enables the flow control operation to Tx pause packets. When `EMAC_Q[i]_TXFLOW_CTL.TFE` = 0, the flow control operation in the MAC is disabled, and the MAC does not transmit any pause packets.	
		0	Transmit Flow Control is Disabled
		1	Transmit Flow Control is Enabled

Table 30-202: EMAC_Q[i]_TXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	FCB_BPA	Flow Control Busy. The EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit initiates a PFC packet if EMAC_Q[i]_TXFLOW_CTL.TFE = 1. To initiate a PFC packet, the application must set the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit to 1'b1. During the control packet transfer, the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit continues to be set to indicate that a packet transmission is in progress. When PFC packet transmission is complete, the MAC resets the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit to 1'b0. Do not write to this register until the EMAC_Q[i]_TXFLOW_CTL.FCB_BPA bit is cleared. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Flow Control Busy or Backpressure Activate is Disabled
		1	Flow Control Busy or Backpressure Activate is Enabled

Receive Queue Control 0 Register

The EMAC_RXQ_CTL0 register controls the queue management in the MAC receiver.

Note: In multiple Rx queues configuration, all the queues are disabled by default. Enable the Rx queue by programming the corresponding field in this register.

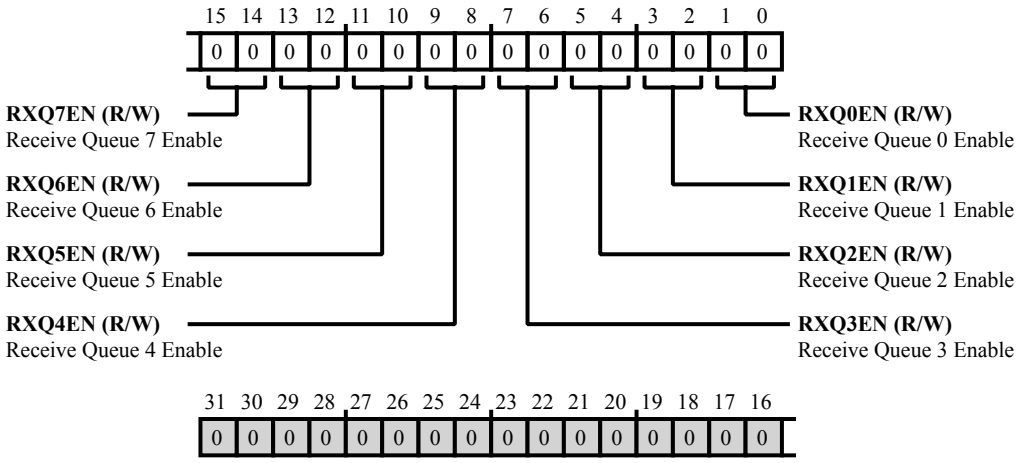


Figure 30-148: EMAC_RXQ_CTL0 Register Diagram

Table 30-203: EMAC_RXQ_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
15:14 (R/W)	RXQ7EN	Receive Queue 7 Enable. The EMAC_RXQ_CTL0.RXQ7EN field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
13:12 (R/W)	RXQ6EN	Receive Queue 6 Enable. The EMAC_RXQ_CTL0.RXQ6EN field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved

Table 30-203: EMAC_RXQ_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:10 (R/W)	RXQ5EN	Receive Queue 5 Enable. The `EMAC_RXQ_CTL0.RXQ5EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
9:8 (R/W)	RXQ4EN	Receive Queue 4 Enable. The `EMAC_RXQ_CTL0.RXQ4EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
7:6 (R/W)	RXQ3EN	Receive Queue 3 Enable. The `EMAC_RXQ_CTL0.RXQ3EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
5:4 (R/W)	RXQ2EN	Receive Queue 2 Enable. The `EMAC_RXQ_CTL0.RXQ2EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved
3:2 (R/W)	RXQ1EN	Receive Queue 1 Enable. The `EMAC_RXQ_CTL0.RXQ1EN` field is similar to the RXQ0EN field.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved

Table 30-203: EMAC_RXQ_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (R/W)	RXQ0EN	Receive Queue 0 Enable. The `EMAC_RXQ_CTL0.RXQ0EN` field indicates whether Rx queue 0 is enabled for AV or DCB.	
		0	Queue Not Enabled
		1	Queue Enabled for AV
		2	Queue Enabled for DCB/Generic
		3	Reserved

Receive Queue Control 1 Register

The EMAC_RXQ_CTL1 register controls the routing of multicast, broadcast, AV, DCB, and untagged packets to the Rx queues.

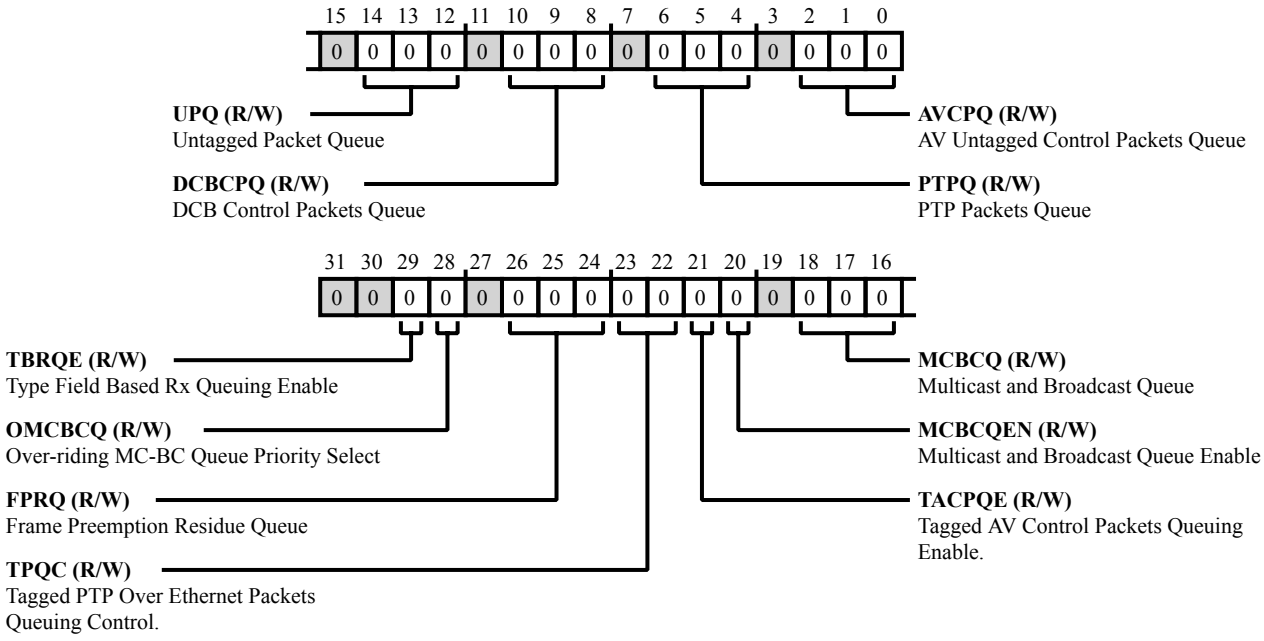


Figure 30-149: EMAC_RXQ_CTL1 Register Diagram

Table 30-204: EMAC_RXQ_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	TBRQE	Type Field Based Rx Queuing Enable. When the EMAC_RXQ_CTL1.TBRQE bit is set, it enables the Type field based Rx queuing where the Type field of received packet is compared with programmed TYP field in MAC_TMRQ_Regs(n) and if a match occurs the packet is routed to the corresponding TMRQ field.	
28 (R/W)	OMCBCQ	Over-riding MC-BC Queue Priority Select. When set (=1), the EMAC_RXQ_CTL1.OMCBCQ bit indicates that the priority of MCBCQ is reduced and the received packet is first routed to PTPQ, AVCPQ, DCBCPQ depending on packet type. When cleared (=0), the EMAC_RXQ_CTL1.OMCBCQ bit indicates that the received multicast/broadcast packet is routed to MCBCQ.	
		0	Overriding MCBCQ Priority Disabled
		1	Overriding MCBCQ Priority Enabled

Table 30-204: EMAC_RXQ_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26:24 (R/W)	FPRQ	Frame Preemption Residue Queue. The EMAC_RXQ_CTL1.FPRQ field holds the Rx queue number to which the residual preemption frames must be forwarded. Preemption frames that are tagged and pass the SA/DA/VLAN filtering are routed based on PSRQ and all other frames are treated as residual frames and is routed to the queue number mentioned in this field. The queue 0 is used as a default queue for express frames, so this field cannot be programmed to a value 0.	
23:22 (R/W)	TPQC	Tagged PTP Over Ethernet Packets Queuing Control.. The EMAC_RXQ_CTL1.TPQC field controls the routing of the VLAN Tagged PTPoE packets. The following programmable options are allowed: - 2'b00: VLAN Tagged PTPoE packets are routed as generic VLAN Tagged packet (based on PSRQ for only non-AV enabled Rx queue). - 2'b01: VLAN Tagged PTPoE packets are routed to Rx queue specified by PTPQ field (Rx queue can be enabled for AV or non-AV traffic). - 2'b10: VLAN Tagged PTPoE packets are routed to only AV enabled Rx queues based on PSRQ. - 2'b11: Reserved If DWC_EQOS_AV_ENABLE is not selected in the configuration, the following programmable options are allowed. - 1'b0: VLAN Tagged PTPoE packets are routed as generic VLAN Tagged packet (based on PSRQ for DCB/Generic enabled Rx queues). - 1'b1: VLAN Tagged PTPoE packets are routed to Rx queue specified by PTPQ field.	
21 (R/W)	TACPQE	Tagged AV Control Packets Queuing Enable.. When the EMAC_RXQ_CTL1.TACPQE bit is set, the MAC routes the received Tagged AV Control packets to the Rx queue specified by AVCPQ field. When EMAC_RXQ_CTL1.TACPQE = 0, the MAC routes the received Tagged AV Control packets based on the tag priority matching the PSRQn fields in EMAC_RXQ_CTL2 and EMAC_RXQ_CTL3 registers.	
		0	Tagged AV Control Packets Queuing is Disabled
		1	Tagged AV Control Packets Queuing is Enabled
20 (R/W)	MCBCQEN	Multicast and Broadcast Queue Enable. The EMAC_RXQ_CTL1.MCBCQEN bit specifies that multicast or broadcast packets routing to the Rx queue is enabled and the multicast or broadcast packets must be routed to Rx queue specified in the EMAC_RXQ_CTL1.MCBCQ field.	
		0	Multicast and Broadcast Queue is Disabled
		1	Multicast and Broadcast Queue is Enabled

Table 30-204: EMAC_RXQ_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18:16 (R/W)	MCBCQ	Multicast and Broadcast Queue. The `EMAC_RXQ_CTL1.MCBCQ` field specifies the Rx queue onto which Multicast or Broadcast Packets are routed. Any Rx queue enabled for Generic/DCB/AV traffic can be used to route the Multicast or Broadcast Packets.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7
14:12 (R/W)	UPQ	Untagged Packet Queue. The `EMAC_RXQ_CTL1.UPQ` field indicates the Rx queue to which untagged packets are to be routed. Any Rx queue enabled for generic/DCB/AV traffic can be used to route the untagged packets.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7
10:8 (R/W)	DCBCPQ	DCB Control Packets Queue. The `EMAC_RXQ_CTL1.DCBCPQ` field specifies the Rx queue on which the received DCB control packets are routed. The DCB data packets are routed based on the EMAC_Q[n]_TXFLOW_CTL.PRSQ field of corresponding queue.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4

Table 30-204: EMAC_RXQ_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7
6:4 (R/W)	PTPQ	PTP Packets Queue. The `EMAC_RXQ_CTL1.PTPQ` field specifies the Rx queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed. When EMAC_TM_CTL.AV8021ASMEN = 1, only untagged PTP over Ethernet packets are routed on an Rx queue. If the bit is not set, then based on programming of the `EMAC_RXQ_CTL1.TPQC` field, both tagged and untagged PTPoE packets can be routed to this Rx queue.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7
2:0 (R/W)	AVCPQ	AV Untagged Control Packets Queue. The `EMAC_RXQ_CTL1.AVCPQ` field specifies the receive queue on which the received AV tagged and untagged control packets are routed. The AV tagged (when `EMAC_RXQ_CTL1.TACPQE`= 1) and untagged control packets are routed to receive queue specified by this field.	
		0	Receive Queue 0
		1	Receive Queue 1
		2	Receive Queue 2
		3	Receive Queue 3
		4	Receive Queue 4
		5	Receive Queue 5
		6	Receive Queue 6
		7	Receive Queue 7

Receive Queue Control 2 Register

The EMAC_RXQ_CTL2 register controls the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queues 0 to 3.

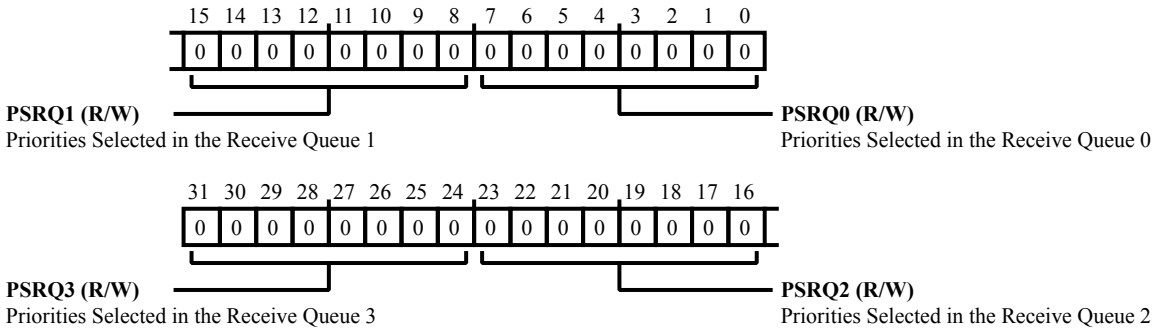


Figure 30-150: EMAC_RXQ_CTL2 Register Diagram

Table 30-205: EMAC_RXQ_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PSRQ3	Priorities Selected in the Receive Queue 3. The EMAC_RXQ_CTL2.PSRQ3 field selects the priorities assigned to Rx queue 3. All packets with priorities that match the values set in this field are routed to Rx queue 3. For example, if PSRQ3[6, 3] are set, packets with USP field equal to 3 or 6 are routed to Rx queue 3. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 3 crosses the flow control threshold settings.
23:16 (R/W)	PSRQ2	Priorities Selected in the Receive Queue 2. The EMAC_RXQ_CTL2.PSRQ2 field selects the priorities assigned to Rx queue 2. All packets with priorities that match the values set in this field are routed to Rx queue 2. For example, if PSRQ2[1, 0] are set, packets with USP field equal to 1 or 0 are routed to Rx queue 2. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 2 crosses the flow control threshold settings.

Table 30-205: EMAC_RXQ_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	PSRQ1	Priorities Selected in the Receive Queue 1. The `EMAC_RXQ_CTL2.PSRQ1` field selects the priorities assigned to Rx queue 1. All packets with priorities that match the values set in this field are routed to Rx queue 1. For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 1 crosses the flow control threshold settings.
7:0 (R/W)	PSRQ0	Priorities Selected in the Receive Queue 0. The `EMAC_RXQ_CTL2.PSRQ0` field selects the priorities assigned to Rx queue 0. All packets with priorities that match the values configured in this field are routed to Rx queue 0. For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 0 crosses the flow control threshold settings.

Receive Queue Control 3 Register

The EMAC_RXQ_CTL3 register controls the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queues 4 to 7.

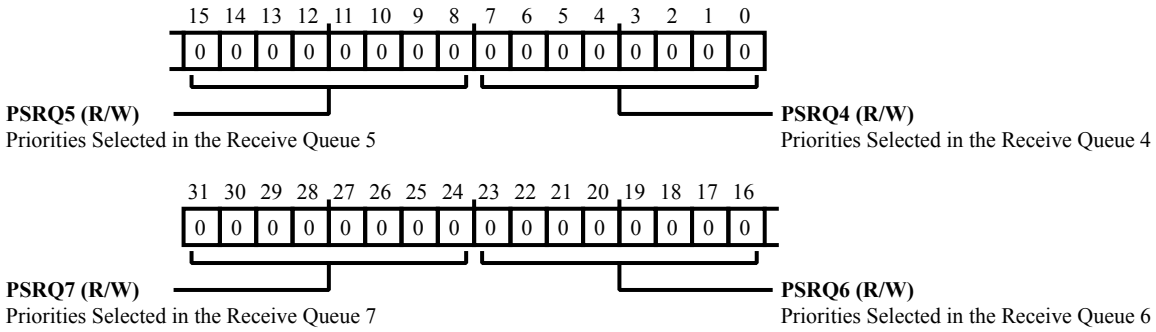


Figure 30-151: EMAC_RXQ_CTL3 Register Diagram

Table 30-206: EMAC_RXQ_CTL3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PSRQ7	Priorities Selected in the Receive Queue 7. The EMAC_RXQ_CTL3.PSRQ7 field selects the priorities assigned to Rx queue 7. All packets with priorities that match the values set in this field are routed to Rx queue 7. For example, if PSRQ7[7, 4] are set, packets with USP field equal to 7 or 4 are routed to Rx queue 7. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 7 crosses the flow control threshold settings.
23:16 (R/W)	PSRQ6	Priorities Selected in the Receive Queue 6. The EMAC_RXQ_CTL3.PSRQ6 field selects the priorities assigned to Rx queue 6. All packets with priorities that match the values set in this field are routed to Rx queue 6. For example, if PSRQ6[5] are set, packets with USP field equal to 5 are routed to Rx queue 6. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 6 crosses the flow control threshold settings.

Table 30-206: EMAC_RXQ_CTL3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	PSRQ5	Priorities Selected in the Receive Queue 5. The `EMAC_RXQ_CTL3.PSRQ5` field selects the priorities assigned to Rx queue 5. All packets with priorities that match the values set in this field are routed to Rx queue 5. For example, if PSRQ5[6] is set, packets with USP field equal to 6 are routed to Rx queue 5. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 5 crosses the flow control threshold settings.
7:0 (R/W)	PSRQ4	Priorities Selected in the Receive Queue 4. The `EMAC_RXQ_CTL3.PSRQ4` field selects the priorities assigned to Rx queue 4. All packets with priorities that match the values set in this field are routed to Rx queue 4. For example, if PSRQ4[7:4] is set, packets with USP field equal to 7, 6, 5, or 4 are routed to Rx queue 4. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. This field also determines the priorities to be included in the PFC packet sent to remote station when Rx queue 4 crosses the flow control threshold settings.

Receive Queue Control 4 Register

The EMAC_RXQ_CTL4 register controls the routing of unicast and multicast packets that fail the Destination or Source address filter to the Rx queues.

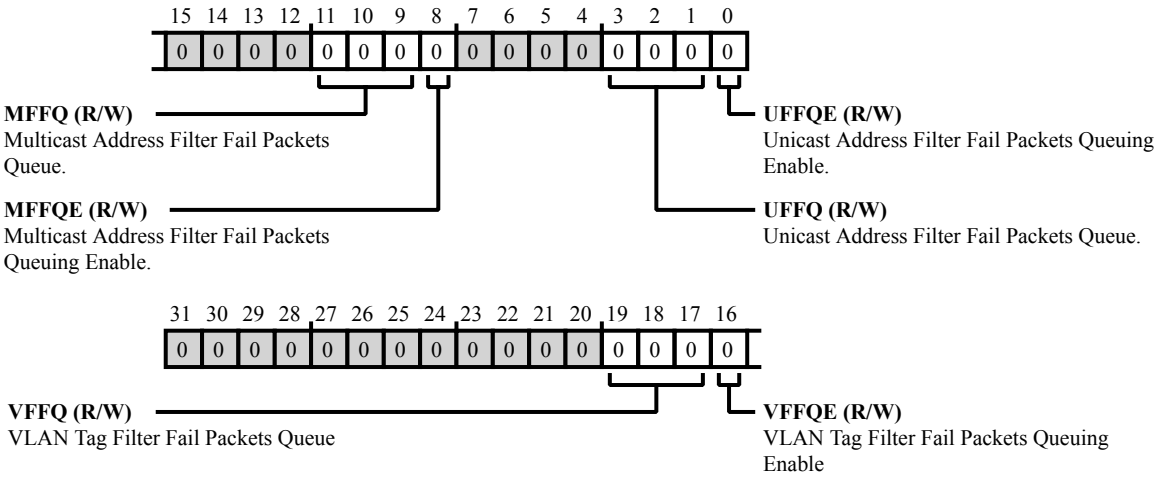


Figure 30-152: EMAC_RXQ_CTL4 Register Diagram

Table 30-207: EMAC_RXQ_CTL4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
19:17 (R/W)	VFFQ	VLAN Tag Filter Fail Packets Queue. The EMAC_RXQ_CTL4.VFFQ field holds the Rx queue number to which the tagged packets failing the Destination or Source Address filter (and EMAC_RXQ_CTL4.UFFQE/ EMAC_RXQ_CTL4.MFFQE = 0) or failing the VLAN tag filter must be routed to. The EMAC_RXQ_CTL4.VFFQ field is valid only when EMAC_RXQ_CTL4.VFFQE = 1.	
16 (R/W)	VFFQE	VLAN Tag Filter Fail Packets Queuing Enable. When the EMAC_RXQ_CTL4.VFFQE bit is set, the tagged packets which fail the Destination or Source address filter or fail the VLAN tag filter, are routed to the Rx queue number programmed in the EMAC_RXQ_CTL4.VFFQ field. When EMAC_RXQ_CTL4.VFFQE = 0, the tagged packets which fail the Destination or Source address filter or fail the VLAN tag filter are routed based on other routing options. The EMAC_RXQ_CTL4.VFFQE bit is valid only when the EMAC_MACPKT_FILT.RA = 1.	
		0	VLAN Tag Filter Fail Packets Queuing is Disabled
		1	VLAN Tag Filter Fail Packets Queuing is Enabled

Table 30-207: EMAC_RXQ_CTL4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:9 (R/W)	MFFQ	Multicast Address Filter Fail Packets Queue.. The EMAC_RXQ_CTL4.MFFQ field holds the Rx queue number to which the Multicast packets failing the Destination or Source Address filter are routed to. The EMAC_RXQ_CTL4.MFFQ field is valid only when EMAC_RXQ_CTL4.MFFQE = 1.	
8 (R/W)	MFFQE	Multicast Address Filter Fail Packets Queuing Enable.. When the EMAC_RXQ_CTL4.MFFQE bit is set, the Multicast packets which fail the Destination or Source address filter is routed to the Rx queue Number programmed in the EMAC_RXQ_CTL4.MFFQ. When EMAC_RXQ_CTL4.MFFQE = 0, the Multicast packets which fail the Destination or Source address filter is routed based on other routing options. The EMAC_RXQ_CTL4.MFFQE bit is valid only when EMAC_MACPKT_FILT.RA = 1.	
		0	Multicast Address Filter Fail Packets Queuing is Disabled
		1	Multicast Address Filter Fail Packets Queuing is Enabled
3:1 (R/W)	UFFQ	Unicast Address Filter Fail Packets Queue.. The EMAC_RXQ_CTL4.UFFQ field holds the Rx queue number to which the Unicast packets failing the Destination or Source Address filter are routed to. The EMAC_RXQ_CTL4.UFFQ field is valid only when EMAC_RXQ_CTL4.UFFQE = 1.	
0 (R/W)	UFFQE	Unicast Address Filter Fail Packets Queuing Enable.. When the EMAC_RXQ_CTL4.UFFQE bit is set, the Unicast packets which fail the Destination or Source address filter is routed to the Rx queue number programmed in the EMAC_RXQ_CTL4.UFFQ field. When EMAC_RXQ_CTL4.UFFQE = 0, the Unicast packets which fail the Destination or Source address filter is routed based on other routing options. The EMAC_RXQ_CTL4.UFFQE bit is valid only when EMAC_MACPKT_FILT.RA = 1.	
		0	Unicast Address Filter Fail Packets Queuing is Disabled
		1	Unicast Address Filter Fail Packets Queuing is Enabled

Receive Flow Control Register

The EMAC_RXFLOW_CTL register controls the pausing of MAC transmit based on the received pause packet.

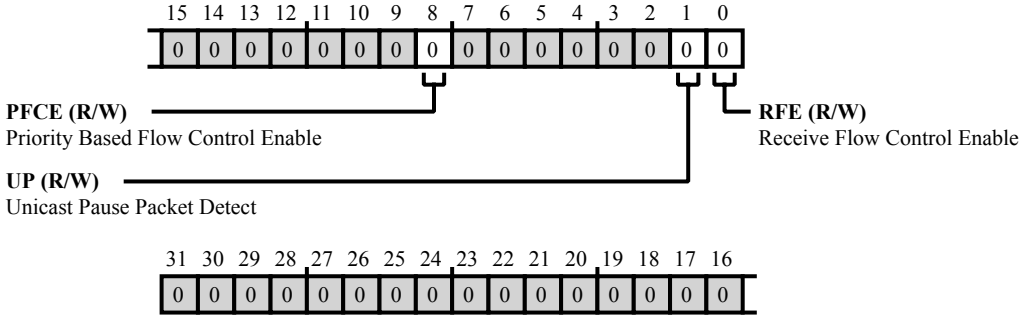


Figure 30-153: EMAC_RXFLOW_CTL Register Diagram

Table 30-208: EMAC_RXFLOW_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	PFCE	Priority Based Flow Control Enable. When the EMAC_RXFLOW_CTL.PFCE bit is set, it enables generation and reception of priority-based flow control (PFC) packets. When EMAC_RXFLOW_CTL.PFCE = 0, it enables generation and reception of 802.3x Pause control packets.	
		0	Priority Based Flow Control is Disabled
		1	Priority Based Flow Control is Enabled
1 (R/W)	UP	Unicast Pause Packet Detect. A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When the EMAC_RXFLOW_CTL.UP bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in EMAC_ADDR0_HI and EMAC_ADDR0_LO. When EMAC_RXFLOW_CTL.UP = 0, the MAC only detects Pause packets with unique multicast address. Note: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01_80_C2_00_00_01) is as specified in IEEE 802.1 Qbb-2011.	
		0	Unicast Pause Packet Detect Disabled
		1	Unicast Pause Packet Detect Enabled

Table 30-208: EMAC_RXFLOW_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	RFE	Receive Flow Control Enable. When the `EMAC_RXFLOW_CTL.RFE` bit is set and the MAC is operating in full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When `EMAC_RXFLOW_CTL.RFE` = 0 or the MAC is operating in half-duplex mode, the decode function of the Pause packet is disabled. When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.	
		0	Receive Flow Control is Disabled
		1	Receive Flow Control is Enabled

Receive Transmit Status Register

The EMAC_RXTX_STAT register contains the receive and transmit error status.

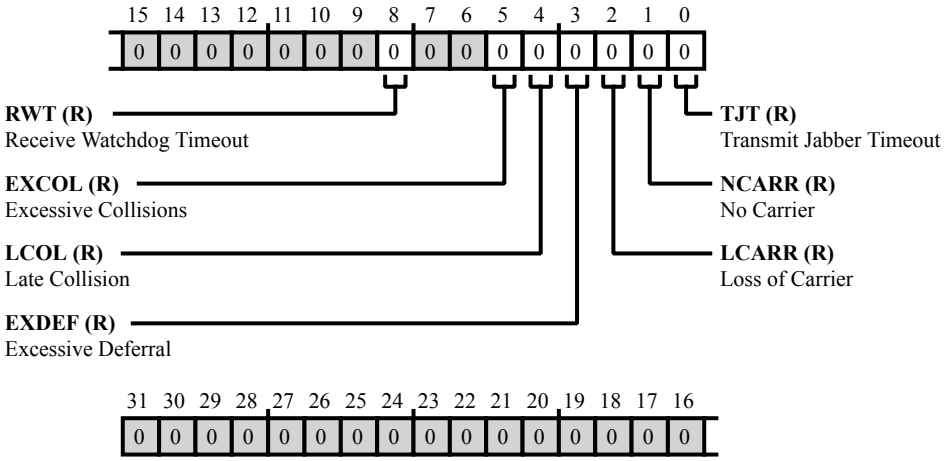


Figure 30-154: EMAC_RXTX_STAT Register Diagram

Table 30-209: EMAC_RXTX_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/NW)	RWT	Receive Watchdog Timeout. The EMAC_RXTX_STAT.RWT bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1) and EMAC_MAC_CFG.WD = 0. The EMAC_RXTX_STAT.RWT bit is set when a packet with length greater than 16,383 bytes is received and EMAC_MAC_CFG.WD = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Receive Watchdog Timeout
		1	Receive Watchdog Timed Out
5 (R/NW)	EXCOL	Excessive Collisions. When EMAC_MTL_OPMODE.DTXSTS = 1, the EMAC_RXTX_STAT.EXCOL bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. When EMAC_MAC_CFG.DR = 1, the EMAC_RXTX_STAT.EXCOL bit is set after the first collision and the packet transmission is aborted. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Collision
		1	Excessive Collision is Sensed

Table 30-209: EMAC_RXTX_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	LCOL	Late Collision. When EMAC_MTL_OPMODE.DTXSTS = 1, the `EMAC_RXTX_STAT.LCOL` bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode; 512 bytes including Preamble and Carrier Extension in GMII mode). The `EMAC_RXTX_STAT.LCOL` bit is not valid if the Underflow error occurs. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Collision
		1	Late Collision is Sensed
3 (R/NW)	EXDEF	Excessive Deferral. When EMAC_MTL_OPMODE.DTXSTS = 1 and EMAC_MAC_CFG.DC = 1, the `EMAC_RXTX_STAT.EXDEF` bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 in 1000/2500 Mbps mode or when jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1)). Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Excessive Deferral
		1	Excessive Deferral
2 (R/NW)	LCARR	Loss of Carrier. When EMAC_MTL_OPMODE.DTXSTS = 1, the `EMAC_RXTX_STAT.LCARR` bit indicates that the loss of carrier occurred during packet transmission, that is, the ETHx_CRS signal was inactive for one or more transmission clock periods during packet transmission. The `EMAC_RXTX_STAT.LCARR` bit is valid only for packets transmitted without collision. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Carrier is Present
		1	Loss of Carrier
1 (R/NW)	NCARR	No Carrier. When EMAC_MTL_OPMODE.DTXSTS = 1, the `EMAC_RXTX_STAT.NCARR` bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Carrier is Present
		1	No Carrier

Table 30-209: EMAC_RXTX_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	TJT	Transmit Jabber Timeout. The `EMAC_RXTX_STAT.TJT` bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1) and EMAC_MAC_CFG.JD = 0. The `EMAC_RXTX_STAT.TJT` bit is set when the packet size exceeds 16,383 bytes and the EMAC_MAC_CFG.JD = 1. Access restriction applies. Clears on read (or write of 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	No Transmit Jabber Timeout
		1	Transmit Jabber Timeout Occurred

Source Port Identity 0 Register

The EMAC_SRCPRT_IDNTY0 register contains bits[31:0] of the 80-bit source port identity of the PTP node.

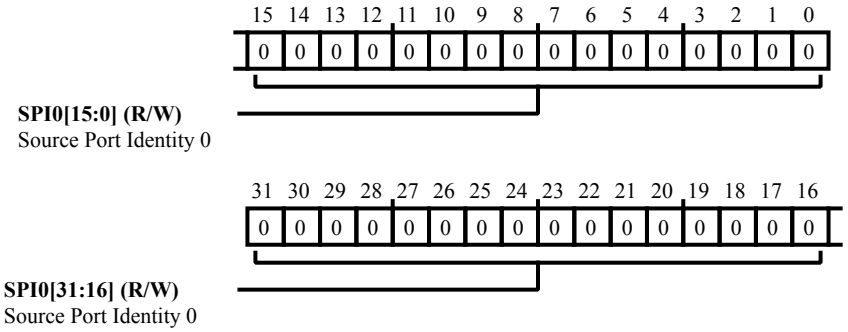


Figure 30-155: EMAC_SRCPRT_IDNTY0 Register Diagram

Table 30-210: EMAC_SRCPRT_IDNTY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	SPI0	Source Port Identity 0. The EMAC_SRCPRT_IDNTY0.SPI0 field indicates bits [31:0] of sourcePortIdentity of PTP node.

Source Port Identity 1 Register

The EMAC_SRCPRT_IDNTY1 register contains bits[63:32] of the 80-bit Source Port Identity of the PTP node.

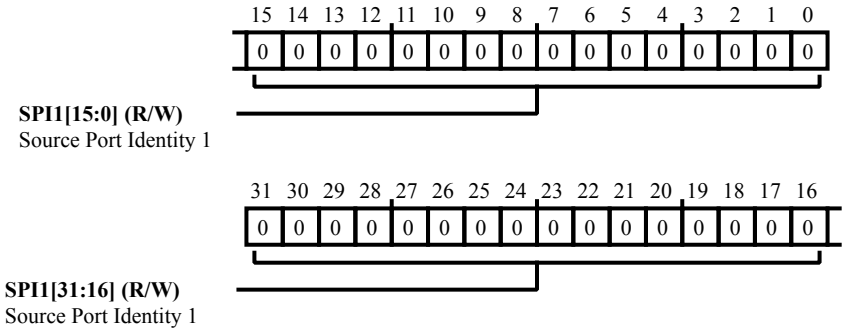


Figure 30-156: EMAC_SRCPRT_IDNTY1 Register Diagram

Table 30-211: EMAC_SRCPRT_IDNTY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	SPI1	Source Port Identity 1. The EMAC_SRCPRT_IDNTY1.SPI1 field indicates bits [63:32] of sourcePortIdentity of PTP node.

Source Port Identity 2 Register

The EMAC_SRCPRT_IDNTY2 register contains bits[79:64] of the 80-bit Source Port Identity of the PTP node.

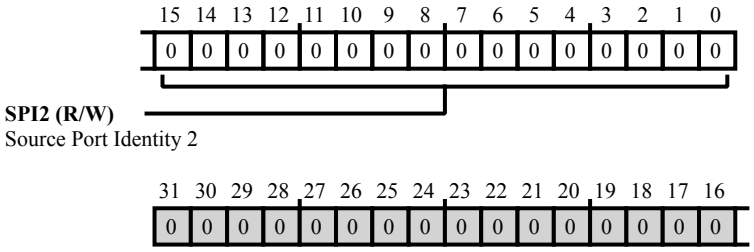


Figure 30-157: EMAC_SRCPRT_IDNTY2 Register Diagram

Table 30-212: EMAC_SRCPRT_IDNTY2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	SPI2	Source Port Identity 2. The EMAC_SRCPRT_IDNTY2.SPI2 field indicates bits [79:64] of sourcePortIdentity of PTP node.

Subsecond Increment Register

The EMAC_TM_SUBSEC register specifies the value to be added to the internal system time register every cycle of PTP clock.

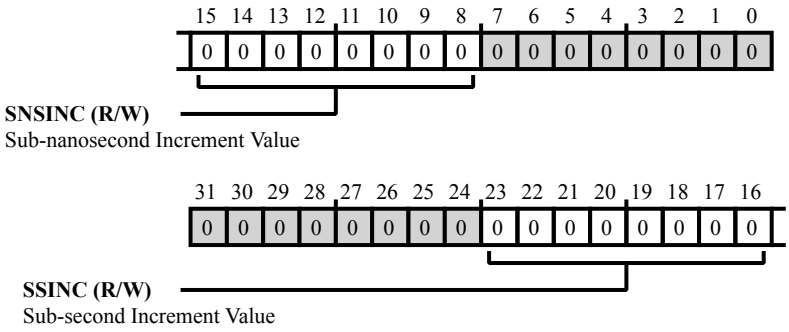


Figure 30-158: EMAC_TM_SUBSEC Register Diagram

Table 30-213: EMAC_TM_SUBSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	SSINC	Sub-second Increment Value. The value programmed in the EMAC_TM_SUBSEC.SSINC field is accumulated every PTP clock cycle with the contents of the sub-second register. For example, when the PTP clock is 50 MHz (period is 20 ns), program 20 (0x14) when the EMAC_TM_NSEC register has an accuracy of 1 ns (EMAC_TM_CTL.TSCTRLSSR = 1). When EMAC_TM_CTL.TSCTRLSSR = 0, the EMAC_TM_NSEC register has a resolution of ~0.465 ns. In this case, program a value of 43 (0x2B) which is derived by 20 ns/0.465.
15:8 (R/W)	SNSINC	Sub-nanosecond Increment Value. The EMAC_TM_SUBSEC.SNSINC field contains the sub-nanosecond increment value, represented in nanoseconds multiplied by 2^8. This value is accumulated with the sub-nanoseconds field of the subsecond register. For example, when EMAC_TM_CTL.TSCTRLSSR = 1. and if the required increment is 5.3ns, then EMAC_TM_SUBSEC.SSINC should be 0x05 and EMAC_TM_SUBSEC.SNSINC should be 0x4C.

System Time Higher Word Seconds Register

System Time - Higher Word Seconds register.

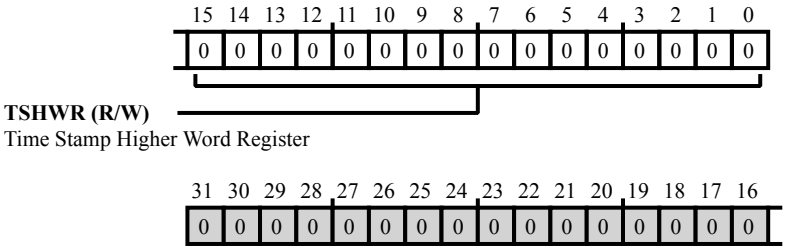


Figure 30-159: EMAC_TM_HISEC Register Diagram

Table 30-214: EMAC_TM_HISEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	TSHWR	Time Stamp Higher Word Register. The `EMAC_TM_HISEC.TSHWR` field contains the most-significant 16-bits of time stamp seconds value. The `EMAC_TM_HISEC` register is optional. Add this register by selecting the Add IEEE 1588 Higher Word Register option. The `EMAC_TM_HISEC` register is directly written to initialize the value and it is incremented when there is an overflow from 32 bits of the EMAC_TM_SEC register. Access restriction applies. Updated based on the event. Write 1 to set. Write 0 to clear.

System Time Nanoseconds Register

The EMAC_TM_NSEC register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

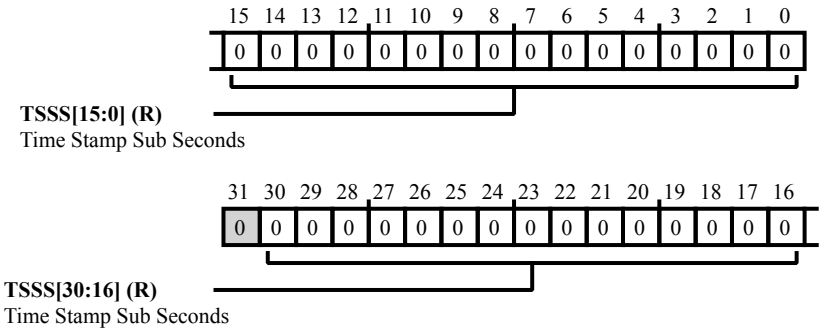


Figure 30-160: EMAC_TM_NSEC Register Diagram

Table 30-215: EMAC_TM_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
30:0 (R/NW)	TSSS	Time Stamp Sub Seconds. The value in the EMAC_TM_NSEC.TSSS field has the sub-second representation of time, with an accuracy of 0.46 ns. (EMAC_TM_CTL.TSCTRLSSR = 1), each bit represents 1 ns. The maximum value is 0x3B9A_C9FF after which it rolls over to zero.

System Time Nanoseconds Update Register

MAC System Time Nanoseconds Update register.

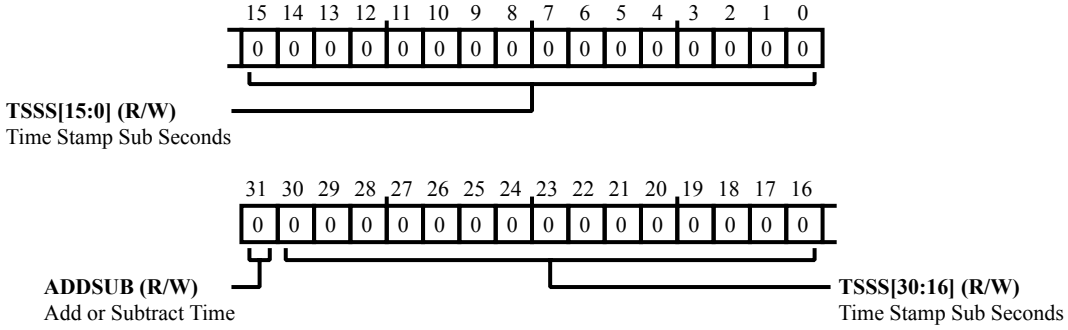


Figure 30-161: EMAC_TM_NSECUPDT Register Diagram

Table 30-216: EMAC_TM_NSECUPDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ADDSUB	Add or Subtract Time. When the `EMAC_TM_NSECUPDT.ADDSUB` bit is set, the time value is subtracted with the contents of the update register. When `EMAC_TM_NSECUPDT.ADDSUB` = 0, the time value is added with the contents of the update register.	
		0	Add Time
		1	Subtract Time

Table 30-216: EMAC_TM_NSECUPDT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30:0 (R/W)	TSSS	Time Stamp Sub Seconds. The `EMAC_TM_NSECUPDT.TSSS` field is the sub-seconds part of the update. When EMAC_TM_NSECUPDT.ADDSUB = 1, the `EMAC_TM_NSECUPDT.TSSS` field must be programmed with the complement of the sub-seconds part of the update value as described. When EMAC_TM_NSECUPDT.ADDSUB = 0, the `EMAC_TM_NSECUPDT.TSSS` field must be programmed with the sub-seconds part of the update value, with an accuracy based on the EMAC_TM_CTL.TSCTRLSSR bit. For EMAC_TM_CTL.TSCTRLSSR = 1: -- The programmed value must be 10^9 - <sub-second value>. -- Each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF. For EMAC_TM_CTL.TSCTRLSSR = 0: -- The programmed value must be 2^{31} - <sub-second_value>. -- Each bit represents an accuracy of 0.46 ns. For example, to subtract 2.000000001 seconds from the system time, then the EMAC_TM_NSECUPDT.TSSS field must be 0x7FFF_FFFF (that is, 2^{31} - 1), when EMAC_TM_CTL.TSCTRLSSR = 0 and 0x3B9A_C9FF (that is, 10^9 - 1), when EMAC_TM_CTL.TSCTRLSSR = 1.

System Time Seconds Register

The EMAC_TM_SEC register, along with the EMAC_TM_NSEC register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from PTP clock to SCLK0).

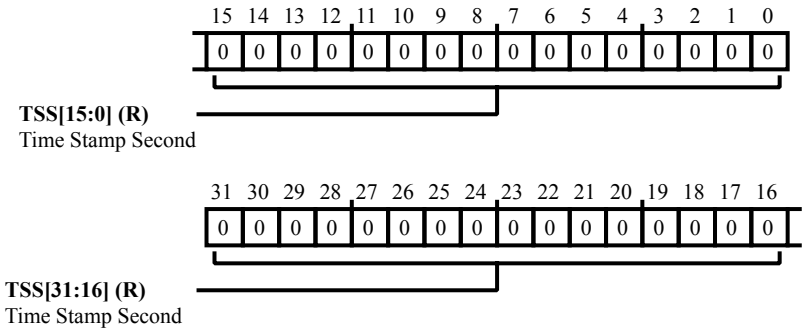


Figure 30-162: EMAC_TM_SEC Register Diagram

Table 30-217: EMAC_TM_SEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TSS	Time Stamp Second. The value in the EMAC_TM_SEC.TSS field indicates the current value in seconds of the System Time maintained by the MAC.

System Time Seconds Update Register

The EMAC_TM_SECUPDT register, along with the EMAC_TM_NSECUPDT register, initializes or updates the system time maintained by the MAC. Write both registers before setting the EMAC_TM_CTL.TSINIT or EMAC_TM_CTL.TSUPDT bits.

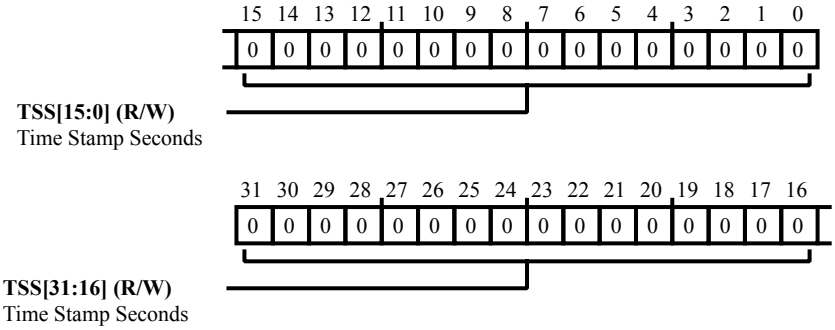


Figure 30-163: EMAC_TM_SECUPDT Register Diagram

Table 30-218: EMAC_TM_SECUPDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSS	Time Stamp Seconds. The value in the EMAC_TM_SECUPDT.TSS field is the seconds part of the update. When EMAC_TM_NSECUPDT.ADDSUB = 0, the EMAC_TM_SECUPDT.TSS field must be programmed with the seconds part of the update value. When EMAC_TM_NSECUPDT.ADDSUB = 1, the EMAC_TM_SECUPDT.TSS field must be programmed with the complement of the seconds part of the update value. For example, to subtract 2.000000001 seconds from the system time, the EMAC_TM_SECUPDT.TSS field must be 0xFFFF_FFFE (that is, 2^{32} - 2).

Time Stamp Addend Register

The EMAC_TM_ADDEND register value is used only when the system time is configured for Fine Update mode (EMAC_TM_CTL.TSCFUPDT). The content of this register is added to a 32-bit accumulator in every clock cycle (PTP clock) and the system time is updated when the accumulator overflows.

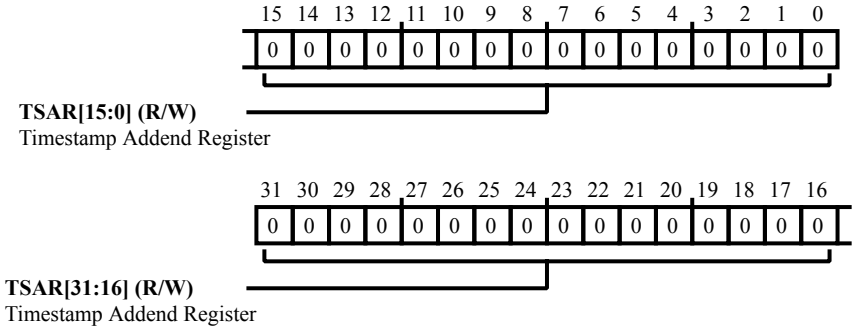


Figure 30-164: EMAC_TM_ADDEND Register Diagram

Table 30-219: EMAC_TM_ADDEND Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSAR	Timestamp Addend Register. The EMAC_TM_ADDEND.TSAR field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

Time Stamp Control Register

The EMAC_TM_CTL register controls the operation of the System Time generator and processing of PTP packets for time stamping in the receiver.

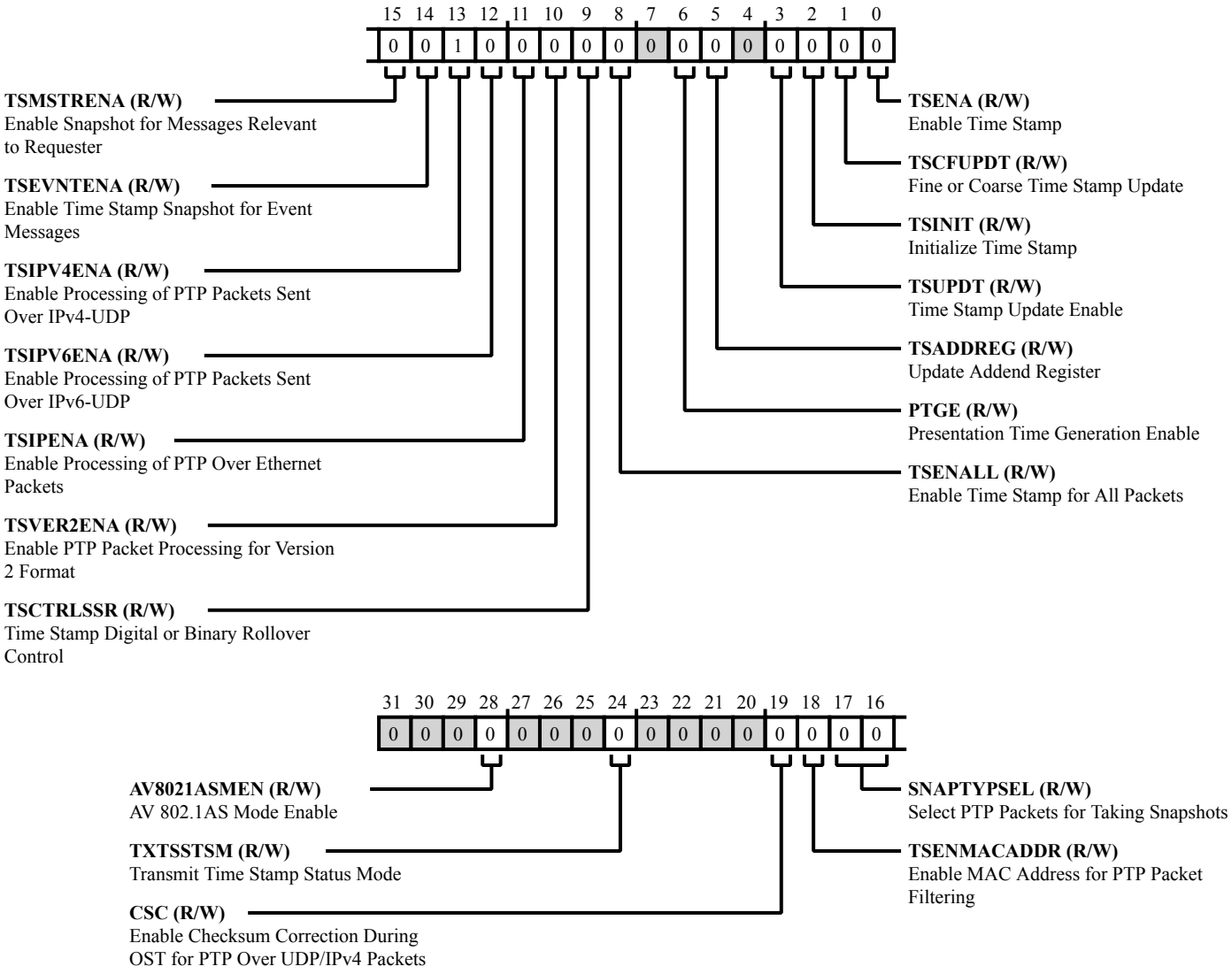


Figure 30-165: EMAC_TM_CTL Register Diagram

Table 30-220: EMAC_TM_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	AV8021ASMEN	AV 802.1AS Mode Enable. When the EMAC_TM_CTL.AV8021ASMEN bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing time stamp snapshots, that is, IEEE 802.1AS mode of operation. When PTP offload feature is enabled, for the purpose of PTP offload, the transport specific field in the PTP header is generated and checked based on the value of the EMAC_TM_CTL.AV8021ASMEN bit.	
		0	AV 802.1AS Mode is Disabled
		1	AV 802.1AS Mode is Enabled
24 (R/W)	TXTSSTSM	Transmit Time Stamp Status Mode. When the EMAC_TM_CTL.TXTSSTSM bit is set, the MAC overwrites the earlier transmit time stamp status even if it is not read by the software. The MAC indicates this by setting the EMAC_TXTMSTAT_NSEC.TXTSSMIS bit. When EMAC_TM_CTL.TXTSSTSM = 0, the MAC ignores the time stamp status of current packet if the time stamp status of previous packet is not read by the software. The MAC indicates this by setting the EMAC_TXTMSTAT_NSEC.TXTSSMIS bit.	
		0	Disable
		1	Enable
19 (R/W)	CSC	Enable Checksum Correction During OST for PTP Over UDP/IPv4 Packets. When the EMAC_TM_CTL.CSC bit is set, the last two bytes of PTP message sent over UDP/IPv4 is updated to keep the UDP checksum correct, for changes made to origin time stamp and/or correction field as part of one step time stamp operation. The application forms the packet with these two dummy bytes. When cleared (=0), no updates are done to keep the UDP checksum correct. The application forms the packet with UDP checksum configured to 0.	
		0	Checksum Correction During OST for PTP Over UDP/IPv4 Packets is Disabled
		1	Checksum Correction During OST for PTP Over UDP/IPv4 Packets is Enabled

Table 30-220: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	TSENMACADDR	Enable MAC Address for PTP Packet Filtering. When the EMAC_TM_CTL.TSENMACADDR bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet. When the EMAC_TM_CTL.TSENMACADDR bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated, when PTP is directly sent over Ethernet. For normal time stamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching. For PTP offload, only MAC address register 0 is considered for unicast destination address matching.	
		0	MAC Address for PTP Packet Filtering is Disabled
		1	MAC Address for PTP Packet Filtering is Enabled
17:16 (R/W)	SNAPTYPSEL	Select PTP Packets for Taking Snapshots. The EMAC_TM_CTL.SNAPTYPSEL field, along with bits 15 and 14, decide the set of PTP packet types for which snapshot needs to be taken.	
15 (R/W)	TSMSTRENA	Enable Snapshot for Messages Relevant to Requester. When the EMAC_TM_CTL.TSMSTRENA bit is set, the snapshot is taken only for the messages that are relevant to the requester. Otherwise, the snapshot is taken for the messages relevant to the target node.	
		0	Disable
		1	Enable
14 (R/W)	TSEVNTENA	Enable Time Stamp Snapshot for Event Messages. When the EMAC_TM_CTL.TSEVNTENA bit is set, the time stamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When EMAC_TM_CTL.TSEVNTENA = 0, the snapshot is taken for all messages except Announce, Management, and Signaling.	
		0	Timestamp Snapshot for Event Messages is Disabled
		1	Timestamp Snapshot for Event Messages is Enabled

Table 30-220: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	TSIPV4ENA	Enable Processing of PTP Packets Sent Over IPv4-UDP. When the EMAC_TM_CTL.TSIPV4ENA bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When EMAC_TM_CTL.TSIPV4ENA = 0, the MAC ignores the PTP transported over IPv4 UDP packets. The EMAC_TM_CTL.TSIPV4ENA bit is set by default.	
		0	Processing of PTP Packets Sent Over IPv4-UDP is Disabled
		1	Processing of PTP Packets Sent Over IPv4-UDP is Enabled
12 (R/W)	TSIPV6ENA	Enable Processing of PTP Packets Sent Over IPv6-UDP. When the EMAC_TM_CTL.TSIPV6ENA bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When the EMAC_TM_CTL.TSIPV6ENA bit is clear, the MAC ignores the PTP transported over IPv6 UDP packets.	
		0	Processing of PTP Packets Sent Over IPv6-UDP is Disabled
		1	Processing of PTP Packets Sent Over IPv6-UDP is Enabled
11 (R/W)	TSIPENA	Enable Processing of PTP Over Ethernet Packets. When the EMAC_TM_CTL.TSIPENA bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When EMAC_TM_CTL.TSIPENA = 0, the MAC ignores the PTP over Ethernet packets.	
		0	Processing of PTP Over Ethernet Packets is Disabled
		1	Processing of PTP Over Ethernet Packets is Enabled
10 (R/W)	TSVER2ENA	Enable PTP Packet Processing for Version 2 Format. When the EMAC_TM_CTL.TSVER2ENA bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When EMAC_TM_CTL.TSVER2ENA = 0, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.	
		0	PTP Packet Processing for Version 2 Format is Disabled
		1	PTP Packet Processing for Version 2 Format is Enabled

Table 30-220: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	TSCTRLSSR	Time Stamp Digital or Binary Rollover Control. When the EMAC_TM_CTL.TSCTRLSSR bit is set, the time stamp low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the time stamp (high) seconds. When EMAC_TM_CTL.TSCTRLSSR = 0, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment must be programmed correctly depending on the PTP reference clock frequency and the value of the EMAC_TM_CTL.TSCTRLSSR bit.	
		0	Disable
		1	Enable
8 (R/W)	TSENALL	Enable Time Stamp for All Packets. When the EMAC_TM_CTL.TSENALL bit is set, the time stamp snapshot is enabled for all packets received by the MAC.	
		0	Disable
		1	Enable
6 (R/W)	PTGE	Presentation Time Generation Enable. When the EMAC_TM_CTL.PTGE bit is set, presentation time generation is enabled.	
		0	Presentation Time Generation is Disabled
		1	Presentation Time Generation is Enabled
5 (R/W)	TSADDREG	Update Addend Register. When the EMAC_TM_CTL.TSADDREG bit is set, the content of the EMAC_TM_ADDEND register is updated in the PTP block for fine correction. The EMAC_TM_CTL.TSADDREG bit is cleared when the update is complete. The EMAC_TM_CTL.TSADDREG bit should be zero before it is set. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Addend Register is Not Updated
		1	Addend Register is Updated

Table 30-220: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	TSUPDT	Time Stamp Update Enable. When EMAC_TM_CTL.TSUPDT = 1, the system time is updated (added or subtracted) with the value specified in the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT registers. The EMAC_TM_CTL.TSUPDT bit should be zero before updating it. The EMAC_TM_CTL.TSUPDT bit is reset when the update is complete in hardware. The EMAC_TM_HISEC register (if enabled while configuring the IP) is not updated. When Media Clock Generation and Recovery is enabled, the EMAC_PRSNTM_UPDT register should also be updated before setting this field. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Disable
		1	Enable
2 (R/W)	TSINIT	Initialize Time Stamp. When the EMAC_TM_CTL.TSINIT bit is set, the system time is initialized (overwritten) with the value specified in the EMAC_TM_SECUPDT and EMAC_TM_NSECUPDT registers. The EMAC_TM_CTL.TSINIT bit should be zero before it is updated. The EMAC_TM_CTL.TSINIT bit is cleared when the initialization is complete. The EMAC_TM_HISEC register (if enabled while configuration the IP) can only be initialized. When media clock generation and recovery is enabled the EMAC_PRSNTM_UPDT register must also be updated before setting this field. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Disable
		1	Enable
1 (R/W)	TSCFUPDT	Fine or Coarse Time Stamp Update. When the EMAC_TM_CTL.TSCFUPDT bit is set, the fine method is used to update system time stamp. When EMAC_TM_CTL.TSCFUPDT = 0, coarse method is used to update the system time stamp.	
		0	Coarse Method is Used to Update System Timestamp
		1	Fine Method is Used to Update System Timestamp

Table 30-220: EMAC_TM_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	TSENA	Enable Time Stamp. When the EMAC_TM_CTL.TSENA bit is set, the time stamp is added for transmit and receive packets. When disabled, time stamp is not added for transmit and receive packets and the time stamp generator is also suspended. Initialize the time stamp (system time) after enabling this mode. On the receive side, the MAC processes the 1588 packets only if the EMAC_TM_CTL.TSENA bit is set.	
		0	Disable
		1	Enable

Time Stamp Egress Asymmetry Correction Register

The EMAC_TM_EGASYM_CORR register contains the egress asymmetry correction value to be used while updating the correction field in PDelay_Req PTP messages.

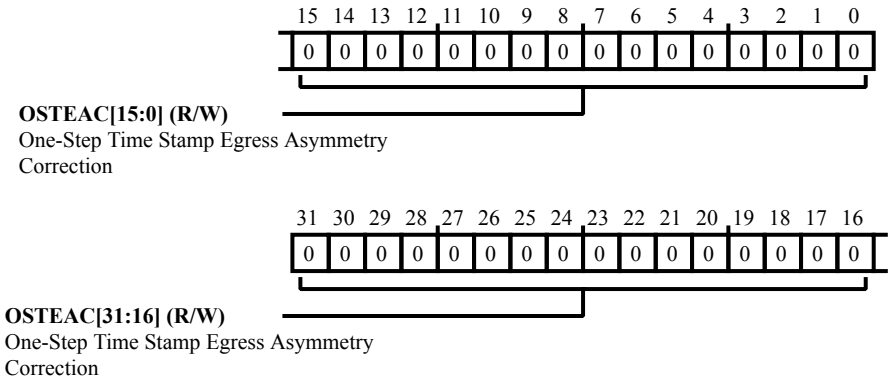


Figure 30-166: EMAC_TM_EGASYM_CORR Register Diagram

Table 30-221: EMAC_TM_EGASYM_CORR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	OSTEAC	One-Step Time Stamp Egress Asymmetry Correction. The EMAC_TM_EGASYM_CORR.OSTEAC field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^{16}. For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFD_8000, which is the 2's complement of 0x0002_8000(2.5 * 216). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003_4CCC (3.3 * 216).

Time Stamp Egress Correction Nanosecond Register

The EMAC_TM_EGCORR_NSEC register contains the correction value in nanoseconds to be used with the captured time stamp value in the egress path.

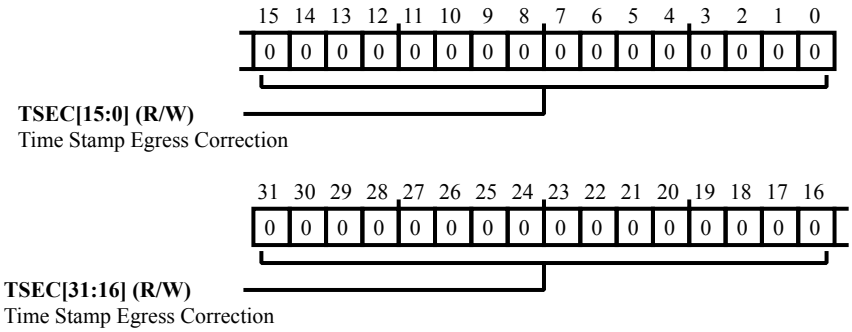


Figure 30-167: EMAC_TM_EGCORR_NSEC Register Diagram

Table 30-222: EMAC_TM_EGCORR_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSEC	Time Stamp Egress Correction. The EMAC_TM_EGCORR_NSEC.TSEC field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

Time Stamp Egress Correction Sub Nanosecond Register

The EMAC_TM_EGCORR_SNSEC register contains the sub-nanosecond part of the correction value to be used with the captured time stamp value, for egress direction.

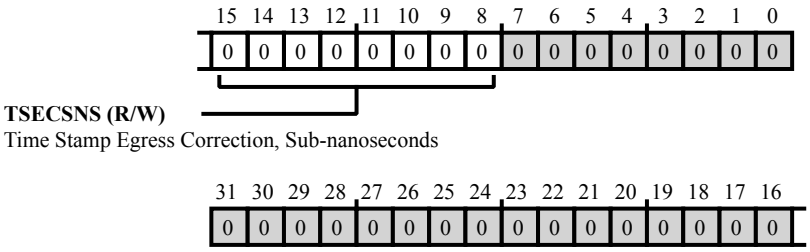


Figure 30-168: EMAC_TM_EGCORR_SNSEC Register Diagram

Table 30-223: EMAC_TM_EGCORR_SNSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	TSECSNS	Time Stamp Egress Correction, Sub-nanoseconds. The EMAC_TM_EGCORR_SNSEC.TSECSNS field contains the sub-nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

Time Stamp Egress Latency Register

The EMAC_TM_EG_LTNCY register holds the egress MAC latency.

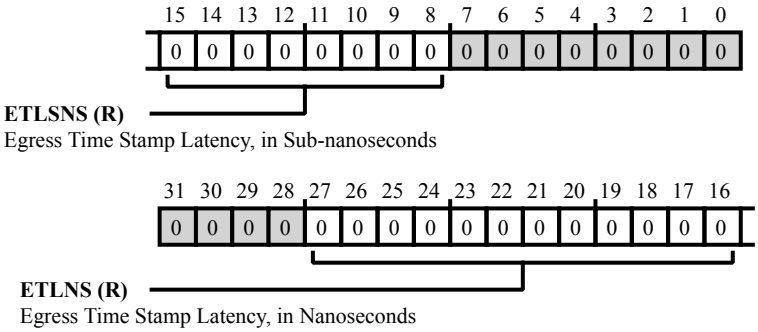


Figure 30-169: EMAC_TM_EG_LTNCY Register Diagram

Table 30-224: EMAC_TM_EG_LTNCY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:16 (R/NW)	ETLNS	Egress Time Stamp Latency, in Nanoseconds. The EMAC_TM_EG_LTNCY register holds the average latency in nanoseconds between the actual point (GMII/MII) where the egress time stamp is taken and the output ports (ETHx_TXDx) of the MAC.
15:8 (R/NW)	ETLSNS	Egress Time Stamp Latency, in Sub-nanoseconds. The EMAC_TM_EG_LTNCY register holds the average latency in sub-nanoseconds between the actual point (GMII/MII) where the egress time stamp is taken and the output ports (phy_txd_o) of the MAC.

Time Stamp Ingress Asymmetry Correction Register

The EMAC_TM_INGASYM_CORR register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

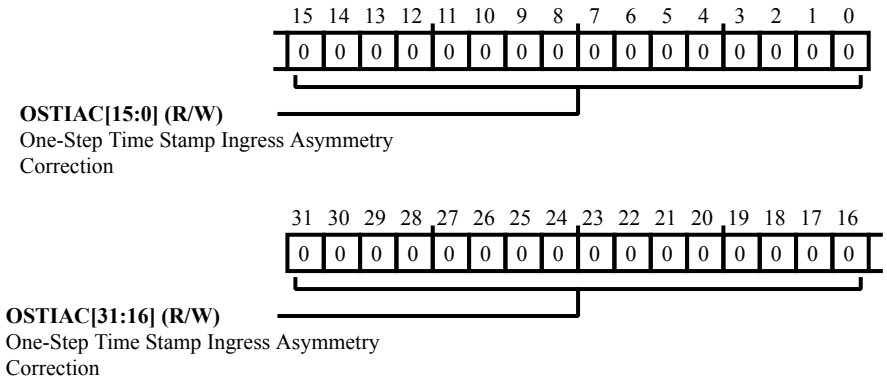


Figure 30-170: EMAC_TM_INGASYM_CORR Register Diagram

Table 30-225: EMAC_TM_INGASYM_CORR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	OSTIAC	One-Step Time Stamp Ingress Asymmetry Correction. The EMAC_TM_INGASYM_CORR.OSTIAC field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^{16}. For example, 2.5 ns is represented as 0x00028000. The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

Time Stamp Ingress Correction Nanosecond Register

The EMAC_TM_INGCORR_NSEC register contains the correction value in nanoseconds to be used with the captured time stamp value in the ingress path.

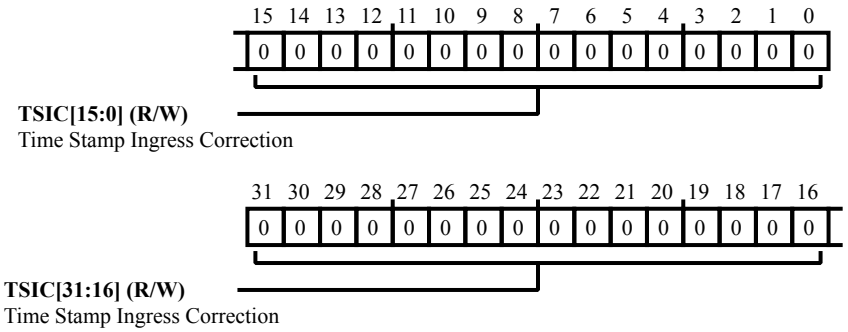


Figure 30-171: EMAC_TM_INGCORR_NSEC Register Diagram

Table 30-226: EMAC_TM_INGCORR_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	TSIC	Time Stamp Ingress Correction. The EMAC_TM_INGCORR_NSEC.TSIC field contains the ingress path correction value as defined by the Ingress Correction expression.

Time Stamp Ingress Correction Sub Nanosecond Register

The EMAC_TM_INGCORR_SNSEC register contains the sub-nanosecond part of the correction value to be used with the captured time stamp value, for ingress direction.

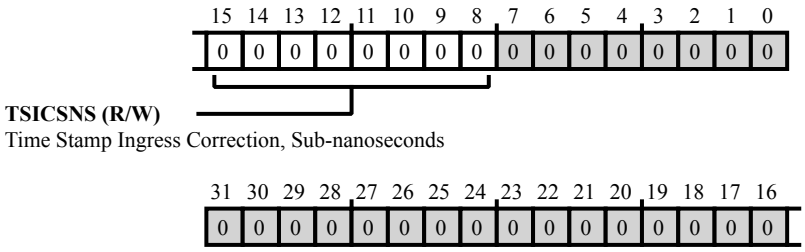


Figure 30-172: EMAC_TM_INGCORR_SNSEC Register Diagram

Table 30-227: EMAC_TM_INGCORR_SNSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	TSICSNS	Time Stamp Ingress Correction, Sub-nanoseconds. The EMAC_TM_INGCORR_SNSEC.TSICSNS field contains the sub-nanoseconds part of the ingress path correction value as defined by the Ingress Correction expression.

Time Stamp Ingress Latency Register

The EMAC_TM_ING_LTNCY register holds the ingress MAC latency.

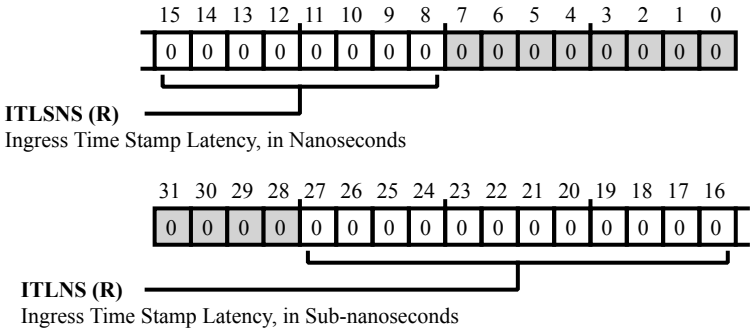


Figure 30-173: EMAC_TM_ING_LTNCY Register Diagram

Table 30-228: EMAC_TM_ING_LTNCY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:16 (R/NW)	ITLNS	Ingress Time Stamp Latency, in Sub-nanoseconds. The EMAC_TM_ING_LTNCY register holds the average latency in sub-nanoseconds between the input ports (ETHx_RXDx) of MAC and the actual point (GMII/MII) where the ingress time stamp is taken. Ingress correction value is computed as described in the section, "Ingress Correction".
15:8 (R/NW)	ITLSNS	Ingress Time Stamp Latency, in Nanoseconds. The EMAC_TM_ING_LTNCY register holds the average latency in nanoseconds between the input ports (phy_rxd_i) of MAC and the actual point (GMII/MII) where the ingress time stamp is taken. Ingress correction value is computed as described in the section, "Ingress Correction".

Time Stamp Status Register

All bits except bits[27:25] are cleared when the application reads the EMAC_TM_STAT register.

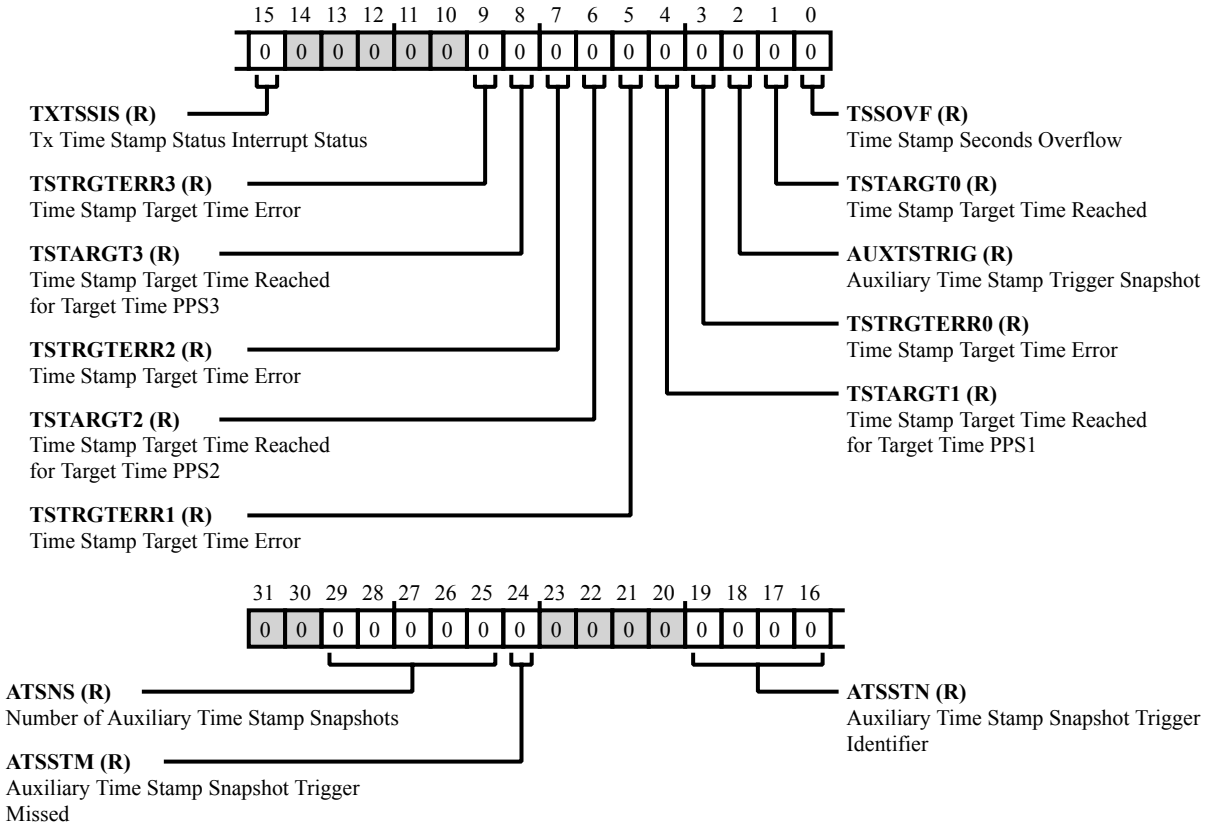


Figure 30-174: EMAC_TM_STAT Register Diagram

Table 30-229: EMAC_TM_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/NW)	ATSNS	Number of Auxiliary Time Stamp Snapshots. The EMAC_TM_STAT.ATSNS field indicates the number of snapshots available in the FIFO. A value equal to the selected depth of FIFO (4, 8, or 16) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set. The EMAC_TM_STAT.ATSNS bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected.

Table 30-229: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/NW)	ATSSTM	Auxiliary Time Stamp Snapshot Trigger Missed. The EMAC_TM_STAT.ATSSTM bit is set when the auxiliary time stamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO. The EMAC_TM_STAT.ATSSTM bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected.	
		0	Status Not Detected
		1	Status Detected
19:16 (R/NW)	ATSSTN	Auxiliary Time Stamp Snapshot Trigger Identifier. The EMAC_TM_STAT.ATSSTN field identifies the auxiliary trigger inputs for which the time stamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list: - Bit 16: Auxiliary trigger 0 - Bit 17: Auxiliary trigger 1 - Bit 18: Auxiliary trigger 2 - Bit 19: Auxiliary trigger 3 The software can read this register to find the triggers that are set when the time stamp is taken.	
15 (R/NW)	TXTSSIS	Tx Time Stamp Status Interrupt Status. When the drop transmit status is enabled in MTL, the EMAC_TM_STAT.TXTSSIS bit is set when the captured transmit time stamp is updated in the EMAC_TXTMSTAT_NSEC and EMAC_TXTMSTAT_SEC registers. When PTP offload feature is enabled, the EMAC_TM_STAT.TXTSSIS bit is set when the captured transmit time stamp is updated in the EMAC_TXTMSTAT_NSEC and EMAC_TXTMSTAT_SEC registers, for PTO generated Delay Request and Pdelay request packets. The EMAC_TM_STAT.TXTSSIS bit is cleared when the EMAC_TXTMSTAT_SEC register is read (or write to EMAC_TXTMSTAT_SEC register when EMAC_CSR_SW_CTL.RCWE = 1).	
		0	Status Not Detected
		1	Status Detected

Table 30-229: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/NW)	TSTRGTERR3	Time Stamp Target Time Error. The EMAC_TM_STAT.TSTRGTERR3 bit is set when the latest target time programmed in the EMAC_PPS[3]_TGTM_SEC and EMAC_PPS[3]_TGTM_NSEC registers elapses. The EMAC_TM_STAT.TSTRGTERR3 bit is cleared when the application reads the EMAC_TM_STAT.TSTRGTERR3 bit. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTRGTERR3 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
8 (R/NW)	TSTARGT3	Time Stamp Target Time Reached for Target Time PPS3. When the EMAC_TM_STAT.TSTARGT3 bit is set and EMAC_PPS_CTL.MCGREN3 = 0, it indicates that the value of system time is greater than or equal to the value specified in the EMAC_PPS[3]_TGTM_SEC and EMAC_PPS[3]_TGTM_NSEC registers. When the EMAC_TM_STAT.TSTARGT3 bit is set and EMAC_PPS_CTL.MCGREN3 = 1, it indicates that EMAC0_MCGR3 trigger request is asserted. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTARGT3 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
7 (R/NW)	TSTRGTERR2	Time Stamp Target Time Error. The EMAC_TM_STAT.TSTRGTERR2 bit is set when the latest target time programmed in the EMAC_PPS[2]_TGTM_SEC and EMAC_PPS[2]_TGTM_NSEC registers elapses. The EMAC_TM_STAT.TSTRGTERR2 bit is cleared when the application reads the EMAC_TM_STAT.TSTRGTERR2 bit. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTRGTERR2 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected

Table 30-229: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/NW)	TSTARGT2	Time Stamp Target Time Reached for Target Time PPS2. When the EMAC_TM_STAT.TSTARGT2 bit is set and EMAC_PPS_CTL.MCGREN2 = 0, it indicates that the value of system time is greater than or equal to the value specified in the EMAC_PPS[2]_TGTM_SEC and EMAC_PPS[2]_TGTM_NSEC registers. When the EMAC_TM_STAT.TSTARGT2 bit is set and EMAC_PPS_CTL.MCGREN2 = 1, it indicates that EMAC0_MCGR2 trigger request is asserted. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTARGT2 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
5 (R/NW)	TSTRGTERR1	Time Stamp Target Time Error. The EMAC_TM_STAT.TSTRGTERR1 bit is set when the latest target time programmed in the EMAC_PPS[1]_TGTM_SEC and EMAC_PPS[1]_TGTM_NSEC registers elapses. The EMAC_TM_STAT.TSTRGTERR1 bit is cleared when the application reads the EMAC_TM_STAT.TSTRGTERR1 bit. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTRGTERR1 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
4 (R/NW)	TSTARGT1	Time Stamp Target Time Reached for Target Time PPS1. When the EMAC_TM_STAT.TSTARGT1 bit is set and EMAC_PPS_CTL.MCGREN1 = 0, it indicates that the value of system time is greater than or equal to the value specified in the EMAC_PPS[1]_TGTM_SEC and EMAC_PPS[1]_TGTM_NSEC registers. When the EMAC_TM_STAT.TSTARGT1 bit is set and EMAC_PPS_CTL.MCGREN1 = 1, it indicates that EMAC0_MCGR1 trigger request is asserted. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTARGT1 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected

Table 30-229: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	TSTRGTERR0	Time Stamp Target Time Error. The EMAC_TM_STAT.TSTRGTERR0 bit is set when the latest target time programmed in the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers elapses. The EMAC_TM_STAT.TSTRGTERR0 bit is cleared when the application reads the EMAC_TM_STAT.TSTRGTERR0 bit. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTRGTERR0 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
2 (R/NW)	AUXTSTRIG	Auxiliary Time Stamp Trigger Snapshot. The EMAC_TM_STAT.AUXTSTRIG bit is set (=1) when the auxiliary snapshot is written to the FIFO. The EMAC_TM_STAT.AUXTSTRIG bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected. Access restriction applies. Clears on read (or the EMAC_TM_STAT.AUXTSTRIG bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Auxiliary Timestamp Trigger Snapshot Status Not Detected
		1	Auxiliary Timestamp Trigger Snapshot Status Detected
1 (R/NW)	TSTARGT0	Time Stamp Target Time Reached. When the EMAC_TM_STAT.TSTARGT0 bit is set and EMAC_PPS_CTL.MCGREN0 = 0, it indicates that the value of system time is greater than or equal to the value specified in the EMAC_PPS[n]_TGTM_SEC and EMAC_PPS[n]_TGTM_NSEC registers. When the EMAC_TM_STAT.TSTARGT0 bit is set and EMAC_PPS_CTL.MCGREN0 = 1, it indicates that EMAC0_MCGR trigger request is asserted. Access restriction applies. Clears on read (or the EMAC_TM_STAT.TSTARGT0 bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected

Table 30-229: EMAC_TM_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	TSSOVF	Time Stamp Seconds Overflow. When the `EMAC_TM_STAT.TSSOVF` bit is set, it indicates that the seconds value of the time stamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF. Access restriction applies. Clears on read (or the `EMAC_TM_STAT.TSSOVF` bit is written to 1 when EMAC_CSR_SW_CTL.RCWE = 1). Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected

Transmit Queue Priority Mapping 0 Register

The EMAC_TXQ_PRTY_MAP0 register contains the priority values assigned to Tx queue 0 through Tx queue 3.

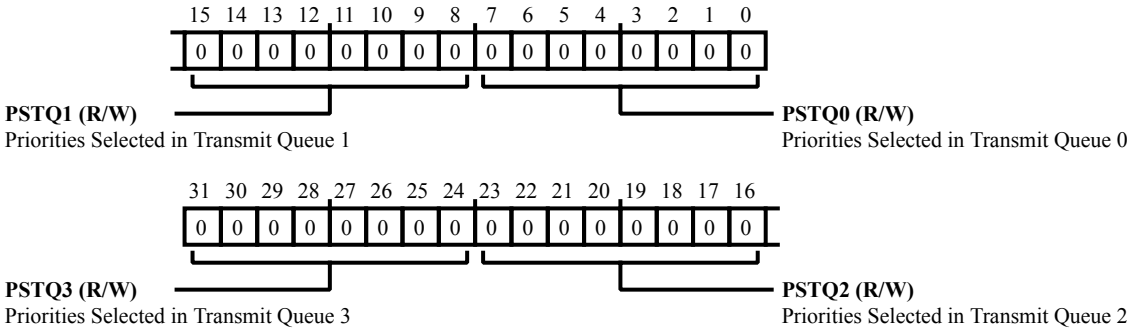


Figure 30-175: EMAC_TXQ_PRTY_MAP0 Register Diagram

Table 30-230: EMAC_TXQ_PRTY_MAP0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PSTQ3	Priorities Selected in Transmit Queue 3. The EMAC_TXQ_PRTY_MAP0.PSTQ3 bit is similar to the PSTQ0 bit.
23:16 (R/W)	PSTQ2	Priorities Selected in Transmit Queue 2. The EMAC_TXQ_PRTY_MAP0.PSTQ2 bit is similar to the PSTQ0 bit.
15:8 (R/W)	PSTQ1	Priorities Selected in Transmit Queue 1. The EMAC_TXQ_PRTY_MAP0.PSTQ1 bit is similar to the PSTQ0 bit.
7:0 (R/W)	PSTQ0	Priorities Selected in Transmit Queue 0. The EMAC_TXQ_PRTY_MAP0.PSTQ0 field holds the priorities assigned to Tx queue 0 by the software. The EMAC_TXQ_PRTY_MAP0.PSTQ0 field determines if Tx queue 0 should be blocked from transmitting specified pause time when a PFC packet is received with priorities matching the priorities programmed in this field. If the content of this field is not mutually exclusive to the corresponding fields of other Transmit queues, that is, same priority is mapped to multiple Tx queues, the MAC blocks all queues with matching priority, for the specified time.

Transmit Queue Priority Mapping 1 Register

The EMAC_TXQ_PRTY_MAP1 register contains the priority values assigned to Tx queue 4 through Tx queue 7.

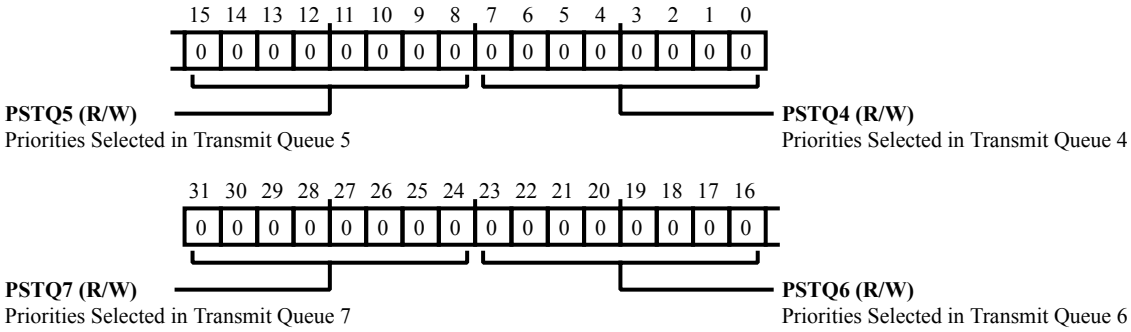


Figure 30-176: EMAC_TXQ_PRTY_MAP1 Register Diagram

Table 30-231: EMAC_TXQ_PRTY_MAP1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PSTQ7	Priorities Selected in Transmit Queue 7. The EMAC_TXQ_PRTY_MAP1.PSTQ7 bit is similar to the PSTQ4 bit.
23:16 (R/W)	PSTQ6	Priorities Selected in Transmit Queue 6. The EMAC_TXQ_PRTY_MAP1.PSTQ6 bit is similar to the PSTQ4 bit.
15:8 (R/W)	PSTQ5	Priorities Selected in Transmit Queue 5. The EMAC_TXQ_PRTY_MAP1.PSTQ5 bit is similar to the PSTQ4 bit.
7:0 (R/W)	PSTQ4	Priorities Selected in Transmit Queue 4. The EMAC_TXQ_PRTY_MAP1.PSTQ4 field holds the priorities assigned to Tx queue 4 by the software. The EMAC_TXQ_PRTY_MAP1.PSTQ4 field determines if Tx queue 4 should be blocked from transmitting specified pause time when a PFC packet is received with priorities matching the priorities programmed in this field. If the content of this field is not mutually exclusive to corresponding fields of other Transmit queues, that is, same priority is mapped to multiple Tx queues, the MAC blocks all queues with matching priority for specified time.

Transmit Time Stamp Status Nanoseconds Register

The EMAC_TXTMSTAT_NSEC register contains the nanosecond part of time stamp captured for transmit packets when Tx status is disabled. The EMAC_TXTMSTAT_NSEC register, along with EMAC_TXTMSTAT_SEC, gives the 64-bit time stamp captured for the PTP packet successfully transmitted by the MAC. This value is considered to be read by the application when the last byte of the EMAC_TXTMSTAT_NSEC register is read. In the little-endian mode, this means when bits[31:24] are read; in big-endian mode, bits[7:0] are read. If the application does not read these registers and time stamp of another packet is captured, then either the current time stamp is lost (overwritten) or the new time stamp is lost (dropped), depending on the configuration of the EMAC_TM_CTL.TXTSSTSM bit. The status bit EMAC_TM_STAT.TXTSSIS is set when the MAC transmitter captures the time stamp.

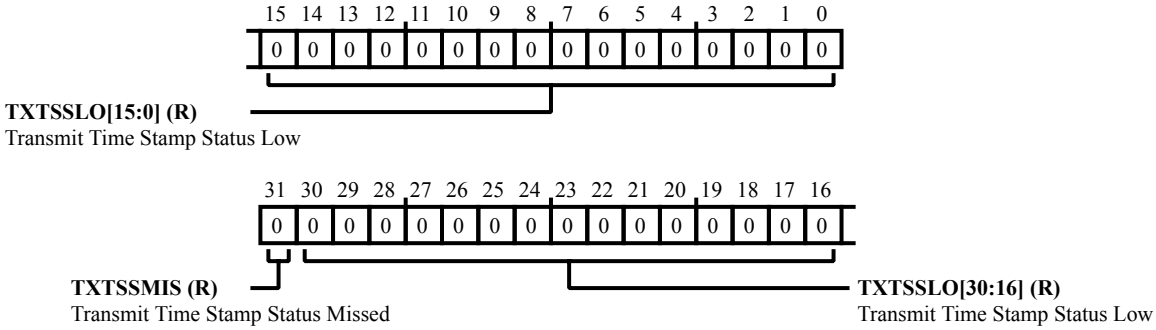


Figure 30-177: EMAC_TXTMSTAT_NSEC Register Diagram

Table 30-232: EMAC_TXTMSTAT_NSEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	TXTSSMIS	Transmit Time Stamp Status Missed. When the EMAC_TXTMSTAT_NSEC.TXTSSMIS bit is set, it indicates one of the following: The time stamp of the current packet is ignored if EMAC_TM_CTL.TXTSSTSM = 0. The time stamp of the previous packet is overwritten with time stamp of the current packet if EMAC_TM_CTL.TXTSSTSM = 1. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Status Not Detected
		1	Status Detected
30:0 (R/NW)	TXTSSLO	Transmit Time Stamp Status Low. The EMAC_TXTMSTAT_NSEC.TXTSSLO field contains the 31 bits of the nanoseconds field of the transmit packet's captured time stamp.	

Transmit Time Stamp Status Seconds Register

The EMAC_TXTMSTAT_SEC register contains the higher 32 bits of the time stamp (in seconds) captured when a PTP packet is transmitted.

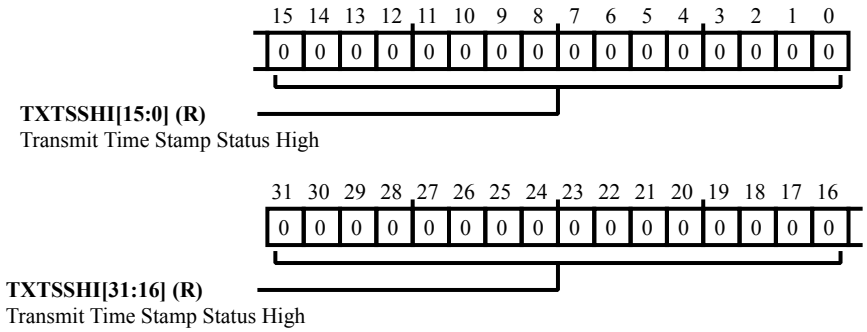


Figure 30-178: EMAC_TXTMSTAT_SEC Register Diagram

Table 30-233: EMAC_TXTMSTAT_SEC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXTSSHI	Transmit Time Stamp Status High. The EMAC_TXTMSTAT_SEC.TXTSSHI field contains the lower 32 bits of the seconds field of transmit packet's captured time stamp.

VLAN Hash Table Register

When EMAC_VLANTAG_CTL.VTHM = 1, the 16-bit EMAC_VLAN_HASHTBL register is used for group address filtering based on the VLAN tag. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on EMAC_VLANTAG_CTL.ETV) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated hash value are used to index the contents of the VLAN hash table. For example, hash value of 4b'1000 selects bit 8 of the VLAN hash table.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).

2. Perform bitwise reversal for the value obtained in step 1.

3. Take the upper four bits from the value obtained in step 2. The VLAN hash table register is configured to be double-synchronized to the (G)MII clock domain, the synchronization is triggered only when bits[15:8] (in little-endian mode) or bits[7:0] (in big-endian mode) of this register are written.

Consecutive writes to this register should be performed after at least four clock cycles in the (G)MII clock domain.

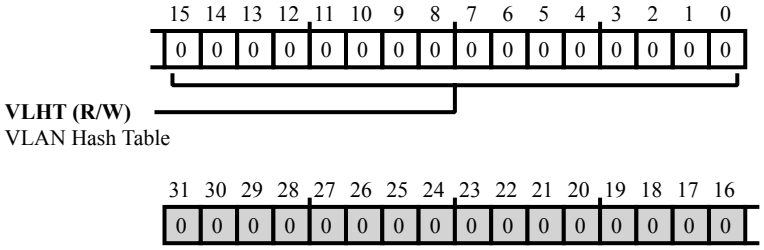


Figure 30-179: EMAC_VLAN_HASHTBL Register Diagram

Table 30-234: EMAC_VLAN_HASHTBL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VLHT	VLAN Hash Table. The EMAC_VLAN_HASHTBL.VLHT field contains the 16-bit VLAN Hash Table.

VLAN Tag Inclusion or Replacement Register

The EMAC_VLAN_INCL register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

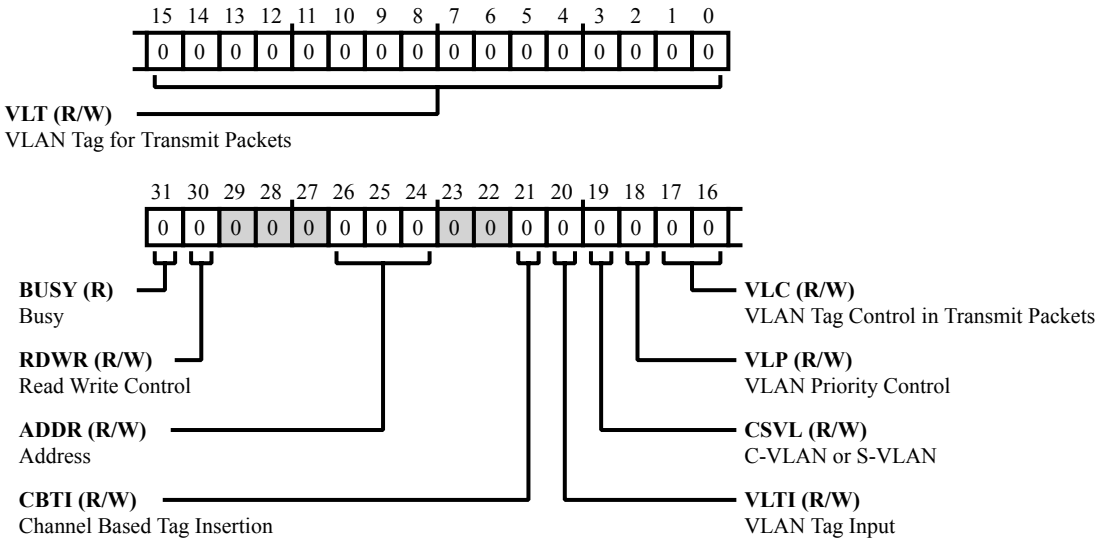


Figure 30-180: EMAC_VLAN_INCL Register Diagram

Table 30-235: EMAC_VLAN_INCL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/NW)	BUSY	Busy. The EMAC_VLAN_INCL.BUSY bit indicates the status of the read/write operation of indirect access to the queue/channel specific VLAN inclusion register. For write operation write to a register is complete when EMAC_VLAN_INCL.BUSY = 0. For read operation the read data is valid when the bit is reset. The application must make sure that EMAC_VLAN_INCL.BUSY = 0 before attempting subsequent access to this register.	
		0	Busy Status Not Detected
		1	Busy Status Detected
30 (R/W)	RDWR	Read Write Control. The EMAC_VLAN_INCL.RDWR bit controls the read or write operation for indirectly accessing the queue/channel specific VLAN Inclusion register. When set indicates write operation and when cleared (=0) indicates read operation. This does not have any effect when CBTI is reset.	
		0	Read Operation of Indirect Access
		1	Write Operation of Indirect Access

Table 30-235: EMAC_VLAN_INCL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26:24 (R/W)	ADDR	Address. The EMAC_VLAN_INCL.ADDR field selects one of the queue/channel specific VLAN Inclusion register for read/write access. This does not have any effect when CBTI is reset.	
21 (R/W)	CBTI	Channel Based Tag Insertion. When the EMAC_VLAN_INCL.CBTI bit is set, outer VLAN tag is inserted for every packets transmitted by the MAC. The tag value is taken from the queue/channel specific VLAN tag register. The VLTI, VLP, VLC, and VLT fields of this register are ignored when the EMAC_VLAN_INCL.CBTI bit is set. When the EMAC_VLAN_INCL.CBTI bit is set, a write operation to byte 3 of this register initiates the read/write access to the indirect register. When cleared (=0), outer VLAN operation is based on the setting of VLTI, VLP, VLC and VLT fields of this register.	
		0	Channel Based Tag Insertion is Disabled
		1	Channel Based Tag Insertion is Enabled
20 (R/W)	VLTI	VLAN Tag Input. When the EMAC_VLAN_INCL.VLTI bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from: - The Tx descriptor	
		0	VLAN Tag Input is Disabled
		1	VLAN Tag Input is Enabled
19 (R/W)	CSVL	C-VLAN or S-VLAN. When the EMAC_VLAN_INCL.CSVL bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When EMAC_VLAN_INCL.CSVL = 0, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.	
		0	C-VLAN Type (0x8100) is Inserted or Replaced
		1	S-VLAN Type (0x88A8) is Inserted or Replaced
18 (R/W)	VLP	VLAN Priority Control. When the EMAC_VLAN_INCL.VLP bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When EMAC_VLAN_INCL.VLP = 0, the mti_vlan_ctrl_i control input is used and bits[17:16] are ignored.	
		0	VLAN Priority Control is Disabled
		1	VLAN Priority Control is Enabled

Table 30-235: EMAC_VLAN_INCL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17:16 (R/W)	VLC	VLAN Tag Control in Transmit Packets. The `EMAC_VLAN_INCL.VLC` field indicates the VLAN tag control in transmit packets. 2'b00: No VLAN tag deletion, insertion, or replacement 2'b01: VLAN tag deletion The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags. 2'b10: VLAN tag insertion The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 2'b11: VLAN tag replacement The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8). Note: Changes to this field take effect only on the start of a packet. If this field is written when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.	
		0	No VLAN Tag Deletion, Insertion, or Replacement
		1	VLAN Tag Deletion
		2	VLAN Tag Insertion
		3	VLAN Tag Replacement
15:0 (R/W)	VLT	VLAN Tag for Transmit Packets. The `EMAC_VLAN_INCL.VLT` field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and Bits[11:0] are the VID field in the VLAN tag. The following list describes the bits of this field: Bits[15:13]: User Priority Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) Bits[11:0]: VLAN Identifier (VID) field of VLAN tag	

VLAN Tag Control Register

The EMAC_VLANTAG_CTL register contains control and status bits for VLAN filtering operation. It is also used for indirect addressing. It contains the address offset, command type and busy bit for CSR access of the per VLAN tag registers.

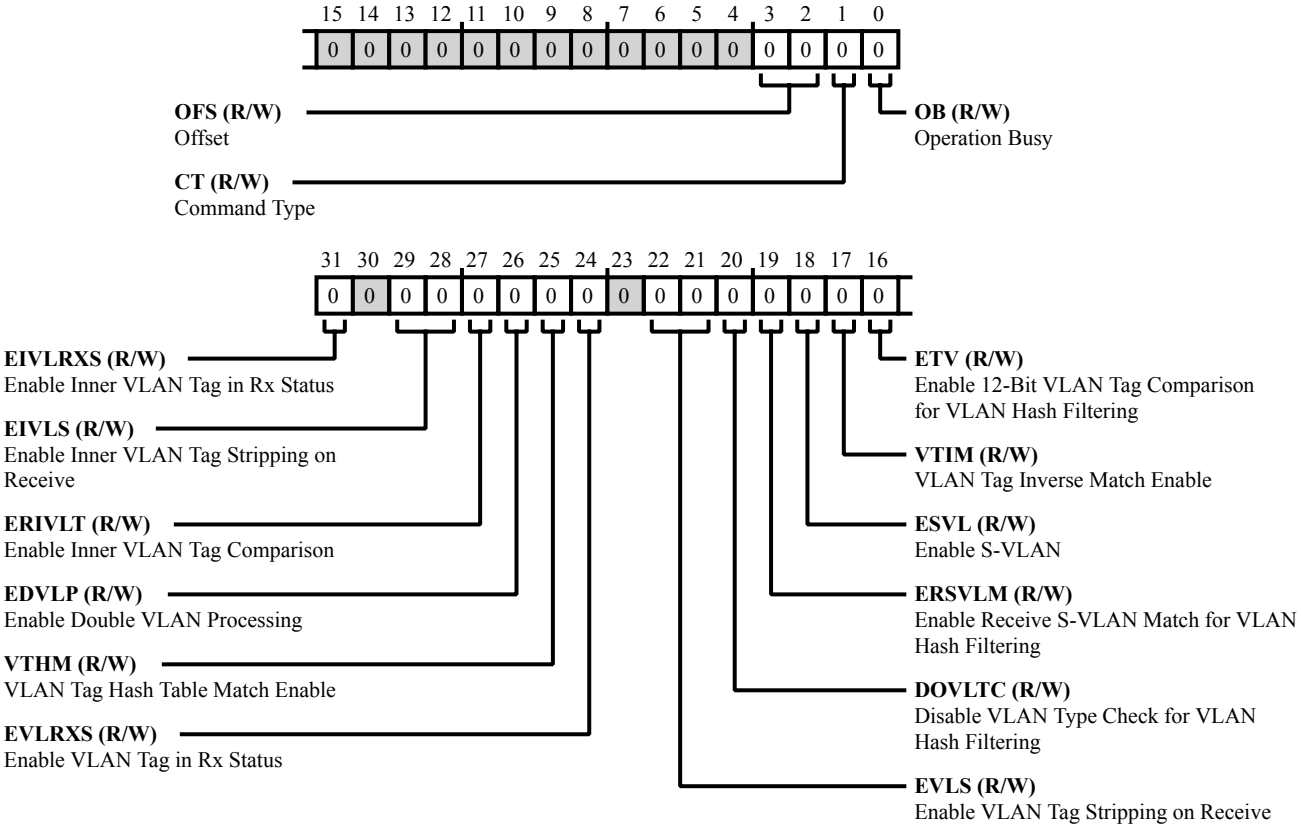


Figure 30-181: EMAC_VLANTAG_CTL Register Diagram

Table 30-236: EMAC_VLANTAG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EIVLRXS	Enable Inner VLAN Tag in Rx Status. When the EMAC_VLANTAG_CTL.EIVLRXS bit is set, the MAC provides the inner VLAN tag in the Rx status. When EMAC_VLANTAG_CTL.EIVLRXS = 0, the MAC does not provide the inner VLAN tag in Rx status.	
		0	Inner VLAN Tag in Rx Status is Disabled
		1	Inner VLAN Tag in Rx Status is Enabled

Table 30-236: EMAC_VLANTAG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29:28 (R/W)	EIVLS	Enable Inner VLAN Tag Stripping on Receive. The EMAC_VLANTAG_CTL.EIVLS field indicates the stripping operation on inner VLAN tag in received packet.	
		0	Do Not Strip
		1	Strip If VLAN Filter Passes
		2	Strip If VLAN Filter Fails
		3	Always Strip
27 (R/W)	ERIVLT	Enable Inner VLAN Tag Comparison. When the EMAC_VLANTAG_CTL.ERIVLT bit, EMAC_VLANTAG_CTL.VTHM bit, and the EMAC_VLANTAG_CTL.EDVLP field are set, the MAC receiver enables VLAN hash filtering operation on the inner VLAN tag (if present). When EMAC_VLANTAG_CTL.ERIVLT = 0 and EMAC_VLANTAG_CTL.VTHM = 1, the MAC receiver enables VLAN hash filtering operation on the outer VLAN tag (if present). The EMAC_VLANTAG_CTL.ERSVLM bit and EMAC_VLANTAG_CTL.DOVLTC bit determines which VLAN type is enabled for filtering.	
		0	Inner VLAN Tag is Disabled
		1	Inner VLAN Tag is Enabled
26 (R/W)	EDVLP	Enable Double VLAN Processing. When the EMAC_VLANTAG_CTL.EDVLP bit is set, the MAC enables processing of up to two VLAN tags on Tx and Rx (if present). When EMAC_VLANTAG_CTL.EDVLP = 0, the MAC enables processing of up to one VLAN tag on Tx and Rx (if present).	
		0	Double VLAN Processing is Disabled
		1	Double VLAN Processing is Enabled

Table 30-236: EMAC_VLANTAG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/W)	VTHM	VLAN Tag Hash Table Match Enable. When the EMAC_VLANTAG_CTL.VTHM bit is set, the most significant four bits of CRC of VLAN tag are used to index the content of the EMAC_VLAN_HASHTBL_REGn. A value of 1 in the VLAN hash table register, corresponding to the index, indicates that the packet matched the VLAN hash table. When EMAC_VLANTAG_CTL.ETV= 1, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison. When EMAC_VLANTAG_CTL.ETV= 0, the CRC of the 16-bit VLAN tag is used for comparison. When EMAC_VLANTAG_CTL.VTHM = 0, the VLAN Hash match operation is not performed.	
		0	VLAN Tag Hash Table Match is Disabled
		1	VLAN Tag Hash Table Match is Enabled
24 (R/W)	EVLRXS	Enable VLAN Tag in Rx Status. When the EMAC_VLANTAG_CTL.EVLRXS bit is set, MAC provides the outer VLAN Tag in the Rx status. When EMAC_VLANTAG_CTL.EVLRXS = 0, the MAC does not provide the outer VLAN tag in Rx status.	
		0	VLAN Tag in Rx Status is Disabled
		1	VLAN Tag in Rx Status is Enabled
22:21 (R/W)	EVLS	Enable VLAN Tag Stripping on Receive. The EMAC_VLANTAG_CTL.EVLS field indicates the stripping operation on the outer VLAN tag in received packet.	
		0	Do Not Strip
		1	Strip If VLAN Filter Passes
		2	Strip If VLAN Filter Fails
		3	Always Strip
20 (R/W)	DOVLTC	Disable VLAN Type Check for VLAN Hash Filtering. When the EMAC_VLANTAG_CTL.DOVLTC bit is set, the MAC VLAN hash filter does not check whether the VLAN tag specified by the EMAC_VLANTAG_CTL.ERIVLT bit is of type S-VLAN or C-VLAN. When EMAC_VLANTAG_CTL.DOVLTC = 0, the MAC VLAN hash filter filters or matches the VLAN tag specified by the EMAC_VLANTAG_CTL.ERIVLT bit only when VLAN tag type is similar to the one specified by the EMAC_VLANTAG_CTL.ERSVLM bit.	
		0	VLAN Type Check is Enabled
		1	VLAN Type Check is Disabled

Table 30-236: EMAC_VLANTAG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	ERSVLM	Enable Receive S-VLAN Match for VLAN Hash Filtering. When the EMAC_VLANTAG_CTL.ERSVLM bit is set, the MAC receiver enables VLAN hash filtering or matching for S-VLAN (type = 0x88A8) packets. When EMAC_VLANTAG_CTL.ERSVLM = 0, the MAC receiver enables VLAN hash filtering or matching for C-VLAN (type = 0x8100) packets. The EMAC_VLANTAG_CTL.ERIVLT bit determines the VLAN tag position considered for VLAN hash filtering or matching.	
		0	Receive S-VLAN Match is Disabled
		1	Receive S-VLAN Match is Enabled
18 (R/W)	ESVL	Enable S-VLAN. When the EMAC_VLANTAG_CTL.ESVL bit is set, the MAC transmitter and receiver consider the S-VLAN packets (type = 0x88A8) as valid VLAN tagged packets.	
		0	S-VLAN is Disabled
		1	S-VLAN is Enabled
17 (R/W)	VTIM	VLAN Tag Inverse Match Enable. When the EMAC_VLANTAG_CTL.VTIM bit is set, the EMAC_VLANTAG_CTL.VTIM bit enables the VLAN tag inverse matching. The packets without matching VLAN tag are marked as matched. When cleared (=0), the EMAC_VLANTAG_CTL.VTIM bit enables the VLAN tag perfect matching. The packets with matched VLAN tag are marked as matched.	
		0	VLAN Tag Inverse Match is Disabled
		1	VLAN Tag Inverse Match is Enabled
16 (R/W)	ETV	Enable 12-Bit VLAN Tag Comparison for VLAN Hash Filtering. When the EMAC_VLANTAG_CTL.ETV bit is set, a 12-bit VLAN identifier is used for VLAN hash filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag in the received VLAN-tagged packet are used for hash-based VLAN filtering. When EMAC_VLANTAG_CTL.ETV = 0, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for VLAN hash filtering.	
		0	12-Bit VLAN Tag Comparison is Disabled
		1	12-Bit VLAN Tag Comparison is Enabled
3:2 (R/W)	OFS	Offset. The EMAC_VLANTAG_CTL.OFS field holds the address offset of the MAC VLAN Tag Filter Register which the application is trying to access. The width of the field depends on the number of MAC VLAN tag registers enabled.	

Table 30-236: EMAC_VLANTAG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	CT	Command Type. The EMAC_VLANTAG_CTL.CT bit indicates if the current register access is a read or a write. When set, the EMAC_VLANTAG_CTL.CT bit indicates a read operation. When cleared (=0), it indicates a write operation.	
		0	Write Operation
		1	Read Operation
0 (R/W)	OB	Operation Busy. The EMAC_VLANTAG_CTL.OB bit is set along with a read or write command for initiating the indirect access to per VLAN Tag Filter register. The EMAC_VLANTAG_CTL.OB bit is reset when the read or write command to per VLAN Tag Filter indirect access register is complete. The next indirect register access can be initiated only after EMAC_VLANTAG_CTL.OB = 0. During a write operation, the bit is reset only after the data has been written into the Per VLAN Tag register. During a read operation, the data should be read from the EMAC_VLANTAG_DAT register only after EMAC_VLANTAG_CTL.OB = 0.	
		0	Operation Busy is Disabled
		1	Operation Busy is Enabled

VLAN Tag Indirect Access Data Register

The EMAC_VLANTAG_DAT register holds the read/write data for Indirect Access of the Per VLAN Tag registers. During the read access, this field contains valid read data only after the OB bit is reset. During the write access, this field should be valid prior to EMAC_VLANTAG_CTL.OB = 1

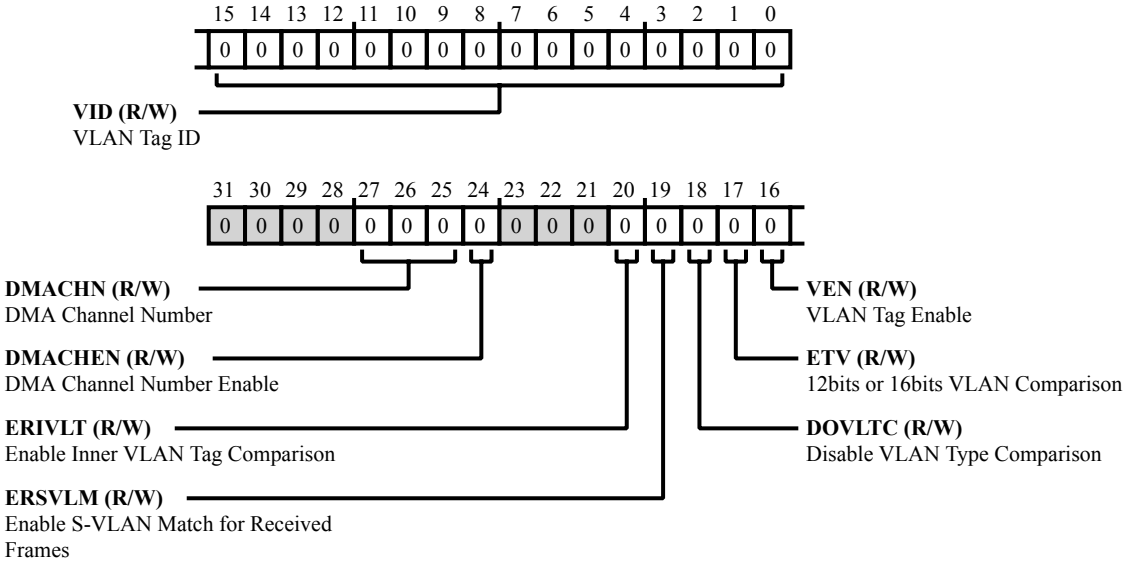


Figure 30-182: EMAC_VLANTAG_DAT Register Diagram

Table 30-237: EMAC_VLANTAG_DAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27:25 (R/W)	DMACHN	DMA Channel Number. The EMAC_VLANTAG_DAT.DMACHN field indicates the DMA channel number to which the VLAN tagged frame is to be routed, when the routing is based on the VLAN tag filter. If the routing is not based on the VLAN tag filter, this field does not need to be programmed.	
24 (R/W)	DMACHEN	DMA Channel Number Enable. The EMAC_VLANTAG_DAT.DMACHEN bit is the enable for the DMA channel number value programmed in the field DMACH. When EMAC_VLANTAG_DAT.DMACHEN = 0, the routing does not occur based on VLAN filter result. The frame is routed based on DA-based DMA channel routing.	
		0	DMA Channel Number is Disabled
		1	DMA Channel Number is Enabled

Table 30-237: EMAC_VLANTAG_DAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	ERIVLT	Enable Inner VLAN Tag Comparison. The EMAC_VLANTAG_DAT.ERIVLT bit is valid only when the Double VLAN Tag Enable of the filter is set. When the EMAC_VLANTAG_DAT.ERIVLT bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN tag (if present). When EMAC_VLANTAG_DAT.ERIVLT = 0, the MAC receiver enables operation on the outer VLAN tag (if present).	
		0	Inner VLAN Tag Comparison is Disabled
		1	Inner VLAN Tag Comparison is Enabled
19 (R/W)	ERSVLM	Enable S-VLAN Match for Received Frames. The EMAC_VLANTAG_DAT.ERSVLM bit is valid only when VLAN Tag enable of the filter is set. When the EMAC_VLANTAG_DAT.ERSVLM bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When EMAC_VLANTAG_DAT.ERSVLM = 0, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.	
		0	Receive S-VLAN Match is Disabled
		1	Receive S-VLAN Match is Enabled
18 (R/W)	DOVLTC	Disable VLAN Type Comparison. The EMAC_VLANTAG_DAT.DOVLTC bit is valid only when VLAN tag enable of the filter is set. When the EMAC_VLANTAG_DAT.DOVLTC bit is set, the MAC does not check whether the VLAN tag specified by the Enable Inner VLAN Tag Comparison bit is of type S-VLAN or C-VLAN. When EMAC_VLANTAG_DAT.DOVLTC = 0, the MAC filters or matches the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit only when VLAN Tag type is similar to the one specified by the Enable S-VLAN match for received frames bit.	
		0	VLAN Type Comparison is Enabled
		1	VLAN Type Comparison is Disabled

Table 30-237: EMAC_VLANTAG_DAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	ETV	12bits or 16bits VLAN Comparison. The EMAC_VLANTAG_DAT.ETV bit is valid only when VEN of the filter is set. When the EMAC_VLANTAG_DAT.ETV bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet.	
		0	16 Bit VLAN Comparison
		1	12 Bit VLAN Comparison
16 (R/W)	VEN	VLAN Tag Enable. The EMAC_VLANTAG_DAT.VEN bit is used to enable or disable the VLAN tag. When the EMAC_VLANTAG_DAT.VEN bit is set, the MAC compares the VLAN tag of received packet with the VLAN Tag ID. When EMAC_VLANTAG_DAT.VEN = 0, no comparison is performed irrespective of the programming of the other fields.	
		0	VLAN Tag is Disabled
		1	VLAN Tag is Enabled
15:0 (R/W)	VID	VLAN Tag ID. The EMAC_VLANTAG_DAT.VID field holds the VLAN tag value which is used by the MAC for perfect comparison. It is valid when VLAN Tag Enable is set.	

Watchdog Timeout Register

The EMAC_WDT_TMOUT register controls the watchdog timeout for received packets.

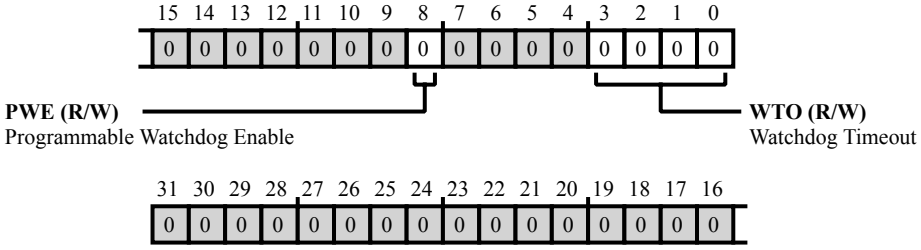


Figure 30-183: EMAC_WDT_TMOUT Register Diagram

Table 30-238: EMAC_WDT_TMOUT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	PWE	Programmable Watchdog Enable. When the EMAC_WDT_TMOUT.PWE bit is set and the EMAC_MAC_CFG.WD = 0, the EMAC_WDT_TMOUT.WTO field is used as watchdog timeout for a received packet. When the EMAC_WDT_TMOUT.PWE bit is cleared, the watchdog timeout for a received packet is controlled by EMAC_MAC_CFG.WD and EMAC_MAC_CFG.JE = 1.	
		0	Programmable Watchdog is Disabled
		1	Programmable Watchdog is Enabled
3:0 (R/W)	WTO	Watchdog Timeout. When the EMAC_WDT_TMOUT.PWE bit is set and EMAC_MAC_CFG.WD = 0, the EMAC_WDT_TMOUT.WTO field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet. Note: When the EMAC_WDT_TMOUT.PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.	
		0	2 KB
		1	3 KB
		2	4 KB
		3	5 KB
		4	6 KB
		5	7 KB
		6	8 KB

Table 30-238: EMAC_WDT_TMOUT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		7	9 KB
		8	10 KB
		9	11 KB
		10	12 KB
		11	13 KB
		12	14 KB
		13	15 KB
		14	16383 Bytes
		15	Reserved

MMC Control Register

The EMAC_MMC_CTRL register establishes the operating mode of MMC.

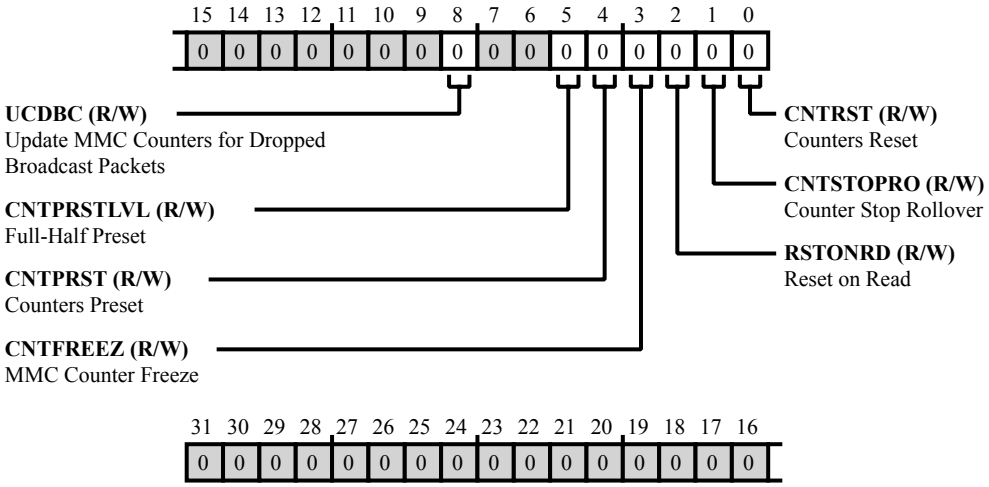


Figure 30-184: EMAC_MMC_CTRL Register Diagram

Table 30-239: EMAC_MMC_CTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	UCDBC	Update MMC Counters for Dropped Broadcast Packets. Note: The EMAC_MMC_CTRL.CNTRST bit has a higher priority than the EMAC_MMC_CTRL.CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the EMAC_MMC_CTRL.CNTPRST bit is not set. When EMAC_MMC_CTRL.UCDBC is set (=1), the MAC updates all related MMC counters for broadcast packets that are dropped because of the EMAC_MACPKT_FILT.DBF bit. When EMAC_MMC_CTRL.UCDBC is cleared (=0), the MMC counters are not updated for dropped broadcast packets.	
		0	Update MMC Counters for Dropped Broadcast Packets is Disabled
		1	Update MMC Counters for Dropped Broadcast Packets is Enabled

Table 30-239: EMAC_MMC_CTRL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	CNTPRSTLVL	Full-Half Preset. When EMAC_MMC_CTRL.CNTPRSTLVL = 0 and EMAC_MMC_CTRL.CNTPRST = 1, all MMC counters are preset to almost-half value. All octet counters are preset to 0x7FFF_F800 (Half 2KB) and all packet-counters gets preset to 0x7FFF_FFF0 (Half 16). When EMAC_MMC_CTRL.CNTPRSTLVL= 1 and EMAC_MMC_CTRL.CNTPRST = 1, all MMC counters are preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (Full 2KB) and all packet-counters gets preset to 0xFFFF_FFF0 (Full 16). For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.	
		0	Full-Half Preset is Disabled
		1	Full-Half Preset is Enabled
4 (R/W)	CNTPRST	Counters Preset. When the EMAC_MMC_CTRL.CNTPRST bit is set, all counters are initialized or preset to almost full or almost half according to the EMAC_MMC_CTRL.CNTPRSTLVL bit. The EMAC_MMC_CTRL.CNTPRST bit is cleared automatically after 1 clock cycle. The EMAC_MMC_CTRL.CNTPRST bit, along with the EMAC_MMC_CTRL.CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full. Access restriction applies. Self-cleared. Write 0 to clear. Write 1 to set.	
		0	Counters Preset is Disabled
		1	Counters Preset is Enabled
3 (R/W)	CNTFREEZ	MMC Counter Freeze. When the EMAC_MMC_CTRL.CNTFREEZ bit is set (=1), it freezes all MMC counters to their current value. Until EMAC_MMC_CTRL.CNTFREEZ = 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the reset on read bit set, then that counter is also cleared in this mode.	
		0	MMC Counter Freeze is Disabled
		1	MMC Counter Freeze is Enabled

Table 30-239: EMAC_MMC_CTRL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	RSTONRD	Reset on Read. When the `EMAC_MMC_CTRL.RSTONRD` bit is set (=1), the MMC counters are reset to zero after read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[7:0]) is read.	
		0	Reset on Read is Disabled
		1	Reset on Read is Enabled
1 (R/W)	CNTSTOPRO	Counter Stop Rollover. When the `EMAC_MMC_CTRL.CNTSTOPRO` bit is set (=1), the counter does not roll over to zero after reaching the maximum value.	
		0	Counter Stop Rollover is Disabled
		1	Counter Stop Rollover is Enabled
0 (R/W)	CNTRST	Counters Reset. When the `EMAC_MMC_CTRL.CNTRST` bit is set (=1), all counters are reset. The `EMAC_MMC_CTRL.CNTRST` bit is cleared automatically after 1 clock cycle. Access restriction applies. Self-cleared. Write 0 to clear. Write 1 to set.	
		0	Counters are Not Reset
		1	All Counters are Reset

MMC Receive FPE Interrupt Status Register

The EMAC_MMC_FPE_RXINT register maintains the interrupts generated from all FPE-related receive statistics counters. The EMAC_MMC_FPE_RXINT register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000_0000 for the 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When the Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones.

The EMAC_MMC_FPE_RXINT register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

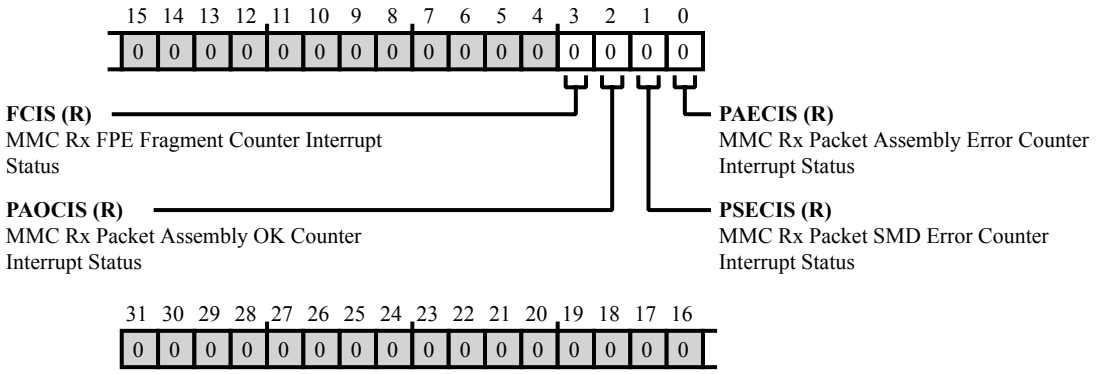


Figure 30-185: EMAC_MMC_FPE_RXINT Register Diagram

Table 30-240: EMAC_MMC_FPE_RXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	FCIS	MMC Rx FPE Fragment Counter Interrupt Status. The EMAC_MMC_FPE_RXINT.FCIS bit is set when the EMAC_RXFPE_FRGMNT_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx FPE Fragment Counter Interrupt Status Not Detected
		1	MMC Rx FPE Fragment Counter Interrupt Status Detected

Table 30-240: EMAC_MMC_FPE_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	PAOCIS	MMC Rx Packet Assembly OK Counter Interrupt Status. The `EMAC_MMC_FPE_RXINT.PAOCIS` bit is set when the EMAC_RXPKT_ASOK_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet Assembly OK Counter Interrupt Status Not Detected
		1	MMC Rx Packet Assembly OK Counter Interrupt Status Detected
1 (R/NW)	PSECIS	MMC Rx Packet SMD Error Counter Interrupt Status. The `EMAC_MMC_FPE_RXINT.PSECIS` bit is set when the EMAC_RXPKT_SMDERR_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet SMD Error Counter Interrupt Status Not Detected
		1	MMC Rx Packet SMD Error Counter Interrupt Status Detected
0 (R/NW)	PAECIS	MMC Rx Packet Assembly Error Counter Interrupt Status. The `EMAC_MMC_FPE_RXINT.PAECIS` bit is set when the EMAC_RXPKT_ASERR_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet Assembly Error Counter Interrupt Status Not Detected
		1	MMC Rx Packet Assembly Error Counter Interrupt Status Detected

MMC Receive FPE Interrupt Mask Register

The EMAC_MMC_FPE_RXIMSK register maintains the masks for interrupts generated from all FPE-related receive statistics counters. The EMAC_MMC_RXIMSK register maintains the masks for the interrupts generated when FPE-related receive statistic counters reach half of their maximum value or the maximum values. The EMAC_MMC_FPE_RXIMSK register is 32 bit wide.

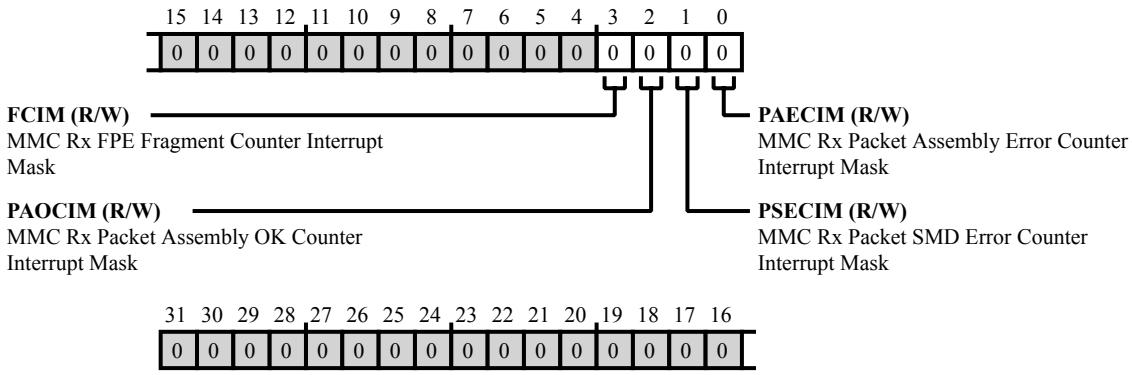


Figure 30-186: EMAC_MMC_FPE_RXIMSK Register Diagram

Table 30-241: EMAC_MMC_FPE_RXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	FCIM	MMC Rx FPE Fragment Counter Interrupt Mask. Setting the EMAC_MMC_FPE_RXIMSK.FCIM bit masks the interrupt when the EMAC_TXFPE_FRGMNT_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx FPE Fragment Counter Interrupt Mask is Disabled
		1	MMC Rx FPE Fragment Counter Interrupt Mask is Enabled

Table 30-241: EMAC_MMC_FPE_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	PAOCIM	MMC Rx Packet Assembly OK Counter Interrupt Mask. Setting the `EMAC_MMC_FPE_RXIMSK.PAOCIM` bit masks the interrupt when the EMAC_RXPKT_ASOK_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.	
		0	MMC Rx Packet Assembly OK Counter Interrupt Mask is Disabled
		1	MMC Rx Packet Assembly OK Counter Interrupt Mask is Enabled
1 (R/W)	PSECIM	MMC Rx Packet SMD Error Counter Interrupt Mask. Setting the `EMAC_MMC_FPE_RXIMSK.PSECIM` bit masks the interrupt when the EMAC_RXPKT_SMDERR_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet SMD Error Counter Interrupt Mask is Disabled
		1	MMC Rx Packet SMD Error Counter Interrupt Mask is Enabled
0 (R/W)	PAECIM	MMC Rx Packet Assembly Error Counter Interrupt Mask. Setting the `EMAC_MMC_FPE_RXIMSK.PAECIM` bit masks the interrupt when the EMAC_RXPKT_ASERR_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Rx Packet Assembly Error Counter Interrupt Mask is Disabled
		1	MMC Rx Packet Assembly Error Counter Interrupt Mask is Enabled

MMC Transmit FPE Interrupt Status Register

The EMAC_MMC_FPE_TXINT register maintains the interrupts generated from all FPE related Transmit statistics counters. The MMC FPE Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones. The MMC FPE Transmit Interrupt register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

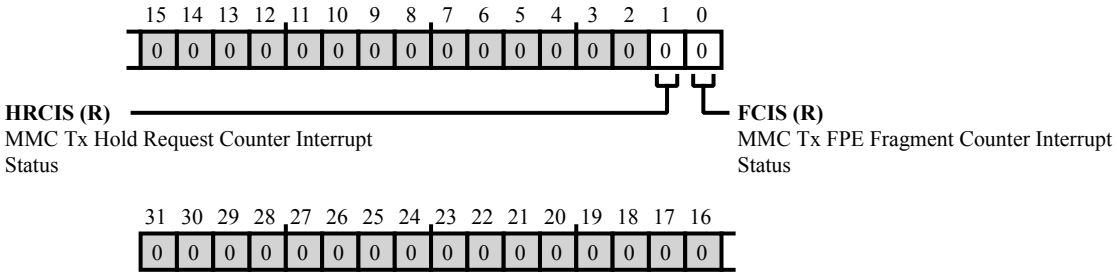


Figure 30-187: EMAC_MMC_FPE_TXINT Register Diagram

Table 30-242: EMAC_MMC_FPE_TXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	HRCIS	MMC Tx Hold Request Counter Interrupt Status. The EMAC_MMC_FPE_TXINT.HRCIS bit is set when the EMAC_TXHLDRQ_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE with AV_EST Enabled configuration.	
		0	MMC Tx Hold Request Counter Interrupt Status Not Detected
		1	MMC Tx Hold Request Counter Interrupt Status Detected

Table 30-242: EMAC_MMC_FPE_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	FCIS	MMC Tx FPE Fragment Counter Interrupt Status. The `EMAC_MMC_FPE_TXINT.FCIS` bit is set when the EMAC_TXFPE_FRGMNT_CNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Tx FPE Fragment Counter Interrupt Status Not Detected
		1	MMC Tx FPE Fragment Counter Interrupt Status Detected

MMC Transmit FPE Interrupt Mask Register

The EMAC_MMC_FPE_TXIMSK register maintains the masks for interrupts generated from all FPE related Transmit statistics counters. The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when FPE related receive statistic counters reach half of their maximum value or the maximum values. The EMAC_MMC_FPE_TXIMSK register is 32 bit wide.

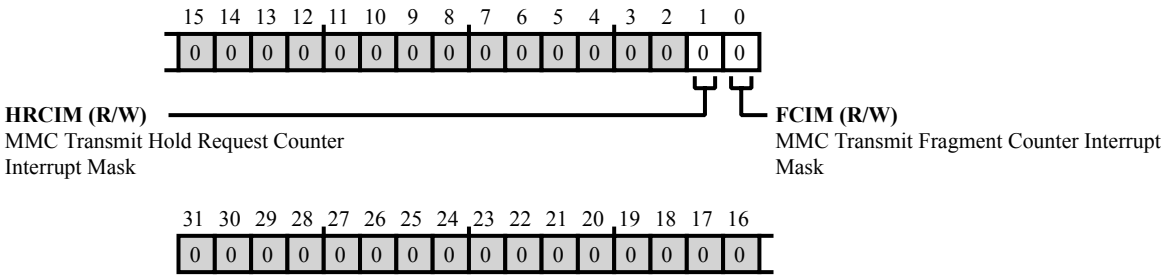


Figure 30-188: EMAC_MMC_FPE_TXIMSK Register Diagram

Table 30-243: EMAC_MMC_FPE_TXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	HRCIM	MMC Transmit Hold Request Counter Interrupt Mask. Setting the EMAC_MMC_FPE_TXIMSK.HRCIM bit masks the interrupt when the EMAC_TXHLDRQ_CNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Hold Request Counter Interrupt Mask is Disabled
		1	MMC Transmit Hold Request Counter Interrupt Mask is Enabled
0 (R/W)	FCIM	MMC Transmit Fragment Counter Interrupt Mask. Setting the EMAC_MMC_FPE_TXIMSK.FCIM bit masks the interrupt when the EMAC_TXFPE_FRGMNT_CNT counter reaches half of the maximum value or the maximum value. Exists when any one of the RX/TX MMC counters are enabled during FPE-enabled configuration.	
		0	MMC Transmit Fragment Counter Interrupt Mask is Disabled
		1	MMC Transmit Fragment Counter Interrupt Mask is Enabled

MMC Receive IPC Interrupt Status Register

The EMAC_MMC_IPC_RXINT register maintains the interrupt that the receive IPC statistic counters generate.

The EMAC_MMC_IPC_RXINT register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32 bit counter and 0xFFFF for 16 bit counter). When Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones.

The EMAC_MMC_IPC_RXINT register is 32 bit wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The counter's least-significant byte lane (Bits[7:0]) must be read to clear the interrupt bit.

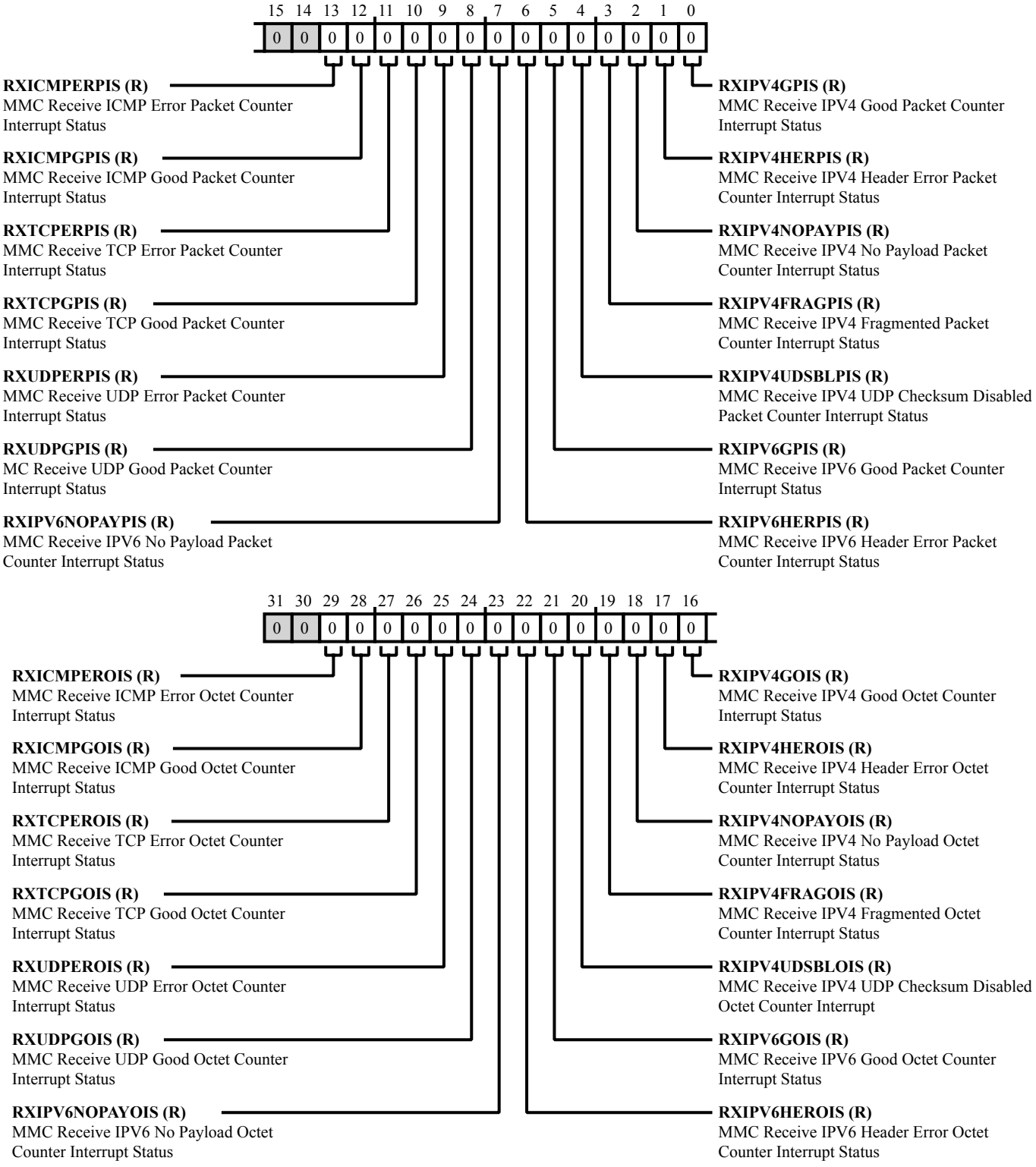


Figure 30-189: EMAC_MMC_IPC_RXINT Register Diagram

Table 30-244: EMAC_MMC_IPC_RXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/NW)	RXICMPEROIS	MMC Receive ICMP Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXICMPEROIS` bit is set when the EMAC_RXICMP_ERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive ICMP Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive ICMP Error Octet Counter Interrupt Status Detected
28 (R/NW)	RXICMPGOIS	MMC Receive ICMP Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXICMPGOIS` bit is set when the EMAC_RXICMP_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive ICMP Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive ICMP Good Octet Counter Interrupt Status Detected
27 (R/NW)	RXTCPEROIS	MMC Receive TCP Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXTCPEROIS` bit is set when the EMAC_RXTCP_ERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive TCP Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive TCP Error Octet Counter Interrupt Status Detected
26 (R/NW)	RXTCPGOIS	MMC Receive TCP Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXTCPGOIS` bit is set when the EMAC_RXTCP_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive TCP Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive TCP Good Octet Counter Interrupt Status Detected

Table 30-244: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/NW)	RXUDPEROIS	MMC Receive UDP Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXUDPEROIS` bit is set when the EMAC_RXUDP_ERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive UDP Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive UDP Error Octet Counter Interrupt Status Detected
24 (R/NW)	RXUDPGOIS	MMC Receive UDP Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXUDPGOIS` bit is set when the EMAC_RXUDP_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive UDP Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive UDP Good Octet Counter Interrupt Status Detected
23 (R/NW)	RXIPV6NOPAYOIS	MMC Receive IPV6 No Payload Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6NOPAYOIS` bit is set when the EMAC_RXIPV6_PYLD_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 No Payload Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 No Payload Octet Counter Interrupt Status Detected
22 (R/NW)	RXIPV6HEROIS	MMC Receive IPV6 Header Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6HEROIS` bit is set when the EMAC_RXIPV6_HDRERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 Header Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 Header Error Octet Counter Interrupt Status Detected

Table 30-244: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/NW)	RXIPV6GOIS	MMC Receive IPV6 Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6GOIS` bit is set when the EMAC_RXIPV6_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 Good Octet Counter Interrupt Status Detected
20 (R/NW)	RXIPV4UDSBLOIS	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt. The `EMAC_MMC_IPC_RXINT.RXIPV4UDSBLOIS` bit is set when the EMAC_RXIPV4_UDPCSD_OCNT reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Status Detected
19 (R/NW)	RXIPV4FRAGOIS	MMC Receive IPV4 Fragmented Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4FRAGOIS` bit is set when the EMAC_RXIPV4_FRGMT_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Fragmented Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Fragmented Octet Counter Interrupt Status Detected
18 (R/NW)	RXIPV4NOPAYOIS	MMC Receive IPV4 No Payload Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4NOPAYOIS` bit is set when the EMAC_RXIPV4_PYLD_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 No Payload Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 No Payload Octet Counter Interrupt Status Detected

Table 30-244: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/NW)	RXIPV4HEROIS	MMC Receive IPV4 Header Error Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4HEROIS` bit is set when the EMAC_RXIPV4_HDRERR_OCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Header Error Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Header Error Octet Counter Interrupt Status Detected
16 (R/NW)	RXIPV4GOIS	MMC Receive IPV4 Good Octet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4GOIS` bit is set when the EMAC_RXIPV4_OCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Good Octet Counter Interrupt Status Detected
13 (R/NW)	RXICMPERPIS	MMC Receive ICMP Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXICMPERPIS` bit is set when the EMAC_RXICMP_ERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive ICMP Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive ICMP Error Packet Counter Interrupt Status Detected
12 (R/NW)	RXICMPGPIS	MMC Receive ICMP Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXICMPGPIS` bit is set when the EMAC_RXICMP_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive ICMP Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive ICMP Good Packet Counter Interrupt Status Detected

Table 30-244: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	RXTCPERPIS	MMC Receive TCP Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXTCPERPIS` bit is set when the EMAC_RXTCP_ERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive TCP Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive TCP Error Packet Counter Interrupt Status Detected
10 (R/NW)	RXTCPGPIS	MMC Receive TCP Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXTCPGPIS` bit is set when the EMAC_RXTCP_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive TCP Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive TCP Good Packet Counter Interrupt Status Detected
9 (R/NW)	RXUDPERPIS	MMC Receive UDP Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXUDPERPIS` bit is set when the EMAC_RXUDP_ERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive UDP Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive UDP Error Packet Counter Interrupt Status Detected
8 (R/NW)	RXUDPGPIS	MC Receive UDP Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXUDPGPIS` bit is set when the EMAC_RXUDP_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive UDP Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive UDP Good Packet Counter Interrupt Status Detected

Table 30-244: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	RXIPV6NOPAYPIS	MMC Receive IPV6 No Payload Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6NOPAYPIS` bit is set when the EMAC_RXIPV6_PYLD_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 No Payload Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 No Payload Packet Counter Interrupt Status Detected
6 (R/NW)	RXIPV6HERPIS	MMC Receive IPV6 Header Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6HERPIS` bit is set when the EMAC_RXIPV6_HDRERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 Header Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 Header Error Packet Counter Interrupt Status Detected
5 (R/NW)	RXIPV6GPIS	MMC Receive IPV6 Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV6GPIS` bit is set when the EMAC_RXIPV6_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV6 Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV6 Good Packet Counter Interrupt Status Detected
4 (R/NW)	RXIPV4UDSBLPIS	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4UDSBLPIS` bit is set when the EMAC_RXIPV4_UDPCSD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Status Detected

Table 30-244: EMAC_MMC_IPC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	RXIPV4FRAGPIS	MMC Receive IPV4 Fragmented Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4FRAGPIS` bit is set when the EMAC_RXIPV4_FRGMT_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Fragmented Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Fragmented Packet Counter Interrupt Status Detected
2 (R/NW)	RXIPV4NOPAYPIS	MMC Receive IPV4 No Payload Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4NOPAYPIS` bit is set when the EMAC_RXIPV4_PYLD_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 No Payload Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 No Payload Packet Counter Interrupt Status Detected
1 (R/NW)	RXIPV4HERPIS	MMC Receive IPV4 Header Error Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4HERPIS` bit is set when the EMAC_RXIPV4_HDRERR_PCNT counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Header Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Header Error Packet Counter Interrupt Status Detected
0 (R/NW)	RXIPV4GPIS	MMC Receive IPV4 Good Packet Counter Interrupt Status. The `EMAC_MMC_IPC_RXINT.RXIPV4GPIS` bit is set when the EMAC_RXIPV4_PCNT_G counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive IPV4 Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive IPV4 Good Packet Counter Interrupt Status Detected

MMC Receive IPC Interrupt Mask Register

The EMAC_MMC_IPC_RXIMSK register maintains the mask for the interrupt generated from the receive IPC statistic counters.

The MMC Receive Checksum Off load Interrupt Mask register maintains the masks for the interrupts generated when the receive IPC (Checksum Off load) statistic counters reach half their maximum value, and when they reach their maximum values. The EMAC_MMC_IPC_RXIMSK register is 32 bits wide.

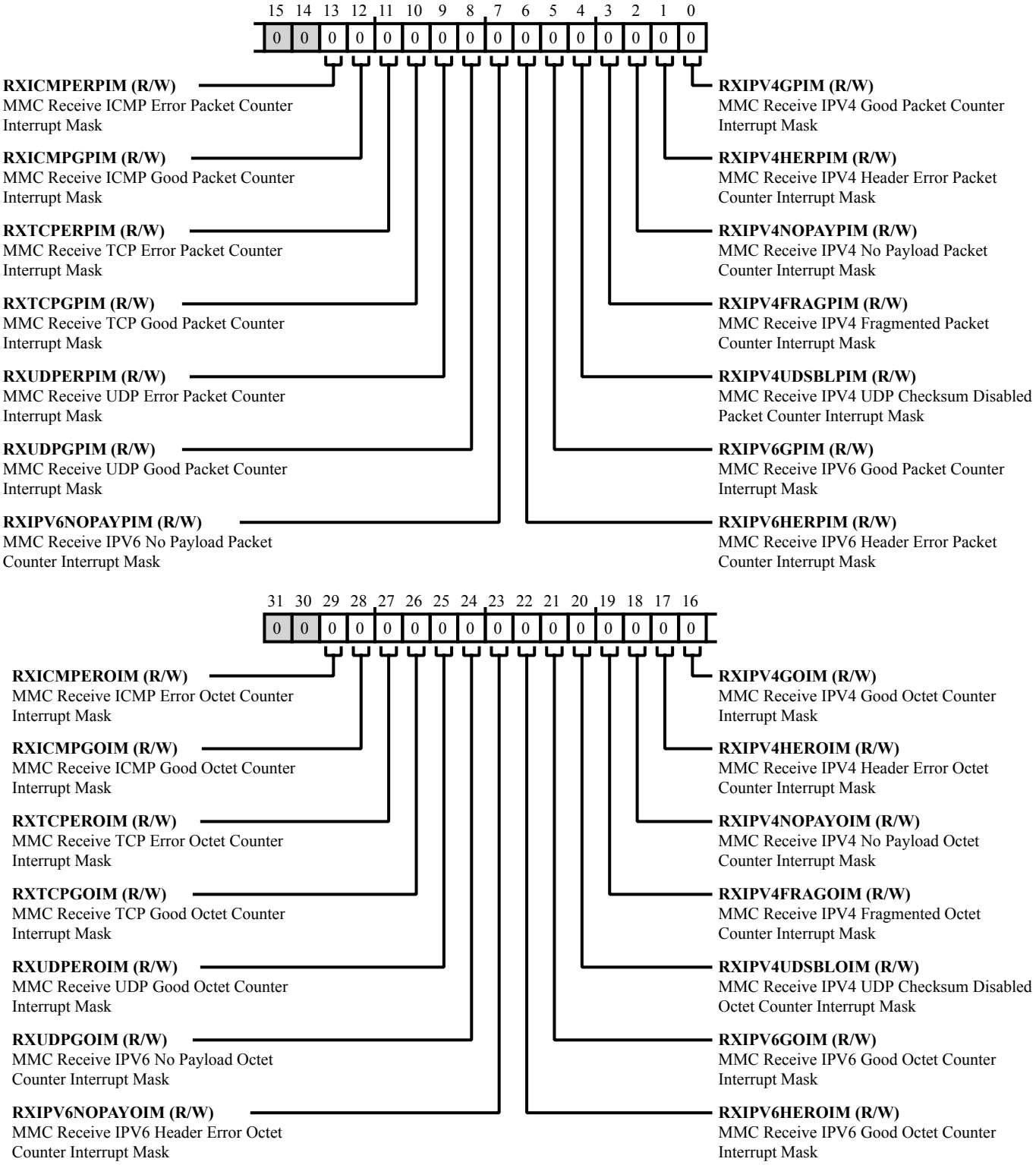


Figure 30-190: EMAC_MMC_IPC_RXIMSK Register Diagram

Table 30-245: EMAC_MMC_IPC_RXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	RXICMPEROIM	MMC Receive ICMP Error Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXICMPEROIM` bit masks the interrupt when the EMAC_RXICMP_ERR_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive ICMP Error Octet Counter Interrupt Mask is Disabled
		1	MMC Receive ICMP Error Octet Counter Interrupt Mask is Enabled
28 (R/W)	RXICMPGOIM	MMC Receive ICMP Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXICMPGOIM` bit masks the interrupt when the EMAC_RXICMP_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive ICMP Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive ICMP Good Octet Counter Interrupt Mask is Enabled
27 (R/W)	RXTCPEROIM	MMC Receive TCP Error Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXTCPEROIM` bit masks the interrupt when the EMAC_RXTCP_ERR_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive TCP Error Octet Counter Interrupt Mask is Disabled
		1	MMC Receive TCP Error Octet Counter Interrupt Mask is Enabled
26 (R/W)	RXTCPGOIM	MMC Receive TCP Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXTCPGOIM` bit masks the interrupt when the EMAC_RXTCP_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive TCP Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive TCP Good Octet Counter Interrupt Mask is Enabled

Table 30-245: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/W)	RXUDPEROIM	MMC Receive UDP Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXUDPEROIM` bit masks the interrupt when the EMAC_RXUDP_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive UDP Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive UDP Good Octet Counter Interrupt Mask is Enabled
24 (R/W)	RXUDPGOIM	MMC Receive IPV6 No Payload Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXUDPGOIM` bit masks the interrupt when the EMAC_RXUDP_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 No Payload Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 No Payload Octet Counter Interrupt Mask is Enabled
23 (R/W)	RXIPV6NOPAYOIM	MMC Receive IPV6 Header Error Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6NOPAYOIM` bit masks the interrupt when the EMAC_RXIPV6_PYLD_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Header Error Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Header Error Octet Counter Interrupt Mask is Enabled
22 (R/W)	RXIPV6HEROIM	MMC Receive IPV6 Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6HEROIM` bit masks the interrupt when the EMAC_RXIPV6_HDRERR_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Good Octet Counter Interrupt Mask is Enabled

Table 30-245: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	RXIPV6GOIM	MMC Receive IPV6 Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6GOIM` bit masks the interrupt when the EMAC_RXIPV6_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Good Octet Counter Interrupt Mask is Enabled
20 (R/W)	RXIPV4UDSBLOIM	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4UDSBLOIM` bit masks the interrupt when the EMAC_RXIPV4_UDPCSD_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask is Enabled
19 (R/W)	RXIPV4FRAGOIM	MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4FRAGOIM` bit masks the interrupt when the EMAC_RXIPV4_FRGMT_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask is Enabled
18 (R/W)	RXIPV4NOPAYOIM	MMC Receive IPV4 No Payload Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4NOPAYOIM` bit masks the interrupt when the EMAC_RXIPV4_PYLD_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 No Payload Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 No Payload Octet Counter Interrupt Mask is Enabled

Table 30-245: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	RXIPV4HEROIM	MMC Receive IPV4 Header Error Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4HEROIM` bit masks the interrupt when the EMAC_RXIPV4_HDRERR_OCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Header Error Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Header Error Octet Counter Interrupt Mask is Enabled
16 (R/W)	RXIPV4GOIM	MMC Receive IPV4 Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4GOIM` bit masks the interrupt when the EMAC_RXIPV4_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Good Octet Counter Interrupt Mask is Enabled
13 (R/W)	RXICMPERPIM	MMC Receive ICMP Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXICMPERPIM` bit masks the interrupt when the EMAC_RXICMP_ERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive ICMP Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive ICMP Error Packet Counter Interrupt Mask is Enabled
12 (R/W)	RXICMPGPIM	MMC Receive ICMP Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXICMPGPIM` bit masks the interrupt when the EMAC_RXICMP_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive ICMP Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive ICMP Good Packet Counter Interrupt Mask is Enabled

Table 30-245: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	RXTCPERPIM	MMC Receive TCP Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXTCPERPIM` bit masks the interrupt when the EMAC_RXTCP_ERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive TCP Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive TCP Error Packet Counter Interrupt Mask is Enabled
10 (R/W)	RXTCPGPIM	MMC Receive TCP Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXTCPGPIM` bit masks the interrupt when the EMAC_RXTCP_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive TCP Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive TCP Good Packet Counter Interrupt Mask is Enabled
9 (R/W)	RXUDPERPIM	MMC Receive UDP Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXUDPERPIM` bit masks the interrupt when the EMAC_RXUDP_ERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive UDP Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive UDP Error Packet Counter Interrupt Mask is Enabled
8 (R/W)	RXUDPGPIM	MMC Receive UDP Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXUDPGPIM` bit masks the interrupt when the EMAC_RXUDP_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive UDP Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive UDP Good Packet Counter Interrupt Mask is Enabled

Table 30-245: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXIPV6NOPAYPIM	MMC Receive IPV6 No Payload Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6NOPAYPIM` bit masks the interrupt when the EMAC_RXIPV4_PYLD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 No Payload Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 No Payload Packet Counter Interrupt Mask is Enabled
6 (R/W)	RXIPV6HERPIM	MMC Receive IPV6 Header Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6HERPIM` bit masks the interrupt when the EMAC_RXIPV6_HDRERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Header Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Header Error Packet Counter Interrupt Mask is Enabled
5 (R/W)	RXIPV6GPIM	MMC Receive IPV6 Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV6GPIM` bit masks the interrupt when the EMAC_RXIPV6_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV6 Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV6 Good Packet Counter Interrupt Mask is Enabled
4 (R/W)	RXIPV4UDSBLPIM	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4UDSBLPIM` bit masks the interrupt when the EMAC_RXIPV4_UDPCSD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Mask is Enabled

Table 30-245: EMAC_MMC_IPC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXIPV4FRAGPIM	MMC Receive IPV4 Fragmented Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4FRAGPIM` bit masks the interrupt when the EMAC_RXIPV4_FRGMT_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Fragmented Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Fragmented Packet Counter Interrupt Mask is Enabled
2 (R/W)	RXIPV4NOPAYPIM	MMC Receive IPV4 No Payload Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4NOPAYPIM` bit masks the interrupt when the EMAC_RXIPV4_PYLD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 No Payload Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 No Payload Packet Counter Interrupt Mask is Enabled
1 (R/W)	RXIPV4HERPIM	MMC Receive IPV4 Header Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4HERPIM` bit masks the interrupt when the EMAC_RXIPV4_HDRERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Header Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Header Error Packet Counter Interrupt Mask is Enabled
0 (R/W)	RXIPV4GPIM	MMC Receive IPV4 Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_IPC_RXIMSK.RXIPV4GPIM` bit masks the interrupt when the EMAC_RXIPV4_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive IPV4 Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive IPV4 Good Packet Counter Interrupt Mask is Enabled

MMC Receive FPE Fragment Counter Register

The EMAC_RXFPE_FRGMNT_CNT register provides the number of additional mPackets received due to preemption.

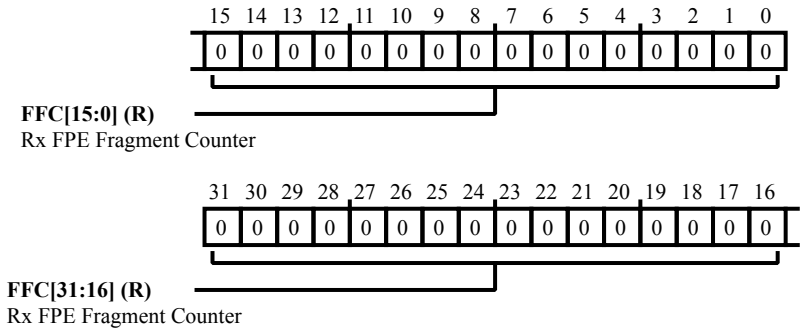


Figure 30-191: EMAC_RXFPE_FRGMNT_CNT Register Diagram

Table 30-246: EMAC_RXFPE_FRGMNT_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	FFC	Rx FPE Fragment Counter. The EMAC_RXFPE_FRGMNT_CNT.FFC field indicates the number of additional mPackets received due to preemption. Exists when at least one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Receive Interrupt Status Register

The EMAC_MMC_RXINT register maintains the interrupts generated from all receive statistics counters when the following occurs:

Receive statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter).

Receive statistic counters cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the Counter Stop Rollover is set, interrupts are set but the counter remains at all-ones. The EMAC_MMC_RXINT register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

Note: R_SS_RC means that this register bit is set internally, and it is cleared when the counter register is read.

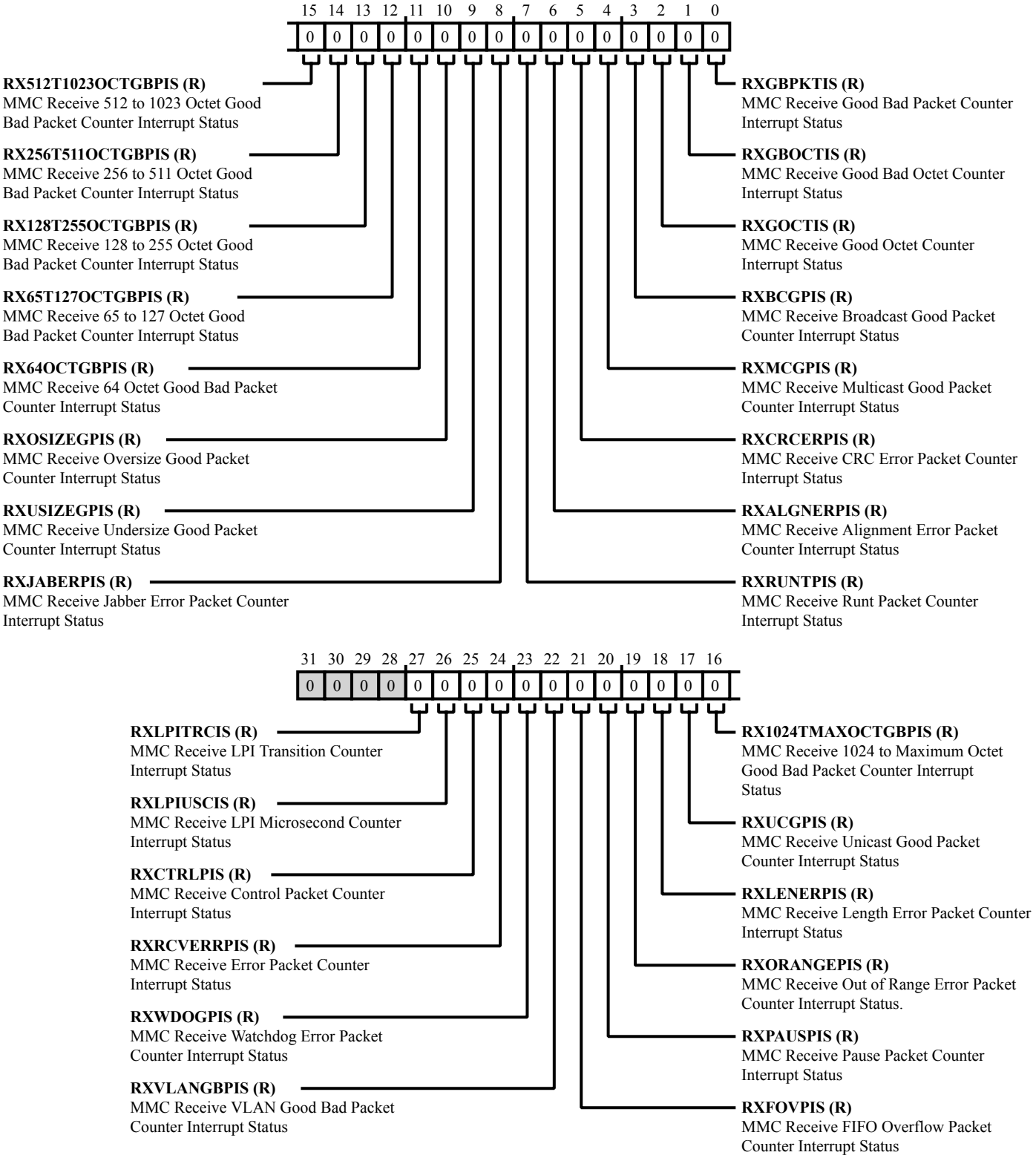


Figure 30-192: EMAC_MMC_RXINT Register Diagram

Table 30-247: EMAC_MMC_RXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	RXLPITRCIS	MMC Receive LPI Transition Counter Interrupt Status. The `EMAC_MMC_RXINT.RXLPITRCIS` bit is set when the EMAC_RX_LPITRAN_CNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive LPI Transition Counter Interrupt Status Not Detected
		1	MMC Receive LPI Transition Counter Interrupt Status Detected
26 (R/NW)	RXLPIUSCIS	MMC Receive LPI Microsecond Counter Interrupt Status. The `EMAC_MMC_RXINT.RXLPIUSCIS` bit is set when the EMAC_RX_LPIUSEC_CNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive LPI Microsecond Counter Interrupt Status Not Detected
		1	MMC Receive LPI Microsecond Counter Interrupt Status Detected
25 (R/NW)	RXCTRLPIS	MMC Receive Control Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXCTRLPIS` bit is set when the EMAC_RX_CTL_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Control Packet Counter Interrupt Status Not Detected
		1	MMC Receive Control Packet Counter Interrupt Status Detected
24 (R/NW)	RXRCVERRPIS	MMC Receive Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXRCVERRPIS` bit is set when the EMAC_RX_ERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Error Packet Counter Interrupt Status Detected

Table 30-247: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/NW)	RXWDOGPIS	MMC Receive Watchdog Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXWDOGPIS` bit is set when the EMAC_RX_WDTERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Watchdog Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Watchdog Error Packet Counter Interrupt Status Detected
22 (R/NW)	RXVLANGBPIS	MMC Receive VLAN Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXVLANGBPIS` bit is set when the EMAC_RX_VLAN_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive VLAN Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive VLAN Good Bad Packet Counter Interrupt Status Detected
21 (R/NW)	RXFOVPIS	MMC Receive FIFO Overflow Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXFOVPIS` bit is set when the EMAC_RX_FIFOOVFL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive FIFO Overflow Packet Counter Interrupt Status Not Detected
		1	MMC Receive FIFO Overflow Packet Counter Interrupt Status Detected
20 (R/NW)	RXPAUSPIS	MMC Receive Pause Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXPAUSPIS` bit is set when the EMAC_RX_PAUSE_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Pause Packet Counter Interrupt Status Not Detected
		1	MMC Receive Pause Packet Counter Interrupt Status Detected

Table 30-247: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	RXORANGEPIS	MMC Receive Out of Range Error Packet Counter Interrupt Status.. The `EMAC_MMC_RXINT.RXORANGEPIS` bit is set when the EMAC_RX_OR-TYPE_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Out of Range Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Out of Range Error Packet Counter Interrupt Status Detected
18 (R/NW)	RXLENERPIS	MMC Receive Length Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXLENERPIS` bit is set when the EMAC_RX_LE-NERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Length Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Length Error Packet Counter Interrupt Status Detected
17 (R/NW)	RXUCGPIS	MMC Receive Unicast Good Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXUCGPIS` bit is set when the EMAC_RX_UCST_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Unicast Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Unicast Good Packet Counter Interrupt Status Detected
16 (R/NW)	RX1024TMAXOCTGBPIS	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX1024TMAXOCTGBPIS` bit is set when the EMAC_RX1024TOMAX_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status Detected

Table 30-247: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	RX512T1023OCTGBPIS	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX512T1023OCTGBPIS` bit is set when the EMAC_RX512TO1023_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status Detected
14 (R/NW)	RX256T511OCTGBPIS	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX256T511OCTGBPIS` bit is set when the EMAC_RX256TO511_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status Detected
13 (R/NW)	RX128T255OCTGBPIS	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX128T255OCTGBPIS` bit is set when the EMAC_RX128TO255_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status Detected
12 (R/NW)	RX65T127OCTGBPIS	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX65T127OCTGBPIS` bit is set when the EMAC_RX65TO127_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status Detected

Table 30-247: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	RX64OCTGBPIS	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RX64OCTGBPIS` bit is set when the EMAC_RX64_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status Detected
10 (R/NW)	RXOSIZEGPIS	MMC Receive Oversize Good Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXOSIZEGPIS` bit is set when the EMAC_RX_OVRSZ_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Oversize Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Oversize Good Packet Counter Interrupt Status Detected
9 (R/NW)	RXUSIZEGPIS	MMC Receive Undersize Good Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXUSIZEGPIS` bit is set when the EMAC_RX_UNDRSZ_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Undersize Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Undersize Good Packet Counter Interrupt Status Detected
8 (R/NW)	RXJABERPIS	MMC Receive Jabber Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXJABERPIS` bit is set when the EMAC_RX_JBERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Jabber Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Jabber Error Packet Counter Interrupt Status Detected

Table 30-247: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	RXRUNTPIS	MMC Receive Runt Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXRUNTPIS` bit is set when the EMAC_RX_RNTERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Runt Packet Counter Interrupt Status Not Detected
		1	MMC Receive Runt Packet Counter Interrupt Status Detected
6 (R/NW)	RXALGNERPIS	MMC Receive Alignment Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXALGNERPIS` bit is set when the EMAC_RX_ALGNERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Alignment Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive Alignment Error Packet Counter Interrupt Status Detected
5 (R/NW)	RXCRCERPIS	MMC Receive CRC Error Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXCRCERPIS` bit is set when the EMAC_RX_CRCERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive CRC Error Packet Counter Interrupt Status Not Detected
		1	MMC Receive CRC Error Packet Counter Interrupt Status Detected
4 (R/NW)	RXMCGPIS	MMC Receive Multicast Good Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXMCGPIS` bit is set when the EMAC_RX_MCST_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Multicast Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Multicast Good Packet Counter Interrupt Status Detected

Table 30-247: EMAC_MMC_RXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	RXBCGPIS	MMC Receive Broadcast Good Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXBCGPIS` bit is set when the EMAC_RX_BCERR_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Broadcast Good Packet Counter Interrupt Status Not Detected
		1	MMC Receive Broadcast Good Packet Counter Interrupt Status Detected
2 (R/NW)	RXGOCTIS	MMC Receive Good Octet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXGOCTIS` bit is set when the EMAC_RX_OCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Good Octet Counter Interrupt Status Not Detected
		1	MMC Receive Good Octet Counter Interrupt Status Detected
1 (R/NW)	RXGBOCTIS	MMC Receive Good Bad Octet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXGBOCTIS` bit is set when the EMAC_RX_OCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Good Bad Octet Counter Interrupt Status Not Detected
		1	MMC Receive Good Bad Octet Counter Interrupt Status Detected
0 (R/NW)	RXGBPKTIS	MMC Receive Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_RXINT.RXGBPKTIS` bit is set when the EMAC_RX_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Receive Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Receive Good Bad Packet Counter Interrupt Status Detected

MMC Receive Interrupt Mask Register

The EMAC_MMC_RXIMSK register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

The EMAC_MMC_RXIMSK register is 32 bit wide.

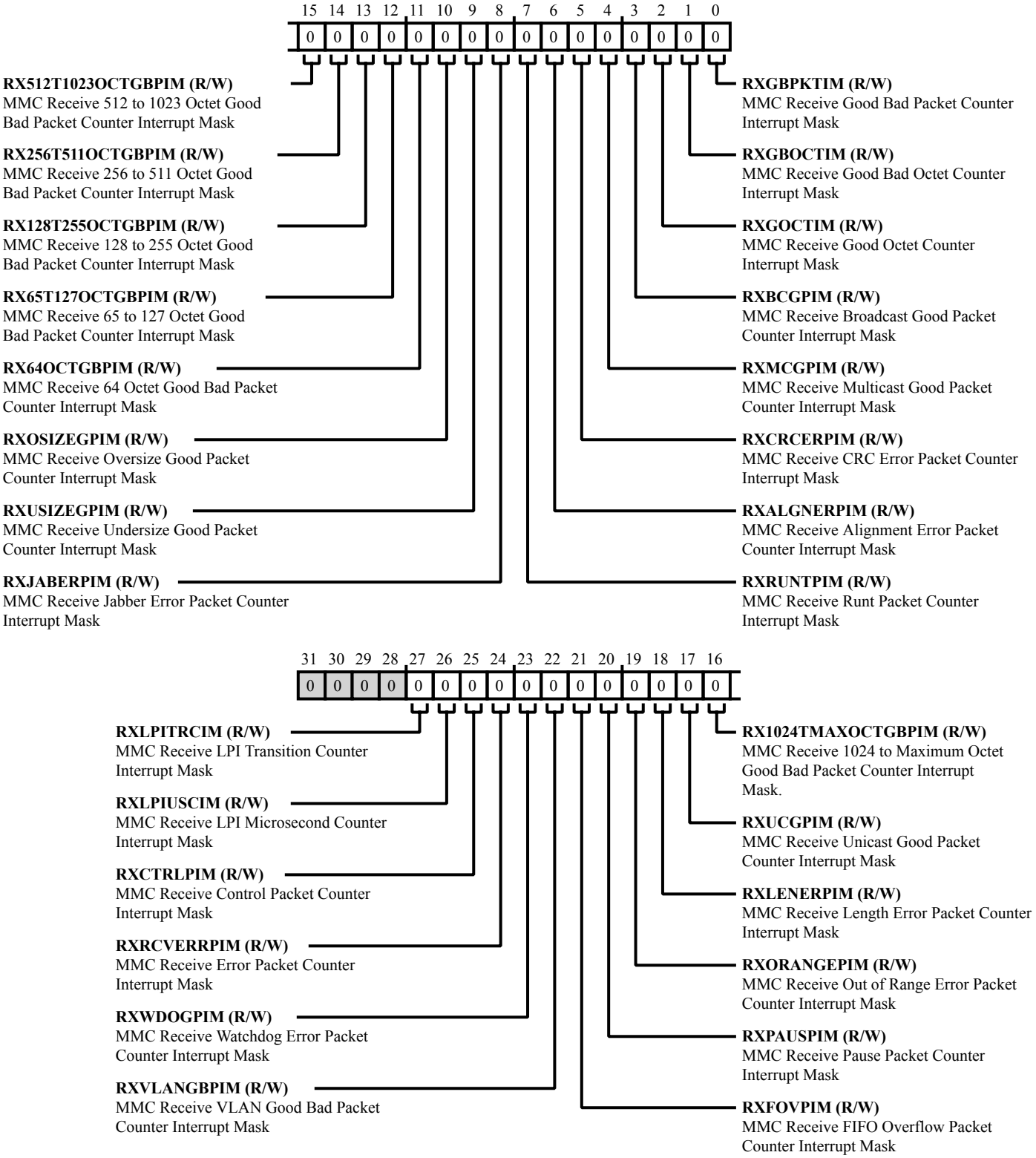


Figure 30-193: EMAC_MMC_RXIMSK Register Diagram

Table 30-248: EMAC_MMC_RXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	RXLPITRCIM	MMC Receive LPI Transition Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXLPITRCIM` bit masks the interrupt when the EMAC_RX_LPITRAN_CNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive LPI Transition Counter Interrupt Mask is Disabled
		1	MMC Receive LPI Transition Counter Interrupt Mask is Enabled
26 (R/W)	RXLPIUSCIM	MMC Receive LPI Microsecond Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXLPIUSCIM` bit masks the interrupt when the EMAC_RX_LPIUSEC_CNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive LPI Microsecond Counter Interrupt Mask is Disabled
		1	MMC Receive LPI Microsecond Counter Interrupt Mask is Enabled
25 (R/W)	RXCTRLPIM	MMC Receive Control Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXCTRLPIM` bit masks the interrupt when the EMAC_RXICMP_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Control Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Control Packet Counter Interrupt Mask is Enabled
24 (R/W)	RXRCVERRPIM	MMC Receive Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXRCVERRPIM` bit masks the interrupt when the EMAC_RX_ERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Error Packet Counter Interrupt Mask is Enabled

Table 30-248: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	RXWDOGPIM	MMC Receive Watchdog Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXWDOGPIM` bit masks the interrupt when the EMAC_RX_WDTERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Watchdog Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Watchdog Error Packet Counter Interrupt Mask is Enabled
22 (R/W)	RXVLANGBPIM	MMC Receive VLAN Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXVLANGBPIM` bit masks the interrupt when the EMAC_RX_VLAN_PCNT_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive VLAN Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive VLAN Good Bad Packet Counter Interrupt Mask is Enabled
21 (R/W)	RXFOVPIM	MMC Receive FIFO Overflow Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXFOVPIM` bit masks the interrupt when the EMAC_RX_FIFOOVFL_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive FIFO Overflow Packet Counter Interrupt Mask is Disabled
		1	MMC Receive FIFO Overflow Packet Counter Interrupt Mask is Enabled
20 (R/W)	RXPAUSPIM	MMC Receive Pause Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXPAUSPIM` bit masks the interrupt when the EMAC_RX_PAUSE_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Pause Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Pause Packet Counter Interrupt Mask is Enabled

Table 30-248: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	RXORANGEPIM	MMC Receive Out of Range Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXORANGEPIM` bit masks the interrupt when the EMAC_RX_ORTYPE_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Out of Range Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Out of Range Error Packet Counter Interrupt Mask is Enabled
18 (R/W)	RXLENERPIM	MMC Receive Length Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXLENERPIM` bit masks the interrupt when the EMAC_RX_LENERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Length Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Length Error Packet Counter Interrupt Mask is Enabled
17 (R/W)	RXUCGPIM	MMC Receive Unicast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXUCGPIM` bit masks the interrupt when the EMAC_RX_UCST_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Unicast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Unicast Good Packet Counter Interrupt Mask is Enabled
16 (R/W)	RX1024TMAXOCTGB-PIM	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask.. Setting the `EMAC_MMC_RXIMSK.RX1024TMAXOCTGBPIM` bit masks the interrupt when the EMAC_RX1024TOMAX_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask is Enabled

Table 30-248: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	RX512T1023OCTGBPIM	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX512T1023OCTGBPIM` bit masks the interrupt when the EMAC_RX512TO1023_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask is Enabled
14 (R/W)	RX256T511OCTGBPIM	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX256T511OCTGBPIM` bit masks the interrupt when the EMAC_RX256TO511_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask is Enabled
13 (R/W)	RX128T255OCTGBPIM	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX128T255OCTGBPIM` bit masks the interrupt when the EMAC_RX128TO255_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask is Enabled
12 (R/W)	RX65T127OCTGBPIM	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX65T127OCTGBPIM` bit masks the interrupt when the EMAC_RX65TO127_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask is Enabled

Table 30-248: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	RX64OCTGBPIM	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RX64OCTGBPIM` bit masks the interrupt when the EMAC_RX64_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask is Enabled
10 (R/W)	RXOSIZEGPIM	MMC Receive Oversize Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXOSIZEGPIM` bit masks the interrupt when the EMAC_RX_OVRSZ_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Oversize Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Oversize Good Packet Counter Interrupt Mask is Enabled
9 (R/W)	RXUSIZEGPIM	MMC Receive Undersize Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXUSIZEGPIM` bit masks the interrupt when the EMAC_RX_UNDRSZ_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Undersize Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Undersize Good Packet Counter Interrupt Mask is Enabled
8 (R/W)	RXJABERPIM	MMC Receive Jabber Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXJABERPIM` bit masks the interrupt when the EMAC_RX_JBERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Jabber Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Jabber Error Packet Counter Interrupt Mask is Enabled

Table 30-248: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	RXRUNTPIM	MMC Receive Runt Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXRUNTPIM` bit masks the interrupt when the EMAC_RX_RNTERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Runt Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Runt Packet Counter Interrupt Mask is Enabled
6 (R/W)	RXALGNERPIM	MMC Receive Alignment Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXALGNERPIM` bit masks the interrupt when the EMAC_RX_ALGNERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Alignment Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Alignment Error Packet Counter Interrupt Mask is Enabled
5 (R/W)	RXCRCERPIM	MMC Receive CRC Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXCRCERPIM` bit masks the interrupt when the EMAC_RX_CRCERR_PCNT register reaches half of the maximum value or the maximum value.	
		0	MMC Receive CRC Error Packet Counter Interrupt Mask is Disabled
		1	MMC Receive CRC Error Packet Counter Interrupt Mask is Enabled
4 (R/W)	RXMCGPIM	MMC Receive Multicast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXMCGPIM` bit masks the interrupt when the EMAC_RX_MCST_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Multicast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Multicast Good Packet Counter Interrupt Mask is Enabled

Table 30-248: EMAC_MMC_RXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXBCGPIM	MMC Receive Broadcast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXBCGPIM` bit masks the interrupt when the EMAC_RX_BCERR_PCNT_G register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Broadcast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Broadcast Good Packet Counter Interrupt Mask is Enabled
2 (R/W)	RXGOCTIM	MMC Receive Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXGOCTIM` bit masks the interrupt when the EMAC_RX_OCNT_G register counter reaches half of the maximum value or the maximum value.	
		0	MMC Receive Good Octet Counter Interrupt Mask is Disabled
		1	MMC Receive Good Octet Counter Interrupt Mask is Enabled
1 (R/W)	RXGBOCTIM	MMC Receive Good Bad Octet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXGBOCTIM` bit masks the interrupt when the EMAC_RX_OCNT_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Good Bad Octet Counter Interrupt Mask is Disabled
		1	MMC Receive Good Bad Octet Counter Interrupt Mask is Enabled
0 (R/W)	RXGBPKTIM	MMC Receive Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_RXIMSK.RXGBPKTIM` bit masks the interrupt when the EMAC_RX_PCNT_GB register reaches half of the maximum value or the maximum value.	
		0	MMC Receive Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Receive Good Bad Packet Counter Interrupt Mask is Enabled

MMC Receive FPE Packet Assembly Error Counter Register

The EMAC_RXPKT_ASERR_CNT register provides the number of MAC frames with reassembly errors on the receiver, due to mismatch in the fragment count value.

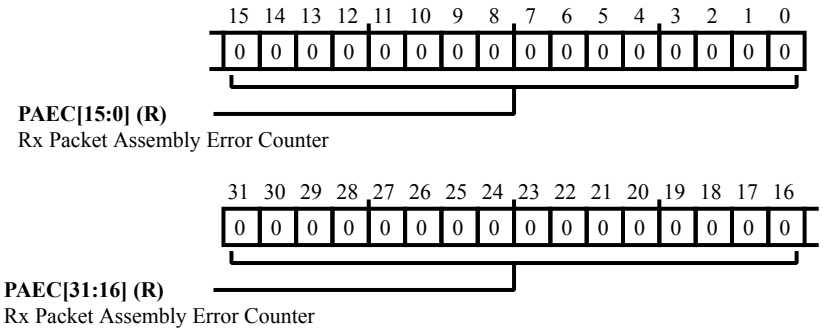


Figure 30-194: EMAC_RXPKT_ASERR_CNT Register Diagram

Table 30-249: EMAC_RXPKT_ASERR_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PAEC	Rx Packet Assembly Error Counter. The EMAC_RXPKT_ASERR_CNT.PAEC field indicates the number of MAC frames with reassembly errors on the receiver, due to mismatch in the Fragment Count value. Exists when any one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Receive FPE Packet Assembly Okay Counter Register

The EMAC_RXPKT_ASOK_CNT register provides the number of MAC frames that were successfully reassembled and delivered to MAC.

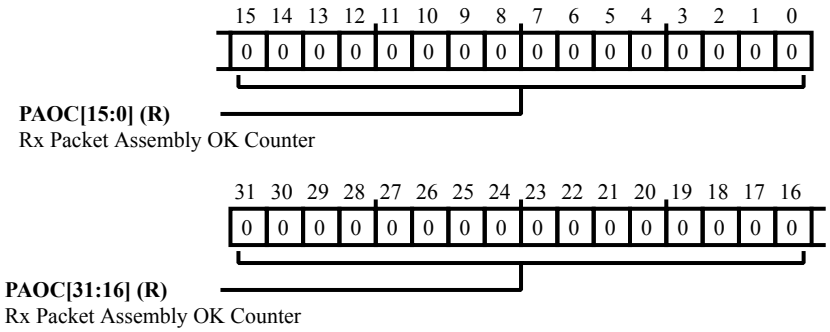


Figure 30-195: EMAC_RXPKT_ASOK_CNT Register Diagram

Table 30-250: EMAC_RXPKT_ASOK_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PAOC	Rx Packet Assembly OK Counter. The EMAC_RXPKT_ASOK_CNT.PAOC field indicates the number of MAC frames that were successfully reassembled and delivered to MAC. Exists when at least one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Receive FPE Packet SMD Error Counter Register

The EMAC_RXPKT_SMDERR_CNT register provides the number of received MAC frames rejected due to unknown SMD value and MAC frame fragments rejected due to arriving with an SMD-C when there was no preceding preempted frame.

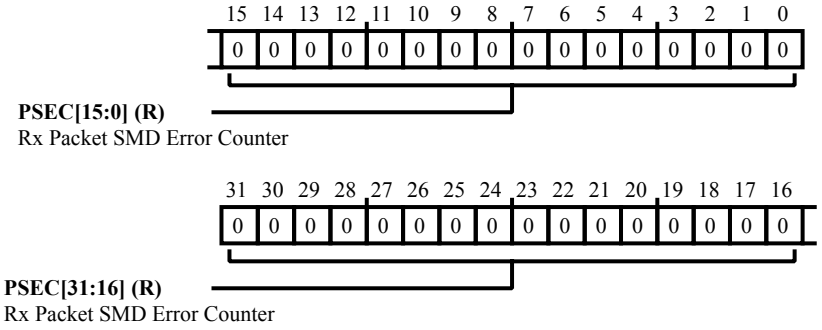


Figure 30-196: EMAC_RXPKT_SMDERR_CNT Register Diagram

Table 30-251: EMAC_RXPKT_SMDERR_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	PSEC	Rx Packet SMD Error Counter. The EMAC_RXPKT_SMDERR_CNT.PSEC field indicates the number of MAC frames rejected due to unknown SMD value and MAC frame fragments rejected due to arriving with an SMD-C when there was no preceding preempted frame. Exists when at least one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Transmit FPE Packet Fragment Counter Register

The EMAC_TXFPE_FRGMNT_CNT register provides the number of additional mPackets transmitted due to pre-emption.

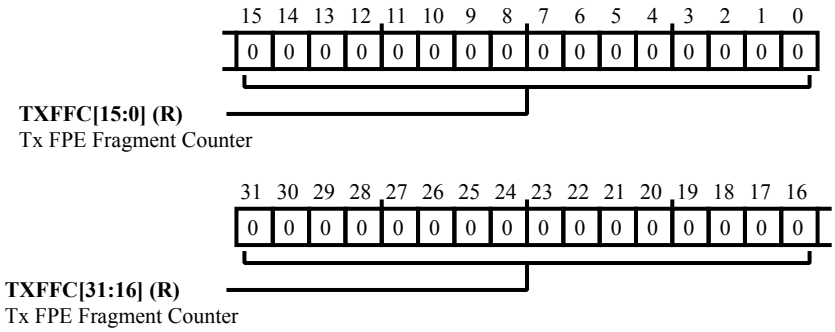


Figure 30-197: EMAC_TXFPE_FRGMNT_CNT Register Diagram

Table 30-252: EMAC_TXFPE_FRGMNT_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXFFC	Tx FPE Fragment Counter. The EMAC_TXFPE_FRGMNT_CNT.TXFFC field indicates the number of additional mPackets that has been transmitted due to preemption Exists when any one of the RX/TX MMC counters are enabled during an FPE-enabled configuration.

MMC Transmit FPE Hold Request Counter Register

The EMAC_TXHLDRQ_CNT register provides the count of number of times a hold request is given to MAC.

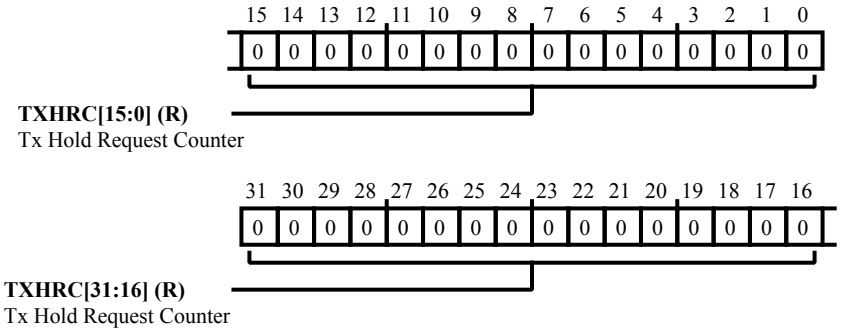


Figure 30-198: EMAC_TXHLDRQ_CNT Register Diagram

Table 30-253: EMAC_TXHLDRQ_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXHRC	Tx Hold Request Counter. The EMAC_TXHLDRQ_CNT.TXHRC field indicates count of number of a hold request is given to MAC.

MMC Transmit Interrupt Status Register

The EMAC_MMC_TXINT register maintains the interrupts generated from all transmit statistics counters when transmit statistic counters reach half their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones.

The EMAC_MMC_TXINT register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (bits[7:0]) of the respective counter must be read to clear the interrupt bit.

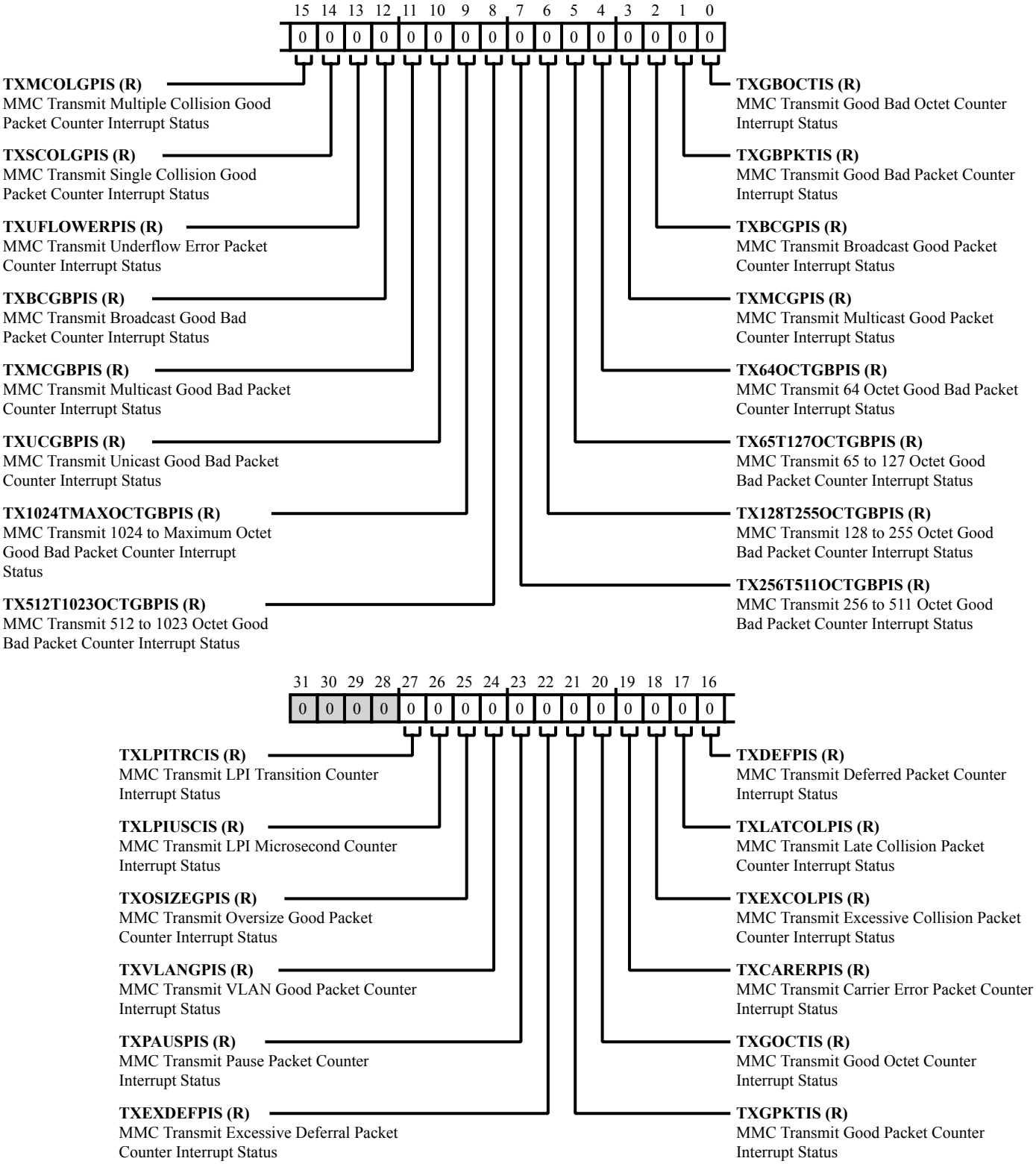


Figure 30-199: EMAC_MMC_TXINT Register Diagram

Table 30-254: EMAC_MMC_TXINT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	TXLPITRCIS	MMC Transmit LPI Transition Counter Interrupt Status. The `EMAC_MMC_TXINT.TXLPITRCIS` bit is set when the EMAC_TX_LPITRAN_CNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit LPI Transition Counter Interrupt Status Not Detected
		1	MMC Transmit LPI Transition Counter Interrupt Status Detected
26 (R/NW)	TXLPIUSCIS	MMC Transmit LPI Microsecond Counter Interrupt Status. The `EMAC_MMC_TXINT.TXLPIUSCIS` bit is set when the EMAC_TX_LPIUSEC_CNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit LPI Microsecond Counter Interrupt Status Not Detected
		1	MMC Transmit LPI Microsecond Counter Interrupt Status Detected
25 (R/NW)	TXOSIZEGPIS	MMC Transmit Oversize Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXOSIZEGPIS` bit is set when the EMAC_TX_OSIZE_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Oversize Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Oversize Good Packet Counter Interrupt Status Detected
24 (R/NW)	TXVLANGPIS	MMC Transmit VLAN Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXVLANGPIS` bit is set when the EMAC_TX_VLAN_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit VLAN Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit VLAN Good Packet Counter Interrupt Status Detected

Table 30-254: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/NW)	TXPAUSPIS	MMC Transmit Pause Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXPAUSPIS` bit is set when the EMAC_TX_PAUSE_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Pause Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Pause Packet Counter Interrupt Status Detected
22 (R/NW)	TXEXDEFPIS	MMC Transmit Excessive Deferral Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXEXDEFPIS` bit is set when the EMAC_TX_EXDFRL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Excessive Deferral Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Excessive Deferral Packet Counter Interrupt Status Detected
21 (R/NW)	TXGPKTIS	MMC Transmit Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXGPKTIS` bit is set when the EMAC_TX_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Good Packet Counter Interrupt Status Detected
20 (R/NW)	TXGOCTIS	MMC Transmit Good Octet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXGOCTIS` bit is set when the EMAC_TX_OCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Good Octet Counter Interrupt Status Not Detected
		1	MMC Transmit Good Octet Counter Interrupt Status Detected

Table 30-254: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	TXCARERPIS	MMC Transmit Carrier Error Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXCARERPIS` bit is set when the EMAC_TX_CRERR_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Carrier Error Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Carrier Error Packet Counter Interrupt Status Detected
18 (R/NW)	TXEXCOLPIS	MMC Transmit Excessive Collision Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXEXCOLPIS` bit is set when the EMAC_TX_EXCOL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Excessive Collision Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Excessive Collision Packet Counter Interrupt Status Detected
17 (R/NW)	TXLATCOLPIS	MMC Transmit Late Collision Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXLATCOLPIS` bit is set when the EMAC_TX_LTCOL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Late Collision Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Late Collision Packet Counter Interrupt Status Detected
16 (R/NW)	TXDEFPIS	MMC Transmit Deferred Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXDEFPIS` bit is set when the EMAC_TX_DFRD_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Deferred Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Deferred Packet Counter Interrupt Status Detected

Table 30-254: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/NW)	TXMCOLGPIS	MMC Transmit Multiple Collision Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXMCOLGPIS` bit is set when the EMAC_TX_MCST_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Multiple Collision Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Multiple Collision Good Packet Counter Interrupt Status Detected
14 (R/NW)	TXSCOLGPIS	MMC Transmit Single Collision Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXSCOLGPIS` bit is set when the EMAC_TX_SNGLCOL_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Single Collision Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Single Collision Good Packet Counter Interrupt Status Detected
13 (R/NW)	TXUFLOWERPIS	MMC Transmit Underflow Error Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXUFLOWERPIS` bit is set when the EMAC_TX_URFL_PCNT register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Underflow Error Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Underflow Error Packet Counter Interrupt Status Detected
12 (R/NW)	TXBCGBPIS	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXBCGBPIS` bit is set when the EMAC_TX_BCST_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status Detected

Table 30-254: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	TXMCGBPIS	MMC Transmit Multicast Good Bad Packet Counter Interrupt Status. The bit is set when the EMAC_TX_MCST_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Multicast Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Multicast Good Bad Packet Counter Interrupt Status Detected
10 (R/NW)	TXUCGBPIS	MMC Transmit Unicast Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXUCGBPIS` bit is set when the EMAC_TX_UCST_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Unicast Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Unicast Good Bad Packet Counter Interrupt Status Detected
9 (R/NW)	TX1024TMAXOCTGBPIS	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX1024TMAXOCTGBPIS` bit is set when the EMAC_TX1024TOMAX_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status Detected
8 (R/NW)	TX512T1023OCTGBPIS	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX512T1023OCTGBPIS` bit is set when the EMAC_TX512TO1023_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status Detected

Table 30-254: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	TX256T511OCTGBPIS	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX256T511OCTGBPIS` bit is set when the EMAC_TX256TO511_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status Detected
6 (R/NW)	TX128T255OCTGBPIS	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX128T255OCTGBPIS` bit is set when the EMAC_TX128TO255_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status Detected
5 (R/NW)	TX65T127OCTGBPIS	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX65T127OCTGBPIS` bit is set when the EMAC_TX65TO127_GB register reaches half the maximum value, and also when it reaches the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status Detected
4 (R/NW)	TX64OCTGBPIS	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TX64OCTGBPIS` bit is set when the EMAC_TX64_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status Detected

Table 30-254: EMAC_MMC_TXINT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	TXMCGPIS	MMC Transmit Multicast Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXMCGPIS` bit is set when the EMAC_TX_MCST_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Multicast Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Multicast Good Packet Counter Interrupt Status Detected
2 (R/NW)	TXBCGPIS	MMC Transmit Broadcast Good Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXBCGPIS` bit is set when the EMAC_TX_BCST_PCNT_G register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Broadcast Good Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Broadcast Good Packet Counter Interrupt Status Detected
1 (R/NW)	TXGBPKTIS	MMC Transmit Good Bad Packet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXGBPKTIS` bit is set when the EMAC_TX_PCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Good Bad Packet Counter Interrupt Status Not Detected
		1	MMC Transmit Good Bad Packet Counter Interrupt Status Detected
0 (R/NW)	TXGBOCTIS	MMC Transmit Good Bad Octet Counter Interrupt Status. The `EMAC_MMC_TXINT.TXGBOCTIS` bit is set when the EMAC_TX_OCNT_GB register reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	MMC Transmit Good Bad Octet Counter Interrupt Status Not Detected
		1	MMC Transmit Good Bad Octet Counter Interrupt Status Detected

MMC Transmit Interrupt Mask Register

The EMAC_MMC_TXIMSK register maintains the masks for interrupts generated from all transmit statistics counters.

The EMAC_MMC_TXIMSK register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. The EMAC_MMC_TXIMSK register is 32 bits wide.

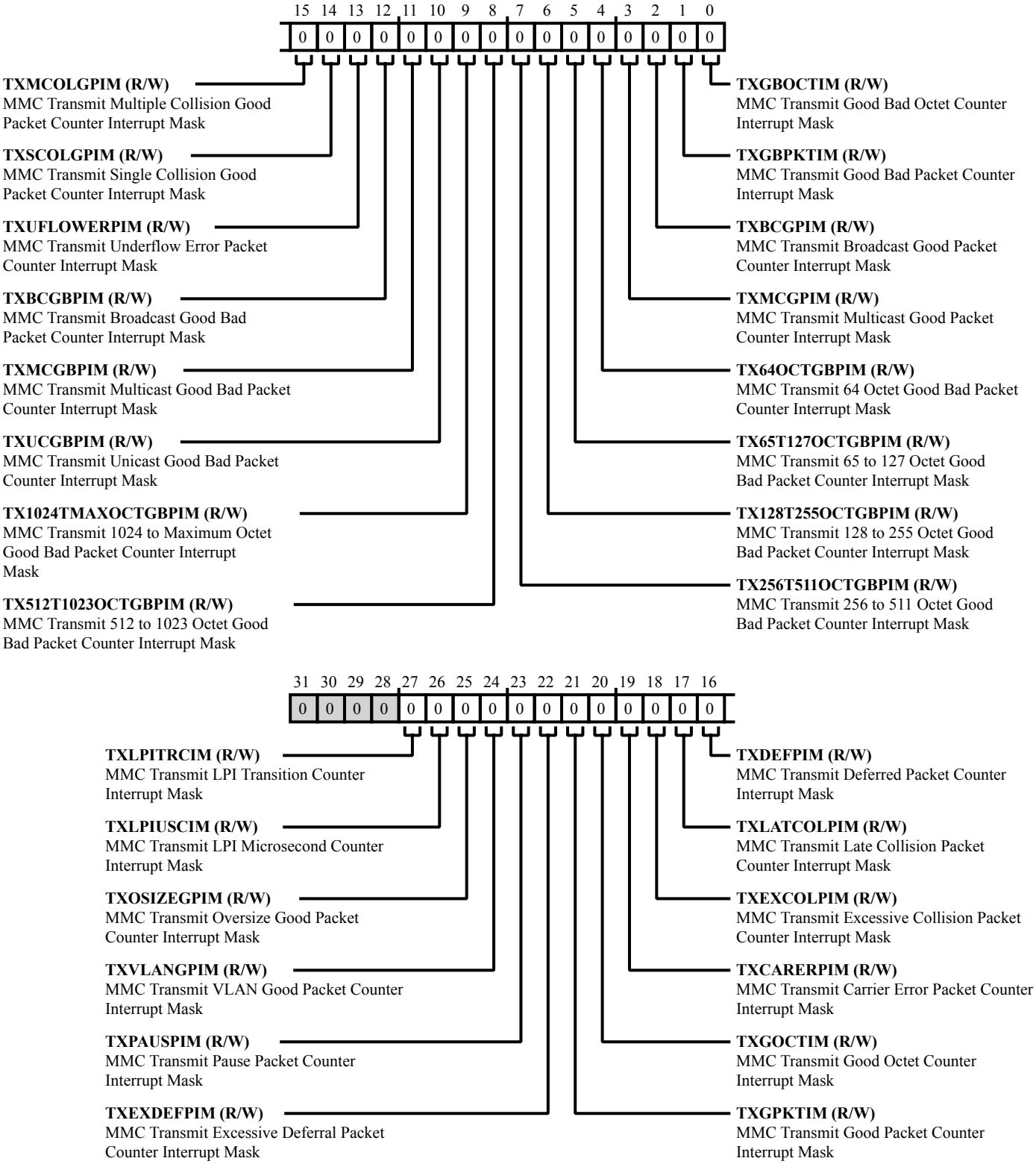


Figure 30-200: EMAC_MMC_TXIMSK Register Diagram

Table 30-255: EMAC_MMC_TXIMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/W)	TXLPITRCIM	MMC Transmit LPI Transition Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXLPITRCIM` bit masks the interrupt when the EMAC_TX_LPITRAN_CNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit LPI Transition Counter Interrupt Mask is Disabled
		1	MMC Transmit LPI Transition Counter Interrupt Mask is Enabled
26 (R/W)	TXLPIUSCIM	MMC Transmit LPI Microsecond Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXLPIUSCIM` bit masks the interrupt when the EMAC_TX_LPIUSEC_CNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit LPI Microsecond Counter Interrupt Mask is Disabled
		1	MMC Transmit LPI Microsecond Counter Interrupt Mask is Enabled
25 (R/W)	TXOSIZEGPIM	MMC Transmit Oversize Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXOSIZEGPIM` bit masks the interrupt when the EMAC_TX_OSIZE_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Oversize Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Oversize Good Packet Counter Interrupt Mask is Enabled
24 (R/W)	TXVLANGPIM	MMC Transmit VLAN Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXVLANGPIM` bit masks the interrupt when the EMAC_TX_VLAN_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit VLAN Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit VLAN Good Packet Counter Interrupt Mask is Enabled

Table 30-255: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23 (R/W)	TXPAUSPIM	MMC Transmit Pause Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXPAUSPIM` bit masks the interrupt when the EMAC_TX_PAUSE_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Pause Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Pause Packet Counter Interrupt Mask is Enabled
22 (R/W)	TXEXDEFPIM	MMC Transmit Excessive Deferral Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXEXDEFPIM` bit masks the interrupt when the EMAC_TX_EXDFRL_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Excessive Deferral Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Excessive Deferral Packet Counter Interrupt Mask is Enabled
21 (R/W)	TXGPKTIM	MMC Transmit Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXGPKTIM` bit masks the interrupt when the EMAC_TX_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Good Packet Counter Interrupt Mask is Enabled
20 (R/W)	TXGOCTIM	MMC Transmit Good Octet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXGOCTIM` bit masks the interrupt when the EMAC_TX_OCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Good Octet Counter Interrupt Mask is Disabled
		1	MMC Transmit Good Octet Counter Interrupt Mask is Enabled

Table 30-255: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/W)	TXCARERPIM	MMC Transmit Carrier Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXCARERPIM` bit masks the interrupt when the EMAC_TX_CRERR_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Carrier Error Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Carrier Error Packet Counter Interrupt Mask is Enabled
18 (R/W)	TXEXCOLPIM	MMC Transmit Excessive Collision Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXEXCOLPIM` bit masks the interrupt when the EMAC_TX_EXCOL_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Excessive Collision Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Excessive Collision Packet Counter Interrupt Mask is Enabled
17 (R/W)	TXLATCOLPIM	MMC Transmit Late Collision Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXLATCOLPIM` bit masks the interrupt when the EMAC_TX_LTCOL_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Late Collision Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Late Collision Packet Counter Interrupt Mask is Enabled
16 (R/W)	TXDEFPIM	MMC Transmit Deferred Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXDEFPIM` bit masks the interrupt when the EMAC_TX_DFRD_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Deferred Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Deferred Packet Counter Interrupt Mask is Enabled

Table 30-255: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	TXMCOLGPIM	MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXMCOLGPIM` bit masks the interrupt when the EMAC_TX_MCOL_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask is Enabled
14 (R/W)	TXSCOLGPIM	MMC Transmit Single Collision Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXSCOLGPIM` bit masks the interrupt when the EMAC_TX_SNGLCOL_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Single Collision Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Single Collision Good Packet Counter Interrupt Mask is Enabled
13 (R/W)	TXUFLOWERPIM	MMC Transmit Underflow Error Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXUFLOWERPIM` bit masks the interrupt when the EMAC_TX_URFL_PCNT counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Underflow Error Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Underflow Error Packet Counter Interrupt Mask is Enabled
12 (R/W)	TXBCGBPIM	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXBCGBPIM` bit masks the interrupt when the EMAC_TX_BCST_PCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask is Enabled

Table 30-255: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	TXMCGBPIM	MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXMCGBPIM` bit masks the interrupt when the EMAC_TX_MCST_PCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask is Enabled
10 (R/W)	TXUCGBPIM	MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXUCGBPIM` bit masks the interrupt when the EMAC_TX_UCST_PCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask is Enabled
9 (R/W)	TX1024TMAXOCTGB-PIM	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX1024TMAXOCTGBPIM` bit masks the interrupt when the EMAC_TX1024TOMAX_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask is Enabled
8 (R/W)	TX512T1023OCTGBPIM	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX512T1023OCTGBPIM` bit masks the interrupt when the EMAC_TX512TO1023_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask is Enabled

Table 30-255: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/W)	TX256T511OCTGBPIM	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX256T511OCTGBPIM` bit masks the interrupt when the EMAC_TX256TO511_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask is Enabled
6 (R/W)	TX128T255OCTGBPIM	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX128T255OCTGBPIM` bit masks the interrupt when the EMAC_TX128TO255_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask is Enabled
5 (R/W)	TX65T127OCTGBPIM	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX65T127OCTGBPIM` bit masks the interrupt when the EMAC_TX65TO127_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask is Enabled
4 (R/W)	TX64OCTGBPIM	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TX64OCTGBPIM` bit masks the interrupt when the EMAC_TX64_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask is Enabled

Table 30-255: EMAC_MMC_TXIMSK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	TXMCGPIM	MMC Transmit Multicast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXMCGPIM` bit masks the interrupt when the EMAC_TX_MCST_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Multicast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Multicast Good Packet Counter Interrupt Mask is Enabled
2 (R/W)	TXBCGPIM	MMC Transmit Broadcast Good Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXBCGPIM` bit masks the interrupt when the EMAC_TX_BCST_PCNT_G counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Broadcast Good Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Broadcast Good Packet Counter Interrupt Mask is Enabled
1 (R/W)	TXGBPKTIM	MMC Transmit Good Bad Packet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXGBPKTIM` bit masks the interrupt when the EMAC_TX_PCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Good Bad Packet Counter Interrupt Mask is Disabled
		1	MMC Transmit Good Bad Packet Counter Interrupt Mask is Enabled
0 (R/W)	TXGBOCTIM	MMC Transmit Good Bad Octet Counter Interrupt Mask. Setting the `EMAC_MMC_TXIMSK.TXGBOCTIM` bit masks the interrupt when the EMAC_TX_OCNT_GB counter reaches half of the maximum value or the maximum value.	
		0	MMC Transmit Good Bad Octet Counter Interrupt Mask is Disabled
		1	MMC Transmit Good Bad Octet Counter Interrupt Mask is Enabled

EST Control Register

The EMAC_EST_CTL register controls the operation of Enhancements to Scheduled Transmission (EST) (IEEE802.1Qbv).

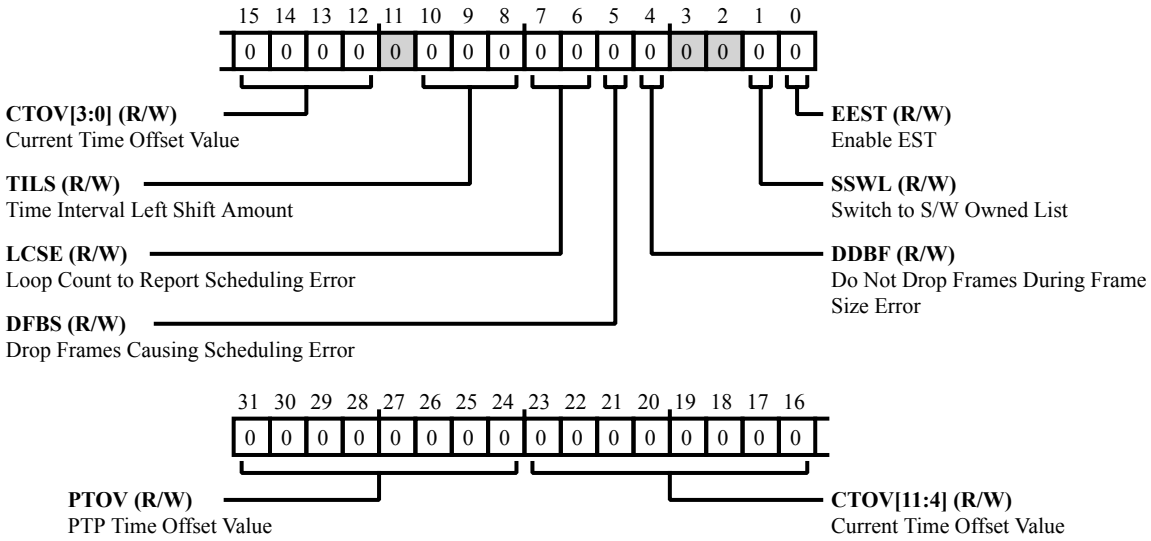


Figure 30-201: EMAC_EST_CTL Register Diagram

Table 30-256: EMAC_EST_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	PTOV	PTP Time Offset Value. The EMAC_EST_CTL.PTOV field indicates the value of the PTP clock period multiplied by 6 in nanoseconds. This value is needed to avoid transmission overruns at the beginning of the installation of a new GCL.
23:12 (R/W)	CTOV	Current Time Offset Value. The EMAC_EST_CTL.CTOV field provides a 12 bit time offset value in nano second that is added to the current time to compensate for all the implementation pipeline delays such as the CDC sync delay, buffering delays, data path delays and so on. This offset helps to ensure that the impact of gate controls is visible on the line exactly at the pre-determined schedule (or as close to the schedule as possible).

Table 30-256: EMAC_EST_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10:8 (R/W)	TILS	Time Interval Left Shift Amount. The EMAC_EST_CTL.TILS field provides the left shift amount for the programmed time interval values used in the Gate Control Lists. - 000: No left shift needed (equal to x1ns) - 001: Left shift TI by 1 bit (equal to x2ns) - 010: Left shift TI by 2 bits (equal to x4ns) - ... - ... - 100: Left shift TI by 7 bits (equal to x128ns) Based on the configuration one or more bits of this field should be treated as reserved/read-only.	
7:6 (R/W)	LCSE	Loop Count to Report Scheduling Error. The EMAC_EST_CTL.LCSE field indicates a programmable number of GCL list iterations before reporting the EMAC_EST_STAT.HLBS bit.	
		0	4 Iterations
		1	8 Iterations
		2	16 Iterations
		3	32 Iterations
5 (R/W)	DFBS	Drop Frames Causing Scheduling Error. When the EMAC_EST_CTL.DFBS bit is set (=1), frames reported to cause HOL blocking due to not getting scheduled (EMAC_EST_STAT.HLBS) after 4,8,16,32 (based on LCSE field of this register) GCL iterations are dropped.	
		0	Do Not Drop Frames Causing Scheduling Error
		1	Drop Frames Causing Scheduling Error
4 (R/W)	DDBF	Do Not Drop Frames During Frame Size Error. When the EMAC_EST_CTL.DDBF bit is set (=1), frames are not dropped during head-of-line blocking due to a frame size error (EMAC_EST_STAT.HLBF).	
		0	Drop Frames During Frame Size Error
		1	Do Not Drop Frames During Frame Size Error

Table 30-256: EMAC_EST_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	SSWL	Switch to S/W Owned List. When set (=1), the EMAC_EST_CTL.SSWL bit indicates that the software has programmed that list that it currently owns (EMAC_EST_STAT.SWOL) and the hardware should switch to the new list based on the new BTR. Hardware clears the EMAC_EST_CTL.SSWL bit when the switch to the EMAC_EST_STAT.SWOL bit happens to indicate the completion of the switch or a BTR error (EMAC_EST_STAT.BTRE = 1). When EMAC_EST_STAT.BTRE = 1, the EMAC_EST_CTL.SSWL bit is cleared, but EMAC_EST_STAT.SWOL is not updated as the switch was not successful. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Switch to S/W Owned List is Disabled
		1	Switch to S/W Owned List is Enabled
0 (R/W)	EEST	Enable EST. When set (=1), the EMAC_EST_CTL.EEST bit enables EST. It must be set for the hardware to start processing the gate control lists. When cleared, the gate control list processing is halted and all gates are assumed to be in the Open state. During the toggle from 0 to 1, the gate control list processing starts only after EMAC_EST_CTL.SSWL =1.	
		0	EST is Disabled
		1	EST is Enabled

EST Extended Control Register

The EMAC_EST_EXT_CTL register indicates the number of overhead bytes for EST related scheduling.

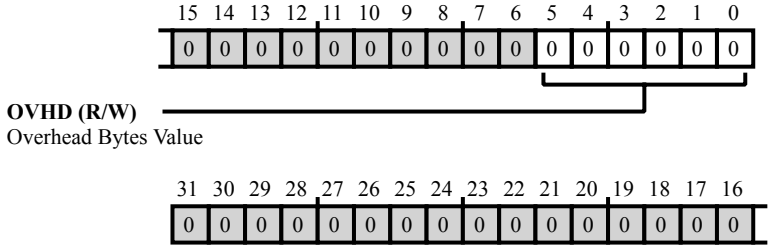


Figure 30-202: EMAC_EST_EXT_CTL Register Diagram

Table 30-257: EMAC_EST_EXT_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5:0 (R/W)	OVHD	Overhead Bytes Value. The EMAC_EST_EXT_CTL.OVHD field indicates the fixed overhead for every packet to account for EST Scheduler Delay, IPG or EIPG, and Preamble bytes.

EST Frame Size Capture Register

The EMAC_EST_FRMSZ_CAP register captures the frame size and queue number of the first occurrence of the frame size-related error. Upon clearing, it captures the data of immediate next occurrence of a similar error.

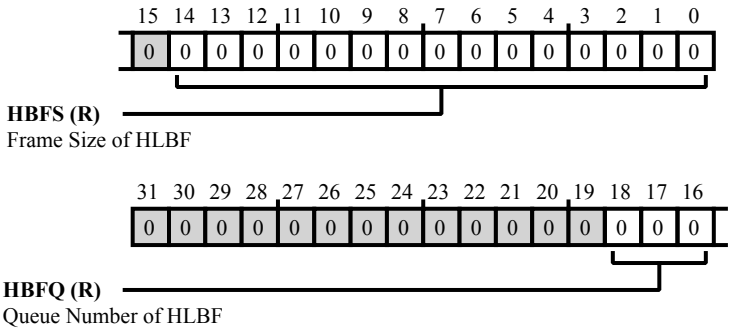


Figure 30-203: EMAC_EST_FRMSZ_CAP Register Diagram

Table 30-258: EMAC_EST_FRMSZ_CAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18:16 (R/NW)	HBFQ	Queue Number of HLBF. The EMAC_EST_FRMSZ_CAP.HBFQ field indicates the binary value of the of the first queue (number) experiencing HLBF error (see HLBF field of status register). Value once written is not altered by any subsequent queue errors of similar nature. Once cleared the queue number of the next occurring HLBF error is captured. The width is based on the number of Tx queues configured; remaining bits are read-only. The EMAC_EST_FRMSZ_CAP.HBFQ field is cleared when the EMAC_EST_FRMSZ_ERR register is all zeros.
14:0 (R/NW)	HBFS	Frame Size of HLBF. The EMAC_EST_FRMSZ_CAP.HBFS field indicates the frame size of the dropped frame related to queue number indicated in EMAC_EST_FRMSZ_CAP.HBFQ field. The contents of the register is considered invalid, if the EMAC_EST_FRMSZ_CAP.HBFS field is zero. The EMAC_EST_FRMSZ_CAP.HBFS field cleared when the EMAC_EST_FRMSZ_ERR register is all zeros.

EST Frame Size Error Register

The EMAC_EST_FRMSZ_ERR register provides the one hot encoded queue numbers that are having the frame size-related error.

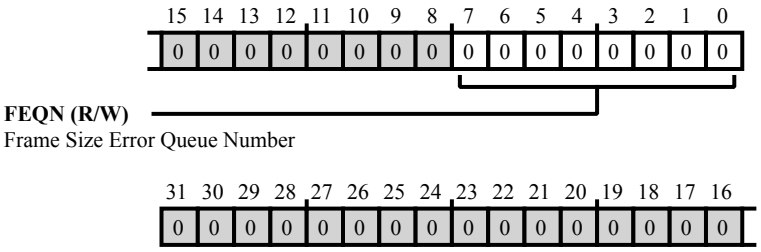


Figure 30-204: EMAC_EST_FRMSZ_ERR Register Diagram

Table 30-259: EMAC_EST_FRMSZ_ERR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FEQN	Frame Size Error Queue Number. The EMAC_EST_FRMSZ_ERR.FEQN field indicates the one hot encoded queue numbers that have experienced the error described in the HLBF field of status register. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.

EST Gate Control List Control Register

The EMAC_EST_GCL_CTL register provides the control information for reading/writing to the gate control lists.

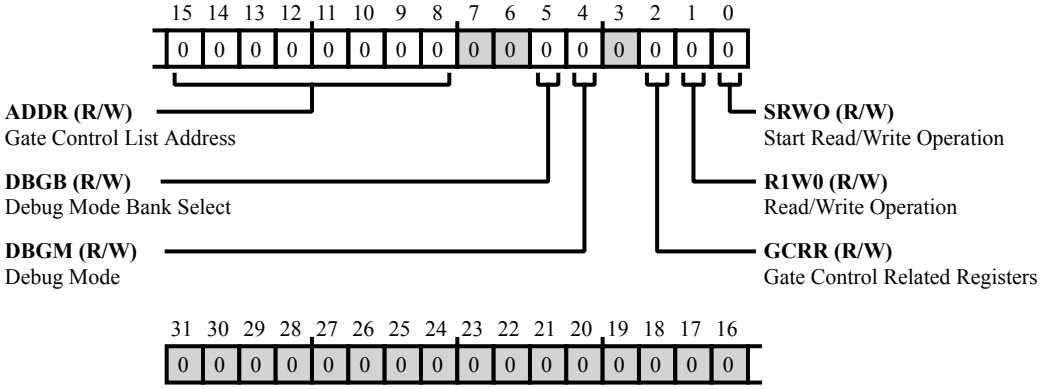


Figure 30-205: EMAC_EST_GCL_CTL Register Diagram

Table 30-260: EMAC_EST_GCL_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	ADDR	Gate Control List Address. When set (=1), the EMAC_EST_GCL_CTL.ADDR field indicates the address (row number) of the GCL at which the R/W operation must be performed. By default, the GCL pointed by EMAC_EST_STAT.SWOL is selected for R/W. However, if EMAC_EST_GCL_CTL.DBGM=1, a debug mode access is given to R/W from EMAC_EST_GCL_CTL.DBGB. The width of this field is dependent on DWC_EQOS_EST_DEP; unused bits should be treated as read only. Gate Control List Related Registers Address: (GCRA when GCRR =1). By default, the GCL-related register set pointed by EMAC_EST_STAT.SWOL is selected for R/W. However, if EMAC_EST_GCL_CTL.DBGM=1, debug mode access is given to R/W from EMAC_EST_GCL_CTL.DBGB. Lower 3 bits are only used in this mode, higher order bits are treated as dont care. 000: BTR Low (31:0) 001: BTR High (63:31) 010: CTR Low (31:0) 011: CTR High (39:32) 100: TER (31:0) 101: LLR (8:0) Others: Reserved

Table 30-260: EMAC_EST_GCL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	DBGB	Debug Mode Bank Select. When the EMAC_EST_GCL_CTL.DBGB bit is cleared (=0), it indicates R/W in debug mode is directed to bank 0 (GCL0 and corresponding time-related registers). When set (=1), the EMAC_EST_GCL_CTL.DBGB bit indicates R/W in debug mode should be directed to bank 1 (GCL1 and corresponding time-related registers). This value is used when EMAC_EST_GCL_CTL.DBGB = 1 and overrides by value of SWOL (which is normally used).	
		0	R/W in Debug Mode Should Be Directed to Bank 0
		1	R/W in Debug Mode Should Be Directed to Bank 1
4 (R/W)	DBGM	Debug Mode. When set (=1), the EMAC_EST_GCL_CTL.DBGM bit indicates R/W in debug mode where the memory bank (for GCL and time related registers) is explicitly provided by DBGB value. When the EMAC_EST_GCL_CTL.DBGM bit is cleared (=0), the SWOL bit is used to determine which bank to use.	
		0	Debug Mode is Disabled
		1	Debug Mode is Enabled
2 (R/W)	GCRR	Gate Control Related Registers. When set (=1), the bit indicates that the R/W access is for the GCL related registers (BTR, CTR, TER, LLR) whose address is provided by GCRA. When cleared (=0), the bit indicates that the R/W is directed to GCL from the address provided by GCLA.	
		0	Gate Control Related Registers are Disabled
		1	Gate Control Related Registers are Enabled
1 (R/W)	R1W0	Read/Write Operation. The EMAC_EST_GCL_CTL.R1W0 bit indicates a read or write operation.	
		0	Write Operation
		1	Read Operation

Table 30-260: EMAC_EST_GCL_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	SRWO	Start Read/Write Operation. When set (=1), the bit indicates the start/progress of a read/write operation. When cleared (=0) by hardware, the bit indicates that the read/write operation has completed or an error has occurred (when bit 20 is set). For read operations, data can be read from EMAC_EST_GCL_DAT register after `EMAC_EST_GCL_CTL.SRWO` = 0. For write operations, the EMAC_EST_GCL_DAT register should be programmed with write data before setting the `EMAC_EST_GCL_CTL.SRWO` bit. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Start Read/Write Op Disabled
		1	Start Read/Write Op Enabled

EST Gate Control List Data Register

The EMAC_EST_GCL_DAT register holds the read data or write data in case of reads and writes respectively.

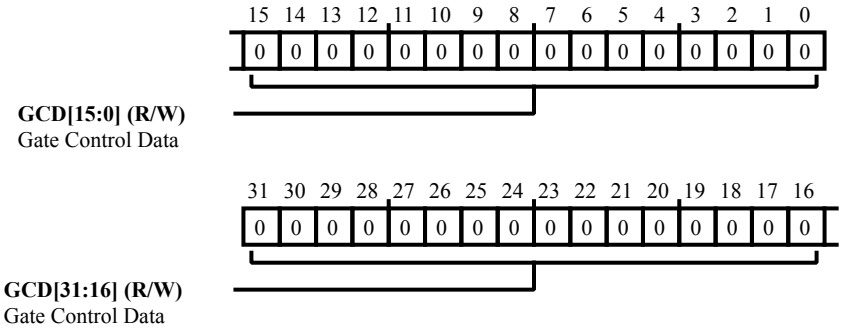


Figure 30-206: EMAC_EST_GCL_DAT Register Diagram

Table 30-261: EMAC_EST_GCL_DAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	GCD	Gate Control Data. The EMAC_EST_GCL_DAT.GCD field contains data corresponding to the address selected in the EMAC_EST_GCL_DAT register. It is used for both read and write operations.

EST Interrupt Enable Register

The EMAC_EST_IEN register implements the interrupt enable bits for the various events that generate an interrupt. Bit positions have a 1-to-1 correlation with the status bit positions in the EMAC_EST_STAT register.

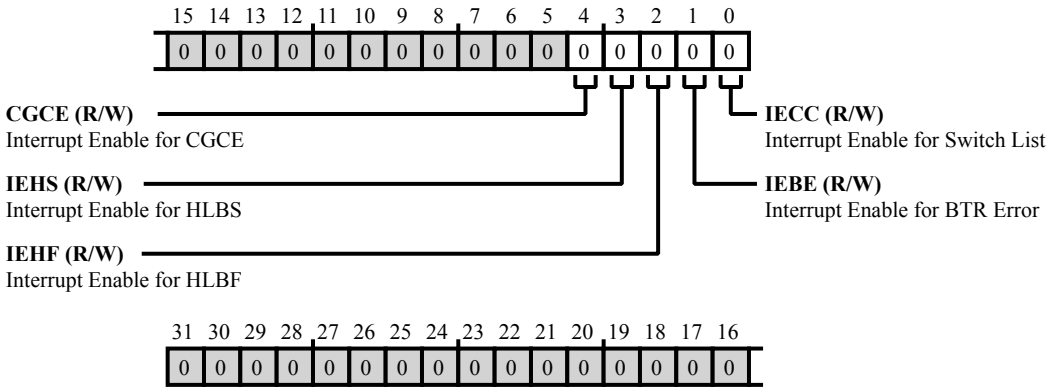


Figure 30-207: EMAC_EST_IEN Register Diagram

Table 30-262: EMAC_EST_IEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	CGCE	Interrupt Enable for CGCE. When the EMAC_EST_IEN.CGCE bit is set (=1), an interrupt is generated when the constant gate control error occurs and is indicated in the status. When the EMAC_EST_IEN.CGCE bit is cleared (= 0), an interrupt is not generated.	
		0	Interrupt for CGCE is Disabled
		1	Interrupt for CGCE is Enabled
3 (R/W)	IEHS	Interrupt Enable for HLBS. When the EMAC_EST_IEN.IEHS bit is set (=1), an interrupt is generated when the head-of-line blocking due to a scheduling issue and is indicated in the status. When the EMAC_EST_IEN.IEHS bit is cleared (= 0), an interrupt is not generated.	
		0	Interrupt for HLBS is Disabled
		1	Interrupt for HLBS is Enabled
2 (R/W)	IEHF	Interrupt Enable for HLBF. When the EMAC_EST_IEN.IEHF bit is set (=1), an interrupt is generated when the head-of-line blocking due to frame size error occurs and is indicated in the status. When the EMAC_EST_IEN.IEHF bit is cleared (= 0), an interrupt is not generated.	
		0	Interrupt for HLBF is Disabled
		1	Interrupt for HLBF is Enabled

Table 30-262: EMAC_EST_IEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	IEBE	Interrupt Enable for BTR Error. When the `EMAC_EST_IEN.IEBE` bit is set (=1), an interrupt is generated when the BTR error occurs and is indicated in the status. When the `EMAC_EST_IEN.IEBE` bit is cleared (= 0), an interrupt is not generated.	
		0	Interrupt for BTR Error is Disabled
		1	Interrupt for BTR Error is Enabled
0 (R/W)	IECC	Interrupt Enable for Switch List. When the `EMAC_EST_IEN.IECC` bit is set (=1), an interrupt is generated when the configuration change is successful, and the hardware has switched to the new list. When cleared (= 0), no interrupt is generated.	
		0	Interrupt for Switch List is Disabled
		1	Interrupt for Switch List is Enabled

EST Scheduling Error Register

The EMAC_EST_SCH_ERR register provides the one hot encoded queue numbers that are having the scheduling-related error (timeout).

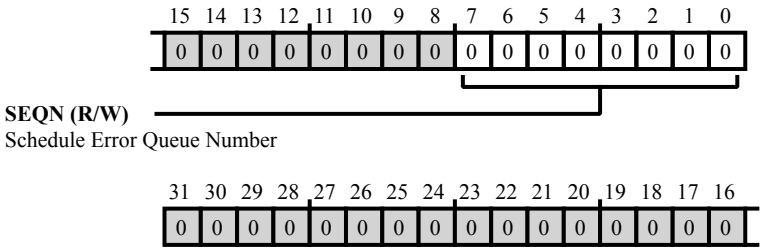


Figure 30-208: EMAC_EST_SCH_ERR Register Diagram

Table 30-263: EMAC_EST_SCH_ERR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	SEQN	Schedule Error Queue Number. The EMAC_EST_SCH_ERR.SEQN field indicates the one hot encoded queue numbers that have experienced error/timeout described in EMAC_EST_STAT.HLBS field. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.

EST Status Register

The EMAC_EST_STAT register provides status information related to Enhancements to Scheduled Transmission (EST) (IEEE802.1Qbv).

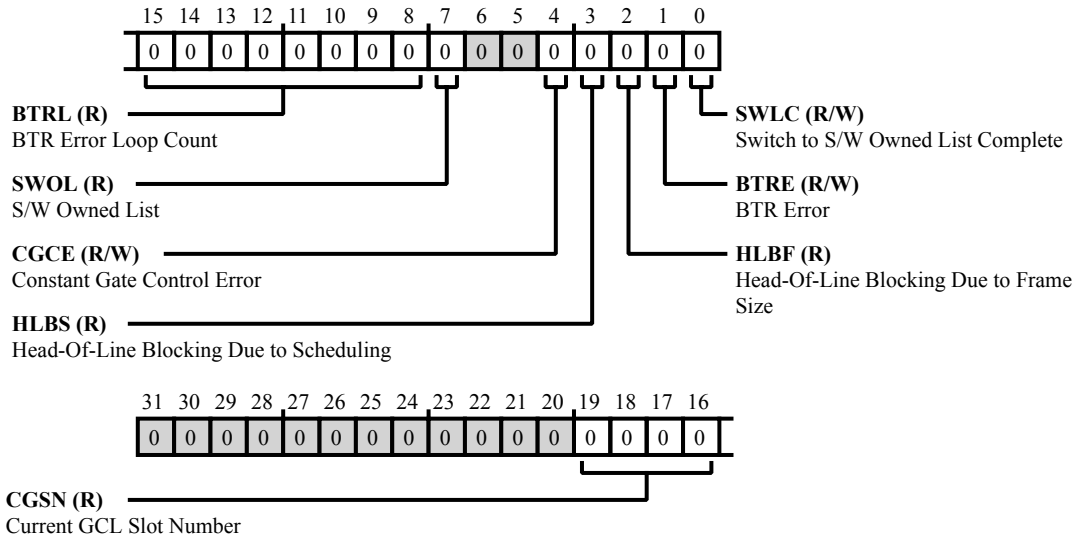


Figure 30-209: EMAC_EST_STAT Register Diagram

Table 30-264: EMAC_EST_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:16 (R/NW)	CGSN	Current GCL Slot Number. The EMAC_EST_STAT.CGSN field indicates the slot number of the GCL list. Slot number is a modulo 16 count of the GCL list loops executed so far. Even if a new GCL list is installed, the count is incremental.
15:8 (R/NW)	BTRL	BTR Error Loop Count. The EMAC_EST_STAT.BTRL field provides the minimum count (N) for which the equation Current Time =< New BTR + (N * New Cycle Time) becomes true. N = "11111111" indicates the iterations exceeded the value of 128 and the hardware was not able to update New BTR to be equal to or greater than Current Time. Software intervention is needed to update the new BTR. Value is cleared when EMAC_EST_STAT.BTRE = 0.

Table 30-264: EMAC_EST_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	SWOL	S/W Owned List. When the EMAC_EST_STAT.SWOL bit is cleared (=0), it indicates that the gate control list number "0" is owned by software. When set (=1), the EMAC_EST_STAT.SWOL bit indicates that Gate Control list "1" is owned by the software. Any reads/writes by the software (using indirect access through GCL_Control) is directed to the list indicated by this value by default. The inverse of this value is treated as HWOL. R/W operations performed by hardware are directed to the list pointed by HWOL by default.	
		0	Gate Control List Number "0" is Owned by Software
		1	Gate Control List Number "1" is Owned by Software
4 (R/W)	CGCE	Constant Gate Control Error. When set (=1), the EMAC_EST_STAT.CGCE bit indicates that an error occurs when the list length (LLR) is 1 and the Cycle Time (CTR) is less than or equal to the programmed Time Interval (TI) value after the optional left shifting. This implies that gates are either always closed or always open based on the Gate Control values; the same effect can be achieved by other simpler (non TSN) programming mechanisms. Since the implementation does not support such a programming an error is reported. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Constant Gate Control Error Not Detected
		1	Constant Gate Control Error Detected
3 (R/NW)	HLBS	Head-Of-Line Blocking Due to Scheduling. When set (=1), the EMAC_EST_STAT.HLBS bit indicates when the frame is not able to win arbitration and get scheduled even after four iterations of the GCL. Indicates to software a potential programming error. The one hot encoded values of the queue numbers that are not able to make progress are indicated in the EMAC_EST_SCH_ERR register. The bit is cleared when the EMAC_EST_SCH_ERR register is all zeros.	
		0	Head-Of-Line Blocking Due to Scheduling Not Detected
		1	Head-Of-Line Blocking Due to Scheduling Detected

Table 30-264: EMAC_EST_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/NW)	HLBF	Head-Of-Line Blocking Due to Frame Size. When set (=1), the EMAC_EST_STAT.HLBF bit indicates HOL Blocking is noticed on one or more queues as a result of none of the time intervals of gate open in the GCL being greater than or equal to the duration needed for frame size (or frame fragment size when preemption is enabled) transmission. The one hot encoded queue numbers that are experiencing HLBF are indicated in the EMAC_EST_FRMSZ_ERR register. Additionally, the first queue number that experienced HLBF along with the frame size is captured in EMAC_EST_FRMSZ_CAP register. The EMAC_EST_STAT.HLBF bit is cleared when EMAC_EST_FRMSZ_ERR register is all zeros.	
		0	Head-Of-Line Blocking Due to Frame Size Not Detected
		1	Head-Of-Line Blocking Due to Frame Size Detected
1 (R/W)	BTRE	BTR Error. When set (=1), the EMAC_EST_STAT.BTRE bit indicates a programming error in the BTR of SWOL where the programmed value is less than current time. If EMAC_EST_STAT.BTRL = "11111111", EMAC_EST_STAT.SWOL is not updated and software should reprogram the BTR to a value greater than current time and then set the SSWL to reinitiate the switch to EMAC_EST_STAT.SWOL. Else, if the value of EMAC_EST_STAT.BTRL < "11111111", the EMAC_EST_STAT.SWOL bit is updated and this field indicates the number of iterations (of + CycleTime) taken by hardware to update the BTR to a value greater than Current Time. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	BTR Error Not Detected
		1	BTR Error Detected
0 (R/W)	SWLC	Switch to S/W Owned List Complete. When EMAC_EST_STAT.SWLC= 1, the hardware has successfully switched to the SWOL, and the EMAC_EST_STAT.SWOL bit has been updated to that effect. The EMAC_EST_STAT.SWLC bit is cleared when the EMAC_EST_CTL.SSWL bit transitions from 0 to 1, or on a software write. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Switch to S/W Owned List Complete Not Detected
		1	Switch to S/W Owned List Complete Detected

Frame Preemption Advance Register

The EMAC_FPE_ADV register contains the hold and release advance time.

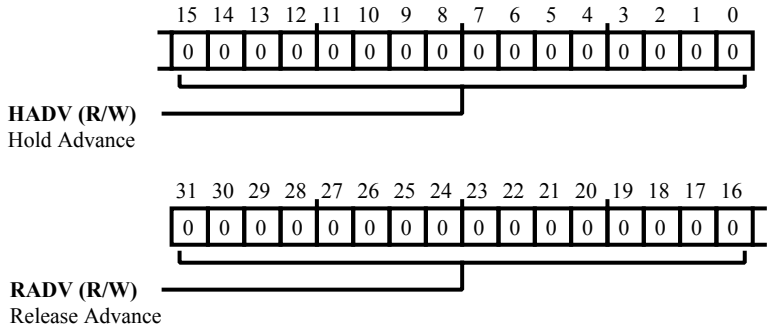


Figure 30-210: EMAC_FPE_ADV Register Diagram

Table 30-265: EMAC_FPE_ADV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	RADV	Release Advance. The EMAC_FPE_ADV.RADV field indicates the maximum time in nanoseconds that can elapse between issuing a RELEASE to the MAC and the MAC being ready to resume transmission of preemptable frames, in the absence of there being any express frames available for transmission.
15:0 (R/W)	HADV	Hold Advance. The EMAC_FPE_ADV.HADV field indicates the maximum time in nanoseconds that can elapse between issuing a HOLD to the MAC, and the MAC ceasing to transmit any preemptable frame that is in the process of transmission or any preemptable frames that are queued for transmission.

MTL Frame Preemption Control and Status Register

The EMAC_FPE_CTRLSTS register controls the operation of, and provides status for frame preemption (IEEE802.1Qbu/802.3br).

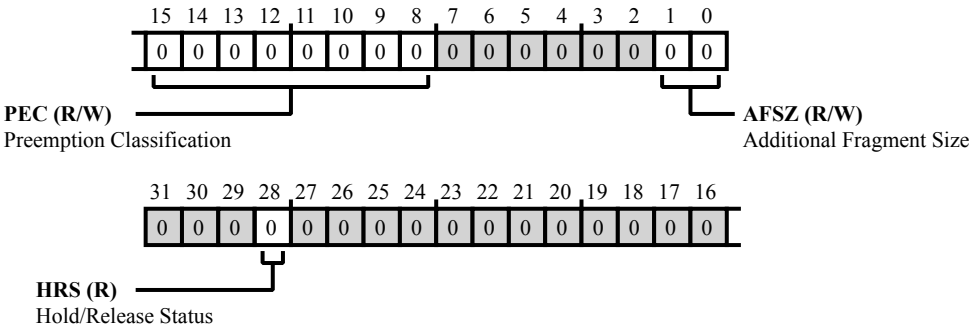


Figure 30-211: EMAC_FPE_CTRLSTS Register Diagram

Table 30-266: EMAC_FPE_CTRLSTS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/NW)	HRS	Hold/Release Status. The EMAC_FPE_CTRLSTS.HRS bit indicates whether a set-and-hold MAC operation was last executed and the pMAC is in hold state or a set-and-release MAC operation was last executed and the pMAC is in the release state.	
		0	Release
		1	Hold
15:8 (R/W)	PEC	Preemption Classification. When set (=1), the EMAC_FPE_CTRLSTS.PEC field indicates that the corresponding queue must be classified as preemptable, when '0' queue is classified as express. When both EST (Qbv) and preemption are enabled, queue 0 is always assumed to be preemptable. When EST (Qbv) is enabled, queues categorized as preemptable here are always assumed to be in "Open" state in the Gate Control List.	
1:0 (R/W)	AFSZ	Additional Fragment Size. The EMAC_FPE_CTRLSTS.AFSZ field indicates, in units of 64 bytes, the minimum number of bytes over 64 bytes required in non-final fragments of preempted frames. The minimum non-final fragment size is (AFSZ +1) * 64 bytes.	

MTL Interrupt Status Register

The software driver (application) reads the EMAC_MTL_ISTAT register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

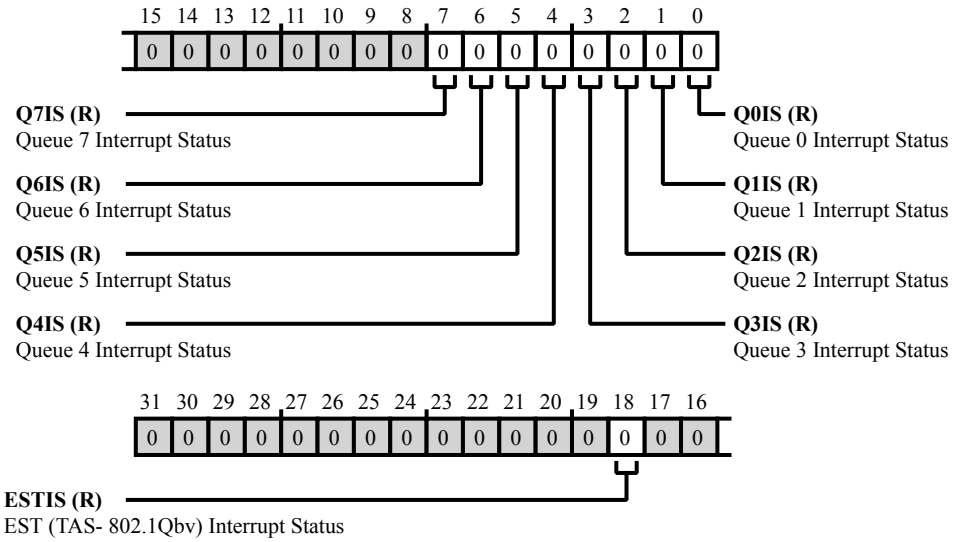


Figure 30-212: EMAC_MTL_ISTAT Register Diagram

Table 30-267: EMAC_MTL_ISTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/NW)	ESTIS	EST (TAS- 802.1Qbv) Interrupt Status. The EMAC_MTL_ISTAT.ESTIS bit indicates an interrupt event during the operation of 802.1Qbv. To reset the EMAC_MTL_ISTAT.ESTIS bit, the application must clear the error/event that has caused the Interrupt.	
		0	EST (TAS- 802.1Qbv) Interrupt Status Not Detected
		1	EST (TAS- 802.1Qbv) Interrupt Status Detected
7 (R/NW)	Q7IS	Queue 7 Interrupt Status. The EMAC_MTL_ISTAT.Q7IS bit indicates that there is an interrupt from queue 7. To reset the EMAC_MTL_ISTAT.Q7IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected

Table 30-267: EMAC_MTL_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/NW)	Q6IS	Queue 6 Interrupt Status. The EMAC_MTL_ISTAT.Q6IS bit indicates that there is an interrupt from queue 6. To reset the EMAC_MTL_ISTAT.Q6IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
5 (R/NW)	Q5IS	Queue 5 Interrupt Status. The EMAC_MTL_ISTAT.Q5IS bit indicates that there is an interrupt from queue 5. To reset the EMAC_MTL_ISTAT.Q5IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
4 (R/NW)	Q4IS	Queue 4 Interrupt Status. The EMAC_MTL_ISTAT.Q4IS bit indicates that there is an interrupt from queue 4. To reset the EMAC_MTL_ISTAT.Q4IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
3 (R/NW)	Q3IS	Queue 3 Interrupt Status. The EMAC_MTL_ISTAT.Q3IS bit indicates that there is an interrupt from queue 3. To reset the EMAC_MTL_ISTAT.Q3IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
2 (R/NW)	Q2IS	Queue 2 Interrupt Status. The EMAC_MTL_ISTAT.Q2IS bit indicates that there is an interrupt from queue 2. To reset the EMAC_MTL_ISTAT.Q2IS bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected

Table 30-267: EMAC_MTL_ISTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	Q1IS	Queue 1 Interrupt Status. The `EMAC_MTL_ISTAT.Q1IS` bit indicates that there is an interrupt from queue 1. To reset the `EMAC_MTL_ISTAT.Q1IS` bit, the application must read the EMAC_Q[n]_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected
0 (R/NW)	Q0IS	Queue 0 Interrupt Status. The `EMAC_MTL_ISTAT.Q0IS` bit indicates that there is an interrupt from queue 0. To reset the `EMAC_MTL_ISTAT.Q0IS` bit, the application must read the EMAC_Q0_INT_CTLSTAT register to get the exact cause of the interrupt and clear its source.	
		0	Status Not Detected
		1	Status Detected

MTL Operation Mode Register

The EMAC_MTL_OPMODE register establishes the transmit and receive operating modes and commands.

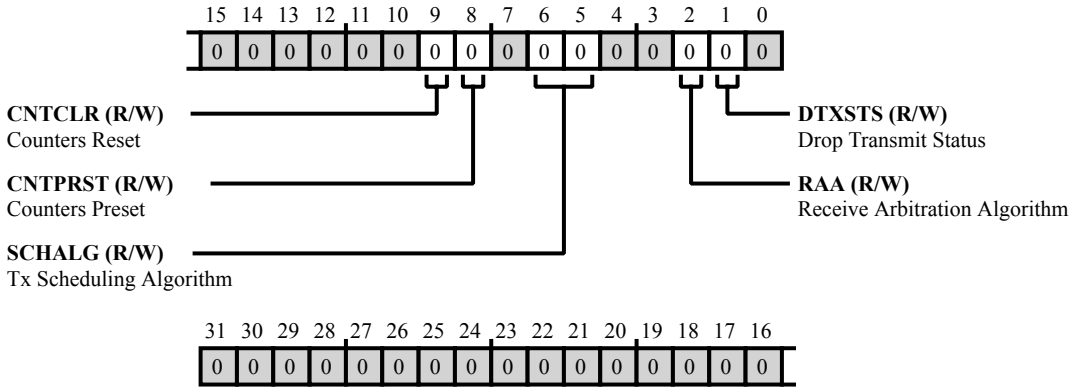


Figure 30-213: EMAC_MTL_OPMODE Register Diagram

Table 30-268: EMAC_MTL_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	CNTCLR	Counters Reset. When EMAC_MTL_OPMODE.CNTCLR bit = 1, all counters are reset. The EMAC_MTL_OPMODE.CNTCLR bit is cleared automatically after 1 clock cycle. If EMAC_MTL_OPMODE.CNTCLR and EMAC_MTL_OPMODE.CNTPRST are both set (=1), the EMAC_MTL_OPMODE.CNTPRST bit has precedence. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Counters are Not Reset
		1	All Counters are Reset
8 (R/W)	CNTPRST	Counters Preset. When set (=1), the EMAC_MTL_OPMODE.CNTPRST bit indicates that the EMAC_TQ[n]_URFL and EMAC_RQ[n]_MSPKTOF_CNT registers are initialized and preset to 12'h7F0. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Disable
		1	Enable

Table 30-268: EMAC_MTL_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:5 (R/W)	SCHALG	Tx Scheduling Algorithm. The `EMAC_MTL_OPMODE.SCHALG` field indicates the algorithm for Tx scheduling.	
		0	WRR Algorithm
		1	WFQ Algorithm
		2	DWRR Algorithm
		3	Strict Priority Algorithm
2 (R/W)	RAA	Receive Arbitration Algorithm. The `EMAC_MTL_OPMODE.RAA` bit indicates the arbitration algorithm for the Rx side.	
		0	Strict Priority (SP). Queue 0 has the lowest priority and the last queue has the highest priority.
		1	Weighted Strict Priority (WSP)
1 (R/W)	DTXSTS	Drop Transmit Status. When set, the `EMAC_MTL_OPMODE.DTXSTS` bit indicates that the Tx packet status received from the MAC is dropped in the MTL. When cleared (=0), the packet status is forwarded to the application.	
		0	Disable
		1	Enable

Queue 0 Interrupt Control and Status Register

The EMAC_Q0_INT_CTLSTAT register contains the interrupt enable and status bits for the queue 0 interrupts.

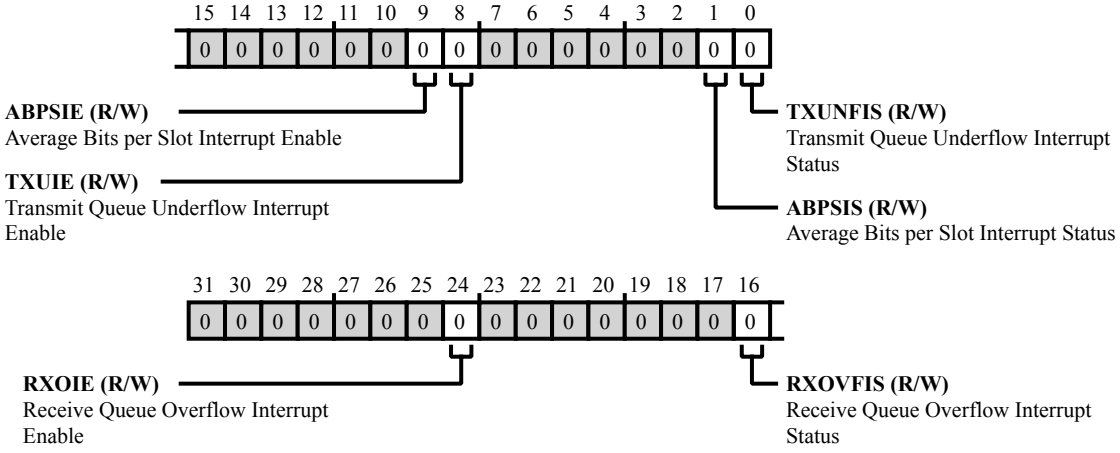


Figure 30-214: EMAC_Q0_INT_CTLSTAT Register Diagram

Table 30-269: EMAC_Q0_INT_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	RXOIE	Receive Queue Overflow Interrupt Enable. When the EMAC_Q0_INT_CTLSTAT.RXOIE bit is set, the Receive Queue Overflow interrupt is enabled. When EMAC_Q0_INT_CTLSTAT.RXOIE = 0, the Receive Queue Overflow interrupt is disabled.	
		0	Receive Queue Overflow Interrupt is Disabled
		1	Receive Queue Overflow Interrupt is Enabled
16 (R/W)	RXOVFIS	Receive Queue Overflow Interrupt Status. The EMAC_Q0_INT_CTLSTAT.RXOVFIS bit indicates that the receive queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. The EMAC_Q0_INT_CTLSTAT.RXOVFIS bit is cleared when the application writes 1 to the EMAC_Q0_INT_CTLSTAT.RXOVFIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Receive Queue Overflow Interrupt Status Not Detected
		1	Receive Queue Overflow Interrupt Status Detected

Table 30-269: EMAC_Q0_INT_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	ABPSIE	Average Bits per Slot Interrupt Enable. When the EMAC_Q0_INT_CTLSTAT.ABPSIE bit is set, the MAC asserts the EMAC_STAT or EMAC_MAC interrupt when the average bits per slot status is updated. When the EMAC_Q0_INT_CTLSTAT.ABPSIE bit is cleared, the interrupt is not asserted for such an event.	
		0	Average Bits per Slot Interrupt is Disabled
		1	Average Bits per Slot Interrupt is Enabled
8 (R/W)	TXUIE	Transmit Queue Underflow Interrupt Enable. When the EMAC_Q0_INT_CTLSTAT.TXUIE bit is set, the Transmit Queue Underflow interrupt is enabled. When EMAC_Q0_INT_CTLSTAT.TXUIE = 0, the Transmit Queue Underflow interrupt is disabled.	
		0	Transmit Queue Underflow Interrupt Status is Disabled
		1	Transmit Queue Underflow Interrupt Status is Enabled
1 (R/W)	ABPSIS	Average Bits per Slot Interrupt Status. When set, the EMAC_Q0_INT_CTLSTAT.ABPSIS bit indicates that the MAC has updated the ABS value. The EMAC_Q0_INT_CTLSTAT.ABPSIS bit is cleared when the application writes 1 to the EMAC_Q0_INT_CTLSTAT.ABPSIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Average Bits per Slot Interrupt Status Not Detected
		1	Average Bits per Slot Interrupt Status Detected
0 (R/W)	TXUNFIS	Transmit Queue Underflow Interrupt Status. The EMAC_Q0_INT_CTLSTAT.TXUNFIS bit indicates that the transmit queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. The EMAC_Q0_INT_CTLSTAT.TXUNFIS bit is cleared when the application writes 1 to the EMAC_Q0_INT_CTLSTAT.TXUNFIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Transmit Queue Underflow Interrupt Status Not Detected
		1	Transmit Queue Underflow Interrupt Status Detected

Queue n Interrupt Control and Status Register

The EMAC_Q[n]_INT_CTLSTAT register contains the interrupt enable and status bits for the queue 1 interrupts.

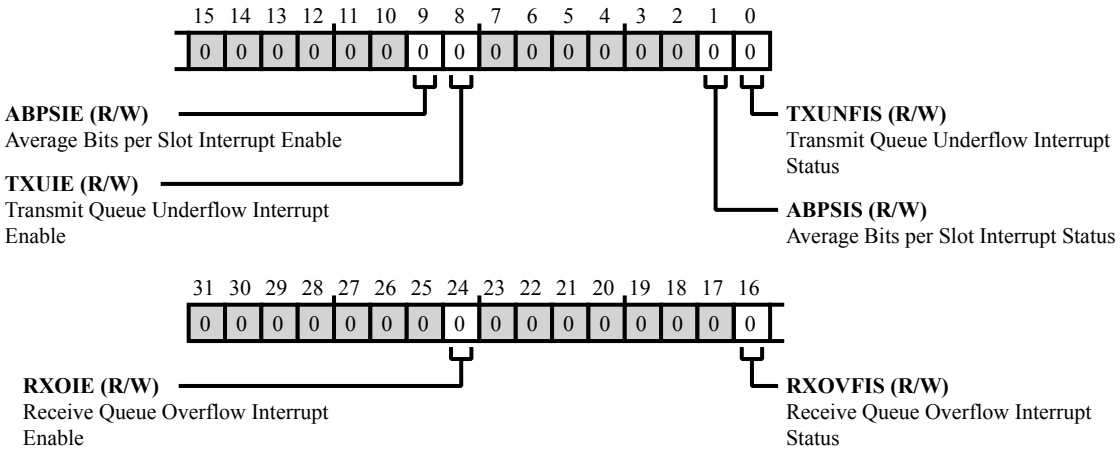


Figure 30-215: EMAC_Q[n]_INT_CTLSTAT Register Diagram

Table 30-270: EMAC_Q[n]_INT_CTLSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	RXOIE	Receive Queue Overflow Interrupt Enable. When the EMAC_Q[n]_INT_CTLSTAT.RXOIE bit is set, the receive queue overflow interrupt interrupt is enabled. When EMAC_Q[n]_INT_CTLSTAT.RXOIE = 0, the receive queue overflow interrupt is disabled.	
		0	Disabled
		1	Enabled
16 (R/W)	RXOVFIS	Receive Queue Overflow Interrupt Status. The EMAC_Q[n]_INT_CTLSTAT.RXOVFIS bit indicates that the receive queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. The EMAC_Q[n]_INT_CTLSTAT.RXOVFIS bit is cleared when the application writes 1 to the EMAC_Q[n]_INT_CTLSTAT.RXOVFIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Status Not Detected
		1	Status Detected

Table 30-270: EMAC_Q[n]_INT_CTLSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	ABPSIE	Average Bits per Slot Interrupt Enable. When the EMAC_Q[n]_INT_CTLSTAT.ABPSIE bit is set, the MAC asserts the EMAC_STAT or EMAC_MAC interrupt when the average bits per slot status is updated. When the EMAC_Q[n]_INT_CTLSTAT.ABPSIE bit is cleared, the interrupt is not asserted for such an event.	
		0	Disabled
		1	Enabled
8 (R/W)	TXUIE	Transmit Queue Underflow Interrupt Enable. When the EMAC_Q[n]_INT_CTLSTAT.TXUIE bit is set, the transmit queue underflow interrupt is enabled. When EMAC_Q[n]_INT_CTLSTAT.TXUIE = 0, the transmit queue underflow interrupt is disabled.	
		0	Disabled
		1	Enabled
1 (R/W)	ABPSIS	Average Bits per Slot Interrupt Status. When set, the EMAC_Q[n]_INT_CTLSTAT.ABPSIS bit indicates that the MAC has updated the ABS value. The EMAC_Q[n]_INT_CTLSTAT.ABPSIS bit is cleared when the application writes 1 to the EMAC_Q[n]_INT_CTLSTAT.ABPSIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Status Not Detected
		1	Status Detected
0 (R/W)	TXUNFIS	Transmit Queue Underflow Interrupt Status. The EMAC_Q[n]_INT_CTLSTAT.TXUNFIS bit indicates that the transmit queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. The EMAC_Q[n]_INT_CTLSTAT.TXUNFIS bit is cleared when the application writes 1 to the EMAC_Q[n]_INT_CTLSTAT.TXUNFIS bit. Access restriction applies. Self-set to 1 on internal event. Write 1 to clear. Writing 0 has no effect.	
		0	Status Not Detected
		1	Status Detected

Receive Queue Control Register

The EMAC_RQ0_CTL register controls the receive arbitration and passing of received packets to the application.

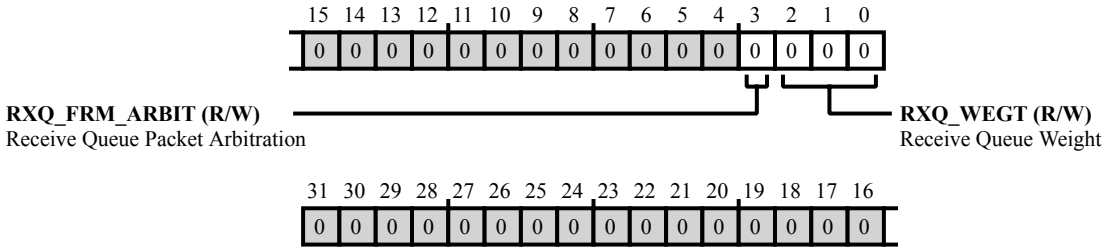


Figure 30-216: EMAC_RQ0_CTL Register Diagram

Table 30-271: EMAC_RQ0_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXQ_FRM_ARBIT	Receive Queue Packet Arbitration. When the EMAC_RQ0_CTL.RXQ_FRM_ARBIT bit is set, the EMAC module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When EMAC_RQ0_CTL.RXQ_FRM_ARBIT = 0, the EMAC module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: - PBL amount of data or - Complete data of a packet The status and the time stamp are not a part of the PBL data. Therefore, the EMAC module drives the complete status (including time stamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in threshold mode).	
		0	Receive Queue Packet Arbitration is Disabled
		1	Receive Queue Packet Arbitration is Enabled

Table 30-271: EMAC_RQ0_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	RXQ_WEGT	Receive Queue Weight. The `EMAC_RQ0_CTL.RXQ_WEGT` field indicates the weight assigned to the Rx queue 0. The field must be programmed with one value less than the required weight, that is, reset value of 0 indicates weight of 1, value of 1 indicates weight of 2, and so on. The weight is used as the number of continuous PBL or packets requests (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle. Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the RXQ_WEGT value to take effect at the start, the EMAC_RQ[n]_CTL registers must be programmed before the EMAC_MTL_OPMODE register.

Receive Queue 0 Debug Register

The EMAC_RQ0_DBG register gives the debug status of various blocks related to the receive queue.

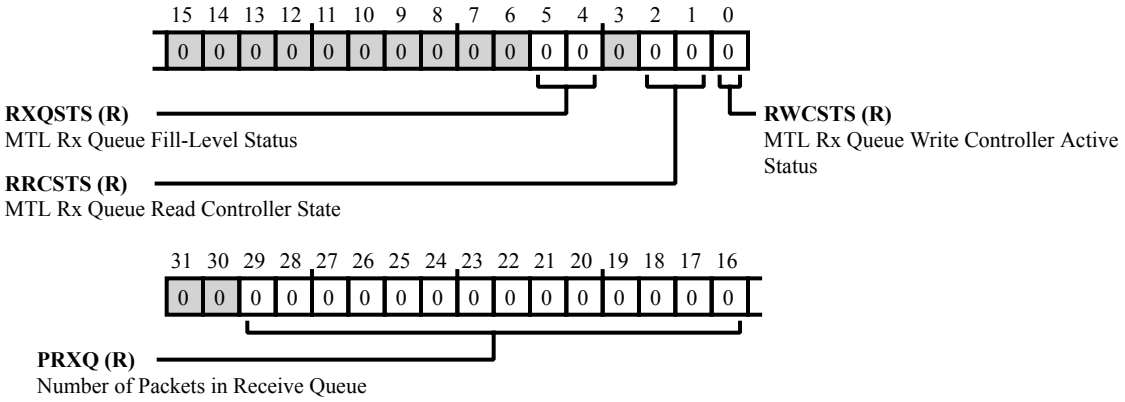


Figure 30-217: EMAC_RQ0_DBG Register Diagram

Table 30-272: EMAC_RQ0_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:16 (R/NW)	PRXQ	Number of Packets in Receive Queue. The EMAC_RQ0_DBG.PRXQ field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.	
5:4 (R/NW)	RXQSTS	MTL Rx Queue Fill-Level Status. The EMAC_RQ0_DBG.RXQSTS field gives the status of the fill-level of the Rx queue:	
		0	Rx Queue Empty
		1	Rx Queue Fill-level Below Flow-control Deactivate Threshold
		2	Rx Queue Fill-level Above Flow-control Activate Threshold
		3	Rx Queue Full
2:1 (R/NW)	RRCSTS	MTL Rx Queue Read Controller State. The EMAC_RQ0_DBG.RRCSTS field gives the state of the Rx queue read controller:	
		0	Idle State
		1	Reading Packet Data
		2	Reading Packet Status (or Timestamp)
		3	Flushing the Packet Data and Status

Table 30-272: EMAC_RQ0_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	RWCSTS	MTL Rx Queue Write Controller Active Status. When high, the `EMAC_RQ0_DBG.RWCSTS` bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.	
		0	MTL Rx Queue Write Controller Active Status Not Detected
		1	MTL Rx Queue Write Controller Active Status Detected

Receive Queue 0 Missed Packet and Overflow Counter Register

The EMAC_RQ0_MSPKTOF_CNT register contains the counter for packets missed because of receive queue packet flush and packets discarded because of receive queue overflow.

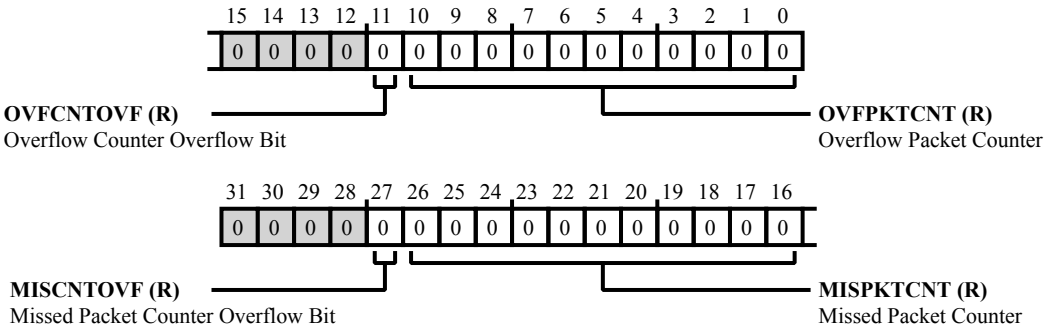


Figure 30-218: EMAC_RQ0_MSPKTOF_CNT Register Diagram

Table 30-273: EMAC_RQ0_MSPKTOF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	MISCNTOVF	Missed Packet Counter Overflow Bit. When set, the EMAC_RQ0_MSPKTOF_CNT.MISCNTOVF bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Missed Packet Counter Overflow Not Detected
		1	Missed Packet Counter Overflow Detected
26:16 (R/NW)	MISPKTCNT	Missed Packet Counter. The EMAC_RQ0_MSPKTOF_CNT.MISPKTCNT field indicates the number of packets missed by the EMAC module because the application asserted EMAC_DMA[n]_RXCTL.RPF for this queue. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
11 (R/NW)	OVFCNTOVF	Overflow Counter Overflow Bit. When set, the EMAC_RQ0_MSPKTOF_CNT.OVFCNTOVF bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Overflow Counter Overflow Not Detected
		1	Overflow Counter Overflow Detected

Table 30-273: EMAC_RQ0_MSPKTOF_CNT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	OVFPKTCNT	Overflow Packet Counter. The `EMAC_RQ0_MSPKTOF_CNT.OVFPKTCNT` field indicates the number of packets discarded by the EMAC module because of receive queue overflow. This counter is incremented each time the EMAC module discards an incoming packet because of overflow. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Receive Queue 0 Operation Mode Register

The EMAC_RQ0_OPMODE register establishes the receive queue operating modes and command.

The RFA and RFD fields are not backward compatible with the RFA and RFD fields of 4.00a release.

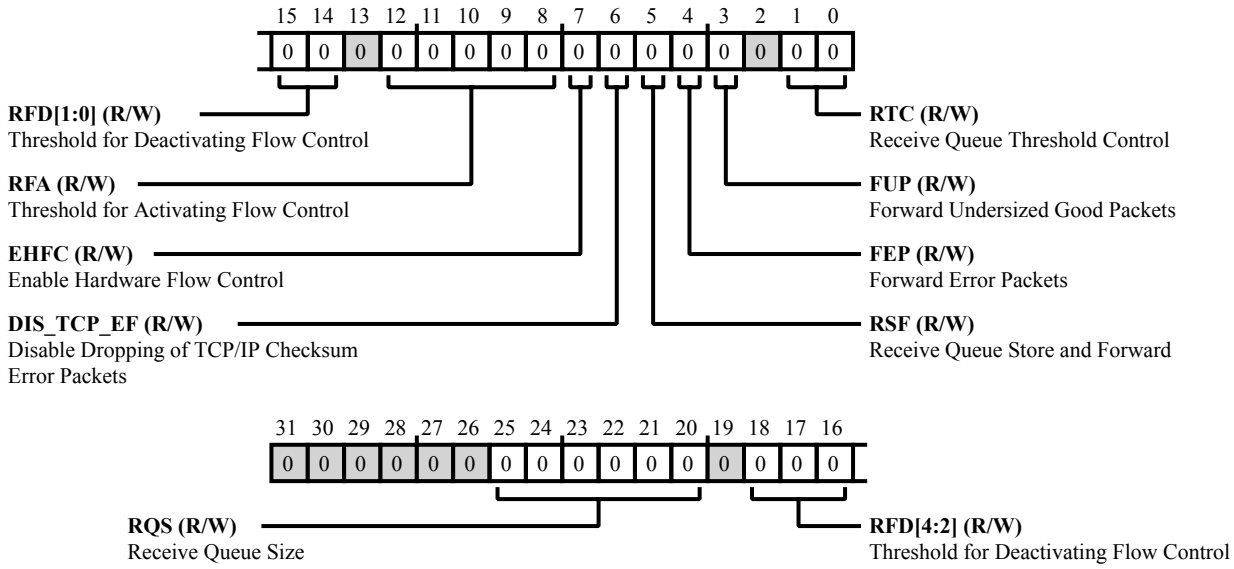


Figure 30-219: EMAC_RQ0_OPMODE Register Diagram

Table 30-274: EMAC_RQ0_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:20 (R/W)	RQS	Receive Queue Size. The EMAC_RQ0_OPMODE.RQS field indicates the size of the allocated receive queues in blocks of 256 bytes. The field is read-write only if the number of Rx queues more than one, the reset value is 0x0 and indicates size of 256 bytes. For example, the value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. So, program RQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the queue = (RQS+1)*256 bytes. When the number of Rx queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value.

Table 30-274: EMAC_RQ0_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18:14 (R/W)	RFD	Threshold for Deactivating Flow Control. The EMAC_RQ0_OPMODE.RFD field indicates the threshold (fill-level of Rx queue) at which the flow control is deasserted after activation. It is valid in half-duplex and full-duplex mode). 0: Full minus 1 KB, that is, FULL 1 KB 1: Full minus 1.5 KB, that is, FULL 1.5 KB 2: Full minus 2 KB, that is, FULL 2 KB 3: Full minus 2.5 KB, that is, FULL 2.5 KB ... 30: Full minus 16 KB, that is, FULL 16 KB 31: Full minus 16.5 KB, that is, FULL 16.5 KB The deassertion is effective only after flow control is asserted. Note: The value must be programmed in such a way to make sure that the threshold is a positive number. When the EMAC_RQ0_OPMODE.EHFC bit is set (=1), these values are applicable only when the Rx queue size determined by the EMAC_RQ0_OPMODE.RQS field, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size determined by the EMAC_RQ0_OPMODE.RQS field.	
12:8 (R/W)	RFA	Threshold for Activating Flow Control. The EMAC_RQ0_OPMODE.RFA field indicates the threshold (fill-level of Rx queue) at which the flow control is activated. It is valid in half-duplex and full-duplex mode). For more information on encoding for this field, see RFD.	
7 (R/W)	EHFC	Enable Hardware Flow Control. When the EMAC_RQ0_OPMODE.EHFC bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When cleared (=0), the flow control operation is disabled.	
		0	Hardware Flow Control is Disabled
		1	Hardware Flow Control is Enabled

Table 30-274: EMAC_RQ0_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets. When the EMAC_RQ0_OPMODE.DIS_TCP_EF bit is set, the MAC does not drop the packets which only have the errors detected by the receive checksum offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When EMAC_RQ0_OPMODE.DIS_TCP_EF = 0, all error packets are dropped if EMAC_RQ0_OPMODE.FEP = 0.	
		0	Dropping of TCP/IP Checksum Error Packets is Enabled
		1	Dropping of TCP/IP Checksum Error Packets is Disabled
5 (R/W)	RSF	Receive Queue Store and Forward. When the EMAC_RQ0_OPMODE.RSF bit is set, the EMAC module reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the EMAC_RQ0_OPMODE.RTC field. When EMAC_RQ0_OPMODE.RSF = 0, the Rx queue operates in the threshold (cut-through) mode, subject to the threshold specified by the EMAC_RQ0_OPMODE.RTC field.	
		0	Receive Queue Store and Forward is Disabled
		1	Receive Queue Store and Forward is Enabled
4 (R/W)	FEP	Forward Error Packets. When EMAC_RQ0_OPMODE.FEP = 0, the Rx queue drops packets with error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in threshold mode), the packet is not dropped. When the EMAC_RQ0_OPMODE.FEP bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the EMAC_RQ0_OPMODE.RSF = 1 and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of the EMAC_RQ0_OPMODE.FEP bit. However, if the EMAC_RQ0_OPMODE.RSF = 0 and the Rx queue overflows when a partial packet is written, a partial packet might be forwarded to the application or DMA.	
		0	Forward Error Packets is Disabled
		1	Forward Error Packets is Enabled

Table 30-274: EMAC_RQ0_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	FUP	Forward Undersized Good Packets. When the EMAC_RQ0_OPMODE.FUP bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When EMAC_RQ0_OPMODE.FUP = 0, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx threshold, for example, RTC = 01.	
		0	Forward Undersized Good Packets is Disabled
		1	Forward Undersized Good Packets is Enabled
1:0 (R/W)	RTC	Receive Queue Threshold Control. The EMAC_RQ0_OPMODE.RTC field indicates the threshold level of the MTL Rx queue (in bytes). The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. The EMAC_RQ0_OPMODE.RTC field is valid only when EMAC_RQ0_OPMODE.RSF = 0. The EMAC_RQ0_OPMODE.RTC field is ignored when EMAC_RQ0_OPMODE.RSF = 1.	
		0	64
		1	32
		2	96
		3	128

Receive Queue n Control Register

The EMAC_RQ[n]_CTL register controls the receive arbitration and passing of received packets to the application.

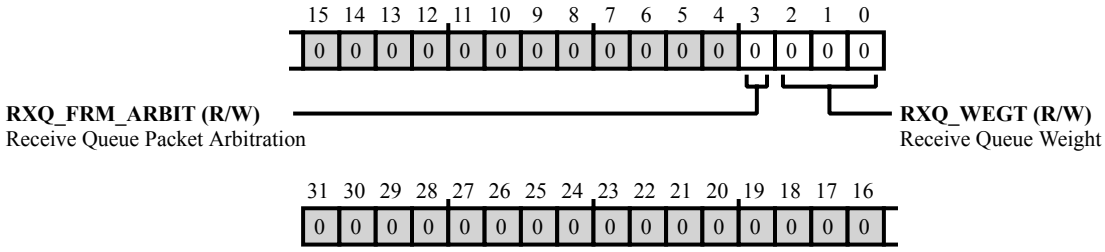


Figure 30-220: EMAC_RQ[n]_CTL Register Diagram

Table 30-275: EMAC_RQ[n]_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RXQ_FRM_ARBIT	Receive Queue Packet Arbitration. When the EMAC_RQ[n]_CTL.RXQ_FRM_ARBIT bit is set, the EMAC module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When EMAC_RQ[n]_CTL.RXQ_FRM_ARBIT = 0, the EMAC module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: - PBL amount of data or - Complete data of a packet The status and the time stamp are not a part of the PBL data. Therefore, the EMAC module drives the complete status (including time stamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in threshold mode).	
		0	Disabled
		1	Enabled

Table 30-275: EMAC_RQ[n]_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	RXQ_WEGT	Receive Queue Weight. The `EMAC_RQ[n]_CTL.RXQ_WEGT` field indicates the weight assigned to the Rx queue 0. The `EMAC_RQ[n]_CTL.RXQ_WEGT` field needs to be programmed with one value less than the required weight. For example, a reset value of 0 indicates a weight of 1, a value of 1 indicates weight of 2, and so on. The weight is used as the number of continuous PBL or packets requests (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle. Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the RXQ_WEGT value to take effect at the start, the EMAC_RQ[n]_CTL registers must be programmed before the EMAC_MTL_OPMODE register.

Receive Queue n Debug Register

The EMAC_RQ[n]_DBG register gives the debug status of various blocks related to the receive queue.

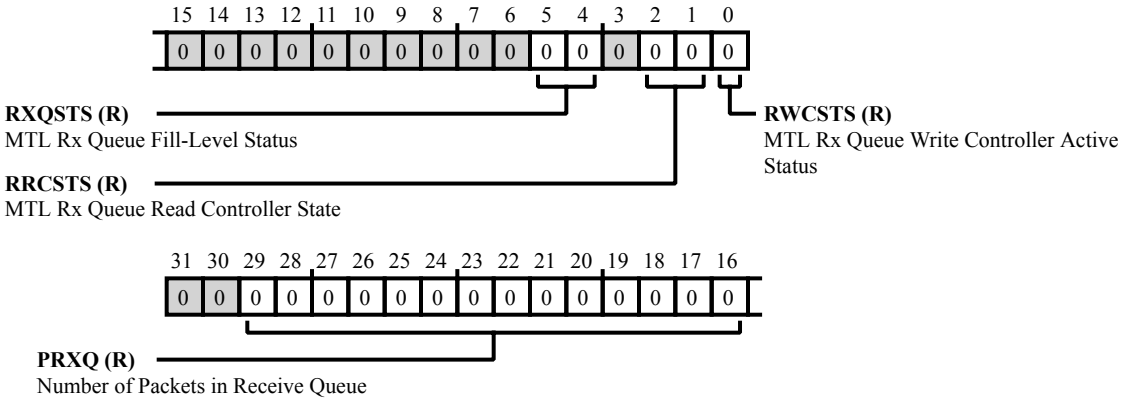


Figure 30-221: EMAC_RQ[n]_DBG Register Diagram

Table 30-276: EMAC_RQ[n]_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29:16 (R/NW)	PRXQ	Number of Packets in Receive Queue. The EMAC_RQ[n]_DBG.PRXQ field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256KB/16B = 16K packets, that is, Max_Queue_Size/Min_Packet_Size.	
5:4 (R/NW)	RXQSTS	MTL Rx Queue Fill-Level Status. The EMAC_RQ[n]_DBG.RXQSTS field gives the status of the fill-level of the Rx queue:	
		0	Empty
		1	Below Flow Control Deactivate Threshold
		2	Above Flow Control Activate Threshold
		3	Full
2:1 (R/NW)	RRCSTS	MTL Rx Queue Read Controller State. The EMAC_RQ[n]_DBG.RRCSTS field gives the state of the Rx queue read controller.	
		0	Idle State
		1	Reading Packet Data
		2	Reading Packet Status (or Time Stamp)
		3	Flushing the Packet Data and Status

Table 30-276: EMAC_RQ[n]_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/NW)	RWCSTS	MTL Rx Queue Write Controller Active Status. When high, the `EMAC_RQ[n]_DBG.RWCSTS` bit indicates that the MTL Rx queue write controller is active, and it is transferring a received packet to the Rx queue.	
		0	Status Not Detected
		1	Status Detected

Receive Queue n Missed Packet and Overflow Counter Register

The EMAC_RQ[n]_MSPKTOF_CNT register contains the counter for packets missed because of receive queue packet flush and packets discarded because of receive queue overflow.

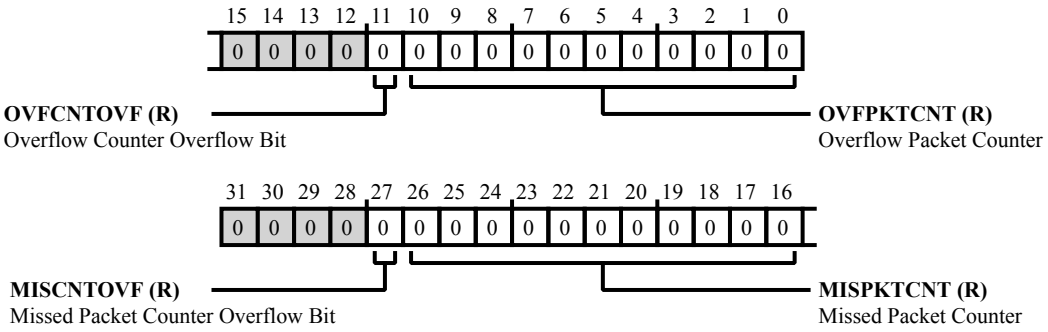


Figure 30-222: EMAC_RQ[n]_MSPKTOF_CNT Register Diagram

Table 30-277: EMAC_RQ[n]_MSPKTOF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
27 (R/NW)	MISCNTOVF	Missed Packet Counter Overflow Bit. When set, the EMAC_RQ[n]_MSPKTOF_CNT.MISCNTOVF bit indicates that the Rx queue missed packet counter crossed the maximum limit. Access restriction applies. Clears (=0) on read. Self-set to 1 on internal event.	
		0	Overflow Not Detected
		1	Overflow Detected
26:16 (R/NW)	MISPKTCNT	Missed Packet Counter. The EMAC_RQ[n]_MSPKTOF_CNT.MISPKTCNT field indicates the number of packets missed by the EMAC module because the application asserted EMAC_DMA[n]_RXCTL.RPF for this queue. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
11 (R/NW)	OVFCNTOVF	Overflow Counter Overflow Bit. When set, the EMAC_RQ[n]_MSPKTOF_CNT.OVFCNTOVF bit indicates that the Rx queue overflow packet counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Overflow Not Detected
		1	Overflow Detected

Table 30-277: EMAC_RQ[n]_MSPKTOF_CNT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	OVFPKTCNT	Overflow Packet Counter. The `EMAC_RQ[n]_MSPKTOF_CNT.OVFPKTCNT` field indicates the number of packets discarded by the EMAC module because of receive queue overflow. This counter is incremented each time the EMAC module discards an incoming packet because of overflow. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Receive Queue n Operation Mode Register

The EMAC_RQ[n]_OPMODE register establishes the receive queue operating modes and command.

The RFA and RFD fields are not backward compatible with the RFA and RFD fields of 4.00a release

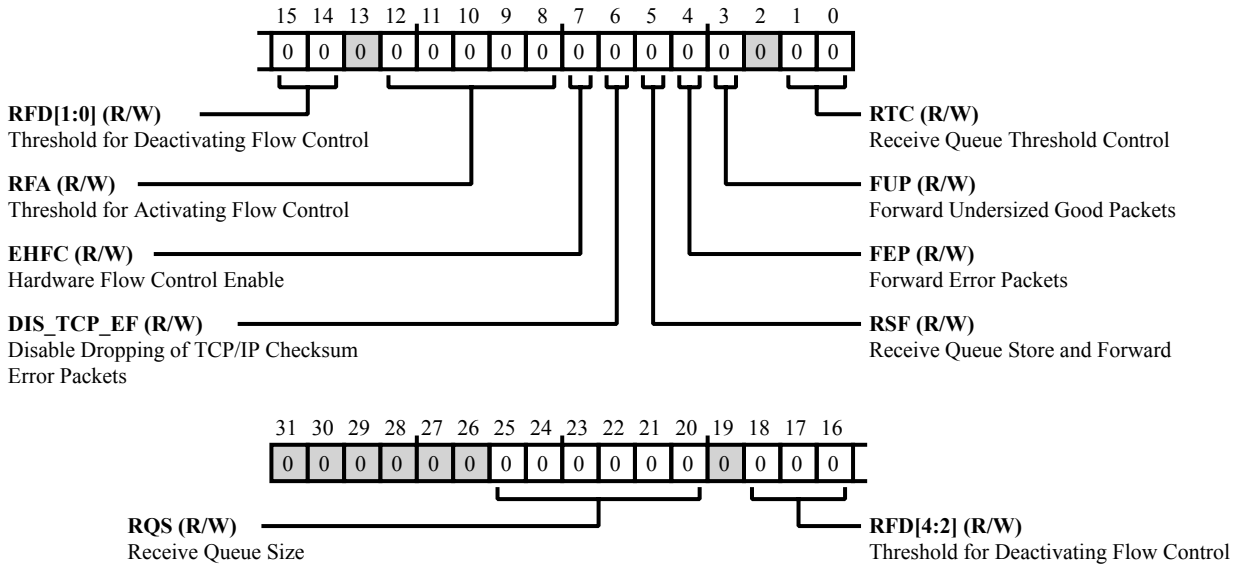


Figure 30-223: EMAC_RQ[n]_OPMODE Register Diagram

Table 30-278: EMAC_RQ[n]_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:20 (R/W)	RQS	Receive Queue Size. The EMAC_RQ[n]_OPMODE.RQS field indicates the size of the allocated receive queues in blocks of 256 bytes. The field is read-write only if the number of Rx queues is more than one; the reset value is 0x0 and indicates size of 256 bytes. For example, a value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. Program RQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the queue = (RQS +1)*256 bytes. When the number of Rx queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value.

Table 30-278: EMAC_RQ[n]_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18:14 (R/W)	RFD	Threshold for Deactivating Flow Control. The EMAC_RQ[n]_OPMODE.RFD field indicates the threshold (fill-level of Rx queue) at which the flow control is deasserted after activation (in half-duplex and full-duplex modes): - 0: Full minus 1 KB, that is, FULL 1 KB - 1: Full minus 1.5 KB, that is, FULL 1.5 KB - 2: Full minus 2 KB, that is, FULL 2 KB - 3: Full minus 2.5 KB, that is, FULL 2.5 KB - ... - 30: Full minus 16 KB, that is, FULL 16 KB - 31: Full minus 16.5 KB, that is, FULL 16.5 KB The deassertion is effective only after flow control is asserted. Note: The value must be programmed in such a way to make sure that the threshold is a positive number. When EMAC_RQ[n]_OPMODE.EHFC =1, these values are applicable only when the Rx queue size determined by the EMAC_RQ[n]_OPMODE.RQS field, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here, the term FULL and QSIZE refers to the queue size determined by the EMAC_RQ[n]_OPMODE.RQS field.	
12:8 (R/W)	RFA	Threshold for Activating Flow Control. The EMAC_RQ[n]_OPMODE.RFA field indicates the threshold (fill-level of Rx queue) at which the flow control is activated (in half-duplex and full-duplex modes). For more information on encoding for this field, see RFD.	
7 (R/W)	EHFC	Hardware Flow Control Enable. When the EMAC_RQ[n]_OPMODE.EHFC bit is set (=1), the flow control signal operation, based on the fill-level of Rx queue, is enabled. When cleared (=0), the flow control operation is disabled.	
		0	Disabled
		1	Enabled

Table 30-278: EMAC_RQ[n]_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W)	DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets. When the EMAC_RQ[n]_OPMODE.DIS_TCP_EF bit is set (=1), the MAC does not drop the packets which only have the errors detected by the receive checksum offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When EMAC_RQ[n]_OPMODE.DIS_TCP_EF = 0, all error packets are dropped if EMAC_RQ[n]_OPMODE.FEP = 0.	
		0	Enabled
		1	Disabled
5 (R/W)	RSF	Receive Queue Store and Forward. When the EMAC_RQ[n]_OPMODE.RSF bit is set (=1), the EMAC module reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the EMAC_RQ[n]_OPMODE.RTC field. When EMAC_RQ[n]_OPMODE.RSF = 0, the Rx queue operates in the threshold (cut-through) mode, subject to the threshold specified by the EMAC_RQ[n]_OPMODE.RTC field.	
		0	Disabled
		1	Enabled
4 (R/W)	FEP	Forward Error Packets. When EMAC_RQ[n]_OPMODE.FEP = 0, the Rx queue drops packets with error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in threshold mode), the packet is not dropped. When the EMAC_RQ[n]_OPMODE.FEP bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of the EMAC_RQ[n]_OPMODE.FEP bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet might be forwarded to the application or DMA.	
		0	Disabled
		1	Enabled

Table 30-278: EMAC_RQ[n]_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	FUP	Forward Undersized Good Packets. When the EMAC_RQ[n]_OPMODE.FUP bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When EMAC_RQ[n]_OPMODE.FUP = 0, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx threshold, for example, RTC = 01.	
		0	Disabled
		1	Enabled
1:0 (R/W)	RTC	Receive Queue Threshold Control. The EMAC_RQ[n]_OPMODE.RTC field indicates the threshold level of the MTL Rx queue (in bytes). The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. The EMAC_RQ[n]_OPMODE.RTC field is valid only when the RSF bit is zero. The EMAC_RQ[n]_OPMODE.RTC field is ignored when the RSF bit is set (=1).	
		0	64
		1	32
		2	96
		3	128

Receive Queue to DMA Channel Mapping Register 0

The EMAC_RXQ_DMA_MAP0 register is reserved in EQOS-CORE and EQOS-MTL configurations.

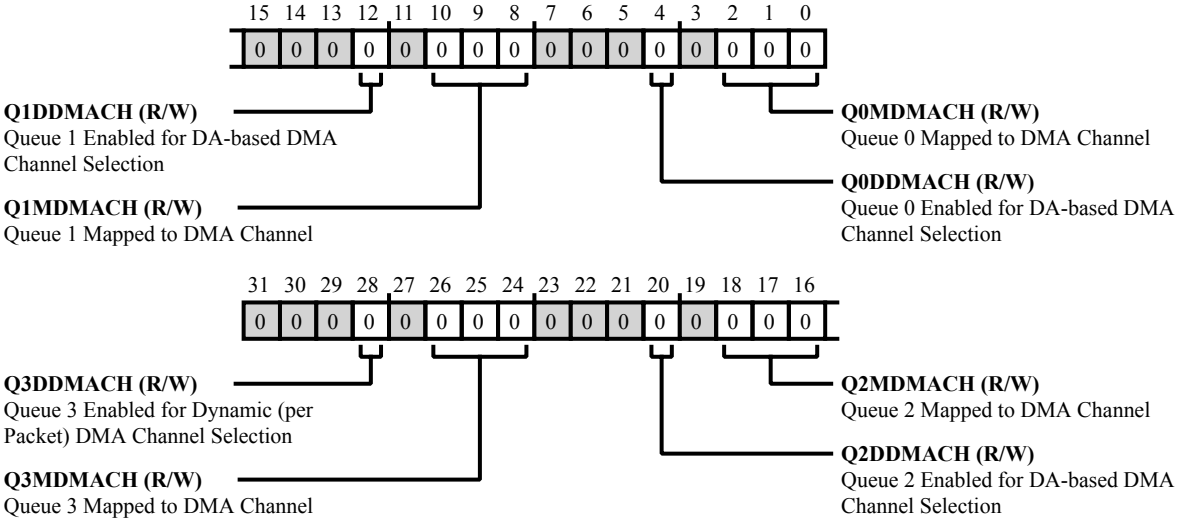


Figure 30-224: EMAC_RXQ_DMA_MAP0 Register Diagram

Table 30-279: EMAC_RXQ_DMA_MAP0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	Q3DDMACH	Queue 3 Enabled for Dynamic (per Packet) DMA Channel Selection. When set, the EMAC_RXQ_DMA_MAP0.Q3DDMACH bit indicates that the packets received in queue 3 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the EMAC_RXQ_DMA_MAP0.Q3DDMACH bit indicates that the packets received in queue 3 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP0.Q3MDMACH field.	
		0	Disable
		1	Enable
26:24 (R/W)	Q3MDMACH	Queue 3 Mapped to DMA Channel. The EMAC_RXQ_DMA_MAP0.Q3MDMACH field controls the routing of the received packet in queue 3 to the DMA channel. The EMAC_RXQ_DMA_MAP0.Q3MDMACH field is valid when the EMAC_RXQ_DMA_MAP0.Q3DDMACH field is reset. Note: The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the others are reserved	
		0	DMA Channel 0

Table 30-279: EMAC_RXQ_DMA_MAP0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		1	DMA Channel 0
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7
20 (R/W)	Q2DDMACH	Queue 2 Enabled for DA-based DMA Channel Selection. When set, the EMAC_RXQ_DMA_MAP0.Q2DDMACH bit indicates that the packets received in queue 2 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the EMAC_RXQ_DMA_MAP0.Q2DDMACH bit indicates that the packets received in queue 2 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP0.Q2MDMACH field.	
		0	Disable
		1	Enable
18:16 (R/W)	Q2MDMACH	Queue 2 Mapped to DMA Channel. The EMAC_RXQ_DMA_MAP0.Q2MDMACH field controls the routing of the received packet in queue 2 to the DMA channel. The EMAC_RXQ_DMA_MAP0.Q2MDMACH field is valid when the EMAC_RXQ_DMA_MAP0.Q2DDMACH field is reset.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7

Table 30-279: EMAC_RXQ_DMA_MAP0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	Q1DDMACH	Queue 1 Enabled for DA-based DMA Channel Selection. When set (=1), the `EMAC_RXQ_DMA_MAP0.Q1DDMACH` bit indicates that the packets received in queue 1 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the `EMAC_RXQ_DMA_MAP0.Q1DDMACH` bit indicates that the packets received in queue 1 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP0.Q1MDMACH field.	
		0	Disable
		1	Enable
10:8 (R/W)	Q1MDMACH	Queue 1 Mapped to DMA Channel. The `EMAC_RXQ_DMA_MAP0.Q1MDMACH` field controls the routing of the received packet in queue 1 to the DMA channel. The `EMAC_RXQ_DMA_MAP0.Q1MDMACH` field is valid when the EMAC_RXQ_DMA_MAP0.Q1DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7
4 (R/W)	Q0DDMACH	Queue 0 Enabled for DA-based DMA Channel Selection. When set, the `EMAC_RXQ_DMA_MAP0.Q0DDMACH` bit indicates that the packets received in queue 0 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the `EMAC_RXQ_DMA_MAP0.Q0DDMACH` bit indicates that the packets received in queue 0 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP0.Q0MDMACH field.	
		0	Disable
		1	Enable

Table 30-279: EMAC_RXQ_DMA_MAP0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	Q0MDMACH	Queue 0 Mapped to DMA Channel. The `EMAC_RXQ_DMA_MAP0.Q0MDMACH` field controls the routing of the packet received in queue 0 to the DMA channel. The `EMAC_RXQ_DMA_MAP0.Q0MDMACH` field is valid when the EMAC_RXQ_DMA_MAP0.Q0DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7

Receive Queue to DMA Channel Mapping Register 1

The EMAC_RXQ_DMA_MAP1 register is reserved in EQOS-CORE and EQOS-MTL configurations.

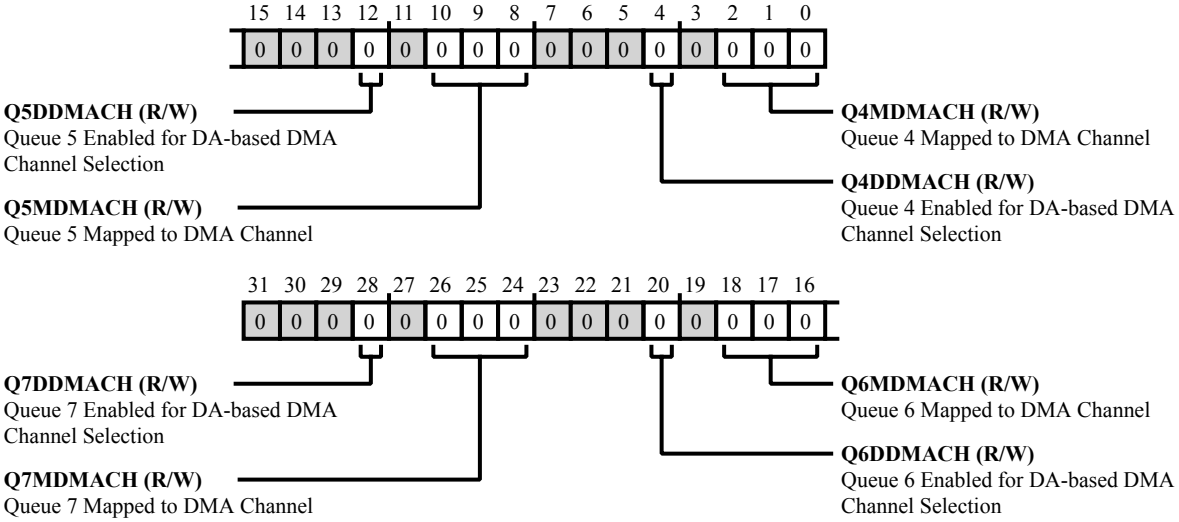


Figure 30-225: EMAC_RXQ_DMA_MAP1 Register Diagram

Table 30-280: EMAC_RXQ_DMA_MAP1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	Q7DDMACH	Queue 7 Enabled for DA-based DMA Channel Selection. When set (=1), the EMAC_RXQ_DMA_MAP1.Q7DDMACH bit indicates that the packets received in queue 7 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (= 0), the EMAC_RXQ_DMA_MAP1.Q7DDMACH bit indicates that the packets received in queue 7 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP1.Q7MDMACH field.	
		0	Disable
		1	Enable
26:24 (R/W)	Q7MDMACH	Queue 7 Mapped to DMA Channel. The EMAC_RXQ_DMA_MAP1.Q7MDMACH field controls the routing of the packet received in queue 7 to the DMA channel. The EMAC_RXQ_DMA_MAP1.Q7MDMACH field is valid when the EMAC_RXQ_DMA_MAP1.Q7DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0

Table 30-280: EMAC_RXQ_DMA_MAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7
20 (R/W)	Q6DDMACH	Queue 6 Enabled for DA-based DMA Channel Selection. When set (=1), the `EMAC_RXQ_DMA_MAP1.Q6DDMACH` bit indicates that the packets received in queue 6 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (=0), the `EMAC_RXQ_DMA_MAP1.Q6DDMACH` bit indicates that the packets received in queue 6 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP1.Q6MDMACH field.	
		0	Disable
		1	Enable
18:16 (R/W)	Q6MDMACH	Queue 6 Mapped to DMA Channel. The `EMAC_RXQ_DMA_MAP1.Q6MDMACH` field controls the routing of the packet received in queue 6 to the DMA channel. The `EMAC_RXQ_DMA_MAP1.Q6MDMACH` field is valid when the EMAC_RXQ_DMA_MAP1.Q6DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7

Table 30-280: EMAC_RXQ_DMA_MAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	Q5DDMACH	Queue 5 Enabled for DA-based DMA Channel Selection. When set (=1), the `EMAC_RXQ_DMA_MAP1.Q5DDMACH` bit indicates that the packets received in queue 5 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (= 0), the `EMAC_RXQ_DMA_MAP1.Q5DDMACH` bit indicates that the packets received in queue 5 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP1.Q5MDMACH field.	
		0	Disable
		1	Enable
10:8 (R/W)	Q5MDMACH	Queue 5 Mapped to DMA Channel. The `EMAC_RXQ_DMA_MAP1.Q5MDMACH` field controls the routing of the packets received in queue 5 to the DMA channel The `EMAC_RXQ_DMA_MAP1.Q5MDMACH` field is valid when the EMAC_RXQ_DMA_MAP1.Q5DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7
4 (R/W)	Q4DDMACH	Queue 4 Enabled for DA-based DMA Channel Selection. When set (=1), the `EMAC_RXQ_DMA_MAP1.Q4DDMACH` bit indicates that the packets received in queue 4 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When cleared (= 0), the `EMAC_RXQ_DMA_MAP1.Q4DDMACH` bit indicates that the packets received in queue 4 are routed to the DMA channel programmed in the EMAC_RXQ_DMA_MAP1.Q4MDMACH field.	
		0	Disable
		1	Enable

Table 30-280: EMAC_RXQ_DMA_MAP1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	Q4MDMACH	Queue 4 Mapped to DMA Channel. The `EMAC_RXQ_DMA_MAP1.Q4MDMACH` field controls the routing of the packet received in queue 4 to the DMA channel. The `EMAC_RXQ_DMA_MAP1.Q4MDMACH` field is valid when the EMAC_RXQ_DMA_MAP1.Q4DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values might be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved.	
		0	DMA Channel 0
		1	DMA Channel 1
		2	DMA Channel 2
		3	DMA Channel 3
		4	DMA Channel 4
		5	DMA Channel 5
		6	DMA Channel 6
		7	DMA Channel 7

MTL Time Based Scheduling Control Register

The EMAC_TBS_CTRL register controls the operation of Time-Based Scheduling (TBS).

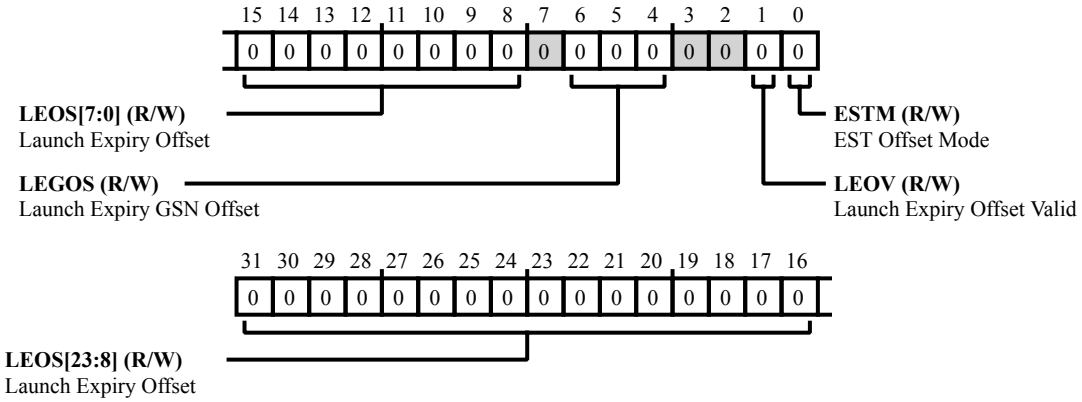


Figure 30-226: EMAC_TBS_CTRL Register Diagram

Table 30-281: EMAC_TBS_CTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:8 (R/W)	LEOS	Launch Expiry Offset. The field contains the value in units of 256 nanoseconds that must be added to the launch time to compute the launch expiry time. The value is valid only when EMAC_TBS_CTRL.LEOV = 1. The max value, 999,999,999 ns, should be smaller than the CTR-1 value when ESTM mode is enabled, because this value is a modulo CTR value.	
6:4 (R/W)	LEGOS	Launch Expiry GSN Offset. The EMAC_TBS_CTRL.LEGOS field indicates the number GSN slots that must be added to the launch GSN to compute the launch expiry time. The value is valid only when EMAC_TBS_CTRL.LEOV = 1.	
1 (R/W)	LEOV	Launch Expiry Offset Valid. When set (=1), the EMAC_TBS_CTRL.LEOV bit indicates that the EMAC_TBS_CTRL.LEOS field is valid. When not set, it indicates that the launch expiry offset is invalid and the MTL must not check for launch expiry time.	
		0	LEOS Field is Invalid
		1	LEOS Field is Valid

Table 30-281: EMAC_TBS_CTRL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	ESTM	EST Offset Mode. When the `EMAC_TBS_CTRL.ESTM` bit is set, the launch time value used in time based scheduling is interpreted as an EST offset value and is added to the Base Time Register (BTR) of the current list. When cleared (=0), the launch time value is used as an absolute value that should be compared with the system time [39:8].	
		0	Disabled
		1	Enabled

Transmit Queue 0 Debug Register

The EMAC_TQ0_DBG register gives the debug status of various blocks related to the transmit queue.

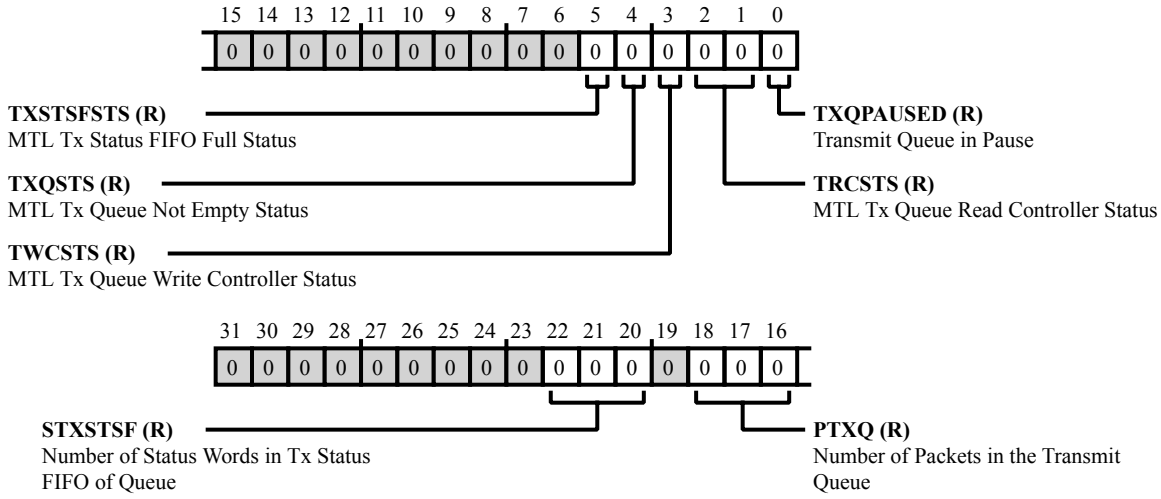


Figure 30-227: EMAC_TQ0_DBG Register Diagram

Table 30-282: EMAC_TQ0_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22:20 (R/NW)	STXSTSF	Number of Status Words in Tx Status FIFO of Queue. The EMAC_TQ0_DBG.STXSTSF field indicates the current number of status in the Tx status FIFO of this queue. When EMAC_MTL_OPMODE.DTXSTS = 1, this field does not reflect the number of status words in Tx status FIFO.	
18:16 (R/NW)	PTXQ	Number of Packets in the Transmit Queue. The EMAC_TQ0_DBG.PTXQ field indicates the current number of packets in the Tx queue. When EMAC_MTL_OPMODE.DTXSTS = 1, this field does not reflect the number of packets in the transmit queue.	
5 (R/NW)	TXSTSFSTS	MTL Tx Status FIFO Full Status. When set (=1), the EMAC_TQ0_DBG.TXSTSFSTS bit indicates that the MTL Tx status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.	
		0	MTL Tx Status FIFO Full Status is Not Detected
		1	MTL Tx Status FIFO Full Status is Detected

Table 30-282: EMAC_TQ0_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	TXQSTS	MTL Tx Queue Not Empty Status. When the EMAC_TQ0_DBG.TXQSTS bit is set (=1), it indicates that the MTL Tx queue is not empty and some data is left for transmission.	
		0	MTL Tx Queue Not Empty Status is Not Detected
		1	MTL Tx Queue Not Empty Status is Detected
3 (R/NW)	TWCSTS	MTL Tx Queue Write Controller Status. When set (=1), the EMAC_TQ0_DBG.TWCSTS bit indicates that the MTL Tx queue write controller is active, and it is transferring the data to the Tx queue.	
		0	MTL Tx Queue Write Controller Status is Not Detected
		1	MTL Tx Queue Write Controller Status is Detected
2:1 (R/NW)	TRCSTS	MTL Tx Queue Read Controller Status. The EMAC_TQ0_DBG.TRCSTS field indicates the state of the Tx queue read controller:	
		0	Idle State
		1	Read State (Transferring Data to the MAC Transmitter)
		2	Waiting for Pending Tx Status from the MAC Transmitter
		3	Flushing the Tx Queue Because of the Packet Abort Request from the MAC
0 (R/NW)	TXQPAUSED	Transmit Queue in Pause. When the EMAC_TQ0_DBG.TXQPAUSED bit is set (=1) and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the full-duplex only mode) because of the following: Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled. Reception of 802.3x Pause packet when PFC is disabled.	
		0	Transmit Queue in Pause Status is Not Detected
		1	Transmit Queue in Pause Status is Detected

Transmit Queue 0 ETS Status Register

The EMAC_TQ0_ETS_STAT register provides the average traffic transmitted in queue 0.

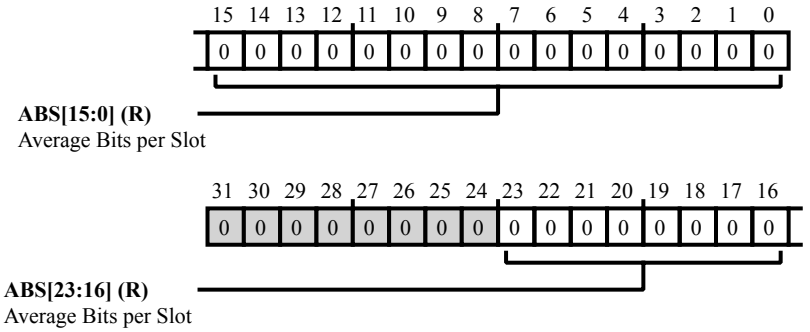


Figure 30-228: EMAC_TQ0_ETS_STAT Register Diagram

Table 30-283: EMAC_TQ0_ETS_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:0 (R/NW)	ABS	Average Bits per Slot. The EMAC_TQ0_ETS_STAT.ABS field contains the average transmitted bits per slot. When the DCB operation is enabled for queue 0, this field is computed over every 10 million bit times slot (4 ms in 2500 Mbps; 10 ms in 1000 Mbps; 100 ms in 100 Mbps). The maximum value is 0x989680.

Transmit Queue 0 Operation Mode Register

The EMAC_TQ0_OPMODE register establishes the transmit queue operating modes and commands.

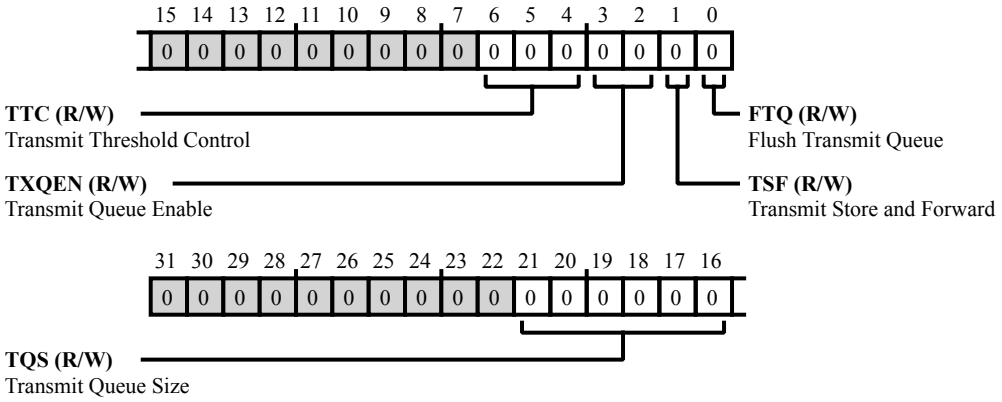


Figure 30-229: EMAC_TQ0_OPMODE Register Diagram

Table 30-284: EMAC_TQ0_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
21:16 (R/W)	TQS	Transmit Queue Size. The EMAC_TQ0_OPMODE.TQS field indicates the size of the allocated transmit queues in blocks of 256 bytes. The field is read-write only if the number of Tx queues more than one, the reset value is 0x0 and indicates size of 256 bytes. For example, a value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. So, program TQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the queue = (TQS+1)*256 bytes. When the number of Tx queues is one, the field is read-only and the configured TX FIFO size in blocks of 256 bytes is reflected in the reset value.	
6:4 (R/W)	TTC	Transmit Threshold Control. These bits control the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset.	
		0	32
		1	64
		2	96
		3	128
		4	192
		5	256
		6	384

Table 30-284: EMAC_TQ0_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		7	512
3:2 (R/W)	TXQEN	Transmit Queue Enable. The EMAC_TQ0_OPMODE.TXQEN field is used to enable/disable the transmit queue 0. The EMAC_TQ0_OPMODE.TXQEN field is read Only in Single queue configurations and read write in multiple queue configurations. Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.	
		0	Not Enabled
		1	Enable in AV Mode (Reserved in Non-AV)
		2	Enabled
		3	Reserved
1 (R/W)	TSF	Transmit Store and Forward. When the EMAC_TQ0_OPMODE.TSF bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When the EMAC_TQ0_OPMODE.TSF bit is set, the EMAC_TQ0_OPMODE.TTC field is ignored. The EMAC_TQ0_OPMODE.TSF bit should be changed only when the transmission is stopped.	
		0	Transmit Store and Forward is Disabled
		1	Transmit Store and Forward is Enabled
0 (R/W)	FTQ	Flush Transmit Queue. When the EMAC_TQ0_OPMODE.FTQ bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. The EMAC_TQ0_OPMODE.FTQ bit is internally reset when the flushing operation is complete. Until EMAC_TQ0_OPMODE.FTQ = 0, do not write to the EMAC_TQ[n]_OPMODE register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Flush Transmit Queue is Disabled
		1	Flush Transmit Queue is Enabled

Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register

The EMAC_TQ0_QTMWGT register contains the quantum value for Deficit Weighted Round Robin (DWRR), weights for the Weighted Round Robin (WRR), and Weighted Fair Queuing (WFQ) for queue 0.

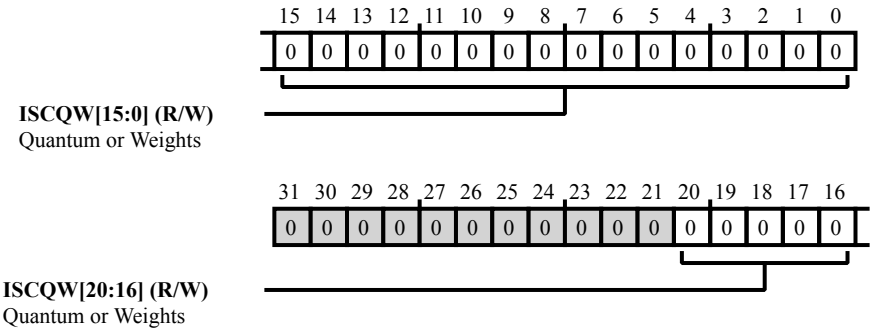


Figure 30-230: EMAC_TQ0_QTMWGT Register Diagram

Table 30-285: EMAC_TQ0_QTMWGT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:0 (R/W)	ISCQW	Quantum or Weights. When the DCB operation is enabled with DWRR algorithm for queue 0 traffic, the EMAC_TQ0_QTMWGT.ISCQW field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. When DCB operation is enabled with WFQ algorithm for queue 0 traffic, the EMAC_TQ0_QTMWGT.ISCQW field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits[20:14] must be written to zero. The higher the programmed weights lesser the bandwidth allocated for the particular transmit queue. This is because the weights are used to compute the packet finish time (weights * packet_size). Lesser the finish time, higher the probability of the packet getting scheduled first and using more bandwidth. When DCB operation or generic queuing operation is enabled with WRR algorithm for queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [20:7] must be written to zero.

Transmit Queue 0 Underflow Packet Counter Register

The EMAC_TQ0_URFL register contains the counter for packets aborted because of transmit queue underflow and packets missed because of receive queue packet flush.

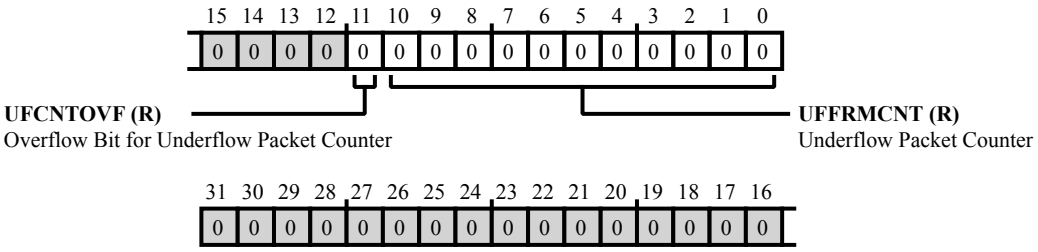


Figure 30-231: EMAC_TQ0_URFL Register Diagram

Table 30-286: EMAC_TQ0_URFL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	UFCNTOVF	Overflow Bit for Underflow Packet Counter. The EMAC_TQ0_URFL.UFCNTOVF bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and the EMAC_TQ0_URFL.UFCNTOVF bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Overflow Not Detected for Underflow Packet Counter
		1	Overflow Detected for Underflow Packet Counter
10:0 (R/NW)	UFFRMCNT	Underflow Packet Counter. The EMAC_TQ0_URFL.UFFRMCNT field indicates the number of packets aborted by the controller because of Tx queue underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. Access restriction applies. Clears on read. Self-set to 1 on internal event.	

Transmit Queue n Debug Register

The EMAC_TQ[n]_DBG register gives the debug status of various blocks related to the transmit queue.

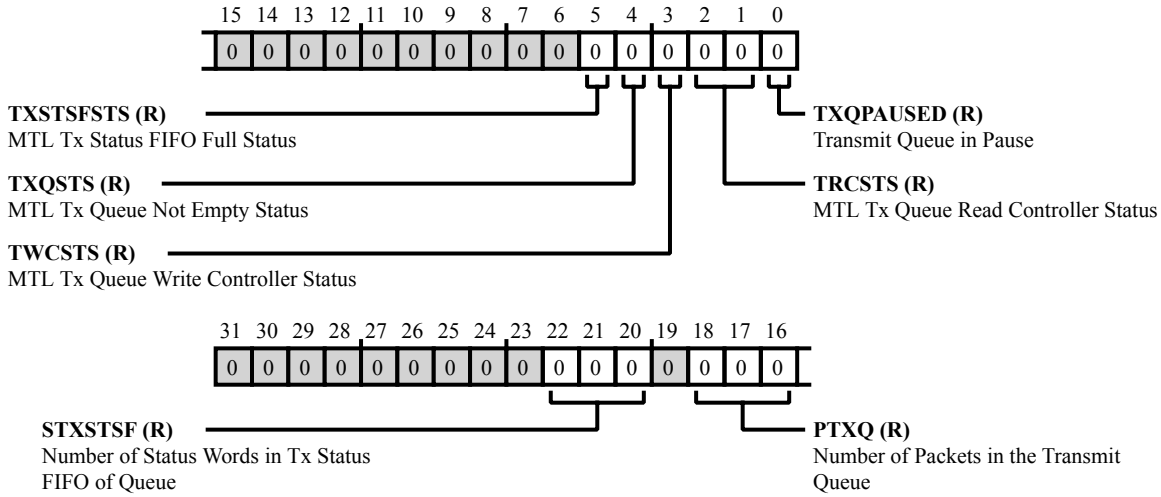


Figure 30-232: EMAC_TQ[n]_DBG Register Diagram

Table 30-287: EMAC_TQ[n]_DBG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22:20 (R/NW)	STXSTSF	Number of Status Words in Tx Status FIFO of Queue. The EMAC_TQ[n]_DBG.STXSTSF field indicates the current number of status in the Tx Status FIFO of this queue. When EMAC_MTL_OPMODE.DTXSTS = 1, this field does not reflect the number of status words in Tx Status FIFO.	
18:16 (R/NW)	PTXQ	Number of Packets in the Transmit Queue. The EMAC_TQ[n]_DBG.PTXQ field indicates the current number of packets in the Tx queue. When EMAC_MTL_OPMODE.DTXSTS = 1, the field does not reflect the number of packets in the transmit queue.	
5 (R/NW)	TXSTSFSTS	MTL Tx Status FIFO Full Status. When set (=1), the EMAC_TQ[n]_DBG.TXSTSFSTS bit indicates that the MTL Tx status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.	
		0	Status Not Detected
		1	Status Detected

Table 30-287: EMAC_TQ[n]_DBG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	TXQSTS	MTL Tx Queue Not Empty Status. When the `EMAC_TQ[n]_DBG.TXQSTS` bit is set (=1), it indicates that the MTL Tx queue is not empty and some data is left for transmission.	
		0	Status Not Detected
		1	Status Detected
3 (R/NW)	TWCSTS	MTL Tx Queue Write Controller Status. When set (=1), the `EMAC_TQ[n]_DBG.TWCSTS` bit indicates that the MTL Tx queue write controller is active (status detected), and it is transferring the data to the Tx queue.	
		0	Status Not Detected
		1	Status Detected
2:1 (R/NW)	TRCSTS	MTL Tx Queue Read Controller Status. The `EMAC_TQ[n]_DBG.TRCSTS` field indicates the state of the Tx queue read controller.	
		0	Idle State
		1	Read State (Transferring data to the MAC transmitter)
		2	Wait for pending Tx status from the MAC transmitter
		3	Flush the Tx queue because of the packet abort request from the MAC
0 (R/NW)	TXQPAUSED	Transmit Queue in Pause. When the `EMAC_TQ[n]_DBG.TXQPAUSED` bit is set (=1) and the Rx flow control is enabled, it indicates that the Tx queue is in the pause condition (in the full-duplex only mode) because of the following: - Reception of the PFC packet for the priorities assigned to the Tx queue, when PFC is enabled. - Reception of the 802.3x pause packet, when PFC is disabled.	
		0	Transmit Queue in Pause Status is Not Detected
		1	Transmit Queue in Pause Status is Detected

Transmit Queue n ETS Control Register

The EMAC_TQ[n]_ETS_CTL register controls the enhanced transmission selection operation.

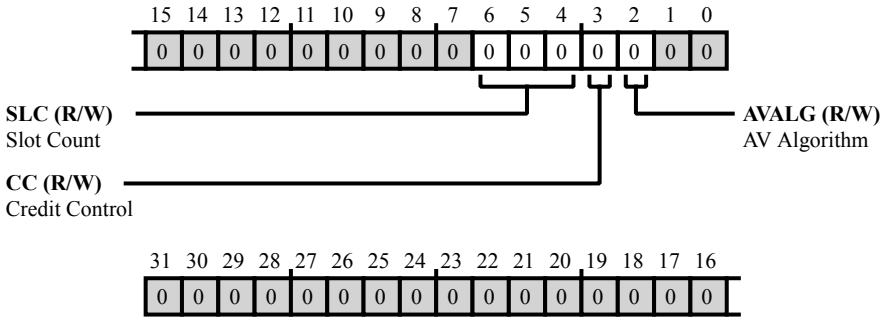


Figure 30-233: EMAC_TQ[n]_ETS_CTL Register Diagram

Table 30-288: EMAC_TQ[n]_ETS_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6:4 (R/W)	SLC	Slot Count. If the credit-based shaper algorithm is enabled, the software can program the EMAC_TQ[n]_ETS_CTL.SLC field (of a duration programmed in EMAC_DMA[n]_SFN_CTLSTAT register). The average transmitted bits per slot must be computed for the queue; it is provided in the EMAC_TQ[n]_ETS_STAT register.	
		0	1 Slot
		1	2 Slots
		2	4 Slots
		3	8 Slots
		4	16 Slots
		5	Reserved
3 (R/W)	CC	Credit Control. When the EMAC_TQ[n]_ETS_CTL.CC bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in channel 1. The credit accumulates even when there is no packet waiting in channel 1 and another channel is transmitting. When EMAC_TQ[n]_ETS_CTL.CC = 0, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in channel 1. When there is no packet waiting in channel 1 and other channel is transmitting, no credit is accumulated.	
		0	Credit Control is Disabled
		1	Credit Control is Enabled

Table 30-288: EMAC_TQ[n]_ETS_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	AVALG	AV Algorithm. When queue 1 is programmed for AV, the `EMAC_TQ[n]_ETS_CTL.AVALG` field configures the scheduling algorithm for the queue. The `EMAC_TQ[n]_ETS_CTL.AVALG` bit when set, a credit-based shaper algorithm (CBS) is selected for queue 1 traffic. When cleared (=0), strict priority is selected.	
		0	CBS Algorithm is Disabled
		1	CBS Algorithm is Enabled

Transmit Queue n ETS Status Register

The EMAC_TQ[n]_ETS_STAT register provides the average traffic transmitted in queue 1.

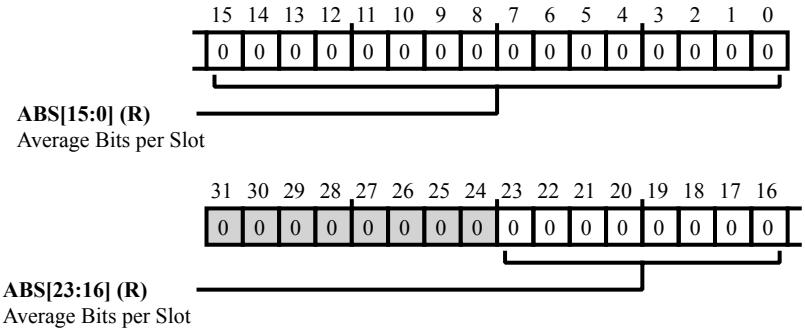


Figure 30-234: EMAC_TQ[n]_ETS_STAT Register Diagram

Table 30-289: EMAC_TQ[n]_ETS_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:0 (R/NW)	ABS	Average Bits per Slot. The EMAC_TQ[n]_ETS_STAT.ABS field contains the average transmitted bits per slot. If AV operation is enabled, the EMAC_TQ[n]_ETS_STAT.ABS field is computed over the number of slots, programmed in EMAC_TQ[n]_ETS_CTL.SLC. The maximum value of this field is 0x6_4000 in 100 Mbps, 0x3E_8000 in 1000 Mbps and 9C_4000 in 2500 Mbps mode respectively. When the DCB operation is enabled for queue, the EMAC_TQ[n]_ETS_STAT.ABS field is computed over every 10 million bit times slot (4 ms in 2500 Mbps; 10 ms in 1000 Mbps; 100 ms in 100 Mbps). The maximum value is 0x989680.

Transmit Queue n CBS HiCredit Register

The EMAC_TQ[n]_HICRDT register contains the HiCredit value required for the credit-based shaper algorithm for the queue.

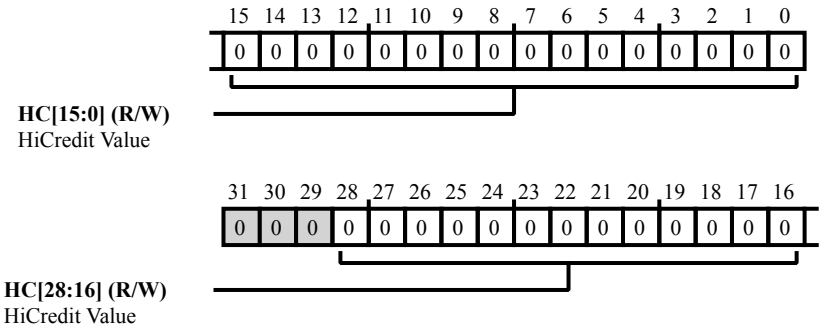


Figure 30-235: EMAC_TQ[n]_HICRDT Register Diagram

Table 30-290: EMAC_TQ[n]_HICRDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	HC	HiCredit Value. When the AV feature is enabled, the EMAC_TQ[n]_HICRDT.HC field contains the hiCredit value required for the credit-based shaper algorithm. This is the maximum value that can be accumulated in the credit parameter. It is specified in bits scaled by 1,024. The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16,384 bytes or 131,072 bits). The value to be specified is 131,072 * 1,024 = 134,217,728 or 0x0800_0000.

Transmit Queue n CBS LoCredit Register

The EMAC_TQ[n]_LOCRDT register contains the loCredit value required for the credit-based shaper algorithm for the queue.

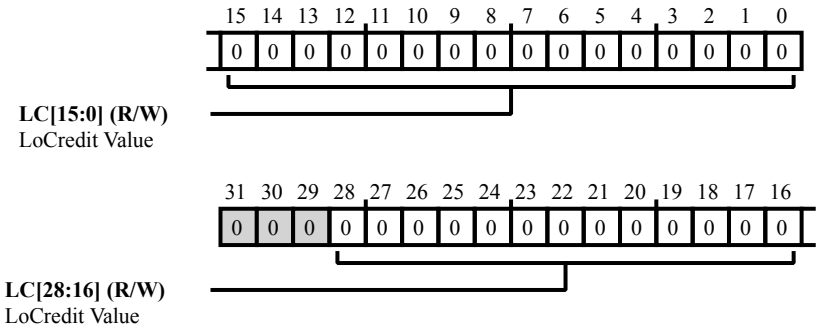


Figure 30-236: EMAC_TQ[n]_LOCRDT Register Diagram

Table 30-291: EMAC_TQ[n]_LOCRDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
28:0 (R/W)	LC	LoCredit Value. When the AV operation is enabled, the EMAC_TQ[n]_LOCRDT.LC field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. It is specified in bits scaled by 1,024. The maximum value to be programmed corresponds to twice the maxFrame-Size transmitted from this queue. If the maxFrameSize is 8192 bytes, then (8192*2) * 8 * 1024 = 134,217,728 or 0x0800_0000. Because it is a negative value, the programmed value is the 2's complement of the value, that is, 0x1800_0000.

Transmit Queue n Operation Mode Register

The EMAC_TQ[n]_OPMODE register establishes the transmit queue operating modes and commands.

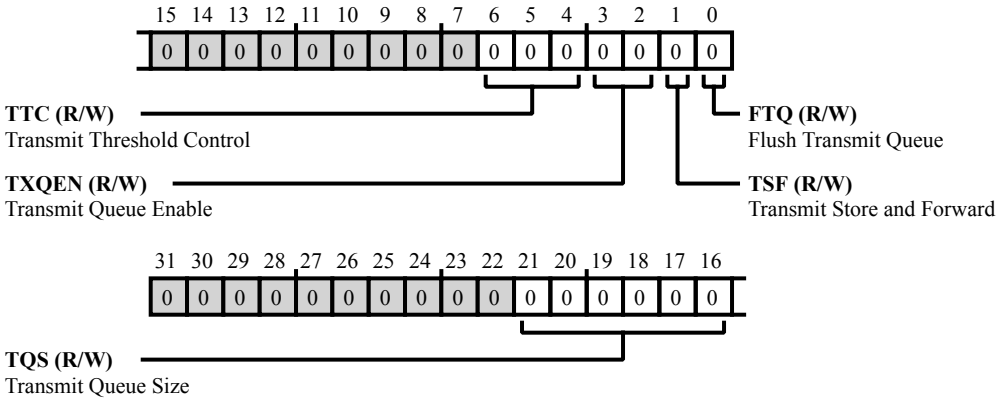


Figure 30-237: EMAC_TQ[n]_OPMODE Register Diagram

Table 30-292: EMAC_TQ[n]_OPMODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
21:16 (R/W)	TQS	Transmit Queue Size. The EMAC_TQ[n]_OPMODE.TQS field indicates the size of the allocated transmit queues in blocks of 256 bytes. The EMAC_TQ[n]_OPMODE.TQS field is read-write only if the number of Tx queues more than one; the reset value is 0x0 and indicates size of 256 bytes. For example, a value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. Program TQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the queue = (TQS+1)*256 bytes. When the number of Tx queues is one, the EMAC_TQ[n]_OPMODE.TQS field is read-only and the configured TX FIFO size in blocks of 256 bytes is reflected in the reset value.	
6:4 (R/W)	TTC	Transmit Threshold Control. The EMAC_TQ[n]_OPMODE.TTC field controls the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. The EMAC_TQ[n]_OPMODE.TTC field is used only when EMAC_TQ[n]_OPMODE.TSF = 0.	
		0	32
		1	64
		2	96
		3	128
		4	192

Table 30-292: EMAC_TQ[n]_OPMODE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		5	256
		6	384
		7	512
3:2 (R/W)	TXQEN	Transmit Queue Enable. The `EMAC_TQ[n]_OPMODE.TXQEN` field is used to enable/disable the transmit queue 0. Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.	
		0	Not Enabled
		1	Enable in AV Mode (Reserved in Non-AV)
		2	Enabled
		3	Reserved
1 (R/W)	TSF	Transmit Store and Forward. When the `EMAC_TQ[n]_OPMODE.TSF` bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When the `EMAC_TQ[n]_OPMODE.TSF` bit is set, the TTC values specified in bits[6:4] of this register are ignored. The `EMAC_TQ[n]_OPMODE.TSF` bit should be changed only when the transmission is stopped.	
		0	Transmit Store and Forward is Disabled
		1	Transmit Store and Forward is Enabled
0 (R/W)	FTQ	Flush Transmit Queue. When the `EMAC_TQ[n]_OPMODE.FTQ` bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. The `EMAC_TQ[n]_OPMODE.FTQ` bit is internally reset when the flushing operation is complete. Until EMAC_TQ[0]_OPMODE= 0, do not write to the EMAC_TQ[1]_OPMODE register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Write 1 to set. Self-cleared. Writing 0 has no effect.	
		0	Flush Transmit Queue is Disabled
		1	Flush Transmit Queue is Enabled

Transmit Queue n IdleSlopeCredit, Quantum or Weights Register

The EMAC_TQ[n]_QTMWGT register provides the average traffic transmitted in queue 1.

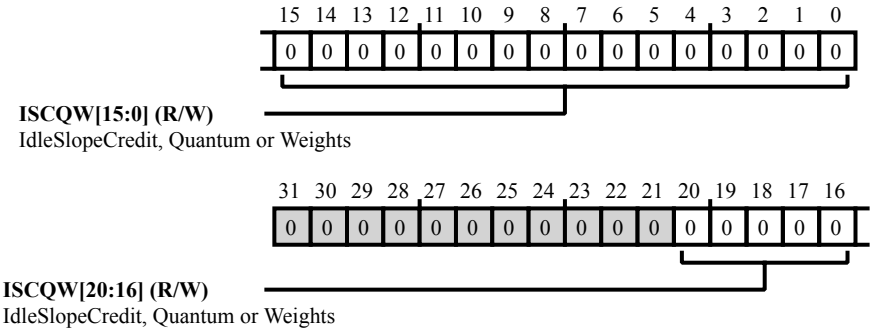


Figure 30-238: EMAC_TQ[n]_QTMWGT Register Diagram

Table 30-293: EMAC_TQ[n]_QTMWGT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
20:0 (R/W)	ISCQW	IdleSlopeCredit, Quantum or Weights. When AV feature is enabled, the `EMAC_TQ[n]_QTMWGT.ISCQW` field contains the idleSlopeCredit value required for the credit-based shaper algorithm for queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1000 Mbps; 3.2 ns for 2500 Mbps) when the credit is increasing. The software should program the `EMAC_TQ[n]_QTMWGT.ISCQW` field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000/2500 Mbps mode and 0x1000 in 100 Mbps mode. Bits[20:14] must be written to zero. Quantum When the DCB operation is enabled with DWRR algorithm for queue 1 traffic, the `EMAC_TQ[n]_QTMWGT.ISCQW` field field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. Weights When DCB operation is enabled with WFQ algorithm for queue 1 traffic, the `EMAC_TQ[n]_QTMWGT.ISCQW` field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits[20:14] must be written to zero. When DCB operation or generic queuing operation is enabled with WRR algorithm for queue 1 traffic, the `EMAC_TQ[n]_QTMWGT.ISCQW` field contains the weight for this queue. The maximum value is 0x64. Bits [20:7] must be written to zero. Note 1: In multiple queue configuration this field in respective per queue register must be programmed to some non-zero value when multiple queues are enabled or single queue other than Q0 is enabled. The `EMAC_TQ[n]_QTMWGT.ISCQW` field need not be programmed when only Q0 is enabled. In general, when WRR algorithm is selected a non-zero value must be programmed on both receive and transmit. In receive, the register is EMAC_MTL_OPMODE register. Note 2: For WFQ algorithm, higher the programmed weights lesser the bandwidth allocated for that transmit queue. The finish time is not a function of particular packet alone but it is as per the formula: (previous_finish_time of particular transmit queue + (weights*packet_size)) Note 3: The weights programmed do not correspond to the number of packets but the fraction of bandwidth or time allocated for particular queue with respect to total BW or time.

Transmit Queue n SendSlopeCredit Register

The EMAC_TQ[n]_SSCRDT register contains the sendSlope credit value required for the credit-based shaper algorithm for the queue.

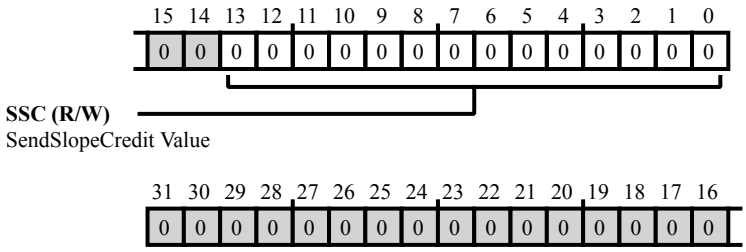


Figure 30-239: EMAC_TQ[n]_SSCRDT Register Diagram

Table 30-294: EMAC_TQ[n]_SSCRDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:0 (R/W)	SSC	SendSlopeCredit Value. When AV operation is enabled, the EMAC_TQ[n]_SSCRDT.SSC field contains the sendSlopeCredit value required for credit-based shaper algorithm for queue 1. This is the rate of change of credit in bits per cycle (40 ns, 8 ns and 3.2 ns for 100 Mbps, 1000 Mbps and 2500 Mbps respectively) when the credit is decreasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000/2500 Mbps mode and 0x1000 in 100 Mbps mode. The EMAC_TQ[n]_SSCRDT.SSC field should be programmed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when channel 1 is selected for transmission.

Transmit Queue n Underflow Packet Counter Register

The EMAC_TQ[n]_URFL register contains the counter for packets aborted because of transmit queue underflow and packets missed because of receive queue packet flush.

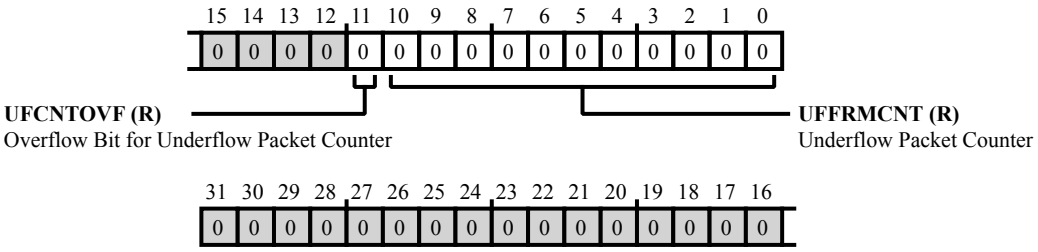


Figure 30-240: EMAC_TQ[n]_URFL Register Diagram

Table 30-295: EMAC_TQ[n]_URFL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/NW)	UFCNTOVF	Overflow Bit for Underflow Packet Counter. The EMAC_TQ[n]_URFL.UFCNTOVF bit is set every time the Tx queue underflow packet counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and the EMAC_TQ[n]_URFL.UFCNTOVF bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event.	
		0	Overflow Not Detected for Underflow Packet Counter
		1	Overflow Detected for Underflow Packet Counter
10:0 (R/NW)	UFFRMCNT	Underflow Packet Counter. The EMAC_TQ[n]_URFL.UFFRMCNT field indicates the number of packets aborted by the controller because of Tx queue underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. Access restriction applies. Clears on read. Self-set to 1 on internal event.	

Receive ICMP Error Octets Counter Register

The EMAC_RXICMP_ERR_OCNT register provides the number of bytes received by the EMAC module in a ICMP segment that had checksum errors. This counter does not count IP header bytes.

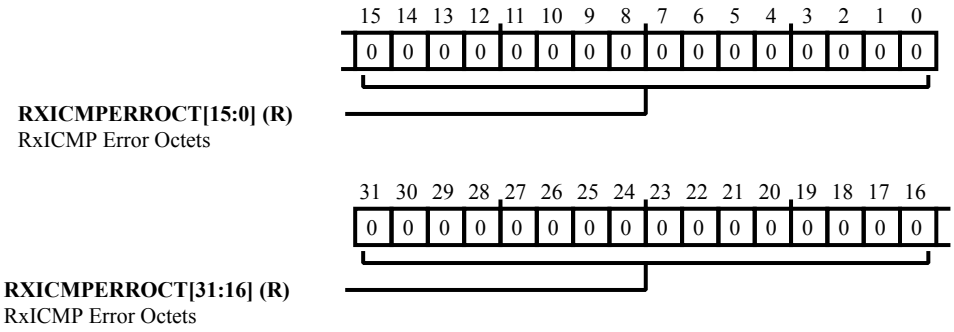


Figure 30-241: EMAC_RXICMP_ERR_OCNT Register Diagram

Table 30-296: EMAC_RXICMP_ERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXICMPERROCT	RxICMP Error Octets. The EMAC_RXICMP_ERR_OCNT.RXICMPERROCT field indicates the number of bytes received in a ICMP segment that had checksum errors. This counter does not count IP header bytes.

Receive ICMP Error Packets Counter Register

The EMAC_RXICMP_ERR_PCNT register provides the number of good IP datagrams received by the EMAC module whose ICMP payload has a checksum error.

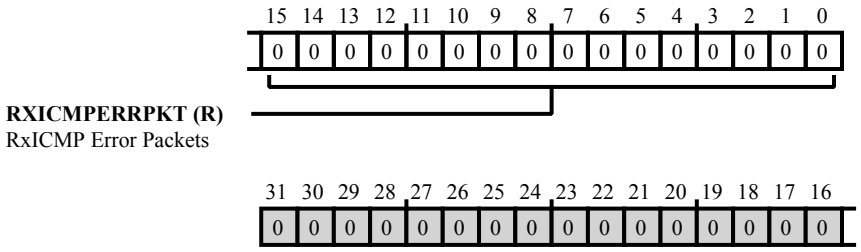


Figure 30-242: EMAC_RXICMP_ERR_PCNT Register Diagram

Table 30-297: EMAC_RXICMP_ERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXICMPERRPKT	RxICMP Error Packets. The EMAC_RXICMP_ERR_PCNT.RXICMPERRPKT field indicates the number of good IP datagrams received whose ICMP payload has a checksum error.

Receive ICMP Good Octets Counter Register

The EMAC_RXICMP_OCNT_G register provides the number of bytes received by the EMAC module in a good ICMP segment. This counter does not count IP header bytes.

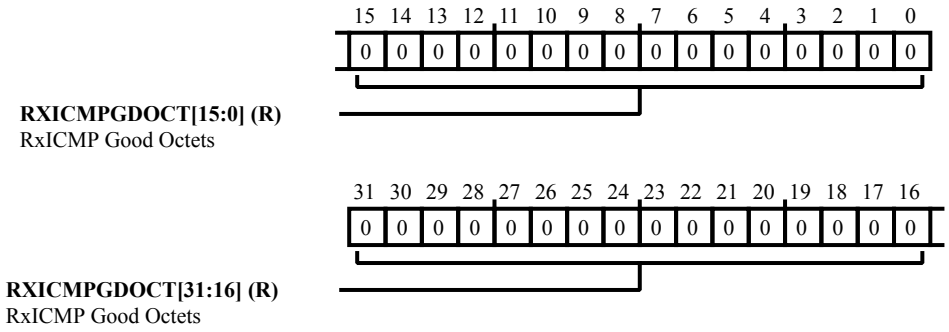


Figure 30-243: EMAC_RXICMP_OCNT_G Register Diagram

Table 30-298: EMAC_RXICMP_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXICMPGDOCT	RxICMP Good Octets. The EMAC_RXICMP_OCNT_G.RXICMPGDOCT field indicates the number of bytes received in a good ICMP segment. This counter does not count IP header bytes.

Receive ICMP Good Packets Counter Register

The EMAC_RXICMP_PCNT_G register provides the number of good IP datagrams received by the EMAC module with a good ICMP payload.

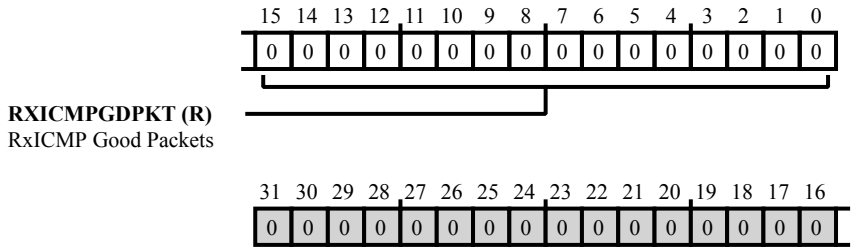


Figure 30-244: EMAC_RXICMP_PCNT_G Register Diagram

Table 30-299: EMAC_RXICMP_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXICMPGDPKT	RxICMP Good Packets. The EMAC_RXICMP_PCNT_G.RXICMPGDPKT field indicates the number of good IP datagrams received with a good ICMP payload.

Receive IPv4 Fragmented Octets Counter Register

The EMAC_RXIPV4_FRGMT_OCNT register provides the number of bytes received by the EMAC module in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

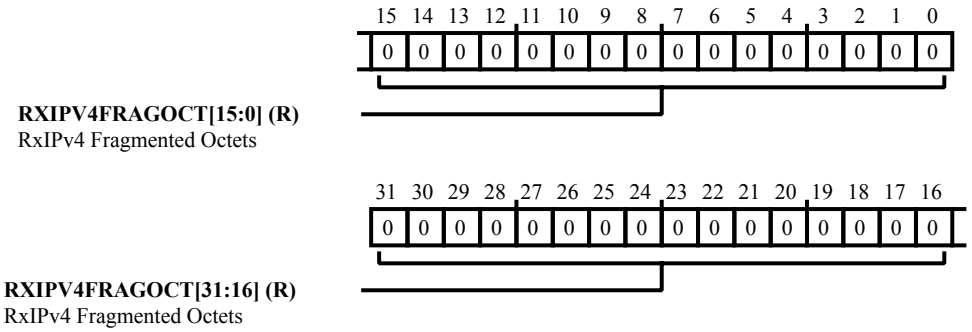


Figure 30-245: EMAC_RXIPV4_FRGMT_OCNT Register Diagram

Table 30-300: EMAC_RXIPV4_FRGMT_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4FRAGOCT	RxIPv4 Fragmented Octets. The EMAC_RXIPV4_FRGMT_OCNT.RXIPV4FRAGOCT field indicates the number of bytes received in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 Fragmented Packets Counter Register

The EMAC_RXIPV4_FRGMT_PCNT register provides the number of good IPv4 datagrams received by the EMAC module with fragmentation.

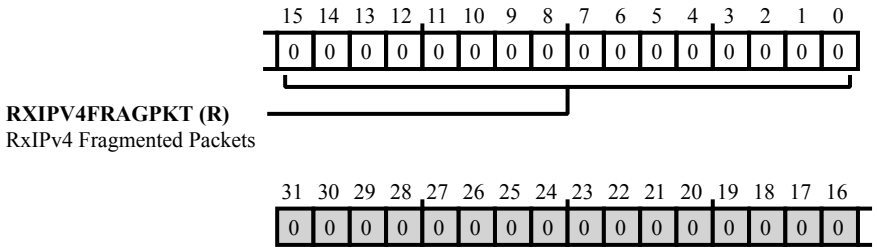


Figure 30-246: EMAC_RXIPV4_FRGMT_PCNT Register Diagram

Table 30-301: EMAC_RXIPV4_FRGMT_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4FRAGPKT	RxIPv4 Fragmented Packets. The EMAC_RXIPV4_FRGMT_PCNT.RXIPV4FRAGPKT field indicates the number of good IPv4 datagrams received with fragmentation.

Receive IPv4 Good Octets Counter Register

The EMAC_RXIPV4_OCNT_G register provides the number of bytes received by the EMAC module in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

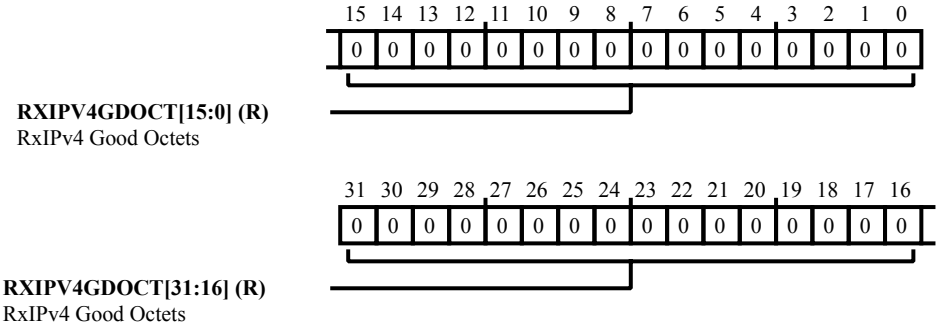


Figure 30-247: EMAC_RXIPV4_OCNT_G Register Diagram

Table 30-302: EMAC_RXIPV4_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4GDOCT	RxIPv4 Good Octets. The EMAC_RXIPV4_OCNT_G.RXIPV4GDOCT field indicates the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 Good Packets Counter Register

The EMAC_RXIPV4_PCNT_G register provides the number of good IPv4 datagrams received by the EMAC module with the TCP, UDP, or ICMP payload.

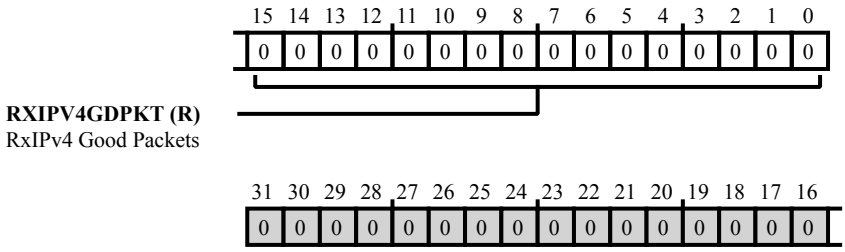


Figure 30-248: EMAC_RXIPV4_PCNT_G Register Diagram

Table 30-303: EMAC_RXIPV4_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4GDPKT	RxIPv4 Good Packets. The EMAC_RXIPV4_PCNT_G.RXIPV4GDPKT field indicates the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload.

Receive IPv4 Header Error Octets Counter Register

The EMAC_RXIPV4_HDRERR_OCNT register provides the number of bytes received by the EMAC module in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

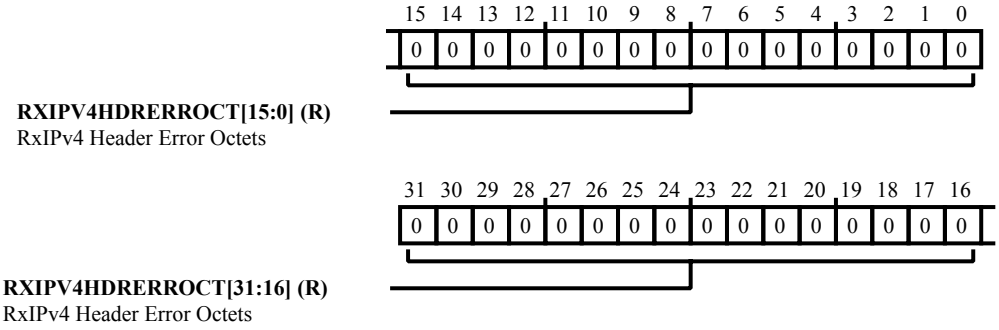


Figure 30-249: EMAC_RXIPV4_HDRERR_OCNT Register Diagram

Table 30-304: EMAC_RXIPV4_HDRERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4HDRERROCT	RxIPv4 Header Error Octets. The EMAC_RXIPV4_HDRERR_OCNT.RXIPV4HDRERROCT field indicates the number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 Header Error Packets Counter Register

The EMAC_RXIPV4_HDRERR_PCNT register provides the number of IPv4 datagrams received by the EMAC module with header (checksum, length, or version mismatch) errors.

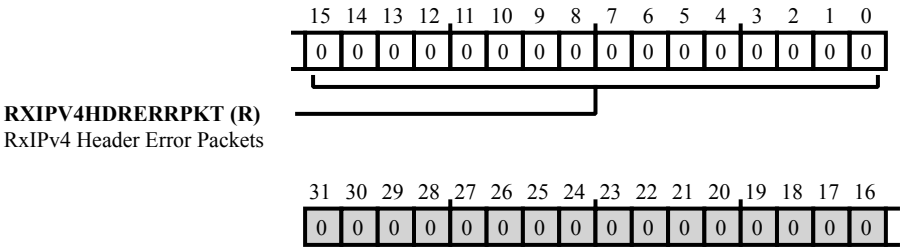


Figure 30-250: EMAC_RXIPV4_HDRERR_PCNT Register Diagram

Table 30-305: EMAC_RXIPV4_HDRERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4HDRERRPKT	RxIPv4 Header Error Packets. The EMAC_RXIPV4_HDRERR_PCNT.RXIPV4HDRERRPKT field indicates the number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors.

Receive IPv4 Payload Octets Counter Register

The EMAC_RXIPV4_PYLD_OCNT register provides the number of bytes received by the EMAC module in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

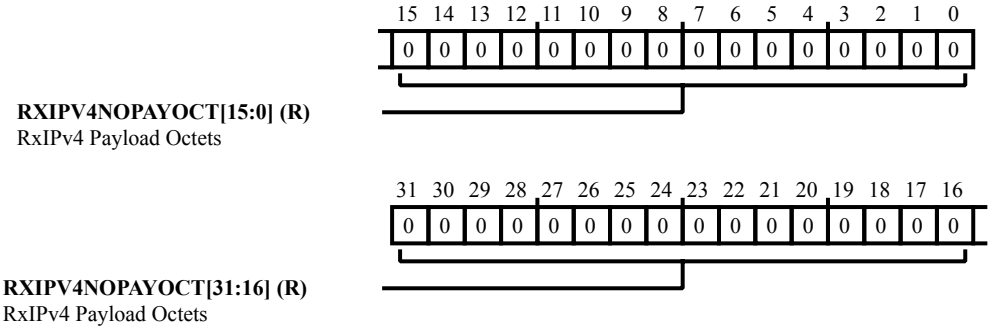


Figure 30-251: EMAC_RXIPV4_PYLD_OCNT Register Diagram

Table 30-306: EMAC_RXIPV4_PYLD_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4NOPAYOCT	RxIPv4 Payload Octets. The EMAC_RXIPV4_PYLD_OCNT.RXIPV4NOPAYOCT field indicates the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 Payload Packets Counter Register

The EMAC_RXIPV4_PYLD_PCNT register provides the number of IPv4 datagram packets received by EMAC module that did not have a TCP, UDP, or ICMP payload.

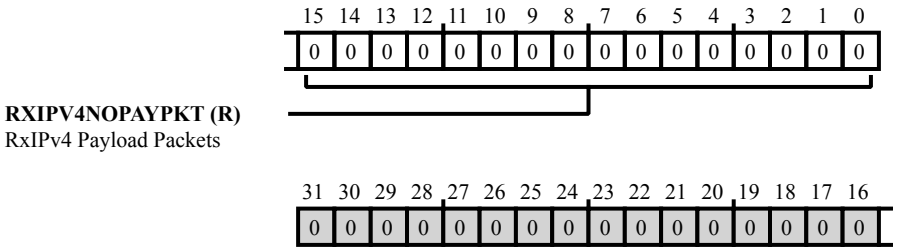


Figure 30-252: EMAC_RXIPV4_PYLD_PCNT Register Diagram

Table 30-307: EMAC_RXIPV4_PYLD_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4NOPAYPKT	RxIPv4 Payload Packets. The EMAC_RXIPV4_PYLD_PCNT.RXIPV4NOPAYPKT field indicates the number of IPv4 datagram packets received that did not have a TCP, UDP, or ICMP payload.

Receive IPv4 UDP Checksum Disable Octets Counter Register

The EMAC_RXIPV4_UDPCSD_OCNT register provides the number of bytes received by the EMAC module in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

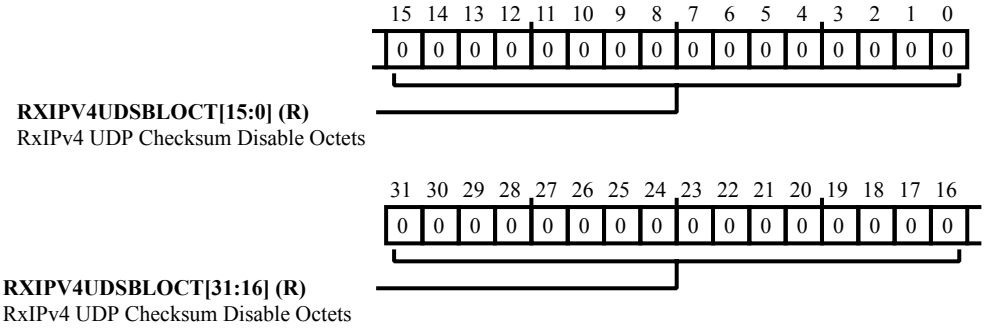


Figure 30-253: EMAC_RXIPV4_UDPCSD_OCNT Register Diagram

Table 30-308: EMAC_RXIPV4_UDPCSD_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV4UDSBLOCT	RxIPv4 UDP Checksum Disable Octets. The EMAC_RXIPV4_UDPCSD_OCNT.RXIPV4UDSBLOCT field indicates the number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv4 UDP Checksum Disable Packets Counter Register

The EMAC_RXIPV4_UDPCSD_PCNT register provides the number of good IPv4 datagrams received by the EMAC module that had a UDP payload with checksum disabled.

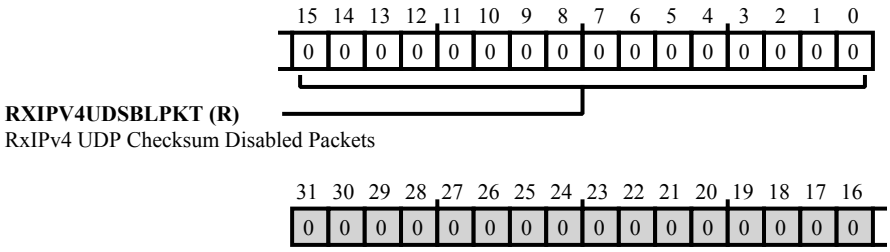


Figure 30-254: EMAC_RXIPV4_UDPCSD_PCNT Register Diagram

Table 30-309: EMAC_RXIPV4_UDPCSD_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV4UDSBLPKT	RxIPv4 UDP Checksum Disabled Packets. The EMAC_RXIPV4_UDPCSD_PCNT.RXIPV4UDSBLPKT field indicates the number of good IPv4 datagrams received that had a UDP payload with checksum disabled.

Receive IPv6 Good Octets Counter Register

The EMAC_RXIPV6_OCNT_G register provides the number of bytes received by the EMAC module in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

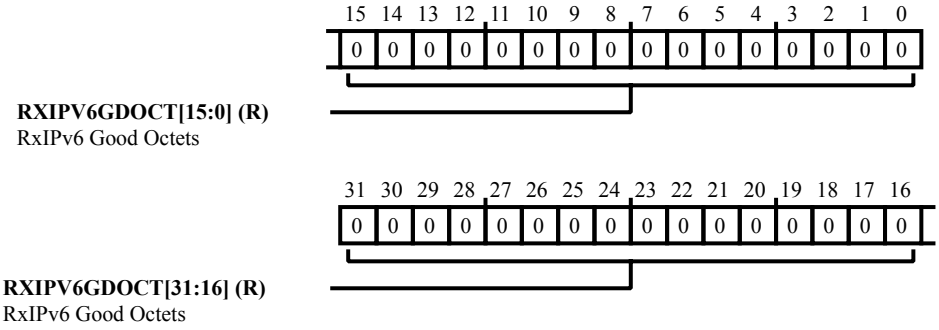


Figure 30-255: EMAC_RXIPV6_OCNT_G Register Diagram

Table 30-310: EMAC_RXIPV6_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV6GDOCT	RxIPv6 Good Octets. The EMAC_RXIPV6_OCNT_G.RXIPV6GDOCT field indicates the number of bytes received in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv6 Good Packets Counter Register

The EMAC_RXIPV6_PCNT_G register provides the number of good IPv6 datagrams received by the EMAC module with the TCP, UDP, or ICMP payload.

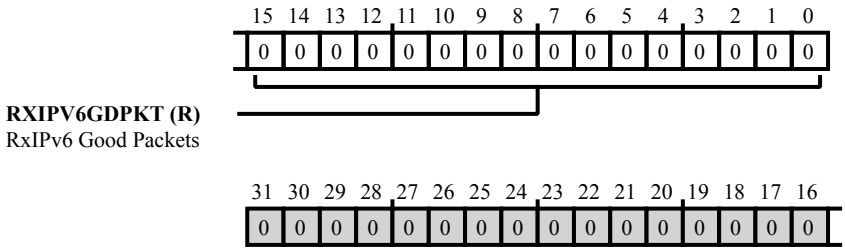


Figure 30-256: EMAC_RXIPV6_PCNT_G Register Diagram

Table 30-311: EMAC_RXIPV6_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV6GDPKT	RxIPv6 Good Packets. The EMAC_RXIPV6_PCNT_G.RXIPV6GDPKT field indicates the number of good IPv6 datagrams received with the TCP, UDP, or ICMP payload.

Receive IPv6 Header Error Octets Counter Register

The EMAC_RXIPV6_HDRERR_OCNT register provides the number of bytes received by the EMAC module in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

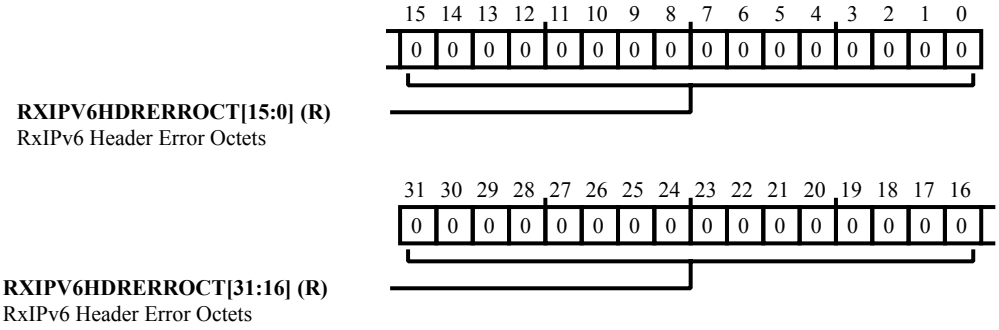


Figure 30-257: EMAC_RXIPV6_HDRERR_OCNT Register Diagram

Table 30-312: EMAC_RXIPV6_HDRERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV6HDRERROCT	RxIPv6 Header Error Octets. The EMAC_RXIPV6_HDRERR_OCNT.RXIPV6HDRERROCT field indicates the number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv6 Header Error Packets Counter Register

The EMAC_RXIPV6_HDRERR_PCNT register provides the number of IPv6 datagrams received by the EMAC module with header (length or version mismatch) errors.

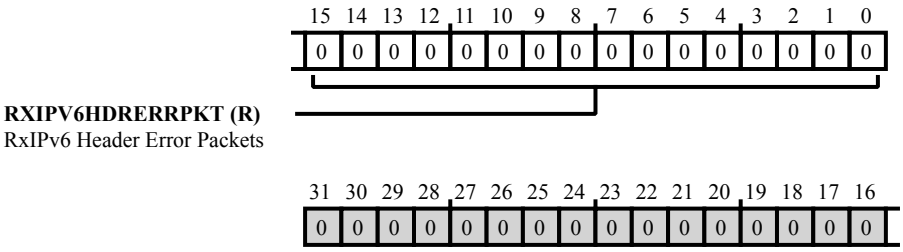


Figure 30-258: EMAC_RXIPV6_HDRERR_PCNT Register Diagram

Table 30-313: EMAC_RXIPV6_HDRERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV6HDRERRPKT	RxIPv6 Header Error Packets. The EMAC_RXIPV6_HDRERR_PCNT.RXIPV6HDRERRPKT field indicates the number of IPv6 datagrams received with header (length or version mismatch) errors.

Receive IPv6 Payload Octets Counter Register

The EMAC_RXIPV6_PYLD_OCNT register provides the number of bytes received by the EMAC module in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

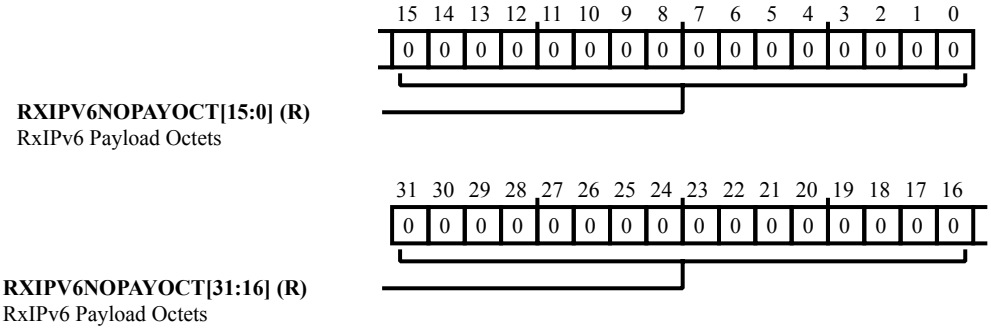


Figure 30-259: EMAC_RXIPV6_PYLD_OCNT Register Diagram

Table 30-314: EMAC_RXIPV6_PYLD_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXIPV6NOPAYOCT	RxIPv6 Payload Octets. The EMAC_RXIPV6_PYLD_OCNT.RXIPV6NOPAYOCT field indicates the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.)

Receive IPv6 Payload Packets Counter Register

The EMAC_RXIPV6_PYLD_PCNT register provides the number of IPv6 datagram packets received by the EMAC module that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

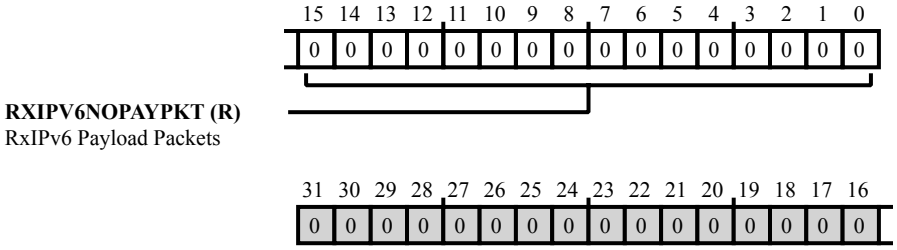


Figure 30-260: EMAC_RXIPV6_PYLD_PCNT Register Diagram

Table 30-315: EMAC_RXIPV6_PYLD_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXIPV6NOPAYPKT	RxIPv6 Payload Packets. The EMAC_RXIPV6_PYLD_PCNT.RXIPV6NOPAYPKT field indicates the number of IPv6 datagram packets received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

Receive TCP Error Octets Counter Register

The EMAC_RXTCP_ERR_OCNT register provides the number of bytes received by the EMAC module in a TCP segment that had checksum errors. This counter does not count IP header bytes.

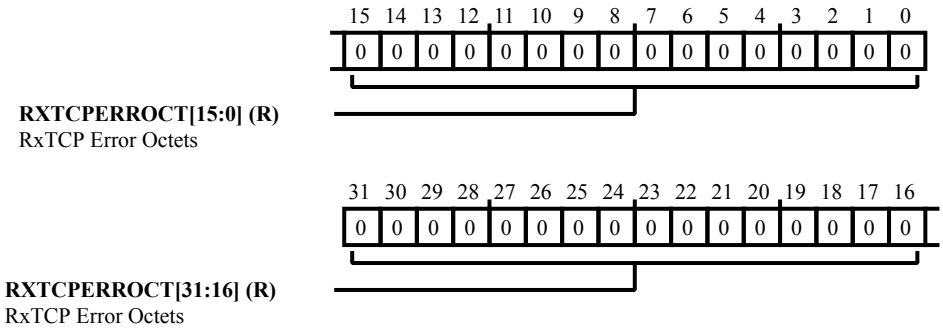


Figure 30-261: EMAC_RXTCP_ERR_OCNT Register Diagram

Table 30-316: EMAC_RXTCP_ERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXTCPERROCT	RxTCP Error Octets. The EMAC_RXTCP_ERR_OCNT.RXTCPERROCT field indicates the number of bytes received in a TCP segment that had checksum errors. This counter does not count IP header bytes.

Receive TCP Error Packets Counter Register

The EMAC_RXTCP_ERR_PCNT register provides the number of good IP datagrams received by the EMAC module whose TCP payload has a checksum error.

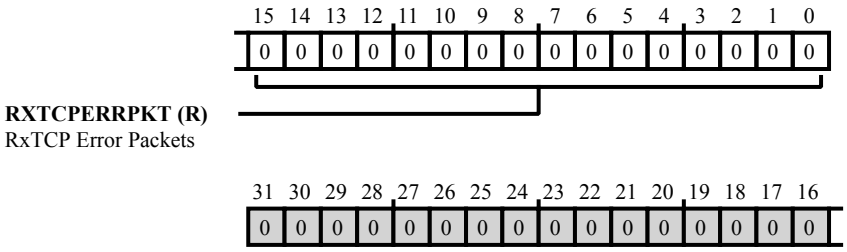


Figure 30-262: EMAC_RXTCP_ERR_PCNT Register Diagram

Table 30-317: EMAC_RXTCP_ERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXTCPERRPKT	RxTCP Error Packets. The EMAC_RXTCP_ERR_PCNT.RXTCPERRPKT field indicates the number of good IP datagrams received whose TCP payload has a checksum error.

Receive TCP Good Octets Counter Register

The EMAC_RXTCP_OCNT_G register provides the number of bytes received by the EMAC module in a good TCP segment. This counter does not count IP header bytes.

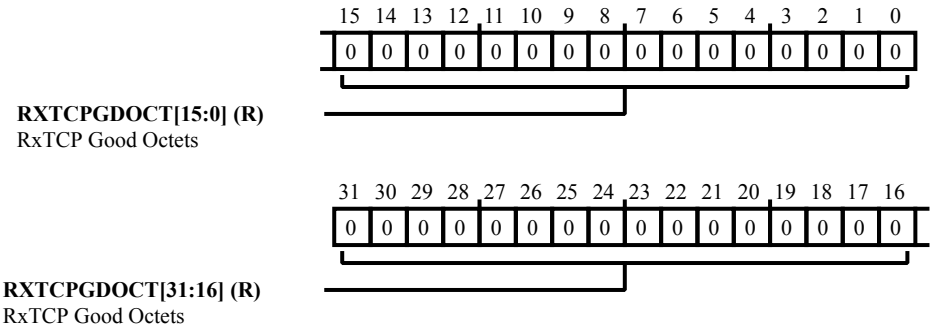


Figure 30-263: EMAC_RXTCP_OCNT_G Register Diagram

Table 30-318: EMAC_RXTCP_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXTCPGDOCT	RxTCP Good Octets. The EMAC_RXTCP_OCNT_G.RXTCPGDOCT field indicates the number of bytes received in a good TCP segment. This counter does not count IP header bytes.

Receive TCP Good Packets Counter Register

The EMAC_RXTCP_PCNT_G register provides the number of good IP datagrams received by the EMAC module with a good TCP payload.

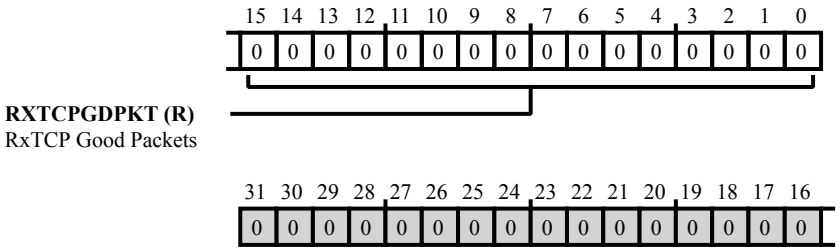


Figure 30-264: EMAC_RXTCP_PCNT_G Register Diagram

Table 30-319: EMAC_RXTCP_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXTCPGDPKT	RxTCP Good Packets. The EMAC_RXTCP_PCNT_G.RXTCPGDPKT field indicates the number of good IP datagrams received with a good TCP payload.

Receive UDP Error Octets Counter Register

The EMAC_RXUDP_ERR_OCNT register provides the number of bytes received by the EMAC module in a UDP segment that had checksum errors. This counter does not count IP header bytes.

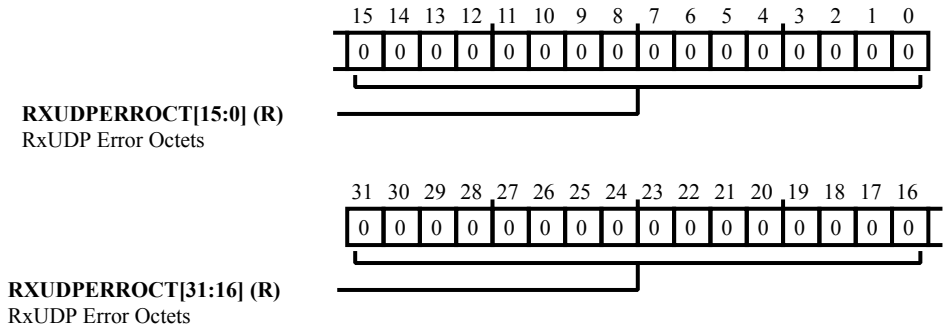


Figure 30-265: EMAC_RXUDP_ERR_OCNT Register Diagram

Table 30-320: EMAC_RXUDP_ERR_OCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXUDPERROCT	RxUDP Error Octets. The EMAC_RXUDP_ERR_OCNT.RXUDPERROCT field indicates the number of bytes received in a UDP segment that had checksum errors. This counter does not count IP header bytes.

Receive UDP Error Packets Counter Register

The EMAC_RXUDP_ERR_PCNT register provides the number of good IP datagrams received by the EMAC module whose UDP payload has a checksum error.

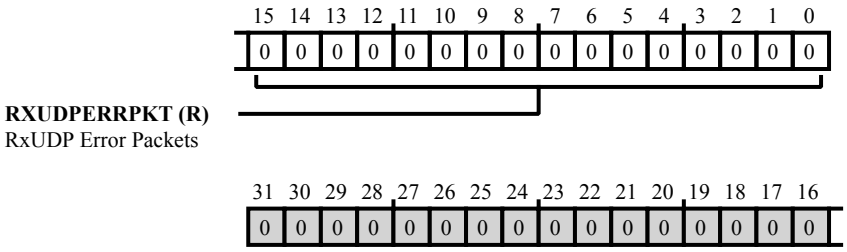


Figure 30-266: EMAC_RXUDP_ERR_PCNT Register Diagram

Table 30-321: EMAC_RXUDP_ERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXUDPERRPKT	RxUDP Error Packets. The EMAC_RXUDP_ERR_PCNT.RXUDPERRPKT field indicates the number of good IP datagrams received whose UDP payload has a checksum error.

Receive UDP Good Octets Counter Register

The EMAC_RXUDP_OCNT_G register provides the number of bytes received by the EMAC module in a good UDP segment. This counter does not count IP header bytes.

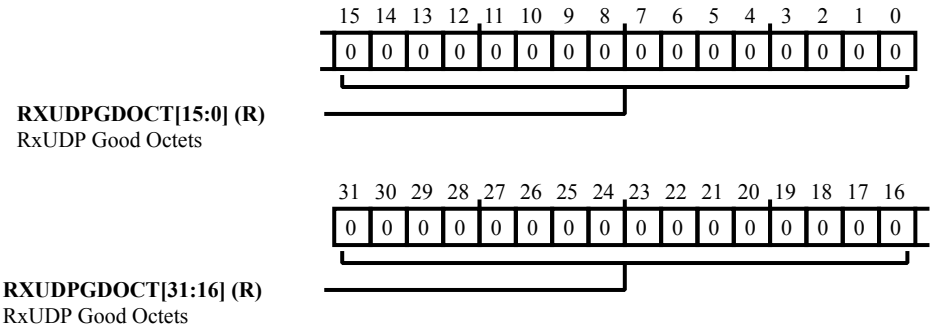


Figure 30-267: EMAC_RXUDP_OCNT_G Register Diagram

Table 30-322: EMAC_RXUDP_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXUDPGDOCT	RxUDP Good Octets. The EMAC_RXUDP_OCNT_G.RXUDPGDOCT field indicates the number of bytes received in a good UDP segment. This counter does not count IP header bytes.

Receive UDP Good Packets Counter Register

The EMAC_RXUDP_PCNT_G register provides the number of good IP datagrams received by the EMAC module with a good UDP payload. This counter is not updated when the EMAC_RXIPV4_UDPCSD_PCNT counter is incremented.

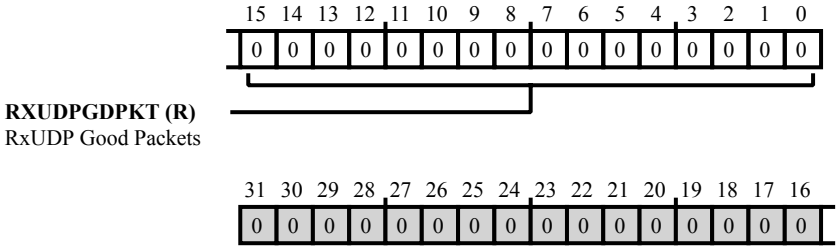


Figure 30-268: EMAC_RXUDP_PCNT_G Register Diagram

Table 30-323: EMAC_RXUDP_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXUDPGDPKT	RxUDP Good Packets. The EMAC_RXUDP_PCNT_G.RXUDPGDPKT field indicates the number of good IP datagrams received with a good UDP payload. This counter is not updated when the EMAC_RXIPV4_UDPCSD_PCNT counter is incremented.

Receive 1024 - Maximum Octets Packets Good Bad Counter Register

The EMAC_RX1024TOMAX_GB register provides the number of good and bad packets received by the EMAC module with length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

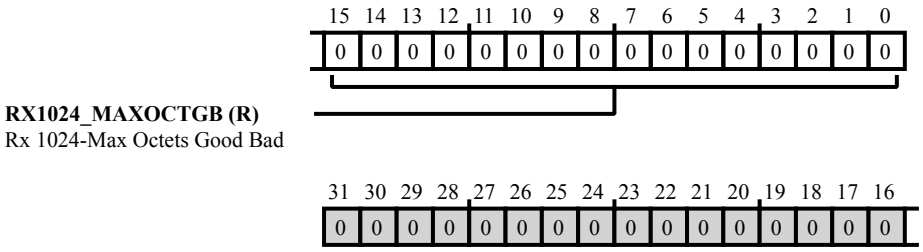


Figure 30-269: EMAC_RX1024TOMAX_GB Register Diagram

Table 30-324: EMAC_RX1024TOMAX_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX1024_MAXOCTGB	Rx 1024-Max Octets Good Bad. The EMAC_RX1024TOMAX_GB.RX1024_MAXOCTGB field indicates the number of good and bad packets received with a length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

Receive 128 - 255 Octets Packets Good Bad Counter Register

The EMAC_RX128TO255_GB register provides the number of good and bad packets received by the EMAC module with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

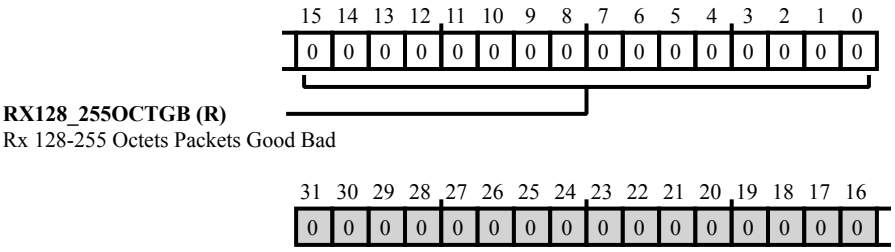


Figure 30-270: EMAC_RX128TO255_GB Register Diagram

Table 30-325: EMAC_RX128TO255_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX128_255OCTGB	Rx 128-255 Octets Packets Good Bad. The EMAC_RX128TO255_GB.RX128_255OCTGB field indicates the number of good and bad packets received with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

Receive 256 - 511 Octets Packets Good Bad Counter Register

The EMAC_RX256TO511_GB register provides the number of good and bad packets received by the EMAC module with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

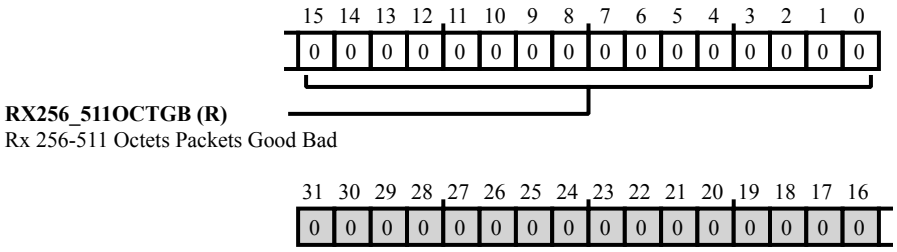


Figure 30-271: EMAC_RX256TO511_GB Register Diagram

Table 30-326: EMAC_RX256TO511_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX256_511OCTGB	Rx 256-511 Octets Packets Good Bad. The EMAC_RX256TO511_GB.RX256_511OCTGB field indicates the number of good and bad packets received with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

Receive 512 - 1023 Octets Packets Good Bad Counter Register

The EMAC_RX512TO1023_GB register provides the number of good and bad packets received by the EMAC module with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

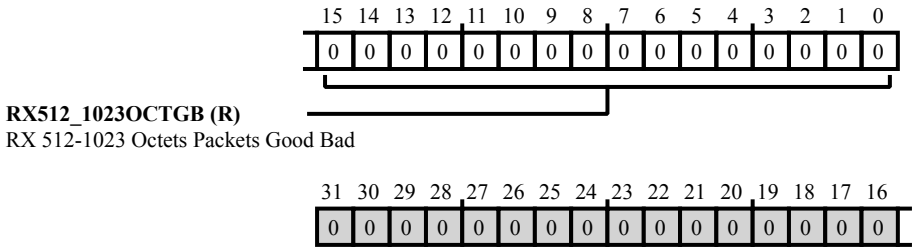


Figure 30-272: EMAC_RX512TO1023_GB Register Diagram

Table 30-327: EMAC_RX512TO1023_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX512_1023OCTGB	RX 512-1023 Octets Packets Good Bad. The EMAC_RX512TO1023_GB.RX512_1023OCTGB field indicates the number of good and bad packets received with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

Receive 64 Octets Packets Good Bad Counter Register

The EMAC_RX64_GB register provides the number of good and bad packets received by the EMAC module with length 64 bytes, exclusive of the preamble.

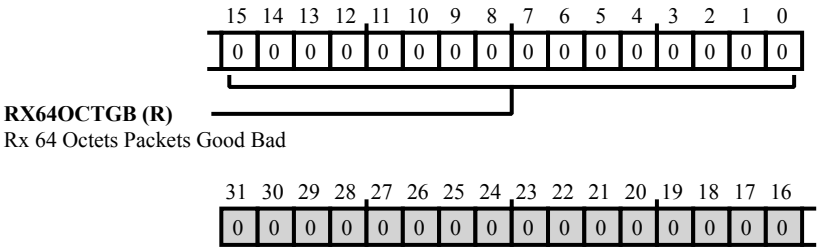


Figure 30-273: EMAC_RX64_GB Register Diagram

Table 30-328: EMAC_RX64_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX64OCTGB	Rx 64 Octets Packets Good Bad. The EMAC_RX64_GB.RX64OCTGB field indicates the number of good and bad packets received with length 64 bytes, exclusive of the preamble.

Receive 65 - 127 Octets Packets Good Bad Counter Register

The EMAC_RX65TO127_GB register provides the number of good and bad packets received by the EMAC module with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

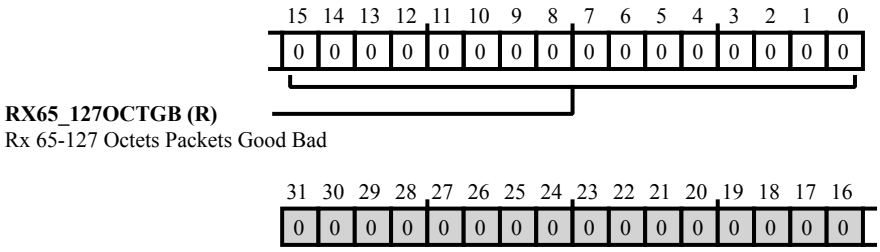


Figure 30-274: EMAC_RX65TO127_GB Register Diagram

Table 30-329: EMAC_RX65TO127_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RX65_127OCTGB	Rx 65-127 Octets Packets Good Bad. The EMAC_RX65TO127_GB.RX65_127OCTGB field indicates the number of good and bad packets received with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

Receive Alignment Error Packets Counter Register

The EMAC_RX_ALGNERR_PCNT register provides the number of packets received by the EMAC module with alignment (dribble) error. It is valid only in 10/100 mode.

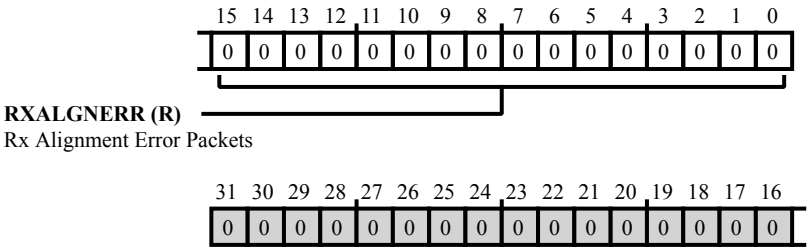


Figure 30-275: EMAC_RX_ALGNERR_PCNT Register Diagram

Table 30-330: EMAC_RX_ALGNERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXALGNERR	Rx Alignment Error Packets. The EMAC_RX_ALGNERR_PCNT.RXALGNERR field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

Receive Broadcast Packets Good Counter Register

The EMAC_RX_BCERR_PCNT_G register provides the number of good broadcast packets received by the EMAC module.

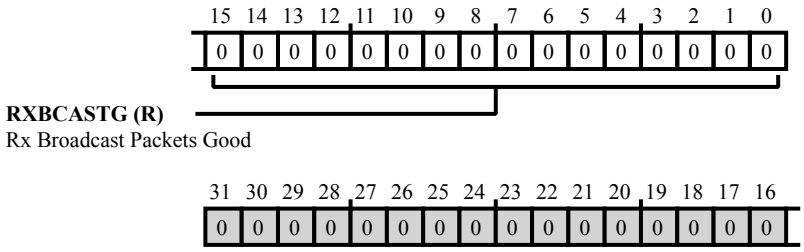


Figure 30-276: EMAC_RX_BCERR_PCNT_G Register Diagram

Table 30-331: EMAC_RX_BCERR_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXBCASTG	Rx Broadcast Packets Good. The EMAC_RX_BCERR_PCNT_G.RXBCASTG field indicates the number of good broadcast packets received.

Receive CRC Error Packets Counter Register

The EMAC_RX_CRCERR_PCNT register provides the number of packets received by the EMAC module with CRC error.

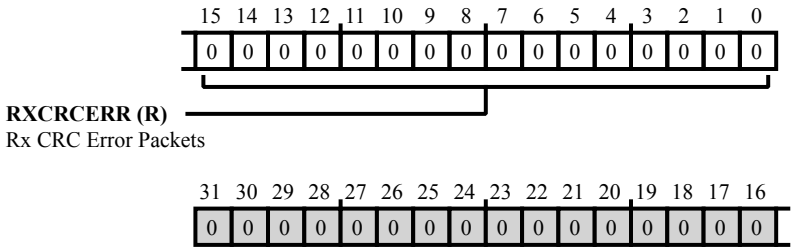


Figure 30-277: EMAC_RX_CRCERR_PCNT Register Diagram

Table 30-332: EMAC_RX_CRCERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXCRCERR	Rx CRC Error Packets. The EMAC_RX_CRCERR_PCNT.RXCRCERR field indicates the number of packets received with CRC error.

Receive Control Packets Good Counter Register

The EMAC_RX_CTL_PCNT_G register provides the number of good control packets received by the EMAC module.

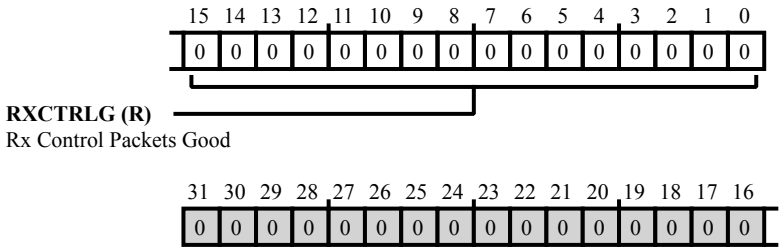


Figure 30-278: EMAC_RX_CTL_PCNT_G Register Diagram

Table 30-333: EMAC_RX_CTL_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXCTRLG	Rx Control Packets Good. The EMAC_RX_CTL_PCNT_G.RXCTRLG field indicates the number of good control packets received.

Receive FIFO Overflow Packets Counter Register

The EMAC_RX_FIFOOVFL_PCNT register provides the number of missed received packets because of FIFO overflow in the EMAC module.

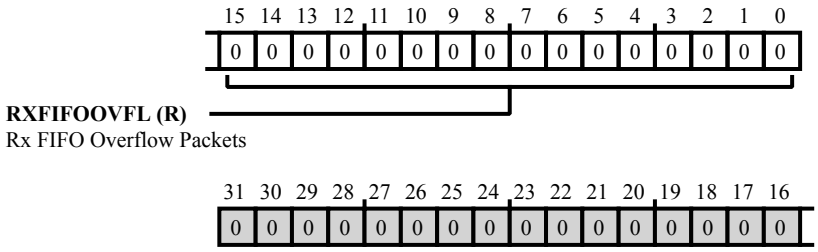


Figure 30-279: EMAC_RX_FIFOOVFL_PCNT Register Diagram

Table 30-334: EMAC_RX_FIFOOVFL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXFIFOOVFL	Rx FIFO Overflow Packets. The EMAC_RX_FIFOOVFL_PCNT.RXFIFOOVFL field indicates the number of missed received packets because of FIFO overflow.

Receive Jabber Error Packets Counter Register

The EMAC_RX_JBERR_PCNT register provides the number of giant packets received by the EMAC module with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. When jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1), packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

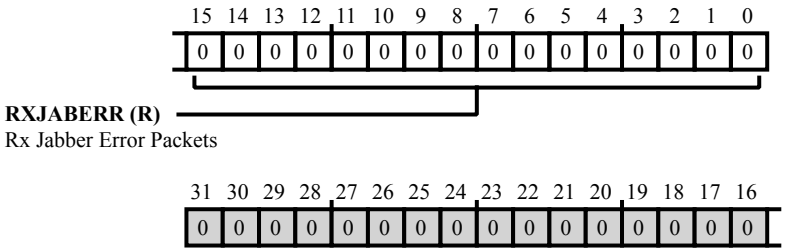


Figure 30-280: EMAC_RX_JBERR_PCNT Register Diagram

Table 30-335: EMAC_RX_JBERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXJABERR	Rx Jabber Error Packets. The EMAC_RX_JBERR_PCNT.RXJABERR field indicates the number of giant packets received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If jumbo packet mode is enabled (EMAC_MAC_CFG.JE = 1), packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

Receive LPI Transition Counter Register

The EMAC_RX_LPITRAN_CNT register provides the number of times the EMAC module has entered Rx LPI.

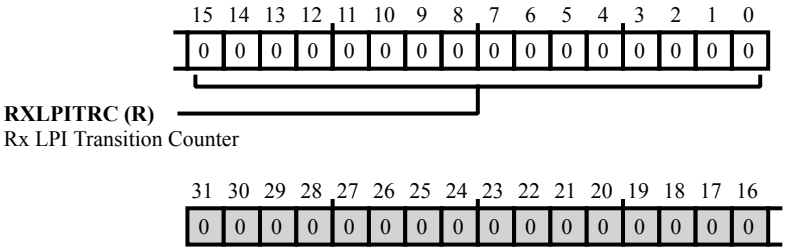


Figure 30-281: EMAC_RX_LPITRAN_CNT Register Diagram

Table 30-336: EMAC_RX_LPITRAN_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXLPITRC	Rx LPI Transition Counter. The EMAC_RX_LPITRAN_CNT.RXLPITRC field indicates the number of times Rx LPI entry has occurred.

Receive LPI Microseconds Counter Register

The EMAC_RX_LPIUSEC_CNT register provides the number of microseconds Rx LPI is sampled by the EMAC module.

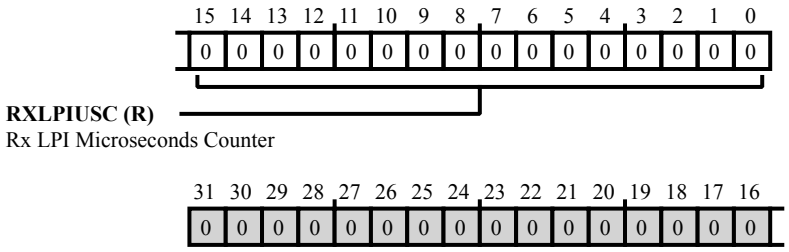


Figure 30-282: EMAC_RX_LPIUSEC_CNT Register Diagram

Table 30-337: EMAC_RX_LPIUSEC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXLPIUSC	Rx LPI Microseconds Counter. The EMAC_RX_LPIUSEC_CNT.RXLPIUSC field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI entry and exit, the timer value can have an error of +/- 1 microsecond.

Receive Length Error Packets Counter Register

The EMAC_RX_LENERR_PCNT register provides the number of packets received by the EMAC module with length error (Length Type field not equal to packet size), for all packets with valid length field.

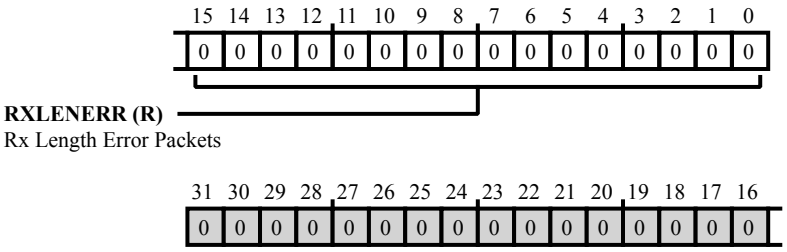


Figure 30-283: EMAC_RX_LENERR_PCNT Register Diagram

Table 30-338: EMAC_RX_LENERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXLENERR	Rx Length Error Packets. The EMAC_RX_LENERR_PCNT.RXLENERR field indicates the number of packets received with length error (length type field not equal to packet size), for all packets with valid length field.

Receive Multicast Packets Good Counter Register

The EMAC_RX_MCST_PCNT_G register provides the number of good multicast packets received by the EMAC module.

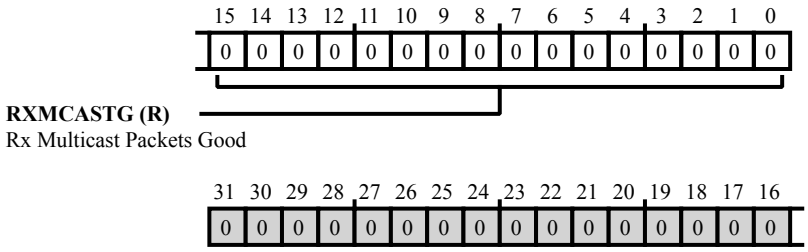


Figure 30-284: EMAC_RX_MCST_PCNT_G Register Diagram

Table 30-339: EMAC_RX_MCST_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXMCASTG	Rx Multicast Packets Good. The EMAC_RX_MCST_PCNT_G.RXMCASTG field indicates the number of good multicast packets received.

Receive Octet Good Counter Register

The EMAC_RX_OCNT_G register provides the number of bytes received by the EMAC module, exclusive of preamble, only in good packets.

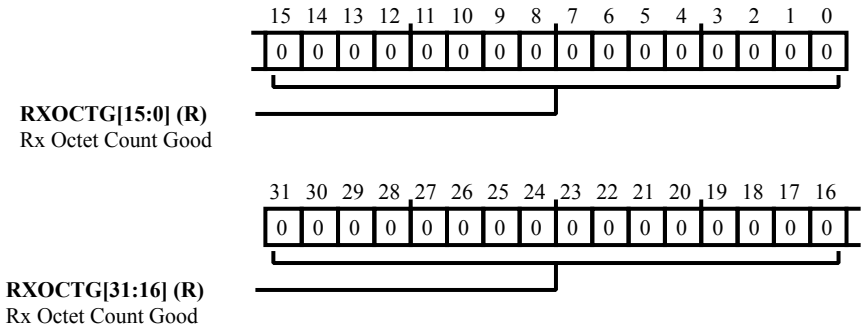


Figure 30-285: EMAC_RX_OCNT_G Register Diagram

Table 30-340: EMAC_RX_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXOCTG	Rx Octet Count Good. The EMAC_RX_OCNT_G.RXOCTG field indicates the number of bytes received, exclusive of preamble, only in good packets.

Receive Octet Good Bad Counter Register

The EMAC_RX_OCNT_GB register provides the number of bytes received by the EMAC module, exclusive of preamble, in good and bad packets.

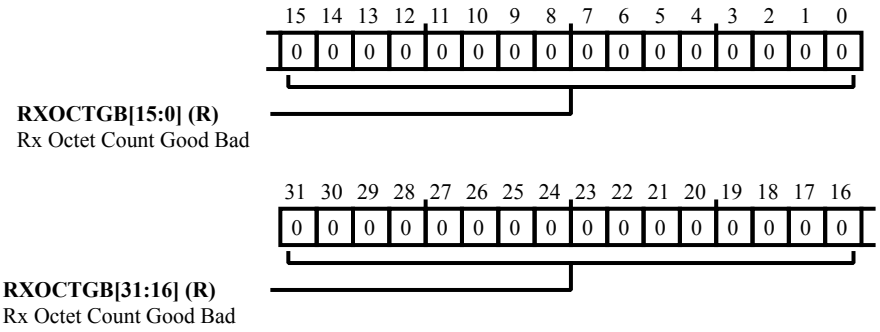


Figure 30-286: EMAC_RX_OCNT_GB Register Diagram

Table 30-341: EMAC_RX_OCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXOCTGB	Rx Octet Count Good Bad. The EMAC_RX_OCNT_GB.RXOCTGB field indicates the number of bytes received, exclusive of preamble, in good and bad packets.

Receive Out of Range Type Packets Counter Register

The EMAC_RX_ORTYPE_PCNT register provides the number of packets received by the EMAC module with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

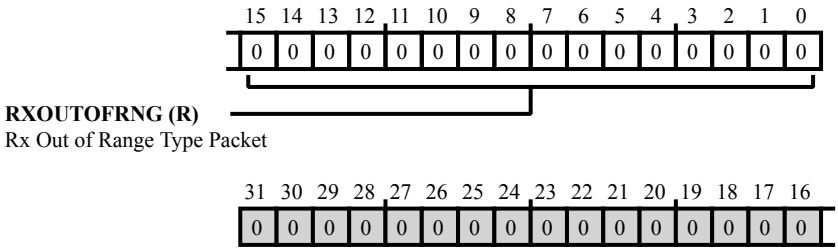


Figure 30-287: EMAC_RX_ORTYPE_PCNT Register Diagram

Table 30-342: EMAC_RX_ORTYPE_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXOUTOFRNG	Rx Out of Range Type Packet. The EMAC_RX_ORTYPE_PCNT.RXOUTOFRNG field indicates the number of packets received with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

Receive Oversize Packets Good Counter Register

The EMAC_RX_OVRSZ_PCNT register provides the number of packets received by the EMAC module without errors, with length greater than the max size (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes when EMAC_MAC_CFG MAC.S2KP = 1).

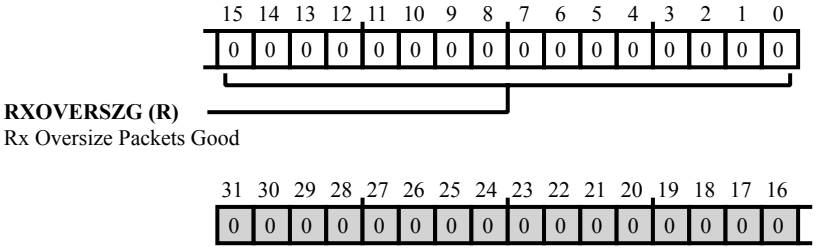


Figure 30-288: EMAC_RX_OVRSZ_PCNT Register Diagram

Table 30-343: EMAC_RX_OVRSZ_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXOVERSZG	Rx Oversize Packets Good. The EMAC_RX_OVRSZ_PCNT.RXOVERSZG field indicates the number of packets received without errors, with length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes when EMAC_MAC_CFG MAC.S2KP = 1).

Receive Packets Good Bad Counter Register

The EMAC_RX_PCNT_GB register provides the number of good and bad packets received by the EMAC module.

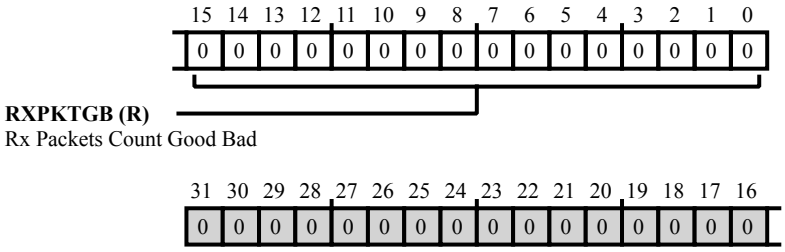


Figure 30-289: EMAC_RX_PCNT_GB Register Diagram

Table 30-344: EMAC_RX_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXPKTGB	Rx Packets Count Good Bad. The EMAC_RX_PCNT_GB.RXPKTGB field indicates the number of good and bad packets received.

Receive Pause Packets Counter Register

The EMAC_RX_PAUSE_PCNT register provides the number of good and valid Pause packets received by the EMAC module.

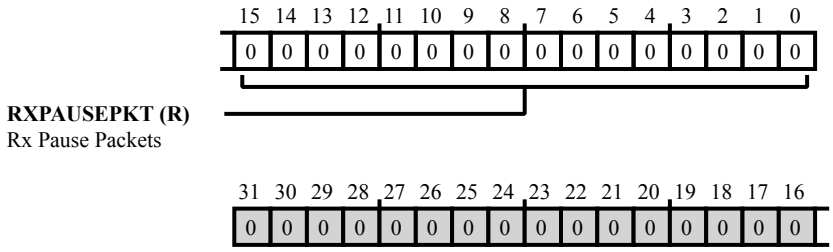


Figure 30-290: EMAC_RX_PAUSE_PCNT Register Diagram

Table 30-345: EMAC_RX_PAUSE_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	RXPAUSEPKT	Rx Pause Packets. The EMAC_RX_PAUSE_PCNT.RXPAUSEPKT field indicates the number of good and valid Pause packets received.

Receive Error Packets Counter Register

The EMAC_RX_ERR_PCNT register provides the number of packets received by the EMAC module with receive error or packet extension error on the GMII or MII interface.

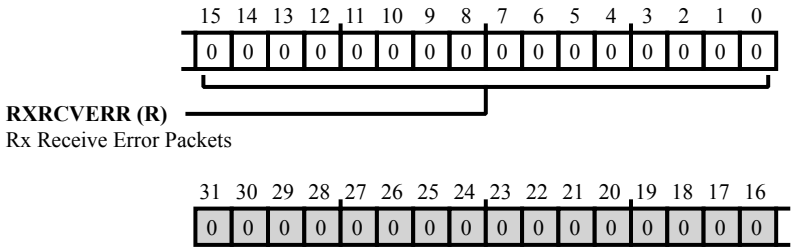


Figure 30-291: EMAC_RX_ERR_PCNT Register Diagram

Table 30-346: EMAC_RX_ERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXRCVERR	Rx Receive Error Packets. The EMAC_RX_ERR_PCNT.RXRCVERR field indicates the number of packets received with Receive error or Packet Extension error on the GMII or MII interface.

Receive Runt Error Packets Counter Register

The EMAC_RX_RNTERR_PCNT register provides the number of packets received by the EMAC module with runt (length less than 64 bytes and CRC error) error.

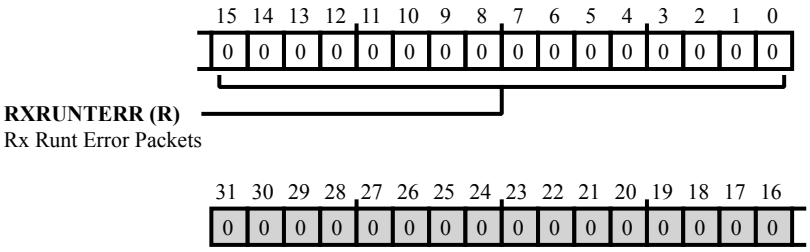


Figure 30-292: EMAC_RX_RNTERR_PCNT Register Diagram

Table 30-347: EMAC_RX_RNTERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXRUNTERR	Rx Runt Error Packets. The EMAC_RX_RNTERR_PCNT.RXRUNTERR field indicates the number of packets received with runt (length less than 64 bytes and CRC error) error.

Receive Undersize Packets Good Counter Register

The EMAC_RX_UNDRSZ_PCNT_G register provides the number of packets received by the EMAC module with length less than 64 bytes, without any errors.

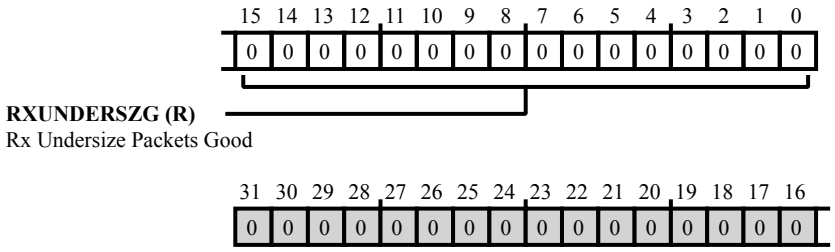


Figure 30-293: EMAC_RX_UNDRSZ_PCNT_G Register Diagram

Table 30-348: EMAC_RX_UNDRSZ_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXUNDERSZG	Rx Undersize Packets Good. The EMAC_RX_UNDRSZ_PCNT_G.RXUNDERSZG field indicates the number of packets received with length less than 64 bytes, without any errors.

Receive Unicast Packets Good Counter Register

The EMAC_RX_UCST_PCNT_G register provides the number of good unicast packets received by the EMAC module.

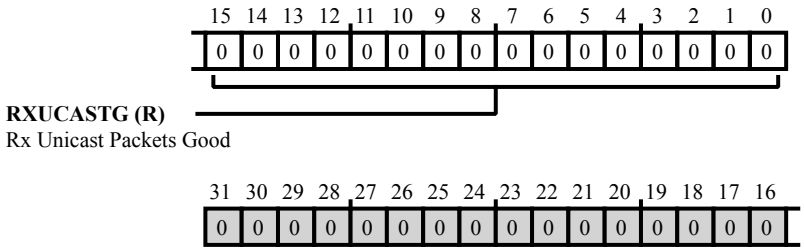


Figure 30-294: EMAC_RX_UCST_PCNT_G Register Diagram

Table 30-349: EMAC_RX_UCST_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXUCASTG	Rx Unicast Packets Good. The EMAC_RX_UCST_PCNT_G.RXUCASTG field indicates the number of good unicast packets received.

Receive VLAN Packets Good Bad Counter Register

The EMAC_RX_VLAN_PCNT_GB register provides the number of good and bad VLAN packets received by the EMAC module.

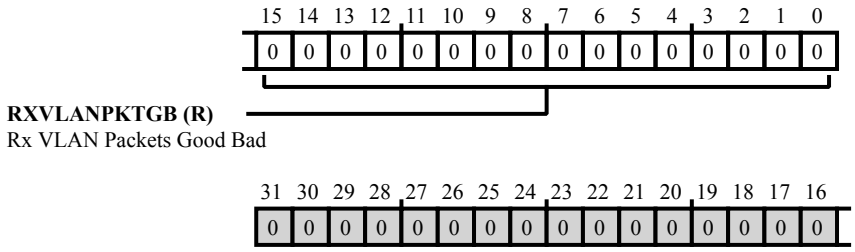


Figure 30-295: EMAC_RX_VLAN_PCNT_GB Register Diagram

Table 30-350: EMAC_RX_VLAN_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXVLANPKTGB	Rx VLAN Packets Good Bad. The EMAC_RX_VLAN_PCNT_GB.RXVLANPKTGB field indicates the number of good and bad VLAN packets received.

Receive Watchdog Error Packets Counter Register

The EMAC_RX_WDTERR_PCNT register provides the number of packets received by the EMAC module with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when EMAC_MAC_CFG.JE and EMAC_MAC_CFG.WD = 0), 10,240 bytes (EMAC_MAC_CFG.JE = 1 and EMAC_MAC_CFG.WD = 0), 16,384 bytes (when EMAC_MAC_CFG.WD = 1) or the value programmed in the EMAC_WDT_TMOUT register).

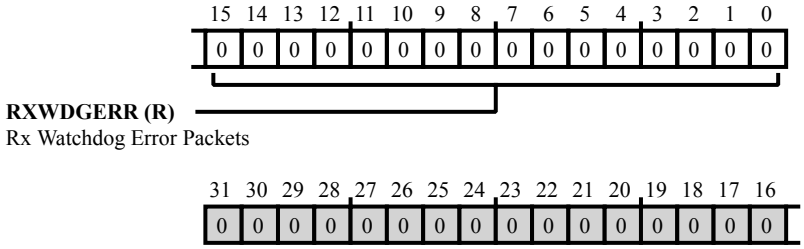


Figure 30-296: EMAC_RX_WDTERR_PCNT Register Diagram

Table 30-351: EMAC_RX_WDTERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	RXWDGERR	Rx Watchdog Error Packets. The EMAC_RX_WDTERR_PCNT.RXWDGERR field indicates the number of packets received with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when EMAC_MAC_CFG MAC.JE = 0 and EMAC_MAC_CFG MAC.WD = 0), 10,240 bytes (when EMAC_MAC_CFG MAC.JE = 1 and EMAC_MAC_CFG MAC.WD = 0), 16,384 bytes (when EMAC_MAC_CFG MAC.WD = 1) or the value programmed in the EMAC_WDT_TMOUT register).

Transmit 1024-Max Octets Packets Good Bad Counter Register

The EMAC_TX1024TOMAX_GB register provides the number of good and bad packets transmitted by the EMAC module with length 1024 to maxsize (inclusive) bytes, exclusive of preamble and retried packets.

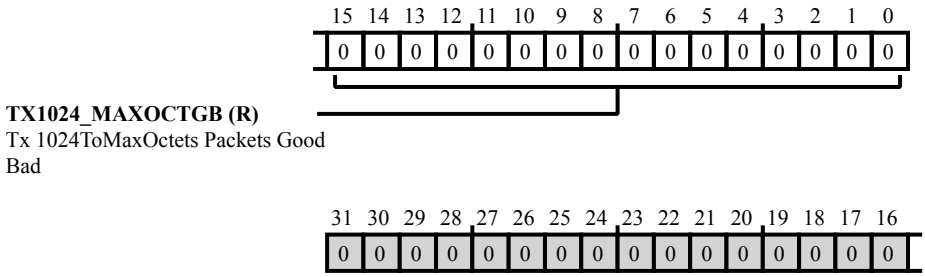


Figure 30-297: EMAC_TX1024TOMAX_GB Register Diagram

Table 30-352: EMAC_TX1024TOMAX_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TX1024_MAXOCTGB	Tx 1024ToMaxOctets Packets Good Bad. The EMAC_TX1024TOMAX_GB.TX1024_MAXOCTGB field indicates the number of good and bad packets transmitted with length between 1024 and maxsize (inclusive) bytes, exclusive of preamble and retried packets.

Transmit 128-255 Octets Packets Good Bad Counter Register

The EMAC_TX128TO255_GB register provides the number of good and bad packets transmitted by the EMAC module with length between 128 to 255 (inclusive) bytes, exclusive of preamble and retried packets.

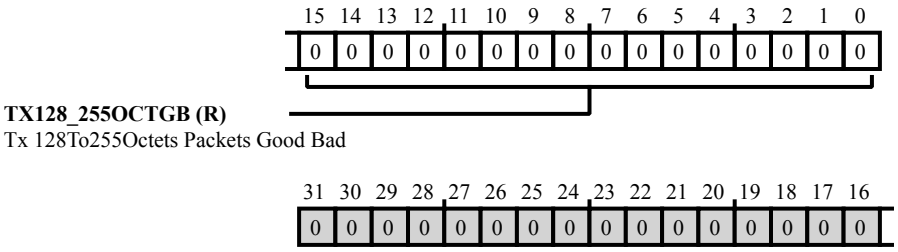


Figure 30-298: EMAC_TX128TO255_GB Register Diagram

Table 30-353: EMAC_TX128TO255_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TX128_255OCTGB	Tx 128To255Octets Packets Good Bad. The EMAC_TX128TO255_GB.TX128_255OCTGB field indicates the number of good and bad packets transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried packets.

Transmit 256-511 Octets Packets Good Bad Counter Register

The EMAC_TX256TO511_GB register provides the number of good and bad packets transmitted by the EMAC module with length between 256 to 511 (inclusive) bytes, exclusive of preamble and retried packets.

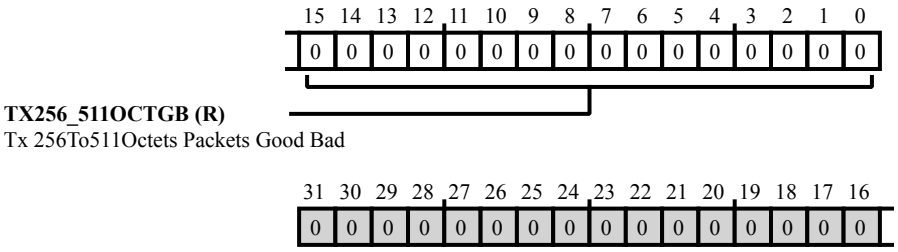


Figure 30-299: EMAC_TX256TO511_GB Register Diagram

Table 30-354: EMAC_TX256TO511_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TX256_511OCTGB	Tx 256To511Octets Packets Good Bad. The EMAC_TX256TO511_GB.TX256_511OCTGB field indicates the number of good and bad packets transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried packets.

Transmit 512-1023 Octets Packets Good Bad Counter Register

The EMAC_TX512TO1023_GB register provides the number of good and bad packets transmitted by the EMAC module with length 512 to 1023 (inclusive) bytes, exclusive of preamble and retried packets.

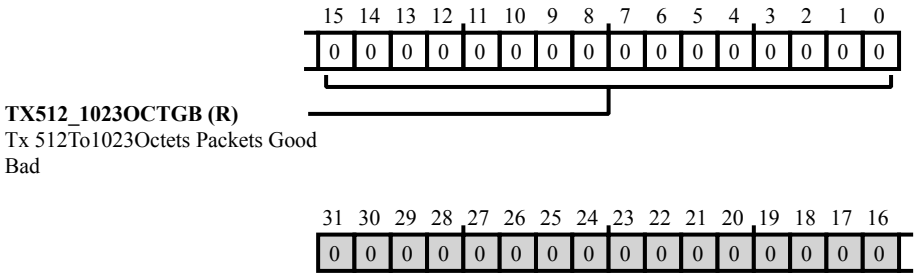


Figure 30-300: EMAC_TX512TO1023_GB Register Diagram

Table 30-355: EMAC_TX512TO1023_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TX512_1023OCTGB	Tx 512To1023Octets Packets Good Bad. The EMAC_TX512TO1023_GB.TX512_1023OCTGB field indicates the number of good and bad packets transmitted with length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried packets.

Transmit 64 Octets Packets Good Bad Counter Register

The EMAC_TX64_GB register provides the number of good and bad packets transmitted by the EMAC module with length 64 bytes, exclusive of preamble and retried packets.

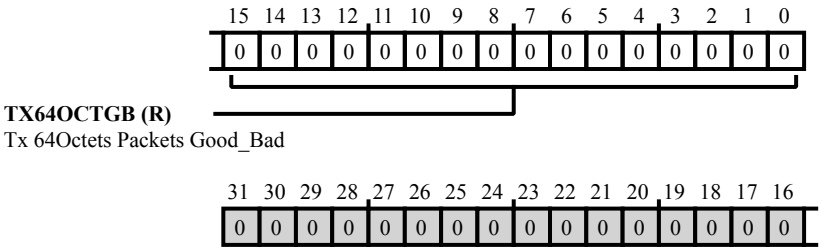


Figure 30-301: EMAC_TX64_GB Register Diagram

Table 30-356: EMAC_TX64_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TX64OCTGB	Tx 64Octets Packets Good_Bad. The EMAC_TX64_GB.TX64OCTGB field indicates the number of good and bad packets transmitted with length 64 bytes, exclusive of preamble and retried packets.

Transmit 65-127 Octets Packets Good Bad Counter Register

The EMAC_TX65TO127_GB register provides the number of good and bad packets transmitted by the EMAC module with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

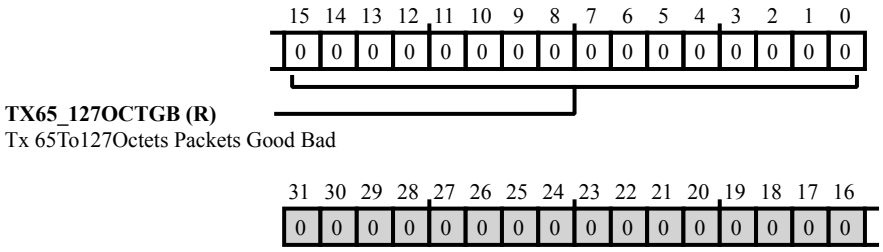


Figure 30-302: EMAC_TX65TO127_GB Register Diagram

Table 30-357: EMAC_TX65TO127_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TX65_127OCTGB	Tx 65To127Octets Packets Good Bad. The EMAC_TX65TO127_GB.TX65_127OCTGB field indicates the number of good and bad packets transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

Transmit Broadcast Packets Good Counter Register

The EMAC_TX_BCST_PCNT_G register provides the number of good broadcast packets transmitted by the EMAC module.

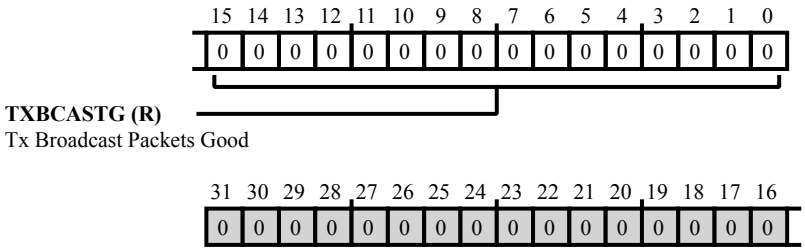


Figure 30-303: EMAC_TX_BCST_PCNT_G Register Diagram

Table 30-358: EMAC_TX_BCST_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXBCASTG	Tx Broadcast Packets Good. The EMAC_TX_BCST_PCNT_G.TXBCASTG field indicates the number of good broadcast packets transmitted.

Transmit Broadcast Packets Good Bad Counter Register

The EMAC_TX_BCST_PCNT_GB register provides the number of good and bad broadcast packets transmitted by the EMAC module.

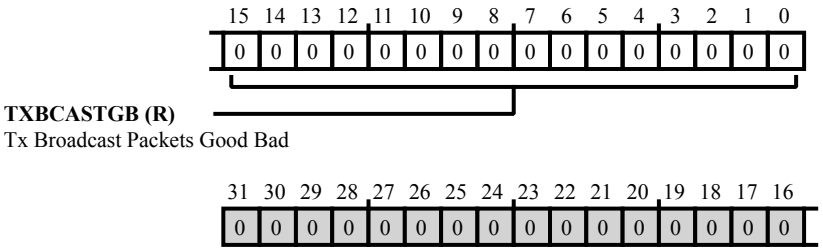


Figure 30-304: EMAC_TX_BCST_PCNT_GB Register Diagram

Table 30-359: EMAC_TX_BCST_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXBCASTGB	Tx Broadcast Packets Good Bad. The EMAC_TX_BCST_PCNT_GB.TXBCASTGB field indicates the number of good and bad broadcast packets transmitted.

Transmit Carrier Error Packets Counter Register

The EMAC_TX_CRERR_PCNT register provides the number of packets aborted by the EMAC module because of carrier sense error (no carrier or loss of carrier).

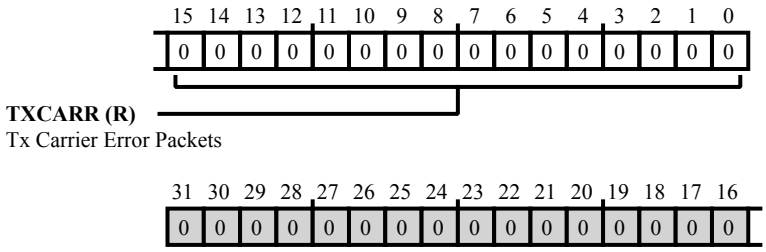


Figure 30-305: EMAC_TX_CRERR_PCNT Register Diagram

Table 30-360: EMAC_TX_CRERR_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXCARR	Tx Carrier Error Packets. The EMAC_TX_CRERR_PCNT.TXCARR field indicates the number of packets aborted because of carrier sense error (no carrier or loss of carrier).

Transmit Deferred Packets Counter Register

The EMAC_TX_DFRD_PCNT register provides the number of successfully transmitted by the EMAC module after a deferral in the half-duplex mode.

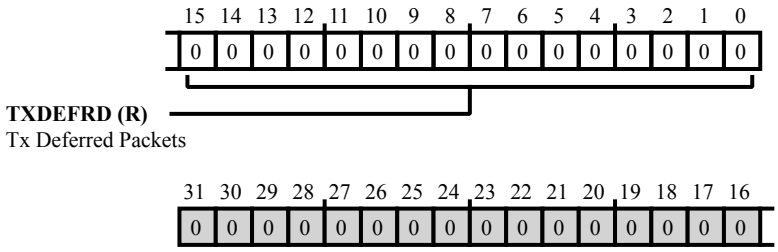


Figure 30-306: EMAC_TX_DFRD_PCNT Register Diagram

Table 30-361: EMAC_TX_DFRD_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXDEFRD	Tx Deferred Packets. The EMAC_TX_DFRD_PCNT.TXDEFRD field indicates the number of successfully transmitted after a deferral in the half-duplex mode.

Transmit Excessive Collision Packets Counter Register

The EMAC_TX_EXCOL_PCNT register provides the number of packets aborted by the EMAC module because of excessive (16) collision errors.

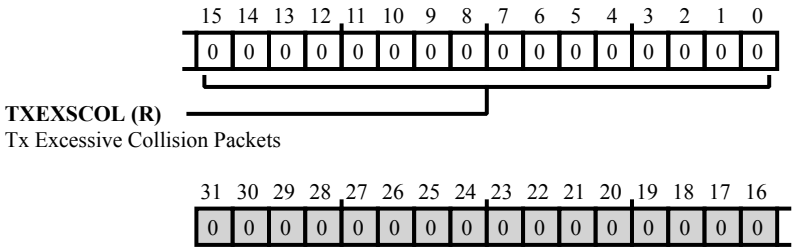


Figure 30-307: EMAC_TX_EXCOL_PCNT Register Diagram

Table 30-362: EMAC_TX_EXCOL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXEXSCOL	Tx Excessive Collision Packets. The EMAC_TX_EXCOL_PCNT.TXEXSCOL field indicates the number of packets aborted because of excessive (16) collision errors.

Transmit Excessive Deferral Error Counter Register

The EMAC_TX_EXDFRL_PCNT register provides the number of packets aborted by the EMAC module because of excessive deferral error (deferred for more than two max-sized packet times).

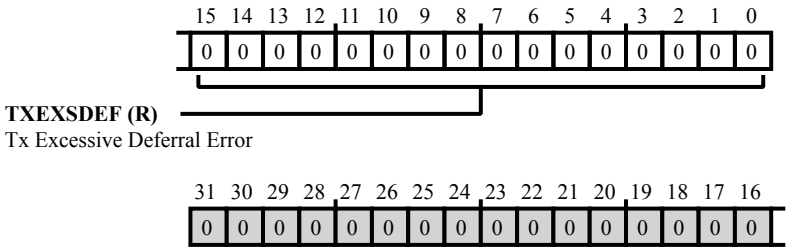


Figure 30-308: EMAC_TX_EXDFRL_PCNT Register Diagram

Table 30-363: EMAC_TX_EXDFRL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXEXSDEF	Tx Excessive Deferral Error. The EMAC_TX_EXDFRL_PCNT.TXEXSDEF field indicates the number of packets aborted because of excessive deferral error (deferred for more than two max-sized packet times).

Transmit LPI Transition Counter Register

The EMAC_TX_LPITRAN_CNT register provides the number of times the EMAC module has entered Tx LPI.

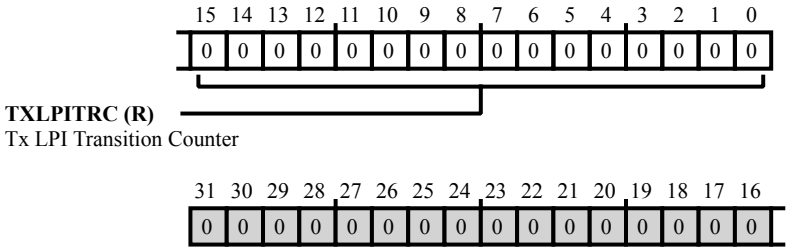


Figure 30-309: EMAC_TX_LPITRAN_CNT Register Diagram

Table 30-364: EMAC_TX_LPITRAN_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXLPITRC	Tx LPI Transition Counter. The EMAC_TX_LPITRAN_CNT.TXLPITRC field indicates the number of times Tx LPI entry has occurred. Even if Tx LPI entry occurs in automate mode (EMAC_LPI_CTLSTAT.LPITXA = 1), the counter increments.

Transmit LPI Microseconds Counter Register

The EMAC_TX_LPIUSEC_CNT register provides the number of microseconds Tx LPI is asserted by the EMAC module.

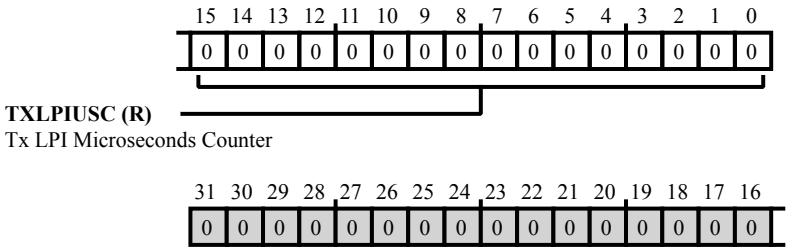


Figure 30-310: EMAC_TX_LPIUSEC_CNT Register Diagram

Table 30-365: EMAC_TX_LPIUSEC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	TXLPIUSC	Tx LPI Microseconds Counter. The EMAC_TX_LPIUSEC_CNT.TXLPIUSC field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI entry and exit, the timer value can have an error of +/- 1 microsecond.

Transmit Late Collision Packets Counter Register

The EMAC_TX_LTCOL_PCNT register provides the number of packets aborted by the EMAC module because of late collision error.

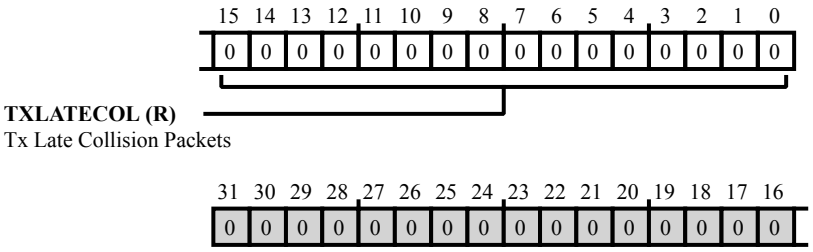


Figure 30-311: EMAC_TX_LTCOL_PCNT Register Diagram

Table 30-366: EMAC_TX_LTCOL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXLATECOL	Tx Late Collision Packets. The EMAC_TX_LTCOL_PCNT.TXLATECOL field indicates the number of packets aborted because of late collision error.

Transmit Multicast Packets Good Counter Register

The EMAC_TX_MCST_PCNT_G register provides the number of good multicast packets transmitted by the EMAC module.

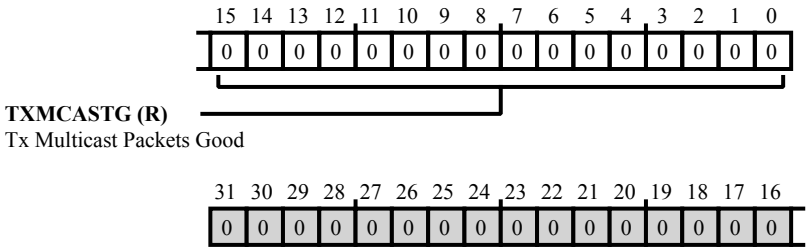


Figure 30-312: EMAC_TX_MCST_PCNT_G Register Diagram

Table 30-367: EMAC_TX_MCST_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXMCASTG	Tx Multicast Packets Good. The EMAC_TX_MCST_PCNT_G.TXMCASTG field indicates the number of good multicast packets transmitted.

Transmit Multicast Packets Good Bad Counter Register

The EMAC_TX_MCST_PCNT_GB register provides the number of good and bad multicast packets transmitted by the EMAC module.

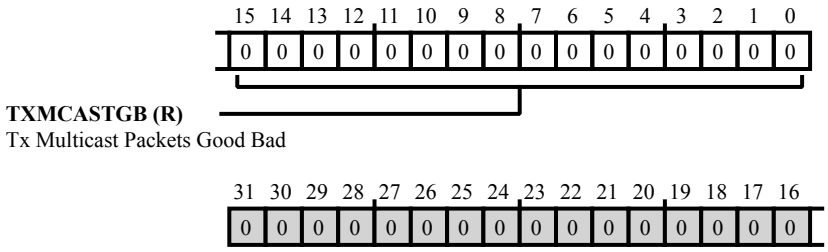


Figure 30-313: EMAC_TX_MCST_PCNT_GB Register Diagram

Table 30-368: EMAC_TX_MCST_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXMCASTGB	Tx Multicast Packets Good Bad. The EMAC_TX_MCST_PCNT_GB.TXMCASTGB field indicates the number of good and bad multicast packets transmitted.

Transmit Multiple Collision Packets Good Counter Register

The EMAC_TX_MCOL_PCNT_G register provides the number of successfully transmitted packets by the EMAC module after multiple collisions in the half-duplex mode.

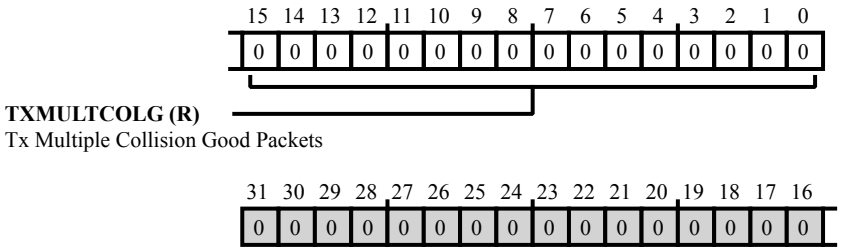


Figure 30-314: EMAC_TX_MCOL_PCNT_G Register Diagram

Table 30-369: EMAC_TX_MCOL_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXMULTCOLG	Tx Multiple Collision Good Packets. The EMAC_TX_MCOL_PCNT_G.TXMULTCOLG field indicates the number of successfully transmitted packets after multiple collisions in the half-duplex mode.

Transmit Osize Packets Good Counter Register

The EMAC_TX_OSIZE_PCNT_G register provides the number of packets transmitted by the EMAC module without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in EMAC_MAC_CFG.S2KP).

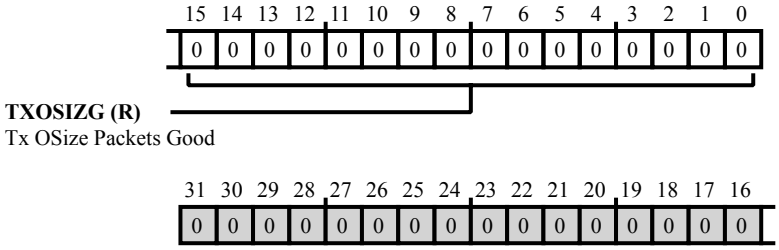


Figure 30-315: EMAC_TX_OSIZE_PCNT_G Register Diagram

Table 30-370: EMAC_TX_OSIZE_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXOSIZG	Tx OSize Packets Good. The EMAC_TX_OSIZE_PCNT_G.TXOSIZG field indicates the number of packets transmitted without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes when EMAC_MAC_CFG MAC.S2KP = 1).

Transmit Octet Good Counter Register

The EMAC_TX_OCNT_G register provides the number of bytes transmitted by the EMAC module, exclusive of preamble, only in good packets.

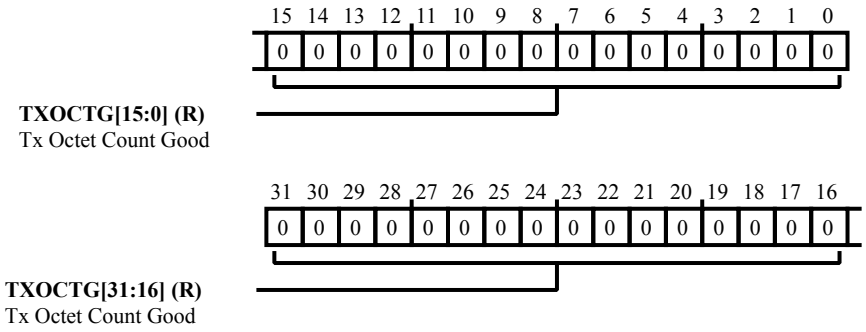


Figure 30-316: EMAC_TX_OCNT_G Register Diagram

Table 30-371: EMAC_TX_OCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXOCTG	Tx Octet Count Good. The EMAC_TX_OCNT_G.TXOCTG field indicates the number of bytes transmitted, exclusive of preamble, only in good packets.

Transmit Octet Good Bad Counter Register

The EMAC_TX_OCNT_GB register provides the number of bytes transmitted by the EMAC module, exclusive of preamble and retried bytes, in good and bad packets.

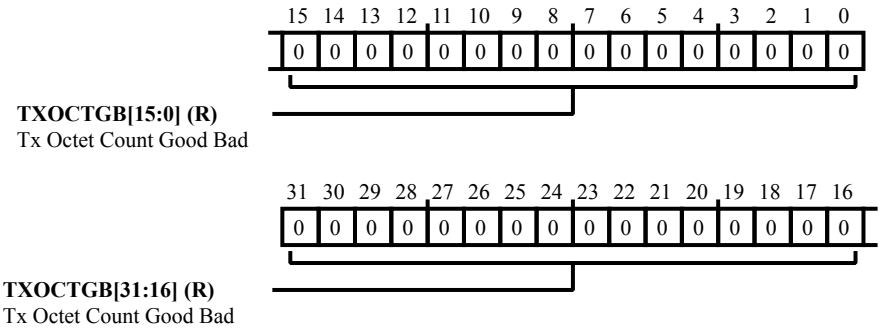


Figure 30-317: EMAC_TX_OCNT_GB Register Diagram

Table 30-372: EMAC_TX_OCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXOCTGB	Tx Octet Count Good Bad. The EMAC_TX_OCNT_GB.TXOCTGB field indicates the number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad packets.

Transmit Packet Good Counter Register

The EMAC_TX_PCNT_G register provides the number of good packets transmitted by the EMAC module.

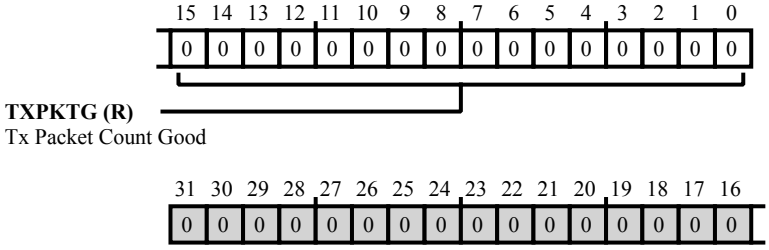


Figure 30-318: EMAC_TX_PCNT_G Register Diagram

Table 30-373: EMAC_TX_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXPKTG	Tx Packet Count Good. The EMAC_TX_PCNT_G.TXPKTG field indicates the number of good packets transmitted.

Transmit Packet Good Bad Counter Register

The EMAC_TX_PCNT_GB register provides the number of good and bad packets transmitted by the EMAC module, exclusive of retried packets.

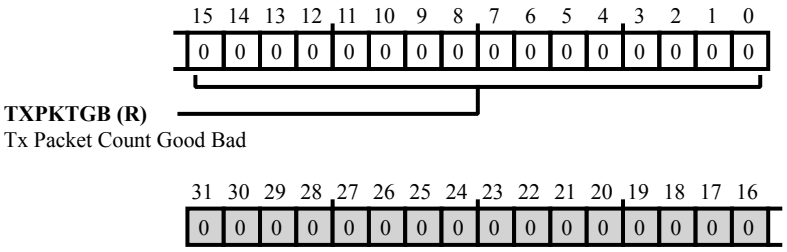


Figure 30-319: EMAC_TX_PCNT_GB Register Diagram

Table 30-374: EMAC_TX_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXPKTGB	Tx Packet Count Good Bad. The EMAC_TX_PCNT_GB.TXPKTGB field indicates the number of good and bad packets transmitted, exclusive of retried packets.

Transmit Pause Packets Counter Register

The EMAC_TX_PAUSE_PCNT register provides the number of good pause packets transmitted by the EMAC module.

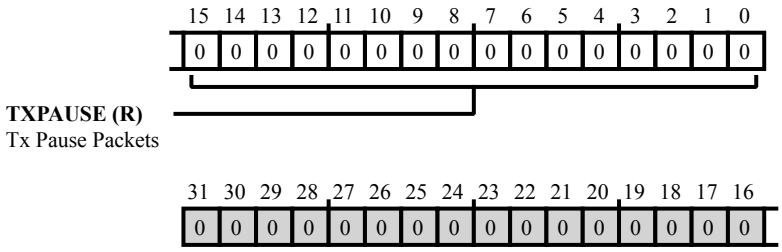


Figure 30-320: EMAC_TX_PAUSE_PCNT Register Diagram

Table 30-375: EMAC_TX_PAUSE_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXPAUSE	Tx Pause Packets. The EMAC_TX_PAUSE_PCNT.TXPAUSE field indicates the number of good Pause packets transmitted.

Transmit Single Collision Packets Good Counter Register

The EMAC_TX_SNGLCOL_PCNT_G register provides the number of successfully transmitted packets by the EMAC module after a single collision in the half-duplex mode.

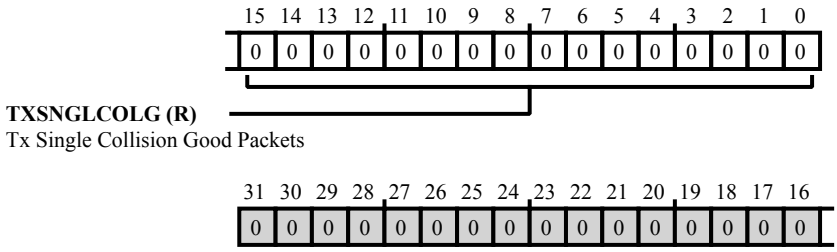


Figure 30-321: EMAC_TX_SNGLCOL_PCNT_G Register Diagram

Table 30-376: EMAC_TX_SNGLCOL_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXSNGLCOLG	Tx Single Collision Good Packets. The EMAC_TX_SNGLCOL_PCNT_G.TXSNGLCOLG field indicates the number of successfully transmitted packets after a single collision in the half-duplex mode.

Transmit Underflow Error Packets Counter Register

The EMAC_TX_URFL_PCNT register provides the number of packets aborted by the EMAC module because of packets underflow error.

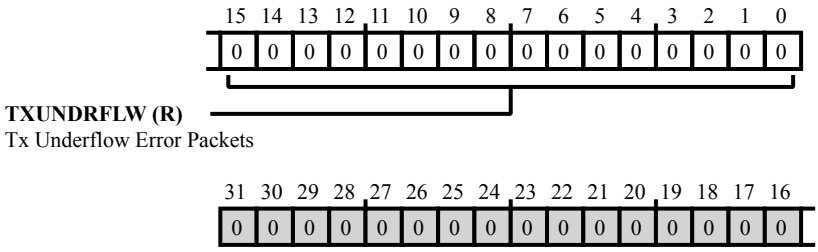


Figure 30-322: EMAC_TX_URFL_PCNT Register Diagram

Table 30-377: EMAC_TX_URFL_PCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXUNDRFLW	Tx Underflow Error Packets. The EMAC_TX_URFL_PCNT.TXUNDRFLW field indicates the number of packets aborted because of packets underflow error.

Transmit Unicast Packets Good Bad Counter Register

The EMAC_TX_UCST_PCNT_GB register provides the number of good and bad unicast packets transmitted by the EMAC module.

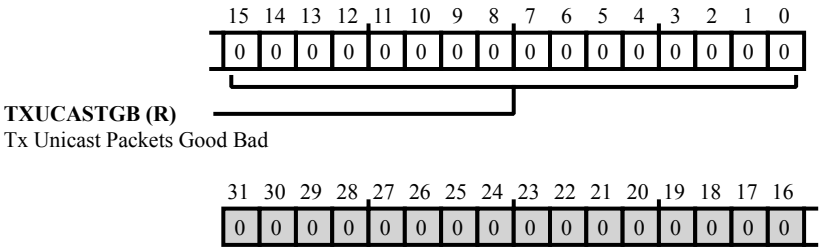


Figure 30-323: EMAC_TX_UCST_PCNT_GB Register Diagram

Table 30-378: EMAC_TX_UCST_PCNT_GB Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXUCASTGB	Tx Unicast Packets Good Bad. The EMAC_TX_UCST_PCNT_GB.TXUCASTGB field indicates the number of good and bad unicast packets transmitted.

Transmit VLAN Packets Good Counter Register

The EMAC_TX_VLAN_PCNT_G register provides the number of good VLAN packets transmitted by the EMAC module.

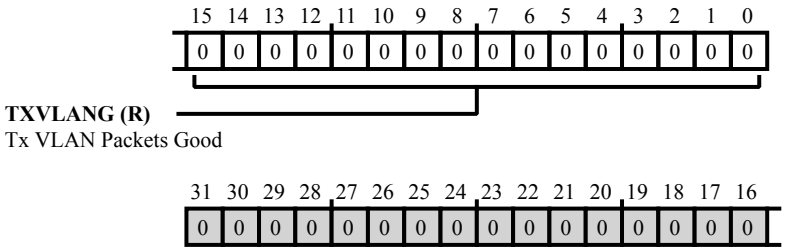


Figure 30-324: EMAC_TX_VLAN_PCNT_G Register Diagram

Table 30-379: EMAC_TX_VLAN_PCNT_G Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	TXVLANG	Tx VLAN Packets Good. The EMAC_TX_VLAN_PCNT_G.TXVLANG field provides the number of good VLAN packets transmitted.

31 Digital Audio Interface (DAI)

The Digital Audio Interfaces (DAIn) are comprised of groups of identical peripherals and their respective Signal Routing Units (SRUn). The SRU connects inputs and outputs of the DAI peripherals with each other and to the external pins. This configuration allows peripherals to be interconnected to accommodate a wide variety of systems without making external pin connections.

In a typical processor, static (multiplexed) pins are assigned to specific peripherals. When certain peripherals are not required for an application, these pins are unnecessary and expensive. The pins may need to be defined as high or low to prevent any illegal conditions. The signal routing units on the SHARC processors addresses this situation by controlling a number of general-purpose pins which can be assigned flexibly (a virtual connectivity between peripherals) depending on system requirements. This virtual connectivity includes pin buffers and routing logic (multiplexer). It also allows the SHARC processors to include an arbitrary number and variety of peripherals while retaining high levels of compatibility without increasing pin count.

SRU Features

The SRU has the following features and capabilities.

- Flexible connections can be made through software and during run time; no hard wiring is required.
- At reset, a default routing scheme is already programmed.
- Connectivity can be made internally between peripherals, externally between pin buffers, or a mix of both.
- Status of the pin buffers can be programmed for conditional execution or interrupts.
- Some pin buffers allow control of signal polarity changes.
- No fan-out limitation, a peripheral or pin buffer output that can be routed to multiple peripheral or pin buffer inputs.

NOTE: The 400-ball HPC BGA package has 2 x 20 DAI pins bonded off-chip (DAIx_PIN01 through DAIx_PIN20).

Functional Description

The fundamental timing clock of the DAI modules is SCLK0.

The DAI Block Diagram shows how the DAI pin buffers are connected through the SRUn. This configuration allows for flexible signal routing.

The DAI units are comprised of four primary blocks:

- Peripherals (A/B/C) associated with DAIn
- Signal Routing Units (SRUn)
- DAIn I/O pin buffers
- Miscellaneous buffers

The peripherals shown in DAI Block Diagram can have up to three connections (if requester or completer capable); one acts as a signal input, one as a signal output and the third as an output enable. The SRUs are based on a group of multiplexers which are controlled by registers to establish the desired interconnections. The DAI pin buffers have three signals which are used for input and output to or from off-chip and the third for output enable.

The miscellaneous buffers have an input and an output and are used for group interconnection.

The figures are a simplified representation of a DAI system. In a real representation, the SRU and DAI would show several types of data being routed from several sources including the following:

- Serial ports (SPORT)
- Precision clock generators (PCG)
- Asynchronous sample rate converters (SRC)
- S/PDIF transmitter
- S/PDIF receiver
- DAI interrupts (miscellaneous)
- PDM interface

ADSP-SC59x DAI Register List

The Digital Audio Interfaces (DAIn) contain groups of identical peripherals which can be connected internally between peripherals, externally between pin buffers, or a mix of both. This module contains the following registers.

Table 31-1: ADSP-SC59x DAI Register List

Name	Description
DAI_CLK0	Clock Routing Control Register 0
DAI_CLK1	Clock Routing Control Register 1
DAI_CLK2	Clock Routing Control Register 2
DAI_CLK3	Clock Routing Control Register 3
DAI_CLK4	Clock Routing Control Register 4

Table 31-1: ADSP-SC59x DAI Register List (Continued)

Name	Description
DAI_CLK5	Clock Routing Control Register 5
DAI_DAT0	Serial Data Routing Control Register 0
DAI_DAT1	Serial Data Routing Control Register 1
DAI_DAT2	Serial Data Routing Control Register 2
DAI_DAT3	Serial Data Routing Control Register 3
DAI_DAT4	Serial Data Routing Control Register 4
DAI_DAT5	Serial Data Routing Control Register 5
DAI_DAT6	Serial Data Routing Control Register 6
DAI_EXTD_CLK0	Extended Clock Routing Control Register 0
DAI_EXTD_CLK1	Extended Clock Routing Control Register 1
DAI_EXTD_CLK2	Extended Clock Routing Control Register 2
DAI_EXTD_CLK3	Extended Clock Routing Control Register 3
DAI_EXTD_CLK4	Extended Clock Routing Control Register 4
DAI_EXTD_CLK5	Extended Clock Routing Control Register 5
DAI_EXTD_DAT0	Extended Serial Data Routing Control Register 0
DAI_EXTD_DAT1	Extended Serial Data Routing Control Register 1
DAI_EXTD_DAT2	Extended Serial Data Routing Control Register 2
DAI_EXTD_DAT3	Extended Serial Data Routing Control Register 3
DAI_EXTD_DAT4	Extended Serial Data Routing Control Register 4
DAI_EXTD_DAT5	Extended Serial Data Routing Control Register 5
DAI_EXTD_DAT6	Extended Serial Data Routing Control Register 6
DAI_EXTD_FS0	Extended Frame Sync Routing Control Register 0
DAI_EXTD_FS1	Extended Frame Sync Routing Control Register 1
DAI_EXTD_FS2	Extended Frame Sync Routing Control Register 2
DAI_EXTD_FS4	Extended Frame Sync Routing Control Register 4
DAI_EXTD_MISC0	Extended Miscellaneous Control Register 0
DAI_EXTD_MISC1	Extended Miscellaneous Control Register 1
DAI_EXTD_MISC2	Extended Miscellaneous Control Register 2
DAI_EXTD_PBEN0	Extended Pin Buffer Enable Register 0
DAI_EXTD_PBEN1	Extended Pin Buffer Enable Register 1
DAI_EXTD_PBEN2	Extended Pin Buffer Enable Register 2

Table 31-1: ADSP-SC59x DAI Register List (Continued)

Name	Description
DAI_EXTD_PBEN3	Extended Pin Buffer Enable Register 3
DAI_EXTD_PIN0	Extended Pin Buffer Assignment Register 0
DAI_EXTD_PIN1	Extended Pin Buffer Assignment Register 1
DAI_EXTD_PIN2	Extended Pin Buffer Assignment Register 2
DAI_EXTD_PIN3	Extended Pin Buffer Assignment Register 3
DAI_EXTD_PIN4	Extended Pin Buffer Assignment Register 4
DAI_FS0	Frame Sync Routing Control Register 0
DAI_FS1	Frame Sync Routing Control Register 1
DAI_FS2	Frame Sync Routing Control Register 2
DAI_FS4	Frame Sync Routing Control Register 4
DAI_GBL_INT_EN	Global SPORT Interrupt Grouping Register
DAI_GBL_PCG_EN	Global PCG Enable Control Register
DAI_GBL_SP_EN	Global SPORT Enable Register
DAI_IMSK_FE	Falling-Edge Interrupt Mask Register
DAI_IMSK_PRI	Core Interrupt Priority Assignment Register
DAI_IMSK_RE	Rising-Edge Interrupt Mask Register
DAI_IRPTL_H	High Priority Interrupt Latch Register
DAI_IRPTL_HS	Shadow High Priority Interrupt Latch Register
DAI_IRPTL_L	Low Priority Interrupt Latch Register
DAI_IRPTL_LS	Shadow Low Priority Interrupt Latch Register
DAI_MISC0	Miscellaneous Control Register 0
DAI_MISC1	Miscellaneous Control Register 1
DAI_MISC2	Miscellaneous Control Register 1
DAI_PBEN0	Pin Buffer Enable Register 0
DAI_PBEN1	Pin Buffer Enable Register 1
DAI_PBEN2	Pin Buffer Enable Register 2
DAI_PBEN3	Pin Buffer Enable Register 3
DAI_PIN0	Pin Buffer Assignment Register 0
DAI_PIN1	Pin Buffer Assignment Register 1
DAI_PIN2	Pin Buffer Assignment Register 2
DAI_PIN3	Pin Buffer Assignment Register 3

Table 31-1: ADSP-SC59x DAI Register List (Continued)

Name	Description
DAI_PIN4	Pin Buffer Assignment Register 4
DAI_PIN_STAT	Pin Status Register

ADSP-SC59x DAI Interrupt List

Table 31-2: ADSP-SC59x DAI Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
44	DAI0_IRQH	DAI0 High priority Interrupt	None	
45	DAI1_IRQH	DAI1 High priority Interrupt	None	
46	DAI0_IRQL	DAI0 Low Priority Interrupt	None	
47	DAI1_IRQL	DAI1 Low Priority Interrupt	None	
113	DAI0_GBL_SPORT_INT0	DAI0 Global SPORT Interrupt 0	None	
114	DAI0_GBL_SPORT_INT1	DAI0 Global SPORT Interrupt 1	None	
115	DAI1_GBL_SPORT_INT0	DAI1 Global SPORT Interrupt 0	None	
116	DAI1_GBL_SPORT_INT1	DAI1 Global SPORT Interrupt 1	None	

ADSP-SC59x DAI Trigger List

Table 31-3: ADSP-SC59x DAI Trigger List Generators

Trigger ID	Name	Description	Sensitivity
101	DAI0_GBL_SPORT_TRG_O0	DAI0 SPORT GROUP0 Trigger Output	None
102	DAI0_GBL_SPORT_TRG_O1	DAI0 SPORT GROUP1 Trigger Output	None
103	DAI1_GBL_SPORT_TRG_O0	DAI1 SPORT GROUP2 Trigger Output	None
104	DAI1_GBL_SPORT_TRG_O1	DAI1 SPORT Group3 Trigger Output	None

Table 31-4: ADSP-SC59x DAI Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
76	DAI0_GBL_SPORT_TRG_I0	DAI0 SPORT GROUP0 Trigger Input	Pulse

Table 31-4: ADSP-SC59x DAI Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
77	DAI0_GBL_SPORT_TRG_I1	DAI0 SPORT GROUP1 Trigger Input	Pulse
78	DAI1_GBL_SPORT_TRG_I0	DAI1 SPORT GROUP2 Trigger Input	Pulse
79	DAI1_GBL_SPORT_TRG_I1	DAI1 SPORT GROUP3 Trigger Input	Pulse

DAI Block Diagram

The ***DAI Functional Block Diagram*** and the ***Digital Audio Interconnect Unit*** figures show the functional blocks within the DAI and the unit connections to the peripherals.

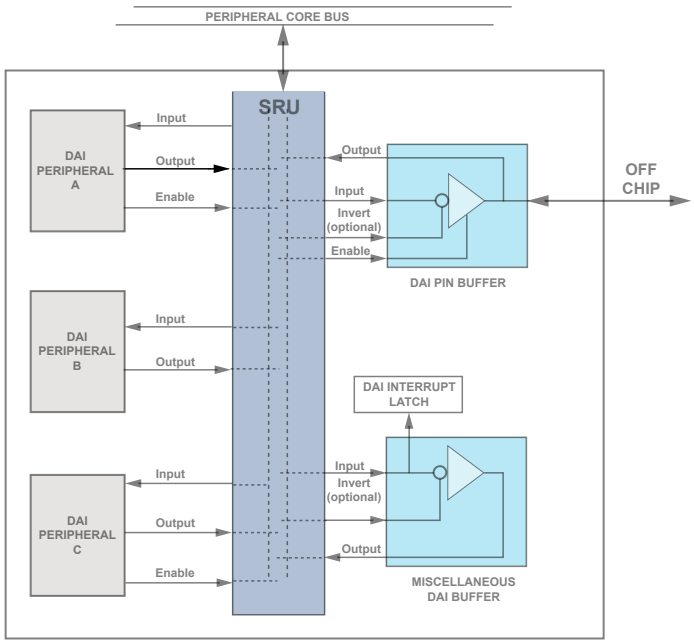


Figure 31-1: DAI Functional Block Diagram

DAI Signal Naming Conventions

The peripherals associated with the DAI do not have any dedicated I/O pins for off-chip communication. Instead, the I/O pin is only accessible in the chip internally and is known as an *internal node*. Every internal node of a DAI peripheral (input or output) is given a unique mnemonic. The convention is to begin the name with an identifier for the peripheral that the signal is coming to or from, followed by the function of the signal.

A number is included if the DAI contains more than one peripheral type (for example, serial ports), or if the peripheral has more than one signal that performs this function (for example, serial ports). The mnemonic always ends with *_I* if the signal is an input, or with *_O* if the signal is an output. An example is shown in the ***Example DAI Mnemonics*** figure.

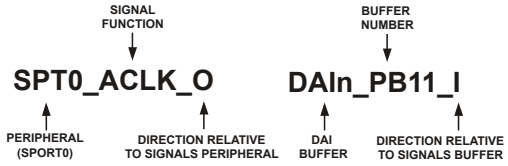


Figure 31-2: Example DAI Mnemonics

I/O Pin Buffers

Within the context of the SRU, physical connections to the DAI pins are replaced by a logical interface known as a *pin buffer*. This three terminal active device is capable of sourcing or sinking output current when its driver is enabled, and passing external input signals when disabled. Each pin has an input, an output, and an enable as shown in the ***Pin Buffer Example*** figure. The inputs and the outputs are defined with respect to the pin, like a peripheral device. This naming convention is consistent with the SRU naming convention.

Pin Buffer Signals

The pin buffer is based on three signals shown in the ***Pin Buffer Example*** figure and described in the following sections.

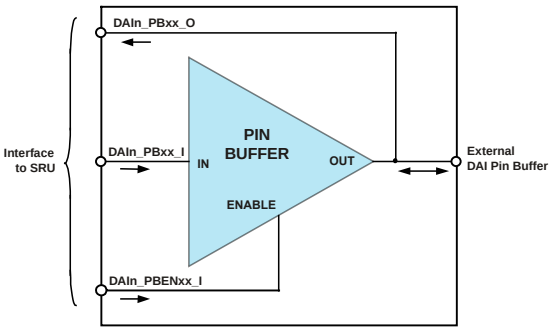


Figure 31-3: Pin Buffer Example

Pin Buffer Input Signal

A pin buffer input (`DAIn_PBxx_I`) is driven as an output from the processor when the pin buffer enable is set (=1). Each physical pin (connected to a bonded pad) can be connected through the SRU to any of the outputs of the DAI peripherals, based on the bit field values. The SRU can also be used to route signals that control the pins in other ways. Many signals can be configured for use as control signals.

Pin Buffer Enable Signal

When a pin buffer enable (`DAIn_PBENxx_I`) is set (=1), the signal present at the corresponding pin buffer input (`DAIn_PBxx_I`) is driven off-chip as an output. When a pin buffer enable is cleared (=0), the signal present at the corresponding pin buffer input is ignored. The pin enable control registers activate the drive buffer for each of the DAI pins. When the pins are not enabled (driven), they can be used as inputs. There are two options to control the pin buffer enable signal; setting the level high for a static solution or connecting the dedicated peripheral's pin buffer output enable signal to its pin buffer, which automatically enables the pin buffer.

Loopback Pin Buffers

Loopback pin buffers provide the ability to route DAI clock signals output from the processor back into the DAI block. There are four loopback pin buffers per DAI block (3/4/5/6) that connect to the external DAI pins with corresponding numbers. The input to a loopback buffer is the same as the input to the DAI pin buffer it corresponds to. Configuring a DAI pin buffer as the output causes the mux output to be from the loopback pin buffer rather than the DAI pin buffer.

For example, when the SPORT0A clock is an output that transmits from DAI Pin 4, then SPORT0A transmits the clock and receives data. In this case, the received data is sampled with respect to the transmit clock that is routed back to the SPORT0A through the loopback pin buffer.

NOTE: For the loopback pin buffers to be applicable, configure the DAI pins when the clock is output and FS/data are inputs.

The *Loopback Pin Buffer* figure shows the connections for the loopback pin buffers in the DAI sub-system.

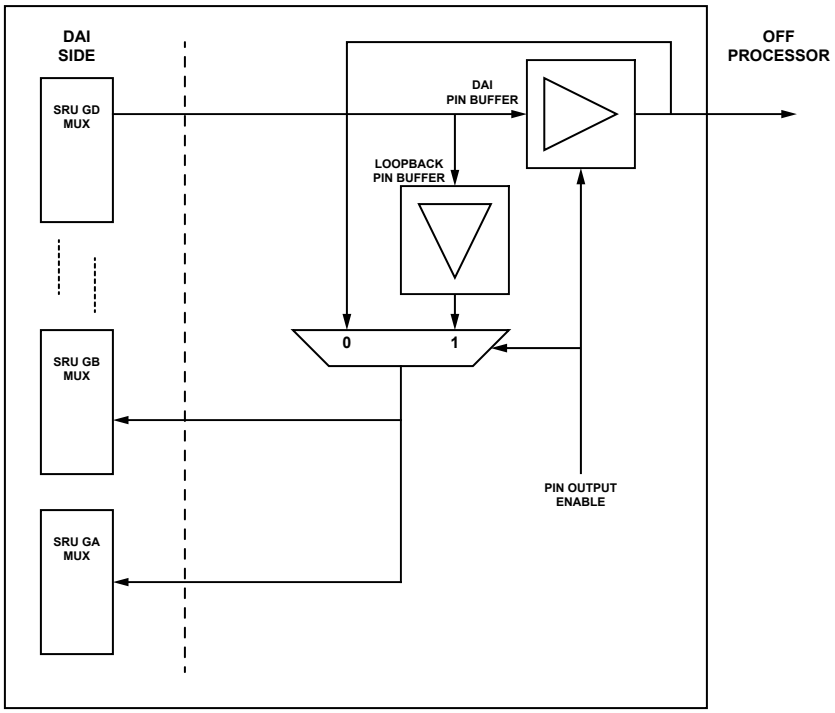


Figure 31-4: Loopback Pin Buffer

Pin Buffer Functions

Pin buffers can be configured as inputs or outputs as described in the following sections.

Pin Buffers as Signal Input

When the DAI pin is used only as an input, connect the corresponding pin buffer enable to logic low as shown in the ***Pin Buffer as Input*** figure. This configuration disables the buffer amplifier and allows an off-chip source to drive the value present on the DAI pin and at the pin buffer output. When the pin buffer enable (for example,

DAI_PBEN0.PB01) is cleared (= 0), the pin buffer output (DAIn_PBxx_O) is the signal driven onto the DAI pin by an external source, and the pin buffer input signal (DAIn_PBxx_I) is not used.

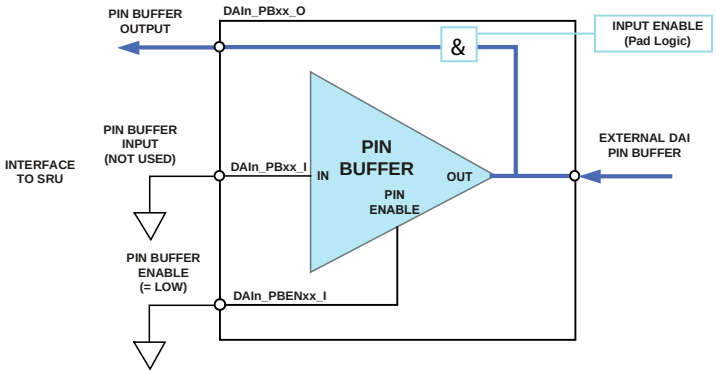


Figure 31-5: Pin Buffer as Input

NOTE: Whether programmed as input or output, a DAI buffer input always routes the same signal to an output internally. DAI pins have programmable internal pull-up resistors. For more information, see the General-Purpose Ports (PORT) chapter for details.

Pin Buffers As Signal Output

In a typical embedded system, most pins are designated as either inputs or outputs when the circuit is designed, even though they can be used in either direction. Each of the DAI pins can be used as either an output or an input. Although the direction of a DAI pin is set simply by writing to a memory-mapped register, most often the direction of the pin is dictated by the designated use of that pin.

When the DAI pin is used only as an output, connect the corresponding pin buffer enable to logic high as shown in the ***Pin Buffer as Output*** figure. This configuration enables the buffer amplifier to operate as a current source and to drive the value present at the pin buffer input onto the DAI pin and off-chip. When the pin buffer enable bits are set (in the DAIn_PBxx_I registers) (=1), the pin buffer output (DAIn_PBxx_O) is the same signal as the pin buffer input (DAIn_PBxx_I), and this signal is driven as an output.

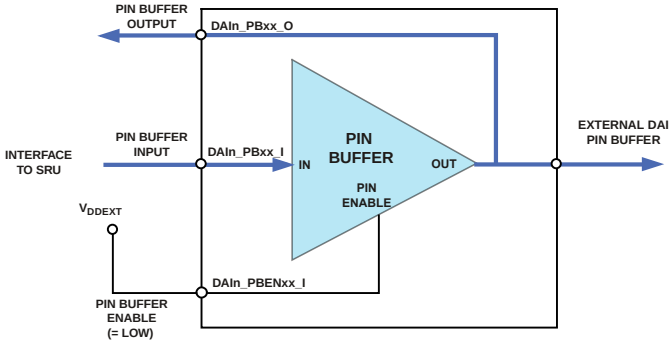


Figure 31-6: Pin Buffer as Output

DAI Pin Buffer Status

The signal levels on the DAI pins can be read with the DAI_PIN_STAT registers.

DAIn Peripherals

There are two categories of peripherals associated with the DAI units. These are described in the following sections.

Output Signals With Pin Buffer Enable Control

Many peripherals within the DAIn that have bidirectional pins generate a corresponding pin enable signal. Typically, the settings within the control registers of a peripheral determine if a bidirectional pin is an input or an output. The pin is then driven accordingly.

Though most peripherals acan operate bidirectionally, it is not required that all of the `_I` and `_O` signals of a peripheral be connected to the pin buffer. When the system design only uses a signal in one direction, it is simpler to connect the pin buffer accordingly.

NOTE: All available pin buffer output enables must be routed to their pin buffer input enable signals in cases where data streaming connections are used. This arrangement guarantees timing requirements.

NOTE: In some cases, it is necessary to use a peripheral's dedicated pin buffer enable signal instead of static levels. For example, SPORT TDM mode requires the SPORT's dedicated data pin buffer enable signal to be used for the SPORT's data pin to three-state the data pins on inactive channels.

Output Signals Without Pin Buffer Enable Control

Some peripherals have signal outputs without an automated pin buffer control enable signal. The operation of these peripherals is simplified. The routing to a DAIn pin buffer enable input requires a static high from the SRUn. To disable the pin buffer output, software must clear the pin buffer enable input.

Signal Routing Units (SRUs)

The following sections provide details specific to the SRUs.

Signal Routing Matrix by Groups

The SRU is similar to a set of patch bays, which contain a bank of inputs and a bank of outputs. For each input (destination), there is a set of permissible output (source) options. Outputs can feed to any number of inputs in parallel, but every input must be patched to exactly one valid output source. Together, the set of inputs and outputs are called a group. The signal's inputs and outputs that comprise each group all serve similar purposes. They are compatible such that almost any output-to-input patch makes functional sense. With the grouping, the multiplexing scheme becomes highly efficient since it does not make sense (for example) to route a frame sync signal to a data signal.

The SRU for the DAI contains six groups named A through F. Each group routes a unique set of signals with a specific purpose:

- Group A routes clock signals
- Group B routes serial data signals
- Group C routes frame sync signals

- Group D routes pin signals
- Group E routes miscellaneous signals
- Group F routes pin output enable signals

Together, the six groups of the SRU include all of the inputs and outputs of the DAI peripherals, a number of additional signals from the core, and all of the connections to the DAI pins.

NOTE: It is not possible to connect a signal in one group directly to a signal in a different group (analogous to wiring from one patch bay to another). However, group D (DAI) is largely devoted to routing in this vein.

DAI Group Routing

Each group has a unique encoding for its associated output signals and a set of configuration registers. For example, DAI group A is used to route clock signals. The memory-mapped group A registers, `DAI_CLK0` through `DAI_CLK5`, contain bit fields corresponding to the clock inputs of various peripherals. The values written to these bit fields specify a signal source that is an output from another peripheral. All of the possible encodings represent sources that are clock signals (or at least could be clock signals in some systems). The ***Example DAI Group A Multiplexing (DAI_CLKx)*** diagrams the input signals that are controlled by the group A registers. All bit fields in the SRU configuration registers correspond to inputs. The value written to the bit field specifies the signal source. This value is also an output from some other component within the SRU.

The SRU is similar to a set of patch bays. Each bay routes a distinct set of outputs to compatible inputs. These connections are implemented as a set of memory-mapped registers with a bit field for each input. The outputs are implemented as a set of bit encodings. Conceptually, a patch cord is used to connect an output to an input. In the SRU, a bit pattern that is associated with a signal output (shown in the ***Example DAI Group A Multiplexing (DAI_CLKx)*** figure) is written to a bit field corresponding to a signal input.

The same encoding can be written to any number of bit fields in the same group. It is not possible to run out of patch points for an output signal.

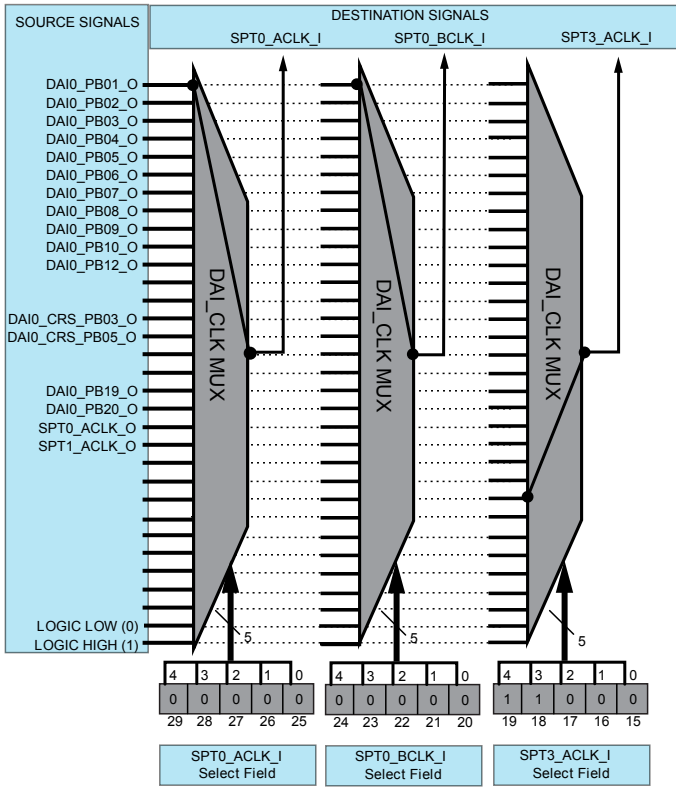


Figure 31-7: Example DAI Group A Multiplexing (DAIn_CLKx)

Just as group A routes clock signals, each of the other groups route a collection of compatible signals. Group B routes serial data streams while group C routes frame sync signals. All of the groups have an encoding that allow a signal to flow from a pin output to the input being specified by the bit field.

Group D routes signals to pins so that they may be driven off-chip (required to route a signal to the pin input). Group F routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input. The input of one pin can be patched to the output of another pin, allowing board-level routing under software control.

Rules for SRU Connections

There are two rules which apply to all routing:

1. One source (output node) can drive different destinations (input nodes).
2. One destination (input node) can only be assigned to one source (output node).

As an example from the ***Example DAI Group A Multiplexing (DAI_CLKx)*** figure:

- `DAI0_PIN02_O` is routed to `SPT0_BCLK_I`
- `SPT2_ACLK_O` is routed to `SPT3_ACLK_I`

NOTE: Inputs may only be connected to outputs.

Miscellaneous Buffers and Functions

The SRU group E provides miscellaneous buffers used for group interconnect.

DAI group E connections are slightly different from the others in that the inputs and outputs being routed vary considerably in function. This group routes control signals and provides a means of connecting signals between groups.

In the ***DAI MISCAx SRU Signal Connections*** table, the `DAIn_MISCAx_I` signals appear as inputs in group E (also connected to the DAI interrupt logic), but do not directly feed any peripheral. Rather, the `MISCAx_O` signals reappear as outputs in group F.

Table 31-5: DAI MISCAx SRU Signal Connections

MISCA Source	DAI Connection	MISCA Destination
	Group E	DAIn_MISCA5-0_I
DAIn_MISCA5-0_O	Group F	

Additional connections among groups provide a great amount of utility. Since the output groups F (DAI) dictate pin direction, these few signal paths enable a number of possible uses and connections for the DAI pins. Other examples include:

- A pin input can be patched to another pin's enable, allowing an off-chip signal to gate an output from the processor.
- Any of the DAI pins can be used as interrupt sources or general-purpose I/O (GPIO) signals.

In summary, the SRU enables many possible functional changes, both within the processor as well as externally. Used creatively, it allows system designers to radically change functionality at run time, and to potentially reuse circuit boards across many products.

DAI Routing Unit (DRU)

The DAI routing unit provides all possible connections across DAIs by interconnecting two DAI units through a multiplexer.

- For every input signal of an SRU, there is a control bit that selects either the original signal from the native DAI or its corresponding signal from the other DAI. For example, either SP0_ACLK_O from DAI0 or SP0_ACLK_O from DAI1 is selected and routed to the SP0_ACLK_O input of the SRU of DAI0 based on the programmed control bit.
- By default, all native signals are connected to their respective SRU (for example, SP0_ACLK_O from DAI0 is connected to SP0_ACLK_O input of the SRU of DAI0.

As a part of fully cross-connected DAI Routing unit, following source signals are added to the existing SRU's source group.

- All the source signals of Group A/B/C/D/E/F from the other DAI are added as source signals to the native group A/B/C/D/E/F's respectively.

The *DAI Routing Unit System* figure depicts the block level view of the DAI Routing Unit connecting DAI0 and DAI1 through a multiplexer.

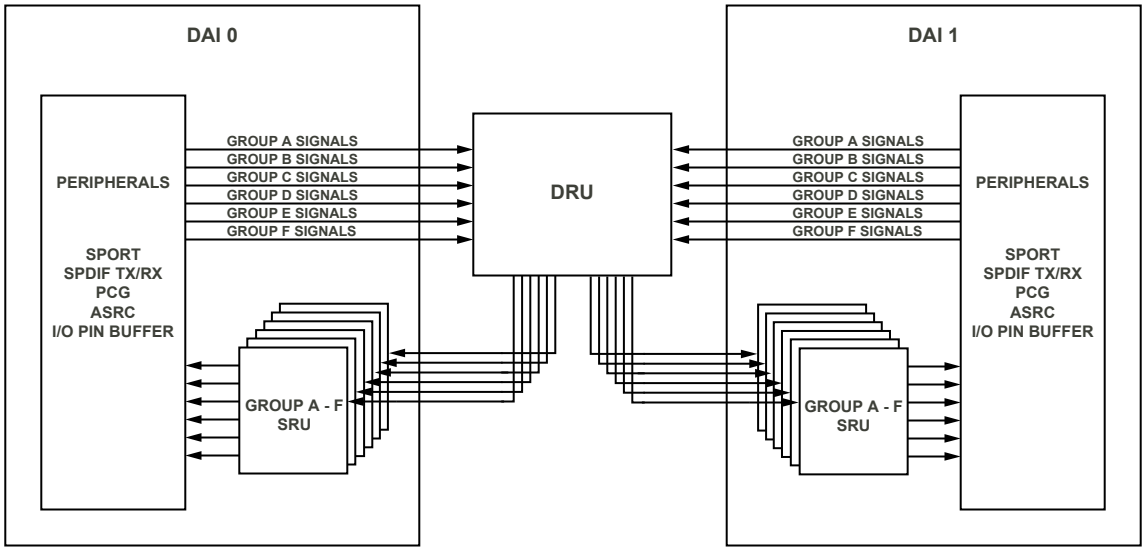


Figure 31-8: DAI Routing Unit System

DAI Routing Capabilities

This section describes the routing options to aid in designing a system using the DAI units. The DAI Default Routing section provides diagrams that show how that DAIs connect at default. The DAI Group tables provide the source signals and selections codes. Finally, the xxx provides information about configuring destinations.

The ***DAI Routing Capabilities*** tables provide an overview of the different routing capabilities for the DAI unit. For information on an individual peripherals routing, see the “SRU Programming” section of the specific peripheral chapter.

Table 31-6: DAI0 Routing Capabilities

Source Signals - Output (xxxx_O)		DAI0 Group	Destination Signals - Input (xxxx_I)
SPORT2B–0A (clocks)	DAI Pin Buffer20–1	A – clocks	SPORT3B–0A (clocks)
PCG A, B	Logic level high		SRC3–0 (clocks)
S/PDIF0 Rx (clock, TDM clock)	Logic level low		PCG A–B ext. clock, ext. sync
SPORT3B-3A,			S/PDIF0 Tx (clock, HF clock)
PCG E, F			PCG E, F
PCG A, B, E, F (Inverted)			PDM0 (clk, bclk)
PDM0			
All Group A source signals from other DAI			

Table 31-6: DAI0 Routing Capabilities (Continued)

Source Signals - Output (xxxx_O)		DAI0 Group	Destination Signals - Input (xxxx_I)
SPORT3B–0A (data) SRC3–0 (data, TDM data) S/PDIF0 Tx/Rx (data) PDM0 All Group B source signals from other DAI	DAI0 pin buffer 20–1 Logic level high Logic level low	B – data	SPORT3B–0A (data) SRC3–0 (data, TDM data) S/PDIF0 Tx/Rx (data) PDM0 (data0/1)
SPORT2B–0A (FS) PCG A, B (FS) S/PDIF0 Rx (FS) SPORT3A-3B, PCG E, F PCG A, B, E, F (Inverted) All Group C source signals from other DAI	DAI0 pin buffer 20–1 Logic level high Logic level low	C – frame sync	SPORT3B–0A (FS) SRC3–0 (FS) PDM0 (lrclk) TMR_ACI14
SPORT3B–0A (clock, FS, TDV, data) PCG A,B (clock, FS) S/PDIF0 Rx (clock, TDM clock, FS, data) S/PDIF0 Tx (data, block start) PCG E, F (clock, FS) PCG A, B, E, F (inverted-Clock, FS) PDM0 (clock, FS) All Group D source signals from other DAI	DAI0 pin buffer 20–1 Logic level high Logic level low	D – pin buffer inputs	DAI0 pin buffer 20–1 Options: DAI0 pin buffer 20–19 Polarity change
SPORT2B–0A (FS) PCG A (clock) PCG B (clock, FS) S/PDIF0 Tx (block start) SPORT3B-3A (FS) PCG E, F (FS) PCG A, B, E, F (completer trigger) All Group E source signals from other DAI	DAI0 pin buffer 20–1 Logic level high Logic level low	E – miscellaneous signals	***DAI Interrupt 31-22*** MISCA5–0 Options: MISCA5–4 Polarity change PCG A, B, E, F (requester trigger)
SPORT3B–0A (clock OE, FS OE, data OE, TDV) MISCA5–0 PDM0 (clock OE, FS OE) All Group F source signals from other DAI	Logic level high Logic level low	F – pin buffer enable	DAI Pin Buffer Enable 20-1

Table 31-7: DAI1 Routing Capabilities

Source Signals - Output (xxxx_O)		DAI1 Group	Destination Signals - Input (xxxx_I)
SPORT6B–4A (clocks) PCG C, D (clocks) S/PDIF1 Rx (clock, TDM clock) SPORT7B-7A, PCG G, H PCG C, D, G, H (Inverted) PDM1 All Group A source signals from other DA	DAI1 Pin Buffer 20-1 Logic level high Logic level low	A – clocks	SPORT7B–4A (clocks) SRC4–7 (clocks) PCG C–D Ext. clock, Ext. sync S/PDIF1 Tx (clock, HF clock) PCG G, H PDM1 (clk, bclk)
SPORT7B–4A (data A, B) SRC4–7 (data, TDM data) S/PDIF1 Tx/Rx (data) PDM1 All Group B source signals from other DA	DAI1 Pin Buffer 20–1 Logic level high Logic level low	B – data	SPORT7B–4A (data) SRC4–7 (data, TDM data) S/PDIF1 Tx/Rx (data) PDM1 (data0/1)
SPORT6B–4A (FS) PCG C, D (FS) S/PDIF1 Rx (FS) SPORT7B-7A, PCG G, H PCG C, D, G, H (Inverted) All Group C source signals from other DA	DAI1 Pin Buffer 20–1 Logic level high Logic level low	C – frame sync	SPORT7B–4A (FS) SRC4–7 (FS) PDM1 (lrclk) TMR_ACI15
SPORT7B–4A (clock, FS, TDV, data) PCG C, D (clock, FS) S/PDIF1 Rx (clock, TDM clock, FS, data) S/PDIF1 Tx (data, block start) PCG G, H (clock, FS) PCG C, D, G, H (inverted-Clock, FS) PDM1 (clock, FS) All Group D source signals from other DAI	DAI1 Pin Buffer 20-1 Logic level high Logic level low	D – pin buffer inputs	DAI1 Pin Buffer 20–1 Options: DAI1 Pin Buffer 20–19 Polarity Change

Table 31-7: DAI1 Routing Capabilities (Continued)

Source Signals - Output (xxxx_O)		DAI1 Group	Destination Signals - Input (xxxx_I)
SPORT6B–4A (FS) PCG C (clock) PCG D (clock, FS) S/PDIF1 Tx (block start) SPORT7B-7A (FS) PCG C, D (FS) PCG C, D, G, H (completer trigger) All Group E source signals from other DAI	DAI1 Pin Buffer 20–1 Logic level high Logic level low	E – miscellaneous signals	***DAI Interrupt 31-22*** MISCA5–0 Options: MISCA5–4 Polarity Change PCG C, D, G, H (requester trigger)
SPORT7B–4A (clock, FS, data, TDV) PDM1 (clock OE, FS OE) All Group F source signals from other DAI.	Logic level high Logic level low	F – pin buffer enable	DAI1 Pin Buffer Enable 20–1

DAI Default Routing

When the processor comes out of reset, the SPORT junctions are bidirectional to the DAI pin buffers. This configuration allows systems to use the SPORTs as either requester or completer (without changing the routing scheme). Therefore, programs only need to use the SPORT control register settings to configure requester or completer operations. Note that all DAI inputs which are not routed by default are tied to signal low.

NOTE: All DAI input buffers which are not routed by default are driven low and all DAI pin buffer enable signals are driven low.

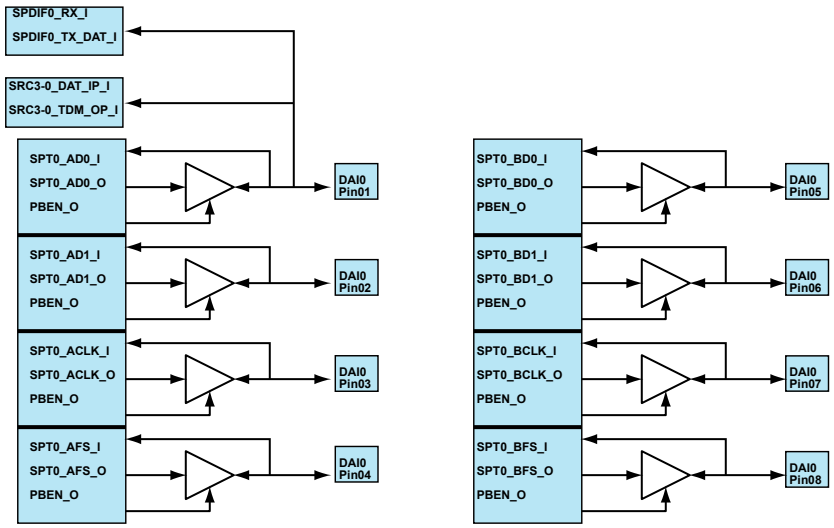


Figure 31-9: DAI0 Default Routing Pins 01-08

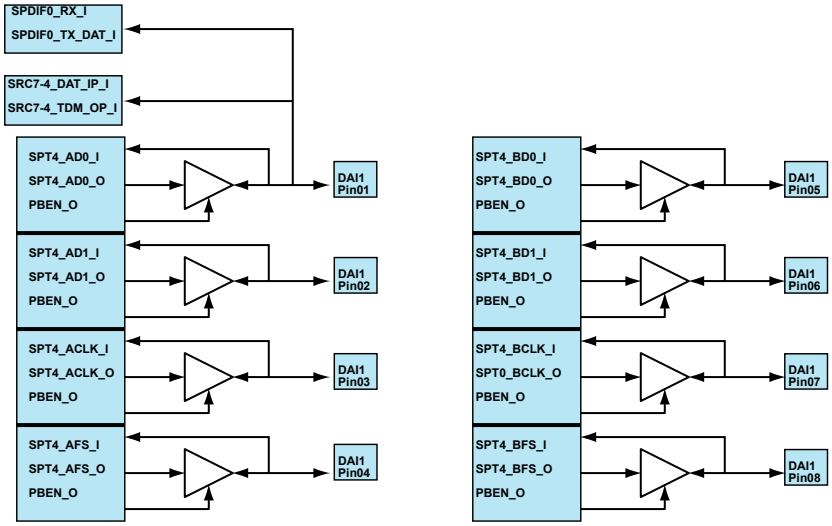


Figure 31-10: DAI1 Default Routing Pins 01-08

Unused DAI Connections

The SRUs have a default general-purpose routing scheme which can be modified to accommodate any number of different system designs. Regardless of the system design, it is good practice to tie all unused inputs to a high or low level to reduce dynamic power consumption.

DAI Operating Modes

Some buffers allow polarity changes, described as follows.

DAI Pin Buffer Polarity

As shown in the ***Pin Buffer Polarity*** figure, the DAI pin buffer 20-19 can change the polarity of the input signal if the corresponding control bits (`DAI_PIN4.INV20`, `DAI_PIN4.INV19`) are set. These bits can be set during run time and the buffer should not loopback to itself.

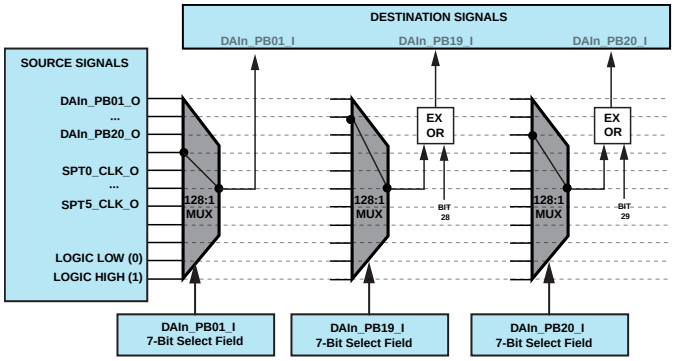


Figure 31-11: Pin Buffer Polarity

DAI Miscellaneous Buffer Polarity

As shown in the ***Pin Buffer Polarity*** figure, the A5-4 miscellaneous buffers can change the polarity of the input signal if the corresponding control bits (`DAI_MISC1.IN5`, `DAI_MISC1.IN4`) are set. Both buffers are not connected to the DAI interrupt latch register. These bits can be set during run time.

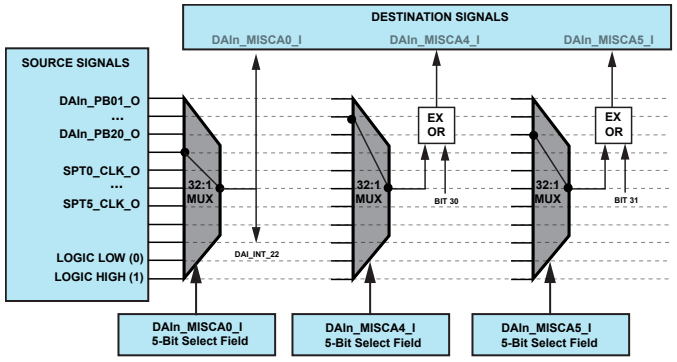


Figure 31-12: Miscellaneous Buffer Polarity

DAI System Interrupt Controller (SIC)

The DAI module incorporates a system interrupt controller (SIC) which is connected to the SEC as seen in the ***DAI System Interrupt Controller*** figure.

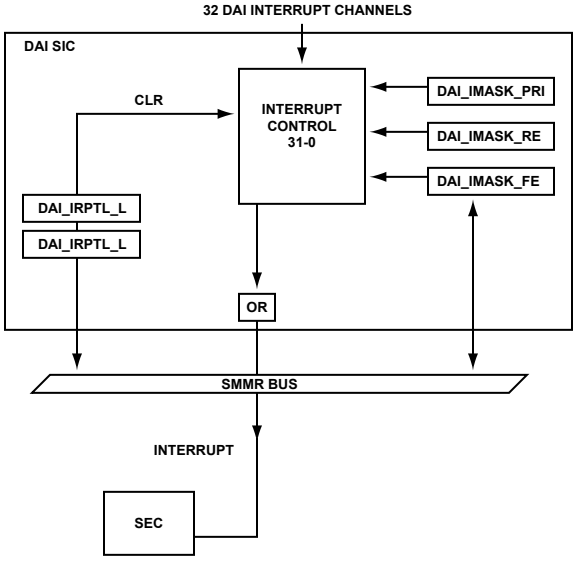


Figure 31-13: DAI System Interrupt Controller

The DAI has its own system interrupt controllers that indicate to the core when DAI audio peripheral-related events have occurred. Since audio events occur infrequently relative to the SHARC core, the DAI interrupt controller reduces all of its interrupts onto two interrupt signals within the core's primary interrupt systems. One interrupt is mapped with DAI low priority. The second interrupt is mapped with DAI high priority. This configuration allows programs to broadly indicate priority. In this way, the DAI SIC provides 32 independently configurable sources or

channels. The output bus interrupt signals are logically OR'ed into one interrupt line and fed to the interrupt controller logic of the core.

Three registers are used to configure the DAI interrupt controller. Each of the 32 interrupt sources can be independently configured to trigger on a rising edge, falling edge, both edges, or neither edge of an incoming signal. All DAI interrupt control registers are memory-mapped registers and are accessed through the peripheral bus.

Interrupt Sources

The DAI's five peripheral sources are multiplexed into 32 interrupt sources and are labeled DAI_INT31-0 (***DAI Interrupt Sources*** table).

NOTE: There are two naming conventions. The DAI interrupt controller register bits are labeled DAI_31-0_INT. Their corresponding SRU routing signals are labeled DAI_INT_31-0_I.

Table 31-8: DAI Interrupt Sources

Interrupt Source	Description	Signal Response
DAI_INT2–0, DAI_INT4	S/PDIF RX, 4 channels	Event
DAI_INT2–0, DAI_INT4	S/PDIF RX, 4 channels	Waveform
DAI_INT21–18	ASRC, 4 channels	
DAI_INT31–22	Miscellaneous, S/PDIF TX, 9 channels	

Interrupt Latch Priority Option

The `DAI_IMSK_PRI` register specifies the priority for the DAI interrupt channels. DAI system interrupt controller has a pair of interrupt latch registers, `DAI_IRPTL_H` and `DAI_IRPTL_L`. The configuration of the `DAI_IMSK_PRI` register also determines the interrupt latch mapping for a particular DAI interrupt. When a DAI interrupt is configured as low priority (`DAI_IMSK_PRI` bit cleared, default setting), interrupts are mapped to the DAI_INTR_IRQL signal and when an interrupt occurs, the corresponding bit of the `DAI_IRPTL_L` register is set. When a DAI interrupt is configured as high priority (`DAI_IMSK_PRI` bit set), interrupts are mapped to the DAI_INTR_IRQH signal and the interrupt is latched to the `DAI_IRPTL_H` register. The low priority DAI interrupt (INTR_DAI_IRQL) and high priority DAI interrupt (INTR_DAI_IRQH) are connected to the SEC .

Interrupt Mask for Waveforms

The `DAI_IMSK_RE` and `DAI_IMSK_FE` registers allow programs to mask or unmask interrupts for specific edges of a signal mapped to the channel. It can be configured for rising edges, falling edges, both rising and falling edges, or neither rising nor falling edges by masking them separately. Signals from the SRU can be used to generate interrupts. For example, when the `DAI_IMSK_FE.MISCINT9` bit is set to one, any falling edge signals from the external channel generate an interrupt and the interrupt latch is set.

Interrupt Mask for Events

The system interrupt controller needs information about a peripheral's interrupt sources that correspond to event signals (see the ***DAI Interrupt Sources*** table). As a result, the rising edge is used as an interrupt source only. For DAI peripherals marked as events, programs may unmask an interrupt source on the rising edge only.

Shadow Interrupt Register

The DAI interrupt controller has shadow registers to simplify debug activities since these registers do not manipulate status control. Any read of the shadow registers (`DAI_IRPTL_HS`, `DAI_IRPTL_LS`), provides the same data as a read of the `DAI_IRPTL_H` and `DAI_IRPTL_L` registers. However, reads of the DAI shadow registers do not change the interrupt acknowledge status to the core interrupt controller.

Interrupt Service

The interrupt acknowledge operates differently when multiple channels are multiplexed into one interrupt output signal. When an interrupt from the DAI must be serviced, any of the two interrupt service routines (INTR_DAI_IRQL, INTR_DAI_IRQH) must query the SIC to determine the source(s). Sources can be any one or more of the DAI channels (DAI_INT31-0).

- When the `DAI_IRPTL_H` register is read, the high priority latched interrupts are cleared.
- When the `DAI_IRPTL_L` register is read, the low priority latched interrupts are cleared.

If an interrupt occurs in the same cycle as a latch register is cleared, the clear mechanism has lower priority and the new interrupt is registered.

Signal Routing Unit Effect Latency

After the DAIy registers are configured the effect latency is 2 *SCLK0* cycles minimum and 3 *SCLK0* cycles maximum.

DAI Programming Model

As discussed in the previous sections, the signal routing unit is controlled by writing values that provide a plug-in tool in CCES so that configuring the SRU is done graphically. Analog Devices offers macros that are included with the CrossCore or VisualDSP++ tools, greatly easing code development in the SRU.

There is a macro that has been created to connect peripherals used in a DAI configuration. This code can be used in both assembly and C code. See the INCLUDE file SRU.H. In practice the macro is provided and forms the style `SRU(source_O, destination_I)` for DAI0. Example appears as: `SRU(DAI0_PB12_O, SPT0_ACLK_I);`

Debug Features

The following section describes the feature that can be used to help in debugging the DAI.

Loopback Routing

The SPORT serial peripheral supports an internal loopback mode. If the loopback bit for each peripheral is enabled, it connects the transmitter with the receiver block internally (does not signal off-chip). The SRU can be used for this purpose. The ***Loopback Routing*** table describes the different possible routings based on the peripheral.

NOTE: The peripheral's loopback mode for debug is independent from both signal routing units.

Table 31-9: Loopback Routing

Peripheral	Loopback Mode	SRU1–0 Internal Routing for Loopback	SRU1–0 External Routing for Loopback
SPORT	Yes	SPTx_xx_O → SPTx_xx_I	SPTx_xx_O → DAIn_PBxx_I DAIn_PBxx_O → SPTx_xx_I
S/PDIF Tx/Rx	No	SPDIFn_TX_O → SPDIFn_RX_I	SPDIFn_TX_O → DAI_PBxx_I DAI_PBxx_O → SPDIFn_RX_I
SRC	No	SRCx_DAT_OP_O → SRCx_DAT_IP_I	SRCx_DAT_OP_O → DAI_PBxx_I DAI_PBxx_O → SRCx_DAT_IP_I

DAI Sources Overview

The following tables provide information of the various DAI sources sorted by group.

Group A – Clock Routing Signals

The group A clock routing sources are listed in the following table.

Table 31-10: Group A – Clock Routing Sources

Extended Selection Code [1:0]		Selection Code [4:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	0	00000 (0x0)	DAI0_PB01_O	Pin Buffer 1
0	0	00001 (0x1)	DAI0_PB02_O	Pin Buffer 2
0	0	00010 (0x2)	DAI0_PB03_O	Pin Buffer 3
0	0	00011 (0x3)	DAI0_PB04_O	Pin Buffer 4
0	0	00100 (0x4)	DAI0_PB05_O	Pin Buffer 5
0	0	00101 (0x5)	DAI0_PB06_O	Pin Buffer 6
0	0	00110 (0x6)	DAI0_PB07_O	Pin Buffer 7
0	0	00111 (0x7)	DAI0_PB08_O	Pin Buffer 8
0	0	01000 (0x8)	DAI0_PB09_O	Pin Buffer 9
0	0	01001 (0x9)	DAI0_PB10_O	Pin Buffer 10
0	0	01010 (0xA)	DAI0_PB11_O	Pin Buffer 11
0	0	01011 (0xB)	DAI0_PB12_O	Pin Buffer 12
0	0	01100 (0xC)	DAI0_PB13_O	Pin Buffer 13
0	0	01101 (0xD)	DAI0_PB14_O	Pin Buffer 14
0	0	01110 (0xE)	DAI0_PB15_O	Pin Buffer 15
0	0	01111 (0xF)	DAI0_PB16_O	Pin Buffer 16
0	0	10000 (0x10)	DAI0_PB17_O	Pin Buffer 17
0	0	10001 (0x11)	DAI0_PB18_O	Pin Buffer 18
0	0	10010 (0x12)	DAI0_PB19_O	Pin Buffer 19
0	0	10011 (0x13)	DAI0_PB20_O	Pin Buffer 20
0	0	10100 (0x14)	SPT0_ACLK_O	SPORT 0 Clock A
0	0	10101 (0x15)	SPT0_BCLK_O	SPORT 0 Clock B
0	0	10110 (0x16)	SPT1_ACLK_O	SPORT 1 Clock A
0	0	10111 (0x17)	SPT1_BCLK_O	SPORT 1 Clock B
0	0	11000 (0x18)	SPT2_ACLK_O	SPORT 2 Clock A
0	0	11001 (0x19)	SPT2_BCLK_O	SPORT 2 Clock B

Table 31-10: Group A – Clock Routing Sources (Continued)

Extended Selection Code [1:0]		Selection Code [4:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	0	11010 (0x1A)	SPDIF0_RX_CLK_O	SPDIF 0 Receive Clock Output
0	0	11011 (0x1B)	SPDIF0_RX_TDMCLK_O	SPDIF 0 Receive TDM Clock Output
0	0	11100 (0x1C)	PCG_CLKA_O	Precision Clock A Output
0	0	11101 (0x1D)	PCG_CLKB_O	Precision Clock B Output
0	0	11110 (0x1E)	LOW	Logic Level Low (0)
0	0	11111 (0x1F)	HIGH	Logic Level High (1)
0	1	00000 (0x0)	SPT3_ACLK_O	SPORT 3 clock A
0	1	00001 (0x1)	SPT3_BCLK_O	SPORT 3 clock B
0	1	00010 (0x2)	PCG_CLKE_O	Precision Clock E output
0	1	00011 (0x3)	PCG_CLKF_O	Precision Clock F output
0	1	00100 (0x4)	PCG_CLKA_INV_O	Inverted Precision Clock A output
0	1	00101 (0x5)	PCG_CLKB_INV_O	Inverted Precision Clock B output
0	1	00110 (0x6)	PCG_CLKE_INV_O	Inverted Precision Clock E output
0	1	00111 (0x7)	PCG_CLKF_INV_O	Inverted Precision Clock F output
0	1	01000 (0x8)	PDM0_CLK0_O	PDM0 Clock0 output
0	1	01001 (0x9) -11111 (0x1F)	Reserved	
1	0	00000 (0x0)	DAI1_PB01_O	Pin Buffer 1 (from other DAI)
1	0	00001(0x1)	DAI1_PB02_O	Pin Buffer 2 (from other DAI)
1	0	00010 (0x2)	DAI1_PB03_O	Pin Buffer 3 (from other DAI). This pin buffer is used for sharing clocks between DAI modules. DAI0_PIN03 and DAI1_PIN03 can be used to share clock signal across DAI. When DAI0_PIN03 is configured as input clock pin then this clock signal is available in Group A of both DAIs. This source signal is DAI0_CRS_PB03_O in group A of DAI 0 and DAI1_CRS_PB03_O in group A of DAI1.
1	0	00011 (0x3)	DAI1_PB04_O	Pin Buffer 4 (from other DAI)

Table 31-10: Group A – Clock Routing Sources (Continued)

Extended Selection Code [1:0]		Selection Code [4:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
1	0	00100 (0x4)	DAI1_PB05_O	Pin Buffer 5 (from other DAI). This pin buffer is used for sharing clocks between DAI modules. DAI0_PIN05 and DAI1_PIN05 can be used to share clock signal across DAI. When DAI0_PIN05 is configured as input clock pin then this clock signal is available in Group A of both DAIs. This source signal is DAI0_CRS_PB05_O in group A of DAI 0 and DAI1_CRS_PB05_O in group A of DAI1.
1	0	00101 (0x5)	DAI1_PB06_O	Pin Buffer 6 (from other DAI)
1	0	00110 (0x6)	DAI1_PB07_O	Pin Buffer 7 (from other DAI)
1	0	00111 (0x7)	DAI1_PB08_O	Pin Buffer 8 (from other DAI)
1	0	01000 (0x8)	DAI1_PB09_O	Pin Buffer 9 (from other DAI)
1	0	01001 (0x9)	DAI1_PB10_O	Pin Buffer 10 (from other DAI)
1	0	01010 (0xA)	DAI1_PB11_O	Pin Buffer 11 (from other DAI)
1	0	01011 (0xB)	DAI1_PB12_O	Pin Buffer 12 (from other DAI)
1	0	01100 (0xC)	DAI1_PB13_O	Pin Buffer 13 (from other DAI)
1	0	01101 (0xD)	DAI1_PB14_O	Pin Buffer 14 (from other DAI)
1	0	01110 (0xE)	DAI1_PB15_O	Pin Buffer 15 (from other DAI)
1	0	01111 (0xF)	DAI1_PB16_O	Pin Buffer 16 (from other DAI)
1	0	10000 (0x10)	DAI1_PB17_O	Pin Buffer 17 (from other DAI)
1	0	10001 (0x11)	DAI1_PB18_O	Pin Buffer 18 (from other DAI)
1	0	10010 (0x12)	DAI1_PB19_O	Pin Buffer 19 (from other DAI)
1	0	10011 (0x13)	DAI1_PB20_O	Pin Buffer 20 (from other DAI)
1	0	10100 (0x14)	SPT4_ACLK_O	SPORT 4 Clock A (from other DAI)
1	0	10101 (0x15)	SPT4_BCLK_O	SPORT 4 Clock B (from other DAI)
1	0	10110 (0x16)	SPT5_ACLK_O	SPORT 5 Clock A (from other DAI)
1	0	10111 (0x17)	SPT5_BCLK_O	SPORT 5 Clock B (from other DAI)
1	0	11000 (0x18)	SPT6_ACLK_O	SPORT 6 Clock A (from other DAI)
1	0	11001 (0x19)	SPT6_BCLK_O	SPORT 6 Clock B (from other DAI)

Table 31-10: Group A – Clock Routing Sources (Continued)

Extended Selection Code [1:0]		Selection Code [4:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	0	11010 (0x1A)	SPDIF1_RX_CLK_O	SPDIF 0 Receive Clock Output (from other DAI)
1	0	11011 (0x1B)	SPDIF1_RX_TDMCLK_O	SPDIF 0 Receive TDM Clock Output (from other DAI)
1	0	11100 (0x1C)	PCG_CLKC_O	Precision Clock C Output (from other DAI)
1	0	11101 (0x1D)	PCG_CLKD_O	Precision Clock D Output (from other DAI)
1	0	11110 (0x1E)	Reserved	
1	0	11111 (0x1F)	Reserved	
1	1	00000 (0x0)	SPT7_ACLK_O	SPORT 3 clock A (from other DAI)
1	1	00001 (0x1)	SPT7_BCLK_O	SPORT 3 clock B (from other DAI)
1	1	00010 (0x2)	PCG_CLKG_O	Precision Clock G output (from other DAI)
1	1	00011 (0x3)	PCG_CLKH_O	Precision Clock H output (from other DAI)
1	1	00100 (0x4)	PCG_CLKC_INV_0	Inverted Precision Clock C output (from other DAI)
1	1	00101 (0x5)	PCG_CLKD_INV_0	Inverted Precision Clock D output (from other DAI)
1	1	00110 (0x6)	PCG_CLKG_INV_0	Inverted Precision Clock G output (from other DAI)
1	1	00111 (0x7)	PCG_CLKH_INV_0	Inverted Precision Clock H output (from other DAI)
1	1	01000 (0x8)	PDM1_CLK0_O	PDM1 Clock0 Output (from other DAI)
1	1	01001 (0x9) -11111 (0x1F)	Reserved	

Group B – Serial Data Source Signals

The group B data sources are listed in the following table. The group B data destinations are configured using DAI_PBEN0 through DAI_PBEN3.

Table 31-11: Group B – Serial Data Signals

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	Reserved	000000 (0x0)	DAI0_PB01_O	Pin Buffer 1
0	Reserved	000001 (0x1)	DAI0_PB02_O	Pin Buffer 2
0	Reserved	000010 (0x2)	DAI0_PB03_O	Pin Buffer 3
0	Reserved	000011 (0x3)	DAI0_PB04_O	Pin Buffer 4
0	Reserved	000100 (0x4)	DAI0_PB05_O	Pin Buffer 5
0	Reserved	000101 (0x5)	DAI0_PB06_O	Pin Buffer 6
0	Reserved	000110 (0x6)	DAI0_PB07_O	Pin Buffer 7
0	Reserved	000111 (0x7)	DAI0_PB08_O	Pin Buffer 8
0	Reserved	001000 (0x8)	DAI0_PB09_O	Pin Buffer 9
0	Reserved	001001 (0x9)	DAI0_PB10_O	Pin Buffer 10
0	Reserved	001010 (0xA)	DAI0_PB11_O	Pin Buffer 11
0	Reserved	001011 (0xB)	DAI0_PB12_O	Pin Buffer 12
0	Reserved	001100 (0xC)	DAI0_PB13_O	Pin Buffer 13
0	Reserved	001101 (0xD)	DAI0_PB14_O	Pin Buffer 14
0	Reserved	001110 (0xE)	DAI0_PB15_O	Pin Buffer 15
0	Reserved	001111 (0xF)	DAI0_PB16_O	Pin Buffer 16
0	Reserved	010000 (0x10)	DAI0_PB17_O	Pin Buffer 17
0	Reserved	010001 (0x11)	DAI0_PB18_O	Pin Buffer 18
0	Reserved	010010 (0x12)	DAI0_PB19_O	Pin Buffer 19
0	Reserved	010011 (0x13)	DAI0_PB20_O	Pin Buffer 20
0	Reserved	010100 (0x14)	SPT0_AD0_O	SPORT 0 Data AD0
0	Reserved	010101 (0x15)	SPT0_AD1_O	SPORT 0 Data AD1
0	Reserved	010110 (0x16)	SPT0_BD0_O	SPORT 0 Data BD0
0	Reserved	010111 (0x17)	SPT0_BD1_O	SPORT 0 Data BD1
0	Reserved	011000 (0x18)	SPT1_AD0_O	SPORT 1 Data AD0
0	Reserved	011001 (0x19)	SPT1_AD1_O	SPORT 1 Data AD1

Table 31-11: Group B – Serial Data Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	011010 (0x1A)	SPT1_BD0_O	SPORT 1 Data BD0
0	Reserved	011011 (0x1B)	SPT1_BD1_O	SPORT 1 Data BD1
0	Reserved	011100 (0x1C)	SPT2_AD0_O	SPORT 2 Data AD0
0	Reserved	011101 (0x1D)	SPT2_AD1_O	SPORT 2 Data AD1
0	Reserved	011110 (0x1E)	SPT2_BD0_O	SPORT 2 Data BD0
0	Reserved	011111 (0x1F)	SPT2_BD1_O	SPORT 2 Data BD1
0	Reserved	100000 (0x20)	SRC0_DAT_OP_O	SRC0 Data Out
0	Reserved	100001 (0x21)	SRC1_DAT_OP_O	SRC1 Data Out
0	Reserved	100010 (0x22)	SRC2_DAT_OP_O	SRC2 Data Out
0	Reserved	100011 (0x23)	SRC3_DAT_OP_O	SRC3 Data Out
0	Reserved	100100 (0x24)	SRC0_TDM_IP_O	SRC0 Data Out
0	Reserved	100101 (0x25)	SRC1_TDM_IP_O	SRC1 Data Out
0	Reserved	100110 (0x26)	SRC2_TDM_IP_O	SRC2 Data Out
0	Reserved	100111 (0x27)	SRC3_TDM_IP_O	SRC3 Data Out
0	Reserved	101000 (0x28)	SPDIF0_RX_DAT_O	SPDIF 0 RX Serial Data Out
0	Reserved	101001 (0x29)–101011 (0x2B)	Reserved	
0	Reserved	101100 (0x2C)	SPT3_AD0_O	SPORT 3 Data AD0
0	Reserved	101101 (0x2D)	SPT3_AD1_O	SPORT 3 Data AD1
0	Reserved	101110 (0x2E)	SPT3_BD0_O	SPORT 3 Data BD0
0	Reserved	101111 (0x2F)	SPT3_BD1_O	SPORT 3 Data BD1
0	Reserved	110000 (0x30)	SPDIF0_TX_O	SPDIF 0 TX Biphase Stream
0	Reserved	110001 (0x31)	Reserved	
0	Reserved	110010 (0x32)	Reserved	
0	Reserved	110011 (0x33)	PDM0_SDATA_O	PDM0 Serial Data Output
0	Reserved	110100 (0x34)–111101 (0x3D)	Reserved	
0	Reserved	111110 (0x3E)	LOW	Logic Level Low (0)
0	Reserved	111111 (0x3F)	HIGH	Logic Level High (1)
1	Reserved	000000 (0x0)	DAI1_PB01_O	Pin Buffer 1 (From other DAI)

Table 31-11: Group B – Serial Data Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	000001 (0x1)	DAI1_PB02_O	Pin Buffer 2 (From other DAI)
1	Reserved	000010 (0x2)	DAI1_PB03_O	Pin Buffer 3 (From other DAI)
1	Reserved	000011 (0x3)	DAI1_PB04_O	Pin Buffer 4 (From other DAI)
1	Reserved	000100 (0x4)	DAI1_PB05_O	Pin Buffer 5 (From other DAI)
1	Reserved	000101 (0x5)	DAI1_PB06_O	Pin Buffer 6 (From other DAI)
1	Reserved	000110 (0x6)	DAI1_PB07_O	Pin Buffer 7 (From other DAI)
1	Reserved	000111 (0x7)	DAI1_PB08_O	Pin Buffer 8 (From other DAI)
1	Reserved	001000 (0x8)	DAI1_PB09_O	Pin Buffer 9 (From other DAI)
1	Reserved	001001 (0x9)	DAI1_PB10_O	Pin Buffer 10 (From other DAI)
1	Reserved	001010 (0xA)	DAI1_PB11_O	Pin Buffer 11 (From other DAI)
1	Reserved	001011 (0xB)	DAI1_PB12_O	Pin Buffer 12 (From other DAI)
1	Reserved	001100 (0xC)	DAI1_PB13_O	Pin Buffer 13 (From other DAI)
1	Reserved	001101 (0xD)	DAI1_PB14_O	Pin Buffer 14 (From other DAI)
1	Reserved	001110 (0xE)	DAI1_PB15_O	Pin Buffer 15 (From other DAI)
1	Reserved	001111 (0xF)	DAI1_PB16_O	Pin Buffer 16 (From other DAI)
1	Reserved	010000 (0x10)	DAI1_PB17_O	Pin Buffer 17 (From other DAI)
1	Reserved	010001 (0x11)	DAI1_PB18_O	Pin Buffer 18 (From other DAI)
1	Reserved	010010 (0x12)	DAI1_PB19_O	Pin Buffer 19 (From other DAI)
1	Reserved	010011 (0x13)	DAI1_PB20_O	Pin Buffer 20 (From other DAI)
1	Reserved	010100 (0x14)	SPT4_AD0_O	SPORT 4 Data AD0 (From other DAI)
1	Reserved	010101 (0x15)	SPT4_AD1_O	SPORT 4 Data AD1 (From other DAI)
1	Reserved	010110 (0x16)	SPT4_BD0_O	SPORT 4 Data BD0 (From other DAI)
1	Reserved	010111 (0x17)	SPT4_BD1_O	SPORT 4 Data BD1 (From other DAI)
1	Reserved	011000 (0x18)	SPT5_AD0_O	SPORT 5 Data AD0 (From other DAI)
1	Reserved	011001 (0x19)	SPT5_AD1_O	SPORT 5 Data AD1 (From other DAI)
1	Reserved	011010 (0x1A)	SPT5_BD0_O	SPORT 5 Data BD0 (From other DAI)
1	Reserved	011011 (0x1B)	SPT5_BD1_O	SPORT 5 Data BD1 (From other DAI)
1	Reserved	011100 (0x1C)	SPT6_AD0_O	SPORT 6 Data AD0 (From other DAI)
1	Reserved	011101 (0x1D)	SPT6_AD1_O	SPORT 6 Data AD1 (From other DAI)
1	Reserved	011110 (0x1E)	SPT6_BD0_O	SPORT 6 Data BD0 (From other DAI)

Table 31-11: Group B – Serial Data Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	011111 (0x1F)	SPT6_BD1_O	SPORT 6 Data BD1 (From other DAI)
1	Reserved	100000 (0x20)	SRC4_DAT_OP_O	SRC4 Data Out (From other DAI)
1	Reserved	100001 (0x21)	SRC5_DAT_OP_O	SRC5 Data Out (From other DAI)
1	Reserved	100010 (0x22)	SRC6_DAT_OP_O	SRC6 Data Out (From other DAI)
1	Reserved	100011 (0x23)	SRC7_DAT_OP_O	SRC7 Data Out (From other DAI)
1	Reserved	100100 (0x24)	SRC4_TDM_IP_O	SRC4 Data Out (From other DAI)
1	Reserved	100101 (0x25)	SRC5_TDM_IP_O	SRC5 Data Out (From other DAI)
1	Reserved	100110 (0x26)	SRC6_TDM_IP_O	SRC6 Data Out (From other DAI)
1	Reserved	100111 (0x27)	SRC7_TDM_IP_O	SRC7 Data Out (From other DAI)
1	Reserved	101000 (0x28)	SPDIF1_RX_DAT_O	SPDIF 1 RX Serial Data Out (From other DAI)
1	Reserved	101001 (0x29)–101011 (0x2B)	Reserved	
1	Reserved	101100 (0x2C)	SPT7_AD0_O	SPORT 7 Data AD0 (From other DAI)
1	Reserved	101101 (0x2D)	SPT7_AD1_O	SPORT 7 Data AD1 (From other DAI)
1	Reserved	101110 (0x2E)	SPT7_BD0_O	SPORT 7 Data BD0 (From other DAI)
1	Reserved	101111 (0x2F)	SPT7_BD1_O	SPORT 7 Data BD1 (From other DAI)
1	Reserved	110000 (0x30)	SPDIF1_TX_O	SPDIF 1 TX Biphase Stream (From other DAI)
1	Reserved	110001 (0x31)	Reserved	
1	Reserved	110010 (0x32)	Reserved	
1	Reserved	110011 (0x33)	PDM1_SDATA_O	PDM1 Serial Data Output
1	Reserved	110100 (0x34)–111111 (0x3F)	Reserved	

Group C – Frame Sync Source Signals

The group C frame sync signal sources are listed in the following table.

Table 31-12: Group C – Frame Sync Signals

Extended Selection Code [1:0]		Selection Code[4:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	0	00000 (0x0)	DAI0_PB01_O	Pin Buffer 1
0	0	00001 (0x1)	DAI0_PB02_O	Pin Buffer 2
0	0	00010 (0x2)	DAI0_PB03_O	Pin Buffer 3
0	0	00011 (0x3)	DAI0_PB04_O	Pin Buffer 4
0	0	00100 (0x4)	DAI0_PB05_O	Pin Buffer 5
0	0	00101 (0x5)	DAI0_PB06_O	Pin Buffer 6
0	0	00110 (0x6)	DAI0_PB07_O	Pin Buffer 7
0	0	00111 (0x7)	DAI0_PB08_O	Pin Buffer 8
0	0	01000 (0x8)	DAI0_PB09_O	Pin Buffer 9
0	0	01001 (0x9)	DAI0_PB10_O	Pin Buffer 10
0	0	01010 (0xA)	DAI0_PB11_O	Pin Buffer 11
0	0	01011 (0xB)	DAI0_PB12_O	Pin Buffer 12
0	0	01100 (0xC)	DAI0_PB13_O	Pin Buffer 13
0	0	01101 (0xD)	DAI0_PB14_O	Pin Buffer 14
0	0	01110 (0xE)	DAI0_PB15_O	Pin Buffer 15
0	0	01111 (0xF)	DAI0_PB16_O	Pin Buffer 16
0	0	10000 (0x10)	DAI0_PB17_O	Pin Buffer 17
0	0	10001 (0x11)	DAI0_PB18_O	Pin Buffer 18
0	0	10010 (0x12)	DAI0_PB19_O	Pin Buffer 19
0	0	10011 (0x13)	DAI0_PB20_O	Pin Buffer 20
0	0	10100 (0x14)	SPT0_AFS_O	SPORT 0 Frame Sync A
0	0	10101 (0x15)	SPT0_BFS_O	SPORT 0 Frame Sync B
0	0	10110 (0x16)	SPT1_AFS_O	SPORT 1 Frame Sync A
0	0	10111 (0x17)	SPT1_BFS_O	SPORT 1 Frame Sync B
0	0	11000 (0x18)	SPT2_AFS_O	SPORT 2 Frame Sync A
0	0	11001 (0x19)	SPT2_BFS_O	SPORT 2 Frame Sync B
0	0	11010 (0x1A)	SPDIF0_FS_O	SPDIF 0 RX Frame Sync Output
0	0	11011 (0x1B)	Reserved	
0	0	11100 (0x1C)	PCG0_FSA_O	Precision Frame Sync A Output
0	0	11101 (0x1D)	PCG0_FSB_O	Precision Frame Sync B Output

Table 31-12: Group C – Frame Sync Signals (Continued)

Extended Selection Code [1:0]		Selection Code[4:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	0	11110 (0x1E)	LOW	Logic Level Low (0)
0	0	11111 (0x1F)	HIGH	Logic Level High (1)
0	1	00000 (0x0)	SPT3_AFS_O	SPORT 3 Frame Sync A
0	1	00001 (0x1)	SPT3_BFS_O	SPORT 3 Frame Sync B
0	1	00010 (0x2)	PCG_FSE_O	Precision Frame Sync E Output
0	1	00011 (0x3)	PCG_FSF_O	Precision Frame Sync F Output
0	1	00100 (0x4)	PCG_FSA_INV_O	Precision Inverted Frame Sync A Output
0	1	00101 (0x5)	PCG_FSB_INV_O	Precision Inverted Frame Sync B Output
0	1	00110 (0x6)	PCG_FSE_INV_O	Precision Inverted Frame Sync E Output
0	1	00111 (0x7)	PCG_FSF_INV_O	Precision Inverted Frame Sync F Output
0	1	01000 (0x8) - 11111 (0x1F)	Reserved	
1	0	00000 (0x0)	DAI1_PB01_O	Pin Buffer 1
1	0	00001 (0x1)	DAI1_PB02_O	Pin Buffer 2
1	0	00010 (0x2)	DAI1_PB03_O	Pin Buffer 3
1	0	00011 (0x3)	DAI1_PB04_O	Pin Buffer 4
1	0	00100 (0x4)	DAI1_PB05_O	Pin Buffer 5
1	0	00101 (0x5)	DAI1_PB06_O	Pin Buffer 6
1	0	00110 (0x6)	DAI1_PB07_O	Pin Buffer 7
1	0	00111 (0x7)	DAI1_PB08_O	Pin Buffer 8
1	0	01000 (0x8)	DAI1_PB09_O	Pin Buffer 9
1	0	01001 (0x9)	DAI1_PB10_O	Pin Buffer 10
1	0	01010 (0xA)	DAI1_PB11_O	Pin Buffer 11
1	0	01011 (0xB)	DAI1_PB12_O	Pin Buffer 12
1	0	01100 (0xC)	DAI1_PB13_O	Pin Buffer 13
1	0	01101 (0xD)	DAI1_PB14_O	Pin Buffer 14
1	0	01110 (0xE)	DAI1_PB15_O	Pin Buffer 15
1	0	01111 (0xF)	DAI1_PB16_O	Pin Buffer 16
1	0	10000 (0x10)	DAI1_PB17_O	Pin Buffer 17
1	0	10001 (0x11)	DAI1_PB18_O	Pin Buffer 18

Table 31-12: Group C – Frame Sync Signals (Continued)

Extended Selection Code [1:0]		Selection Code[4:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	0	10010 (0x12)	DAI1_PB19_O	Pin Buffer 19
1	0	10011 (0x13)	DAI1_PB20_O	Pin Buffer 20
1	0	10100 (0x14)	SPT4_AFS_O	SPORT 4 Frame Sync A
1	0	10101 (0x15)	SPT4_BFS_O	SPORT 4 Frame Sync B
1	0	10110 (0x16)	SPT5_AFS_O	SPORT 5 Frame Sync A
1	0	10111 (0x17)	SPT5_BFS_O	SPORT 5 Frame Sync B
1	0	11000 (0x18)	SPT6_AFS_O	SPORT 6 Frame Sync A
1	0	11001 (0x19)	SPT6_BFS_O	SPORT 6 Frame Sync B
1	0	11010 (0x1A)	SPDIF1_FS_O	SPDIF 1 RX Frame Sync Output
1	0	11011 (0x1B)	Reserved	
1	0	11100 (0x1C)	PCG_FSC_O	Precision Frame Sync C Output
1	0	11101 (0x1D)	PCG_FSD_O	Precision Frame Sync D Output
1	0	11110 (0x1E)	Reserved	
1	0	11111 (0x1F)	Reserved	
1	1	00000 (0x0)	SPT7_AFS_O	SPORT 7 Frame Sync A (From other DAI)
1	1	00001 (0x1)	SPT7_BFS_O	SPORT 7 Frame Sync B (From other DAI)
1	1	00010 (0x2)	PCG_FSG_O	Precision Frame Sync G Output (From other DAI)
1	1	00011 (0x3)	PCG_FSH_O	Precision Frame Sync H Output (From other DAI)
1	1	00100 (0x4)	PCG_FSC_INV_O	Precision Inverted Frame Sync C Output (From other DAI)
1	1	00101 (0x5)	PCG_FSD_INV_O	Precision Inverted Frame Sync D Output (From other DAI)
1	1	00110 (0x6)	PCG_FSG_INV_O	Precision Inverted Frame Sync G Output (From other DAI)
1	1	00111 (0x7)	PCG_FSH_INV_O	Precision Inverted Frame Sync H Output (From other DAI)
1	1	01000 (0x8) - 11111 (0x1F)	Reserved	

Group D – Pin Signal Assignment Source Signals

The group D pin signal assignment sources are listed in the following table.

Table 31-13: Group D – Pin Signal Assignments

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	0000000 (0x0)	DAI0_PB01_O	Pin Buffer 1
0	Reserved	0000001 (0x1)	DAI0_PB02_O	Pin Buffer 2
0	Reserved	0000010 (0x2)	DAI0_PB03_O	Pin Buffer 3
0	Reserved	0000011 (0x3)	DAI0_PB04_O	Pin Buffer 4
0	Reserved	0000100 (0x4)	DAI0_PB05_O	Pin Buffer 5
0	Reserved	0000101 (0x5)	DAI0_PB06_O	Pin Buffer 6
0	Reserved	0000110 (0x6)	DAI0_PB07_O	Pin Buffer 7
0	Reserved	0000111 (0x7)	DAI0_PB08_O	Pin Buffer 8
0	Reserved	0001000 (0x8)	DAI0_PB09_O	Pin Buffer 9
0	Reserved	0001001 (0x9)	DAI0_PB10_O	Pin Buffer 10
0	Reserved	0001010 (0xA)	DAI0_PB11_O	Pin Buffer 11
0	Reserved	0001011 (0xB)	DAI0_PB12_O	Pin Buffer 12
0	Reserved	0001100 (0xC)	DAI0_PB13_O	Pin Buffer 13
0	Reserved	0001101 (0xD)	DAI0_PB14_O	Pin Buffer 14
0	Reserved	0001110 (0xE)	DAI0_PB15_O	Pin Buffer 15
0	Reserved	0001111 (0xF)	DAI0_PB16_O	Pin Buffer 16
0	Reserved	0010000 (0x10)	DAI0_PB17_O	Pin Buffer 17
0	Reserved	0010001 (0x11)	DAI0_PB18_O	Pin Buffer 18
0	Reserved	0010010 (0x12)	DAI0_PB19_O	Pin Buffer 19
0	Reserved	0010011 (0x13)	DAI0_PB20_O	Pin Buffer 20
0	Reserved	0010100 (0x14)	SPT0_AD0_O	SPORT 0 Data AD0
0	Reserved	0010101 (0x15)	SPT0_AD1_O	SPORT 0 Data AD1
0	Reserved	0010110 (0x16)	SPT0_BD0_O	SPORT 0 Data BD0
0	Reserved	0010111 (0x17)	SPT0_BD1_O	SPORT 0 Data BD1
0	Reserved	0011000 (0x18)	SPT1_AD0_O	SPORT 1 Data AD0
0	Reserved	0011001 (0x19)	SPT1_AD1_O	SPORT 1 Data AD1
0	Reserved	0011010 (0x1A)	SPT1_BD0_O	SPORT 1 Data BD0
0	Reserved	0011011 (0x1B)	SPT1_BD1_O	SPORT 1 Data BD1
0	Reserved	0011100 (0x1C)	SPT2_AD0_O	SPORT 2 Data AD0
0	Reserved	0011101 (0x1D)	SPT2_AD1_O	SPORT 2 Data AD1

Table 31-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	0011110 (0x1E)	SPT2_BD0_O	SPORT 2 Data BD0
0	Reserved	0011111 (0x1F)	SPT2_BD1_O	SPORT 2 Data BD1
0	Reserved	0100000 (0x20)	SPT0_ACLK_O	SPORT 0 Clock A
0	Reserved	0100001 (0x21)	SPT0_BCLK_O	SPORT 0 Clock B
0	Reserved	0100010 (0x22)	SPT1_ACLK_O	SPORT 1 Clock A
0	Reserved	0100011 (0x23)	SPT1_BCLK_O	SPORT 1 Clock B
0	Reserved	0100100 (0x24)	SPT2_ACLK_O	SPORT 2 Clock A
0	Reserved	0100101 (0x25)	SPT2_BCLK_O	SPORT 2 Clock B
0	Reserved	0100110 (0x26)	SPT0_AFS_O	SPORT 0 Frame Sync A
0	Reserved	0100111 (0x27)	SPT0_BFS_O	SPORT 0 Frame Sync B
0	Reserved	0101000 (0x28)	SPT1_AFS_O	SPORT 1 Frame Sync A
0	Reserved	0101001 (0x29)	SPT1_BFS_O	SPORT 1 Frame Sync B
0	Reserved	0101010 (0x2A)	SPT2_AFS_O	SPORT 2 Frame Sync A
0	Reserved	0101011 (0x2B)	SPT2_BFS_O	SPORT 2 Frame Sync B
0	Reserved	0101100 (0x2C)	SPT3_AD0_O	SPORT 3 Data AD0
0	Reserved	0101101 (0x2D)	SPT3_AD1_O	SPORT 3 Data AD1
0	Reserved	0101110 (0x2E)	SPT3_BD0_O	SPORT 3 Data BD0
0	Reserved	0101111 (0x2F)	SPT3_BD1_O	SPORT 3 Data BD1
0	Reserved	0110000 (0x30)	MLB0_CLKOUT	MLB PLL clock output
0	Reserved	0110001 (0x31)	SPDIF0_TX_BLKSTART_O	SPDIF 0 TX Block Start Output
0	Reserved	0110010 (0x32)	Reserved	
0	Reserved	0110011 (0x33)	Reserved	
0	Reserved	0110100 (0x34)	SPT3_ACLK_O	SPORT 3 Clock A
0	Reserved	0110101 (0x35)	SPT3_BCLK_O	SPORT 3 Clock B
0	Reserved	0110110 (0x36)	SPT3_AFS_O	SPORT 3 Frame Sync A
0	Reserved	0110111 (0x37)	SPT3_BFS_O	SPORT 3 Frame Sync B
0	Reserved	0111000 (0x38)	PCG0_CLKA_O	Precision Clock A
0	Reserved	0111001 (0x39)	PCG0_CLKB_O	Precision Clock B
0	Reserved	0111010 (0x3A)	PCG0_FSA_O	Precision Frame Sync A
0	Reserved	0111011 (0x3B)	PCG0_FSB_O	Precision Frame Sync B

Table 31-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	0111100 (0x3C)	Reserved	
0	Reserved	0111101 (0x3D)	SRC0_DAT_OP_O	SRC0 Data Output
0	Reserved	0111110 (0x3E)	SRC1_DAT_OP_O	SRC1 Data Output
0	Reserved	0111111 (0x3F)	SRC2_DAT_OP_O	SRC2 Data Output
0	Reserved	1000000 (0x40)	SRC3_DAT_OP_O	SRC3 Data Output
0	Reserved	1000001 (0x41)	SPDIF0_RX_DAT_O	SPDIF 0 RX Data Output
0	Reserved	1000010 (0x42)	SPDIF0_FS_O	SPDIF 0 RX Frame Sync Output
0	Reserved	1000011 (0x43)	SPDIF0_RXCLK_O	SPDIF 0 RX Clock Output
0	Reserved	1000100 (0x44)	SPDIF0_RX_TDMCLK_O	SPDIF 0 RX TDM Clock Output
0	Reserved	1000101 (0x45)	SPDIF0_TX_O	SPDIF 0 TX Biphase Encoded Data Output
0	Reserved	1000110 (0x46)	SPT0_ATDV_O	SPORT0 Transmit A Data Valid Output
0	Reserved	1000111 (0x47)	SPT0_BTDV_O	SPORT0 Transmit B Data Valid Output
0	Reserved	1001000 (0x48)	SPT1_ATDV_O	SPORT1 Transmit A Data Valid Output
0	Reserved	1001001 (0x49)	SPT1_BTDV_O	SPORT1 Transmit B Data Valid Output
0	Reserved	1001010 (0x4A)	SPT2_ATDV_O	SPORT2 Transmit A Data Valid Output
0	Reserved	1001011 (0x4B)	SPT2_BTDV_O	SPORT2 Transmit B Data Valid Output
0	Reserved	1001100 (0x4C)	SPT3_ATDV_O	SPORT3 Transmit A Data Valid Output
0	Reserved	1001101 (0x4D)	SPT3_BTDV_O	SPORT3 Transmit B Data Valid Output
0	Reserved	1001110 (0x4E) – 1010101 (0x55)	Reserved	
0	Reserved	1010110 (0x56)	PCG_CLKE_O	Precision Clock E
0	Reserved	1010111 (0x57)	PCG_CLKF_O	Precision Clock F
0	Reserved	1011000 (0x58)	PCG_FSE_O	Precision Frame Sync E
0	Reserved	1011001 (0x59)	PCG_FSF_O	Precision Frame Sync F
0	Reserved	1011010 (0x5A)	PCG_CLKA_INV_O	Inverted Precision Clock A Output
0	Reserved	1011011 (0x5B)	PCG_CLKB_INV_O	Inverted Precision Clock B Output
0	Reserved	1011100 (0x5C)	PCG_CLKE_INV_O	Inverted Precision Clock E Output
0	Reserved	1011101 (0x5D)	PCG_CLKF_INV_O	Inverted Precision Clock F Output
0	Reserved	1011110 (0x5E)	PCG_FSA_INV_O	Inverted Precision Frame Sync A Output

Table 31-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	1011111 (0x5F)	PCG_FSB_INV_O	Inverted Precision Frame Sync B Output
0	Reserved	1100000 (0x60)	PCG_FSE_INV_O	Inverted Precision Frame Sync E Output
0	Reserved	1100001 (0x61)	PCG_FSF_INV_O	Inverted Precision Frame Sync F Output
0	Reserved	1100010 (0x62)	PDM0_CLK0_O	PDM0 Clock0 Output
0	Reserved	1100011 (0x63)	PDM0_SDATA_O	PDM0 Serial Data Output
0	Reserved	1010110 (0x64) – 1111101 (0x7D)	Reserved	
0	Reserved	1111110 (0x7E)	LOW	Logic Level Low (0)
0	Reserved	1111111 (0x7F)	HIGH	Logic Level High (1)
1	Reserved	0000000 (0x0)	DAI1_PB01_O	Pin Buffer 1 (from other DAI)
1	Reserved	0000001 (0x1)	DAI1_PB02_O	Pin Buffer 2 (from other DAI)
1	Reserved	0000010 (0x2)	DAI1_PB03_O	Pin Buffer 3 (from other DAI)
1	Reserved	0000011 (0x3)	DAI1_PB04_O	Pin Buffer 4 (from other DAI)
1	Reserved	0000100 (0x4)	DAI1_PB05_O	Pin Buffer 5 (from other DAI)
1	Reserved	0000101 (0x5)	DAI1_PB06_O	Pin Buffer 6 (from other DAI)
1	Reserved	0000110 (0x6)	DAI1_PB07_O	Pin Buffer 7 (from other DAI)
1	Reserved	0000111 (0x7)	DAI1_PB08_O	Pin Buffer 8 (from other DAI)
1	Reserved	0001000 (0x8)	DAI1_PB09_O	Pin Buffer 9 (from other DAI)
1	Reserved	0001001 (0x9)	DAI1_PB10_O	Pin Buffer 10 (from other DAI)
1	Reserved	0001010 (0xA)	DAI1_PB11_O	Pin Buffer 11 (from other DAI)
1	Reserved	0001011 (0xB)	DAI1_PB12_O	Pin Buffer 12 (from other DAI)
1	Reserved	0001100 (0xC)	DAI1_PB13_O	Pin Buffer 13 (from other DAI)
1	Reserved	0001101 (0xD)	DAI1_PB14_O	Pin Buffer 14 (from other DAI)
1	Reserved	0001110 (0xE)	DAI1_PB15_O	Pin Buffer 15 (from other DAI)
1	Reserved	0001111 (0xF)	DAI1_PB16_O	Pin Buffer 16 (from other DAI)
1	Reserved	0010000 (0x10)	DAI1_PB17_O	Pin Buffer 17 (from other DAI)
1	Reserved	0010001 (0x11)	DAI1_PB18_O	Pin Buffer 18 (from other DAI)
1	Reserved	0010010 (0x12)	DAI1_PB19_O	Pin Buffer 19 (from other DAI)
1	Reserved	0010011 (0x13)	DAI1_PB20_O	Pin Buffer 20 (from other DAI)
1	Reserved	0010100 (0x14)	SPT4_AD0_O	SPORT 4 Data AD0 (from other DAI)

Table 31-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	0010101 (0x15)	SPT4_AD1_O	SPORT 4 Data AD1 (from other DAI)
1	Reserved	0010110 (0x16)	SPT4_BD0_O	SPORT 4 Data BD0 (from other DAI)
1	Reserved	0010111 (0x17)	SPT4_BD1_O	SPORT 4 Data BD1 (from other DAI)
1	Reserved	0011000 (0x18)	SPT5_AD0_O	SPORT 5 Data AD0 (from other DAI)
1	Reserved	0011001 (0x19)	SPT5_AD1_O	SPORT 5 Data AD1 (from other DAI)
1	Reserved	0011010 (0x1A)	SPT5_BD0_O	SPORT 5 Data BD0 (from other DAI)
1	Reserved	0011011 (0x1B)	SPT5_BD1_O	SPORT 5 Data BD1 (from other DAI)
1	Reserved	0011100 (0x1C)	SPT6_AD0_O	SPORT 6 Data AD0 (from other DAI)
1	Reserved	0011101 (0x1D)	SPT6_AD1_O	SPORT 6 Data AD1 (from other DAI)
1	Reserved	0011110 (0x1E)	SPT6_BD0_O	SPORT 6 Data BD0 (from other DAI)
1	Reserved	0011111 (0x1F)	SPT6_BD1_O	SPORT 6 Data BD1 (from other DAI)
1	Reserved	0100000 (0x20)	SPT4_ACLK_O	SPORT 4 Clock A (from other DAI)
1	Reserved	0100001 (0x21)	SPT4_BCLK_O	SPORT 4 Clock B (from other DAI)
1	Reserved	0100010 (0x22)	SPT5_ACLK_O	SPORT 5 Clock A (from other DAI)
1	Reserved	0100011 (0x23)	SPT5_BCLK_O	SPORT 5 Clock B (from other DAI)
1	Reserved	0100100 (0x24)	SPT6_ACLK_O	SPORT 6 Clock A (from other DAI)
1	Reserved	0100101 (0x25)	SPT6_BCLK_O	SPORT 6 Clock B (from other DAI)
1	Reserved	0100110 (0x26)	SPT4_AFS_O	SPORT 4 Frame Sync A (from other DAI)
1	Reserved	0100111 (0x27)	SPT4_BFS_O	SPORT 4 Frame Sync B (from other DAI)
1	Reserved	0101000 (0x28)	SPT5_AFS_O	SPORT 5 Frame Sync A (from other DAI)
1	Reserved	0101001 (0x29)	SPT5_BFS_O	SPORT 5 Frame Sync B (from other DAI)
1	Reserved	0101010 (0x2A)	SPT6_AFS_O	SPORT 6 Frame Sync A (from other DAI)
1	Reserved	0101011 (0x2B)	SPT6_BFS_O	SPORT 6 Frame Sync B (from other DAI)
1	Reserved	0101100 (0x2C)	SPT7_AD0_O	SPORT 7 Data AD0 (from other DAI)
1	Reserved	0101101 (0x2D)	SPT7_AD1_O	SPORT 7 Data AD1 (from other DAI)
1	Reserved	0101110 (0x2E)	SPT7_BD0_O	SPORT 7 Data BD0 (from other DAI)
1	Reserved	0101111 (0x2F)	SPT7_BD1_O	SPORT 7 Data BD1 (from other DAI)
1	Reserved	0110000 (0x30)	Reserved	
1	Reserved	0110001 (0x31)	SPDIF1_TX_BLKSTART_O	SPDIF 1 TX Block Start Output
1	Reserved	0110010 (0x32)	Reserved	

Table 31-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	0110011 (0x33)	Reserved	
1	Reserved	0110100 (0x34)	SPT7_ACLK_O	SPORT 7 Clock A (from other DAI)
1	Reserved	0110101 (0x35)	SPT7_BCLK_O	SPORT 7 Clock B (from other DAI)
1	Reserved	0110110 (0x36)	SPT7_AFS_O	SPORT 7 Frame Sync A (from other DAI)
1	Reserved	0110111 (0x37)	SPT7_BFS_O	SPORT 7 Frame Sync B (from other DAI)
1	Reserved	0111000 (0x38)	PCG0_CLKC_O	Precision Clock C (from other DAI)
1	Reserved	0111001 (0x39)	PCG0_CLKD_O	Precision Clock D (from other DAI)
1	Reserved	0111010 (0x3A)	PCG0_FSC_O	Precision Frame Sync C (from other DAI)
1	Reserved	0111011 (0x3B)	PCG0_FSD_O	Precision Frame Sync D (from other DAI)
1	Reserved	0111100 (0x3C)	Reserved	
1	Reserved	0111101 (0x3D)	SRC4_DAT_OP_O	SRC0 Data Output (from other DAI)
1	Reserved	0111110 (0x3E)	SRC5_DAT_OP_O	SRC1 Data Output (from other DAI)
1	Reserved	0111111 (0x3F)	SRC6_DAT_OP_O	SRC2 Data Output (from other DAI)
1	Reserved	1000000 (0x40)	SRC7_DAT_OP_O	SRC3 Data Output (from other DAI)
1	Reserved	1000001 (0x41)	SPDIF1_RX_DAT_O	SPDIF 1 RX Data Output (from other DAI)
1	Reserved	1000010 (0x42)	SPDIF1_FS_O	SPDIF 1 RX Frame Sync Output (from other DAI)
1	Reserved	1000011 (0x43)	SPDIF1_RXCLK_O	SPDIF 1 RX Clock Output (from other DAI)
1	Reserved	1000100 (0x44)	SPDIF1_RX_TDMCLK_O	SPDIF 1 RX TDM Clock Output (from other DAI)
1	Reserved	1000101 (0x45)	SPDIF1_TX_O	SPDIF 1 TX Biphase Encoded Data Output (from other DAI)
1	Reserved	1000110 (0x46)	SPT4_ATDV_O	SPORT4 Transmit A Data Valid Output (from other DAI) (from other DAI)
1	Reserved	1000111 (0x47)	SPT4_BTDV_O	SPORT4 Transmit B Data Valid Output (from other DAI)
1	Reserved	1001000 (0x48)	SPT5_ATDV_O	SPORT5 Transmit A Data Valid Output (from other DAI)
1	Reserved	1001001 (0x49)	SPT5_BTDV_O	SPORT5 Transmit B Data Valid Output (from other DAI)

Table 31-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	1001010 (0x4A)	SPT6_ATDV_O	SPORT6 Transmit A Data Valid Output (from other DAI)
1	Reserved	1001011 (0x4B)	SPT6_BTDV_O	SPORT6 Transmit B Data Valid Output (from other DAI)
1	Reserved	1001100 (0x4C)	SPT7_ATDV_O	SPORT7 Transmit A Data Valid Output (from other DAI)
1	Reserved	1001101 (0x4D)	SPT7_BTDV_O	SPORT7 Transmit B Data Valid Output (from other DAI)
1	Reserved	1001110 (0x4E) – 1010101 (0x55)	Reserved	
1	Reserved	1010110 (0x56)	PCG_CLKG_O	Precision Clock G (from other DAI)
1	Reserved	1010111 (0x57)	PCG_CLKH_O	Precision Clock H (from other DAI)
1	Reserved	1011000 (0x58)	PCG_FSG_O	Precision Frame Sync G (from other DAI)
1	Reserved	1011001 (0x59)	PCG_FSH_O	Precision Frame Sync H (from other DAI)
1	Reserved	1011010 (0x5A)	PCG_CLKC_INV_O	Inverted Precision Clock C Output (from other DAI)
1	Reserved	1011011 (0x5B)	PCG_CLKD_INV_O	Inverted Precision Clock D Output (from other DAI)
1	Reserved	1011100 (0x5C)	PCG_CLKG_INV_O	Inverted Precision Clock G Output (from other DAI)
1	Reserved	1011101 (0x5D)	PCG_CLKH_INV_O	Inverted Precision Clock H Output (from other DAI)
1	Reserved	1011110 (0x5E)	PCG_FSC_INV_O	Inverted Precision Frame Sync C Output (from other DAI)
1	Reserved	1011111 (0x5F)	PCG_FSD_INV_O	Inverted Precision Frame Sync D Output (from other DAI)
1	Reserved	1100000 (0x60)	PCG_FSG_INV_O	Inverted Precision Frame Sync G Output (from other DAI)
1	Reserved	1100001 (0x61)	PCG_FSH_INV_O	Inverted Precision Frame Sync H Output (from other DAI)
1	Reserved	1100010 (0x62)	PDM1_CLK0_O	PDM1 Clock0 Output (from other DAI)
1	Reserved	1100011 (0x63)	PDM1_SDATA_O	PDM1 Serial Data Output (from other DAI)

Table 31-13: Group D – Pin Signal Assignments (Continued)

Extended Selection Code [1:0]		Selection Code[6:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
1	Reserved	1010110 (0x64) – 1111111 (0x7F)	Reserved	

Group E – Miscellaneous Source Signals

The following table lists the group E miscellaneous signal sources.

Table 31-14: Group E – Miscellaneous Signals

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	0	00000 (0x0)	DAI0_PB01_O	Pin Buffer 1 Output
0	0	00001 (0x1)	DAI0_PB02_O	Pin Buffer 2 Output
0	0	00010 (0x2)	DAI0_PB03_O	Pin Buffer 3 Output
0	0	00011 (0x3)	DAI0_PB04_O	Pin Buffer 4 Output
0	0	00100 (0x4)	DAI0_PB05_O	Pin Buffer 5 Output
0	0	00101 (0x5)	DAI0_PB06_O	Pin Buffer 6 Output
0	0	00110 (0x6)	DAI0_PB07_O	Pin Buffer 7 Output
0	0	00111 (0x7)	DAI0_PB08_O	Pin Buffer 8 Output
0	0	01000 (0x8)	DAI0_PB09_O	Pin Buffer 9 Output
0	0	01001 (0x9)	DAI0_PB10_O	Pin Buffer 10 Output
0	0	01010 (0xA)	DAI0_PB11_O	Pin Buffer 11 Output
0	0	01011 (0xB)	DAI0_PB12_O	Pin Buffer 12 Output
0	0	01100 (0xC)	DAI0_PB13_O	Pin Buffer 13 Output
0	0	01101 (0xD)	DAI0_PB14_O	Pin Buffer 14 Output
0	0	01110 (0xE)	DAI0_PB15_O	Pin Buffer 15 Output
0	0	01111 (0xF)	DAI0_PB16_O	Pin Buffer 16 Output
0	0	10000 (0x10)	DAI0_PB17_O	Pin Buffer 17 Output
0	0	10001 (0x11)	DAI0_PB18_O	Pin Buffer 18 Output
0	0	10010 (0x12)	DAI0_PB19_O	Pin Buffer 19 Output
0	0	10011 (0x13)	DAI0_PB20_O	Pin Buffer 20 Output
0	0	10100 (0x14)	SPT0_AFS_O	SPORT 0 Frame Sync A

Table 31-14: Group E – Miscellaneous Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	0	10101 (0x15)	SPT0_BFS_O	SPORT 0 Frame Sync B
0	0	10110 (0x16)	SPT1_AFS_O	SPORT 1 Frame Sync A
0	0	10111 (0x17)	SPT1_BFS_O	SPORT 1 Frame Sync B
0	0	11000 (0x18)	SPT2_AFS_O	SPORT 2 Frame Sync A
0	0	11001 (0x19)	SPT2_BFS_O	SPORT 2 Frame Sync B
0	0	11010 (0x1A)	SPDIF0_TX_BLKSTART_O	SPDIF 0 TX Block Start Output
0	0	11011 (0x1B)	PCG0_FSA_O	Precision Frame Sync A
0	0	11100 (0x1C)	PCG0_CLKB_O	Precision Clock B
0	0	11101 (0x1D)	PCG0_FSB_O	Precision Frame Sync B
0	0	11110 (0x1E)	LOW	Logic Level Low (0) as a Source
0	0	11111 (0x1F)	HIGH	Logic Level High (1) as a Source
0	1	00000 (0x0)	TRGS_PCGA_O	Trigger receiver mapped to PCGA
0	1	00001 (0x1)	TRGS_PCGB_O	Trigger receiver mapped to PCGB
0	1	00010 (0x2)	TRGS_PCGE_O	Trigger receiver mapped to PCGE
0	1	00011 (0x3)	TRGS_PCGF_O	Trigger receiver mapped to PCGF
0	1	00100 (0x4)	SPT3_AFS_O	SPORT 3 Frame Sync A
0	1	00101 (0x5)	SPT3_BFS_O	SPORT 3 Frame Sync B
0	1	00110 (0x6)	PCG_FSE_O	Precision Frame Sync E
0	1	00111 (0x7)	PCG_FSF_O	Precision Frame Sync F
0	1	01000 (0x8)-11111 (0x1F)	Reserved	
1	0	00000 (0x0)	DAI1_PB01_O	Pin Buffer 1 Output (from other DAI)
1	0	00001 (0x1)	DAI1_PB02_O	Pin Buffer 2 Output (from other DAI)
1	0	00010 (0x2)	DAI1_PB03_O	Pin Buffer 3 Output (from other DAI)
1	0	00011 (0x3)	DAI1_PB04_O	Pin Buffer 4 Output (from other DAI)
1	0	00100 (0x4)	DAI1_PB05_O	Pin Buffer 5 Output (from other DAI)
1	0	00101 (0x5)	DAI1_PB06_O	Pin Buffer 6 Output (from other DAI)
1	0	00110 (0x6)	DAI1_PB07_O	Pin Buffer 7 Output (from other DAI)
1	0	00111 (0x7)	DAI1_PB08_O	Pin Buffer 8 Output (from other DAI)
1	0	01000 (0x8)	DAI1_PB09_O	Pin Buffer 9 Output (from other DAI)
1	0	01001 (0x9)	DAI1_PB10_O	Pin Buffer 10 Output (from other DAI)

Table 31-14: Group E – Miscellaneous Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
1	0	01010 (0xA)	DAI1_PB11_O	Pin Buffer 11 Output (from other DAI)
1	0	01011 (0xB)	DAI1_PB12_O	Pin Buffer 12 Output (from other DAI)
1	0	01100 (0xC)	DAI1_PB13_O	Pin Buffer 13 Output (from other DAI)
1	0	01101 (0xD)	DAI1_PB14_O	Pin Buffer 14 Output (from other DAI)
1	0	01110 (0xE)	DAI1_PB15_O	Pin Buffer 15 Output (from other DAI)
1	0	01111 (0xF)	DAI1_PB16_O	Pin Buffer 16 Output (from other DAI)
1	0	10000 (0x10)	DAI1_PB17_O	Pin Buffer 17 Output (from other DAI)
1	0	10001 (0x11)	DAI1_PB18_O	Pin Buffer 18 Output (from other DAI)
1	0	10010 (0x12)	DAI1_PB19_O	Pin Buffer 19 Output (from other DAI)
1	0	10011 (0x13)	DAI1_PB20_O	Pin Buffer 20 Output (from other DAI)
1	0	10100 (0x14)	SPT4_AFS_O	SPORT 4 Frame Sync A (from other DAI)
1	0	10101 (0x15)	SPT4_BFS_O	SPORT 4 Frame Sync B (from other DAI)
1	0	10110 (0x16)	SPT5_AFS_O	SPORT 5 Frame Sync A (from other DAI)
1	0	10111 (0x17)	SPT5_BFS_O	SPORT 5 Frame Sync B (from other DAI)
1	0	11000 (0x18)	SPT6_AFS_O	SPORT 6 Frame Sync A (from other DAI)
1	0	11001 (0x19)	SPT6_BFS_O	SPORT 6 Frame Sync B (from other DAI)
1	0	11010 (0x1A)	SPDIF1_TX_BLKSTART_O	SPDIF 1 TX Block Start Output (from other DAI)
1	0	11011 (0x1B)	PCG_FSC_O	Precision Frame Sync C (from other DAI)
1	0	11100 (0x1C)	PCG_CLKD_O	Precision Clock D (from other DAI)
1	0	11101 (0x1D)	PCG_FSD_O	Precision Frame Sync D (from other DAI)
1	0	11110 (0x1E)	Reserved	
1	0	11111 (0x1F)	Reserved	
1	1	00000 (0x0)	TRGS_PCGC_O	Trigger receiver mapped to PCGC
1	1	00001 (0x1)	TRGS_PCGD_O	Trigger receiver mapped to PCGD
1	1	00010 (0x2)	TRGS_PCGG_O	Trigger receiver mapped to PCGG
1	1	00011 (0x3)	TRGS_PCGH_O	Trigger receiver mapped to PCGH
1	1	00100 (0x4)	SPT7_AFS_O	SPORT 7 Frame Sync A
1	1	00101 (0x5)	SPT7_BFS_O	SPORT 7 Frame Sync B
1	1	00110 (0x6)	PCG_FSG_O	Precision Frame Sync G

Table 31-14: Group E – Miscellaneous Signals (Continued)

Extended Selection Code [1:0]		Selection Code[5:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
1	1	00111 (0x7)	PCG_FSH_O	Precision Frame Sync H
1	1	01000 (0x8)-11111 (0x1F)	Reserved	

Group F – Pin Output Enable Source Signals

The group F pin output enable signals are listed in the following table.

Table 31-15: Group F – Pin Output Enable

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	Bit 0			
0	Reserved	000000 (0x0)	LOW	Logic Level Low (0)
0	Reserved	000001 (0x1)	HIGH	Logic Level High (1)
0	Reserved	000010 (0x2)	DAI0_MISCA0_O	DAI0 Miscellaneous Control A0 Output
0	Reserved	000011 (0x3)	DAI0_MISCA1_O	DAI0 Miscellaneous Control A1 Output
0	Reserved	000100 (0x4)	DAI0_MISCA2_O	DAI0 Miscellaneous Control A2 Output
0	Reserved	000101 (0x5)	DAI0_MISCA3_O	DAI0 Miscellaneous Control A3 Output
0	Reserved	000110 (0x6)	DAI0_MISCA4_O	DAI0 Miscellaneous Control A4 Output
0	Reserved	000111 (0x7)	DAI0_MISCA5_O	DAI0 Miscellaneous Control A5 Output
0	Reserved	001000 (0x8)	SPT0_ACLK_PBEN_O	SPORT 0 Clock A Output Enable
0	Reserved	001001 (0x9)	SPT0_AFS_PBEN_O	SPORT 0 Frame Sync A Output Enable
0	Reserved	001010 (0xA)	SPT0_AD0_PBEN_O	SPORT 0 Data AD0 Output Enable
0	Reserved	001011 (0xB)	SPT0_AD1_PBEN_O	SPORT 0 Data AD1 Output Enable
0	Reserved	001100 (0xC)	SPT0_BCLK_PBEN_O	SPORT 0 Clock B Output Enable
0	Reserved	001101 (0xD)	SPT0_BFS_PBEN_O	SPORT 0 Frame Sync B Output Enable
0	Reserved	001110 (0xE)	SPT0_BD0_PBEN_O	SPORT 0 Data BD0 Output Enable
0	Reserved	001111 (0xF)	SPT0_BD1_PBEN_O	SPORT 0 Data BD1 Output Enable
0	Reserved	010000 (0x10)	SPT1_ACLK_PBEN_O	SPORT 1 Clock A Output Enable
0	Reserved	010001 (0x11)	SPT1_AFS_PBEN_O	SPORT 1 Frame Sync A Output Enable
0	Reserved	010010 (0x12)	SPT1_AD0_PBEN_O	SPORT 1 Data AD0 Output Enable
0	Reserved	010011 (0x13)	SPT1_AD1_PBEN_O	SPORT 1 Data AD1 Output Enable

Table 31-15: Group F – Pin Output Enable (Continued)

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	010100 (0x14)	SPT1_BCLK_PBEN_O	SPORT 1 Clock B Output Enable
0	Reserved	010101 (0x15)	SPT1_BFS_PBEN_O	SPORT 1 Frame Sync B Output Enable
0	Reserved	010110 (0x16)	SPT1_BD0_PBEN_O	SPORT 1 Data BD0 Output Enable
0	Reserved	010111 (0x17)	SPT1_BD1_PBEN_O	SPORT 1 Data BD1 Output Enable
0	Reserved	011000 (0x18)	SPT2_ACLK_PBEN_O	SPORT 2 Clock A Output Enable
0	Reserved	011001 (0x19)	SPT2_AFS_PBEN_O	SPORT 2 Frame Sync A Output Enable
0	Reserved	011010 (0x1A)	SPT2_AD0_PBEN_O	SPORT 2 Data AD0 Output Enable
0	Reserved	011011 (0x1B)	SPT2_AD1_PBEN_O	SPORT 2 Data AD1 Output Enable
0	Reserved	011100 (0x1C)	SPT2_BCLK_PBEN_O	SPORT 2 Clock B Output Enable
0	Reserved	011101 (0x1D)	SPT2_BFS_PBEN_O	SPORT 2 Frame Sync B Output Enable
0	Reserved	011110 (0x1E)	SPT2_BD0_PBEN_O	SPORT 2 Data BD0 Output Enable
0	Reserved	011111 (0x1F)	SPT2_BD1_PBEN_O	SPORT 2 Data BD1 Output Enable
0	Reserved	100000 (0x20)	SPT3_ACLK_PBEN_O	SPORT 3 Clock A Output Enable
0	Reserved	100001 (0x21)	SPT3_AFS_PBEN_O	SPORT 3 Frame Sync A Output Enable
0	Reserved	100010 (0x22)	SPT3_AD0_PBEN_O	SPORT 3 Data AD0 Output Enable
0	Reserved	100011 (0x23)	SPT3_AD1_PBEN_O	SPORT 3 Data AD1 Output Enable
0	Reserved	100100 (0x24)	SPT3_BCLK_PBEN_O	SPORT 2 Clock B Output Enable
0	Reserved	100101 (0x25)	SPT3_BFS_PBEN_O	SPORT 3 Frame Sync B Output Enable
0	Reserved	100110 (0x26)	SPT3_BD0_PBEN_O	SPORT 3 Data BD0 Output Enable
0	Reserved	100111 (0x27)	SPT3_BD1_PBEN_O	SPORT 3 Data BD1 Output Enable
0	Reserved	101000 (0x28)	SPT0_ATDV_PBEN_O	SPORT 0 A Transmit Data Valid Output
0	Reserved	101001 (0x29)	SPT0_BTDV_PBEN_O	SPORT 0 B Transmit Data Valid Output
0	Reserved	101010 (0x2A)	SPT1_ATDV_PBEN_O	SPORT 1 A Transmit Data Valid Output
0	Reserved	101011 (0x2B)	SPT1_BTDV_PBEN_O	SPORT 1 B Transmit Data Valid Output
0	Reserved	101100 (0x2C)	SPT2_ATDV_PBEN_O	SPORT 2 A Transmit Data Valid Output
0	Reserved	101101 (0x2D)	SPT2_BTDV_PBEN_O	SPORT 2 B Transmit Data Valid Output
0	Reserved	100111 (0x2E)	SPT3_ATDV_PBEN_O	SPORT 3 A Transmit Data Valid Output
0	Reserved	101110 (0x2F)	SPT3_BTDV_PBEN_O	SPORT 3 B Transmit Data Valid Output
0	Reserved	110000 (0x30)	PDM0_CLK0_OE_O	PDM0 Clock0 Output Enable
0	Reserved	110001 (0x31)	PDM0_SDATA_OE_O	PDM0 Serial Data Output Enable

Table 31-15: Group F – Pin Output Enable (Continued)

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
0	Reserved	110010 (0x32)–1111111 (0x3F)	Reserved	
1	Reserved	000000 (0x0)–000111 (0x7)	Reserved	
1	Reserved	001000 (0x8)	SPT4_ACLK_PBEN_O	SPORT 4 Clock A Output Enable (from other DAI)
1	Reserved	001001 (0x9)	SPT4_AFS_PBEN_O	SPORT 4 Frame Sync A Output Enable (from other DAI)
1	Reserved	001010 (0xA)	SPT4_AD0_PBEN_O	SPORT 4 Data AD0 Output Enable (from other DAI)
1	Reserved	001011 (0xB)	SPT4_AD1_PBEN_O	SPORT 4 Data AD1 Output Enable (from other DAI)
1	Reserved	001100 (0xC)	SPT4_BCLK_PBEN_O	SPORT 4 Clock B Output Enable (from other DAI)
1	Reserved	001101 (0xD)	SPT4_BFS_PBEN_O	SPORT 4 Frame Sync B Output Enable (from other DAI) (from other DAI)
1	Reserved	001110 (0xE)	SPT4_BD0_PBEN_O	SPORT 4 Data BD0 Output Enable (from other DAI)
1	Reserved	001111 (0xF)	SPT4_BD1_PBEN_O	SPORT 4 Data BD1 Output Enable (from other DAI)
1	Reserved	010000 (0x10)	SPT5_ACLK_PBEN_O	SPORT 5 Clock A Output Enable (from other DAI)
1	Reserved	010001 (0x11)	SPT5_AFS_PBEN_O	SPORT 5 Frame Sync A Output Enable (from other DAI)
1	Reserved	010010 (0x12)	SPT5_AD0_PBEN_O	SPORT 5 Data AD0 Output Enable (from other DAI)
1	Reserved	010011 (0x13)	SPT5_AD1_PBEN_O	SPORT 5 Data AD1 Output Enable (from other DAI)
1	Reserved	010100 (0x14)	SPT5_BCLK_PBEN_O	SPORT 5 Clock B Output Enable (from other DAI)
1	Reserved	010101 (0x15)	SPT5_BFS_PBEN_O	SPORT 5 Frame Sync B Output Enable (from other DAI)
1	Reserved	010110 (0x16)	SPT5_BD0_PBEN_O	SPORT 5 Data BD0 Output Enable (from other DAI)

Table 31-15: Group F – Pin Output Enable (Continued)

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	010111 (0x17)	SPT5_BD1_PBEN_O	SPORT 5 Data BD1 Output Enable (from other DAI)
1	Reserved	011000 (0x18)	SPT6_ACLK_PBEN_O	SPORT 6 Clock A Output Enable (from other DAI)
1	Reserved	011001 (0x19)	SPT6_AFS_PBEN_O	SPORT 6 Frame Sync A Output Enable (from other DAI)
1	Reserved	011010 (0x1A)	SPT6_AD0_PBEN_O	SPORT 6 Data AD0 Output Enable (from other DAI)
1	Reserved	011011 (0x1B)	SPT6_AD1_PBEN_O	SPORT 6 Data AD1 Output Enable (from other DAI)
1	Reserved	011100 (0x1C)	SPT6_BCLK_PBEN_O	SPORT 6 Clock B Output Enable (from other DAI)
1	Reserved	011101 (0x1D)	SPT6_BFS_PBEN_O	SPORT 6 Frame Sync B Output Enable (from other DAI)
1	Reserved	011110 (0x1E)	SPT6_BD0_PBEN_O	SPORT 6 Data BD0 Output Enable (from other DAI)
1	Reserved	011111 (0x1F)	SPT6_BD1_PBEN_O	SPORT 6 Data BD1 Output Enable (from other DAI)
1	Reserved	100000 (0x20)	SPT7_ACLK_PBEN_O	SPORT 7 Clock A Output Enable (from other DAI)
1	Reserved	100001 (0x21)	SPT7_AFS_PBEN_O	SPORT 7 Frame Sync A Output Enable (from other DAI)
1	Reserved	100010 (0x22)	SPT7_AD0_PBEN_O	SPORT 7 Data AD0 Output Enable (from other DAI)
1	Reserved	100011 (0x23)	SPT7_AD1_PBEN_O	SPORT 7 Data AD1 Output Enable (from other DAI)
1	Reserved	100100 (0x24)	SPT7_BCLK_PBEN_O	SPORT 6 Clock B Output Enable (from other DAI)
1	Reserved	100101 (0x25)	SPT7_BFS_PBEN_O	SPORT 7 Frame Sync B Output Enable (from other DAI)
1	Reserved	100110 (0x26)	SPT7_BD0_PBEN_O	SPORT 7 Data BD0 Output Enable (from other DAI)
1	Reserved	100111 (0x27)	SPT7_BD1_PBEN_O	SPORT 7 Data BD1 Output Enable (from other DAI)

Table 31-15: Group F – Pin Output Enable (Continued)

Extended Selection Code [1:0]		Selection Code [5:0]	Source Signal	Description (Source Selection)
Bit 1	**Bit 0**			
1	Reserved	101000 (0x28)	SPT4_ATDV_PBEN_O	SPORT 4 A Transmit Data Valid Output (from other DAI)
1	Reserved	101001 (0x29)	SPT4_BTDV_PBEN_O	SPORT 4 B Transmit Data Valid Output (from other DAI)
1	Reserved	101010 (0x2A)	SPT5_ATDV_PBEN_O	SPORT 5 A Transmit Data Valid Output (from other DAI)
1	Reserved	101011 (0x2B)	SPT5_BTDV_PBEN_O	SPORT 5 B Transmit Data Valid Output (from other DAI)
1	Reserved	101100 (0x2C)	SPT6_ATDV_PBEN_O	SPORT 6 A Transmit Data Valid Output (from other DAI)
1	Reserved	101101 (0x2D)	SPT6_BTDV_PBEN_O	SPORT 6 B Transmit Data Valid Output (from other DAI)
1	Reserved	100111 (0x2E)	SPT7_ATDV_PBEN_O	SPORT 7 A Transmit Data Valid Output (from other DAI)
1	Reserved	101110 (0x2F)	SPT7_BTDV_PBEN_O	SPORT 7 B Transmit Data Valid Output (from other DAI)
1	Reserved	110000 (0x30)	PDM1_CLK0_OE_O	PDM1 Clock0 Output Enable (from other DAI)
1	Reserved	110001 (0x31)	PDM1_SDATA_OE_O	PDM1 Serial Data Output Enable
1	Reserved	110010 (0x32)–1111111 (0x3F)	Reserved	

DAI Destination Registers Overview

The tables in the DAI Routing Capabilities section provide high level descriptions that illustrate source (output) connections to destinations (inputs) depending on the routing groups A through F. These tables list the various input fields (INx), which are described in detail in the register descriptions section.

Table 31-16: Clock Destination Registers (Group A)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_CLK0	IN0	SPT0_ACLK_I	SPT4_ACLK_I	SPORT0A or SPORT4A Clock
	IN1	SPT0_BCLK_I	SPT4_BCLK_I	SPORT0B or SPORT4B Clock
	IN2	SPT1_ACLK_I	SPT5_ACLK_I	SPORT1A or SPORT5A Clock

Table 31-16: Clock Destination Registers (Group A) (Continued)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
	IN3	SPT1_BCLK_I	SPT5_BCLK_I	SPORT1B or SPORT5B Clock
	IN4	SPT2_ACLK_I	SPT6_ACLK_I	SPORT2A or SPORT6A Clock
	IN5	SPT2_BCLK_I	SPT6_BCLK_I	SPORT2B or SPORT6B Clock
DAI_CLK1	IN0	SRC0_CLK_IP_I	SRC4_CLK_IP_I	SRC0 or SRC4 Clock Input
	IN1	SRC0_CLK_OP_I	SRC4_CLK_OP_I	SRC0 or SRC4 Clock Output
	IN2	SRC1_CLK_IP_I	SRC5_CLK_IP_I	SRC1 or SRC5 Clock Input
	IN3	SRC1_CLK_OP_I	SRC5_CLK_OP_I	SRC1 or SRC5 Clock Output
	IN4	SRC2_CLK_IP_I	SRC6_CLK_IP_I	SRC2 or SRC6 Clock Input
	IN5	SRC2_CLK_OP_I	SRC6_CLK_OP_I	SRC2 or SRC6 Clock Output
DAI_CLK2	IN0	SRC3_CLK_IP_I	SRC7_CLK_IP_I	SRC3 or SRC7 Clock Input
	IN1	SRC3_CLK_OP_I	SRC7_CLK_OP_I	SRC3 or SRC7 Clock Output
	IN2	SPDIF0_TX_CLK_I	SPDIF1_TX_CLK_I	SPDIF0 or SPDIF1 TX Clock
	IN3	PDM0_CLK0_I	PDM1_CLK0_I	PDM0 or PDM1 Clock0 Input
	IN4	PDM0_BCLK_I	PDM1_BCLK_I	PDM0 or PDM1 Bit Clock Input
DAI_CLK3	IN5	SPDIF0_TX_HFCLK_I	SPDIF1_TX_HFCLK_I	SPDIF0 or SPDIF1 TX HF Clock
DAI_CLK4	IN0	PCG0_EXTCLKA_I	PCG0_EXTCLKC_I	PCG0 External Clock A or C
	IN1	PCG0_EXTCLKB_I	PCG0_EXTCLKD_I	PCG0 External Clock B or D
	IN3	SPDIF0_TX_EXT_SYNC_I	SPDIF1_TX_EXT_SYNC_I	SPDIF0 or SPDIF1 TX External Sync
	IN4	PCG0_SYNC_CLKA_I	PCG0_SYNC_CLKC_I	PCG0 Sync Clock A or C
	IN5	PCG0_SYNC_CLKB_I	PCG0_SYNC_CLKD_I	PCG0 Sync Clock B or D
DAI_CLK5	IN0	SPT3_ACLK_I	SPT7_ACLK_I	SPORT3A or SPORT 7A Clock
	IN1	SPT3_BCLK_I	SPT7_BCLK_I	SPORT3B or SPORT 7B Clock
	IN2	PCG0_SYNC_CLKE_I	PCG0_SYNC_CLKG_I	PCG0 Sync Clock E or G
	IN3	PCG0_SYNC_CLKF_I	PCG0_SYNC_CLKH_I	PCG0 Sync Clock F or H
	IN4	PCG0_EXTCLKE_I	PCG0_EXTCLKG_I	PCG0 External Clock E or G
	IN5	PCG0_EXTCLKF_I	PCG0_EXTCLKH_I	PCG0 External Clock F or H

Table 31-17: Data Destination Registers (Group B)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_DAT0	IN0	SPT0_AD0_I	SPT4_AD0_I	SPORT0A or SPORT4A Primary Data
	IN1	SPT0_AD1_I	SPT4_AD1_I	SPORT0A or SPORT4A Secondary Data
	IN2	SPT0_BD0_I	SPT4_BD0_I	SPORT0B or SPORT4B Primary Data
	IN3	SPT0_BD1_I	SPT4_BD1_I	SPORT0B or SPORT4B Secondary Data
	IN4	SPT1_AD0_I	SPT5_AD0_I	SPORT1A or SPORT5A Primary Data
DAI_DAT1	IN0	SPT1_AD1_I	SPT5_AD1_I	SPORT1A or SPORT5A Secondary Data
	IN1	SPT1_BD0_I	SPT5_BD0_I	SPORT1B or SPORT5B Primary Data
	IN2	SPT1_BD1_I	SPT5_BD1_I	SPORT1B or SPORT5B Secondary Data
	IN3	SPT2_AD0_I	SPT6_AD0_I	SPORT2A or SPORT6A Primary Data
	IN4	SPT2_AD1_I	SPT6_AD1_I	SPORT2A or SPORT6A Secondary Data
DAI_DAT2	IN0	SPT2_BD0_I	SPT6_BD0_I	SPORT2B or SPORT6B Primary Data
	IN1	SPT2_BD1_I	SPT6_BD1_I	SPORT2B or SPORT6B Secondary Data
	IN2	SRC0_DAT_IP_I	SRC4_DAT_IP_I	SRC0 or SRC4 Data
	IN3	SRC1_DAT_IP_I	SRC5_DAT_IP_I	SRC1 or SRC5 Data
	IN4	SRC2_DAT_IP_I	SRC6_DAT_IP_I	SRC2 or SRC6 Data
DAI_DAT3	IN0	SRC3_DAT_IP_I	SRC7_DAT_IP_I	SRC3 or SRC7 Data
	IN1	SRC0_DAT_TDM_OP_I	SRC4_DAT_TDM_OP_I	SRC0 or SRC4 TDM output port
	IN2	SRC1_DAT_TDM_OP_I	SRC5_DAT_TDM_OP_I	SRC1 or SRC5 TDM Output port Data
	IN3	SRC2_DAT_TDM_OP_I	SRC6_DAT_TDM_OP_I	SRC2 or SRC6 TDM Output port Data
	IN4	SRC3_DAT_TDM_OP_I	SRC7_DAT_TDM_OP_I	SRC3 or SRC7 TDM Output port Data
DAI_DAT4	IN0	SPDIF0_TX_DAT_I	SPDIF1_TX_DAT_I	SPDIF0 or SPDIF1 serial transmitter Data
	IN1	PDM0_DAT0_I	PDM1_DAT0_I	PDM0 or PDM1 Data0 Input
	IN2	PDM0_DAT1_I	PDM1_DAT1_I	PDM0 or PDM1 Data1 Input

Table 31-17: Data Destination Registers (Group B) (Continued)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_DAT5	IN4	SPDIF0_RX_I	SPDIF1_RX_I	SPDIF0 or SPDIF1 receiver bi-phase Data
DAI_DAT6	IN0	SPT3_AD0_I	SPT7_AD0_I	SPORT3A or SPORT7A Primary Data
	IN1	SPT3_AD1_I	SPT7_AD1_I	SPORT3A or SPORT7A Secondary Data
	IN2	SPT3_BD0_I	SPT7_BD0_I	SPORT3B or SPORT7B Primary Data
	IN3	SPT3_BD1_I	SPT7_BD1_I	SPORT3B or SPORT7B Secondary Data
	IN4	Reserved	Reserved	

Table 31-18: Frame Sync Destination Registers (Group C)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_FS0	IN0	SPT0_AFS_I	SPT4_AFS_I	SPORT0A or SPORT4A Frame Sync
	IN1	SPT0_BFS_I	SPT4_BFS_I	SPORT0B or SPORT4B Frame Sync
	IN2	SPT1_AFS_I	SPT5_AFS_I	SPORT1A or SPORT5A Frame Sync
	IN3	SPT1_BFS_I	SPT5_BFS_I	SPORT1B or SPORT5B Frame Sync
	IN4	SPT2_AFS_I	SPT6_AFS_I	SPORT2A or SPORT6A Frame Sync
	IN5	SPT2_BFS_I	SPT6_BFS_I	SPORT2B or SPORT6B Frame Sync
DAI_FS1	IN0	SRC0_FS_IP_I	SRC4_FS_IP_I	SRC0 or SRC4 Frame Sync Input
	IN1	SRC0_FS_OP_I	SRC4_FS_OP_I	SRC0 or SRC4 Frame Sync Output
	IN2	SRC1_FS_IP_I	SRC5_FS_IP_I	SRC1 or SRC5 Frame Sync Input
	IN3	SRC1_FS_OP_I	SRC5_FS_OP_I	SRC1 or SRC5 Frame Sync Output
	IN4	SRC2_FS_IP_I	SRC6_FS_IP_I	SRC2 or SRC6 Frame Sync Input
	IN5	SRC2_FS_OP_I	SRC6_FS_OP_I	SRC2 or SRC6 Frame Sync Output
DAI_FS2	IN0	SRC3_FS_IP_I	SRC7_FS_IP_I	SRC3 or SRC7 Frame Sync Input
	IN1	SRC3_FS_OP_I	SRC7_FS_OP_I	SRC3 or SRC7 Frame Sync Output
	IN2	SPDIF0_TX_FS_I	SPDIF1_TX_FS_I	SPDIF0 or SPDIF1 Transmit Frame Sync

Table 31-18: Frame Sync Destination Registers (Group C) (Continued)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_FS4	IN0	SPT3_AFS_I	SPT7_AFS_I	SPORT3A or SPORT7A Frame Sync
	IN1	SPT3_BFS_I	SPT7_BFS_I	SPORT3B or SPORT7B Frame Sync
	IN2	TM0_ACI14_I	TM0_ACI15_I	ACI 14 or ACI 15 Timer Input
	IN3	PDM0_LRCLK_I	PDM1_LRCLK_I	PDM0 or PDM1 Left/Right Clock or Frame Sync Input

Table 31-19: Pin Buffer Assignment Destination Registers (Group D)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_PIN0	IN0	DAI0_PB01_I	DAI1_PB01_I	Pin Buffer 1
	IN1	DAI0_PB02_I	DAI1_PB02_I	Pin Buffer 2
	IN2	DAI0_PB03_I	DAI1_PB03_I	Pin Buffer 3
	IN3	DAI0_PB04_I	DAI1_PB04_I	Pin Buffer 4
DAI_PIN1	IN0	DAI0_PB05_I	DAI1_PB05_I	Pin Buffer 5
	IN1	DAI0_PB06_I	DAI1_PB06_I	Pin Buffer 6
	IN2	DAI0_PB07_I	DAI1_PB07_I	Pin Buffer 7
	IN3	DAI0_PB08_I	DAI1_PB08_I	Pin Buffer 8
DAI_PIN2	IN0	DAI0_PB09_I	DAI1_PB09_I	Pin Buffer 9
	IN1	DAI0_PB10_I	DAI1_PB10_I	Pin Buffer 10
	IN2	DAI0_PB11_I	DAI1_PB11_I	Pin Buffer 11
	IN3	DAI0_PB12_I	DAI1_PB12_I	Pin Buffer 12
DAI_PIN3	IN0	DAI0_PB13_I	DAI1_PB13_I	Pin Buffer 13
	IN1	DAI0_PB14_I	DAI1_PB14_I	Pin Buffer 14
	IN2	DAI0_PB15_I	DAI1_PB15_I	Pin Buffer 15
	IN3	DAI0_PB16_I	DAI1_PB16_I	Pin Buffer 16
DAI_PIN4	IN0	DAI0_PB17_I	DAI1_PB17_I	Pin Buffer 17
	IN1	DAI0_PB18_I	DAI1_PB18_I	Pin Buffer 18
	IN2	DAI0_PB19_I	DAI1_PB19_I	Pin Buffer 19
	IN3	DAI0_PB20_I	DAI1_PB20_I	Pin Buffer 20
	IN4	INV_DAI0_PB19_I	INV_DAI1_PB19_I	Inverted Pin Buffer 19
	IN5	INV_DAI0_PB20_I	INV_DAI1_PB20_I	Inverted Pin Buffer 20

In #c_dai_RegisterDescriptions_NotYoda_2156x/MiscControlDestinationRegistersGroupE , DAI_MISCAx_I is an alias name for DAI_INT_x_I.

Table 31-20: Miscellaneous Control Destination Registers (Group E)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_MISC0	IN6	DAI0_MISCA0_I/ DAI0_INT_6_I	DAI1_MISCA0_I/ DAI1_INT_6_I	DAIx miscellaneous A0 input, or DAIx interrupt 28
	IN7	DAI0_MISCA1_I/ DAI0_INT_7_I	DAI1_MISCA1_I/ DAI1_INT_7_I	DAIx miscellaneous A1 input, or DAIx interrupt 29
	IN8	DAI0_MISCA2_I/ DAI0_INT_8_I	DAI1_MISCA2_I/ DAI1_INT_8_I	DAIx miscellaneous A2 input, or DAIx interrupt 30
	IN9	DAI0_MISCA3_I/ DAI0_INT_9_I	DAI1_MISCA3_I/ DAI1_INT_9_I	DAIx miscellaneous A3 input, or DAIx interrupt 31
	IN10	DAI0_MISCA4_I	DAI1_MISCA4_I	DAIx miscellaneous A4 input
	IN11	DAI0_MISCA5_I	DAI1_MISCA5_I	DAIx miscellaneous A5 input
	IN10	DAI0_INV_MISCA4_I	DAI1_INV_MISCA4_I	DAIx inverted miscellaneous A4 input
	IN11	DAI0_INV_MISCA4_I	DAI1_INV_MISCA5_I	DAIx inverted miscellaneous A5 input
DAI_MISC1	IN0	DAI0_INT_0	DAI1_INT_0	DAIx interrupt 22
	IN1	DAI0_INT_1	DAI1_INT_1	DAIx interrupt 23
	IN2	DAI0_INT_2	DAI1_INT_2	DAIx interrupt 24
	IN3	DAI0_INT_3	DAI1_INT_3	DAIx interrupt 25
	IN4	DAI0_INT_4	DAI1_INT_4	DAIx interrupt 26
	IN5	DAI0_INT_5	DAI1_INT_5	DAIx interrupt 27
DAI_MISC2	IN0	PCG_HWA_TRIG_I	PCG_HWC_TRIG_I	PCG hardware trigger input A or C
	IN1	PCG_HWB_TRIG_I	PCG_HWD_TRIG_I	PCG hardware trigger input B or D
	IN2	PCG_HWE_TRIG_I	PCG_HWG_TRIG_I	PCG hardware trigger input E or G
	IN3	PCG_HWF_TRIG_I	PCG_HWH_TRIG_I	PCG hardware trigger input A or C

Table 31-21: Pin Buffer Enable Destination Registers (Group F)

DAI Register	Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
DAI_PBEN0	IN0	DAI0_PBEN01_I	DAI1_PBEN01_I	Pin Buffer 1 Enable
	IN1	DAI0_PBEN02_I	DAI1_PBEN02_I	Pin Buffer 2 Enable
	IN2	DAI0_PBEN03_I	DAI1_PBEN03_I	Pin Buffer 3 Enable
	IN3	DAI0_PBEN04_I	DAI1_PBEN04_I	Pin Buffer 4 Enable
	IN4	DAI0_PBEN05_I	DAI1_PBEN05_I	Pin Buffer 5 Enable
DAI_PBEN1	IN0	DAI0_PBEN06_I	DAI1_PBEN06_I	Pin Buffer 6 Enable
	IN1	DAI0_PBEN07_I	DAI1_PBEN07_I	Pin Buffer 7 Enable
	IN2	DAI0_PBEN08_I	DAI1_PBEN08_I	Pin Buffer 8 Enable
	IN3	DAI0_PBEN09_I	DAI1_PBEN09_I	Pin Buffer 9 Enable
	IN4	DAI0_PBEN10_I	DAI1_PBEN10_I	Pin Buffer 10 Enable
DAI_PBEN2	IN0	DAI0_PBEN11_I	DAI1_PBEN11_I	Pin Buffer 11 Enable
	IN1	DAI0_PBEN12_I	DAI1_PBEN12_I	Pin Buffer 12 Enable
	IN2	DAI0_PBEN13_I	DAI1_PBEN13_I	Pin Buffer 13 Enable
	IN3	DAI0_PBEN14_I	DAI1_PBEN14_I	Pin Buffer 14 Enable
	IN4	DAI0_PBEN15_I	DAI1_PBEN15_I	Pin Buffer 15 Enable
DAI_PBEN3	IN0	DAI0_PBEN16_I	DAI1_PBEN16_I	Pin Buffer 16 Enable
	IN1	DAI0_PBEN17_I	DAI1_PBEN17_I	Pin Buffer 17 Enable
	IN2	DAI0_PBEN18_I	DAI1_PBEN18_I	Pin Buffer 18 Enable
	IN3	DAI0_PBEN19_I	DAI1_PBEN19_I	Pin Buffer 19 Enable
	IN4	DAI0_PBEN20_I	DAI1_PBEN20_I	Pin Buffer 20 Enable

The ***Interrupt Events*** table presents the information for the following registers.

- DAI_IMSK_PRI (Core Interrupt Priority Assignment) register
- DAI_IMSK_FE (Falling-Edge Interrupt Mask) and DAI_IMSK_RE (Rising-Edge Interrupt Mask) registers
- DAI_IRPTL_H (High-Priority Interrupt Latch) and DAI_IRPTL_L (Low-Priority Interrupt Latch) registers
- DAI_IRPTL_HS (Shadow High Interrupt Latch) and DAI_IRPTL_LS (Shadow Low Interrupt Latch) registers

Table 31-22: Interrupt Events

Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
RXVALID	SPDIF0_RXVALID_INT	SPDIF1_RXVALID_INT	SPDIF0 or SPDIF1 RX Valid interrupt

Table 31-22: Interrupt Events (Continued)

Bit Field Name	DAI0 Mapping	DAI1 Mapping	Description
RXLOCK	SPDIF0_RXLOCK_INT	SPDIF1_RXLOCK_INT	SPDIF0 or SPDIF1 RX Lock interrupt
RXLOSSOFLOCK	SPDIF0_RXLOSSOFLOCK_INT	SPDIF1_RXLOSSOFLOCK_INT	SPDIF0 or SPDIF1 RX Loss of Lock interrupt
RXNONAUDIO	SPDIF0_RXNONAUDIO_INT	SPDIF1_RXNONAUDIO_INT	SPDIF0 or SPDIF1 RX Non-audio interrupt
SRC0MUTE	SRC0_MUTE_INT	SRC4_MUTE_INT	DAI0 or DAI1 ASRC0 Mute interrupt
SRC1MUTE	SRC1_MUTE_INT	SRC5_MUTE_INT	DAI0 or DAI1 ASRC1 Mute interrupt
SRC2MUTE	SRC2_MUTE_INT	SRC6_MUTE_INT	DAI0 or DAI1 ASRC2 Mute interrupt
SRC3MUTE	SRC3_MUTE_INT	SRC7_MUTE_INT	DAI0 or DAI1 ASRC3 Mute interrupt
MISCINT0	DAI0_INT_00	DAI1_INT_00	DAI0 or DAI1 Miscellaneous Interrupt 0
MISCINT1	DAI0_INT_01	DAI1_INT_01	DAI0 or DAI1 Miscellaneous Interrupt 1
MISCINT2	DAI0_INT_02	DAI1_INT_02	DAI0 or DAI1 Miscellaneous Interrupt 2
MISCINT3	DAI0_INT_03	DAI1_INT_03	DAI0 or DAI1 Miscellaneous Interrupt 3
MISCINT4	DAI0_INT_04	DAI1_INT_04	DAI0 or DAI1 Miscellaneous Interrupt 4
MISCINT5	DAI0_INT_05	DAI1_INT_05	DAI0 or DAI1 Miscellaneous Interrupt 5
MISCINT6	DAI0_EXTMISCA0_INT/ DAI0_INT_06	DAI1_EXTMISCA0_INT/ DAI1_INT_06	DAI0 or DAI1 Miscellaneous Interrupt 6 / External Miscellaneous A0 Interrupt
MISCINT7	DAI0_EXTMISCA1_INT/ DAI0_INT_07	DAI1_EXTMISCA1_INT/ DAI1_INT_07	DAI0 or DAI1 Miscellaneous Interrupt 7 / External Miscellaneous A1 Interrupt
MISCINT8	DAI0_EXTMISCA2_INT/ DAI0_INT_08	DAI1_EXTMISCA2_INT/ DAI1_INT_08	DAI0 or DAI1 Miscellaneous Interrupt 8 / External Miscellaneous A2 Interrupt
MISCINT9	DAI0_EXTMISCA3_INT/ DAI0_INT_09	DAI1_EXTMISCA3_INT/ DAI1_INT_09	DAI0 or DAI1 Miscellaneous Interrupt 9 / External Miscellaneous A3 Interrupt

SPORT Grouping with DAI

DAI supports two registers DAI_GBL_SP_EN and DAI_GBL_INT_EN to group enabling/disabling and interrupt/trigger generation/reception for multiple SPORTs.

For more information about this, refer to the Grouping of SPORTs section.

Global PCG Enable

The DAI module includes the DAI_GBL_PCG_EN register that enables and disables multiple PCGs globally at the same time. For more information, see Global PCG Enable section in the PCG chapter.

ADSP-SC59x DAI Register Descriptions

The Digital Audio Interface (DAIn) registers are used to configure DAI destinations for the DAI sources shown in the Group A - F Routing tables. (DAI) contains the following registers.

Table 31-23: ADSP-SC59x DAI Register List

Name	Description
DAI_CLK0	Clock Routing Control Register 0
DAI_CLK1	Clock Routing Control Register 1
DAI_CLK2	Clock Routing Control Register 2
DAI_CLK3	Clock Routing Control Register 3
DAI_CLK4	Clock Routing Control Register 4
DAI_CLK5	Clock Routing Control Register 5
DAI_DAT0	Serial Data Routing Control Register 0
DAI_DAT1	Serial Data Routing Control Register 1
DAI_DAT2	Serial Data Routing Control Register 2
DAI_DAT3	Serial Data Routing Control Register 3
DAI_DAT4	Serial Data Routing Control Register 4
DAI_DAT5	Serial Data Routing Control Register 5
DAI_DAT6	Serial Data Routing Control Register 6
DAI_EXTD_CLK0	Extended Clock Routing Control Register 0
DAI_EXTD_CLK1	Extended Clock Routing Control Register 1
DAI_EXTD_CLK2	Extended Clock Routing Control Register 2
DAI_EXTD_CLK3	Extended Clock Routing Control Register 3
DAI_EXTD_CLK4	Extended Clock Routing Control Register 4

Table 31-23: ADSP-SC59x DAI Register List (Continued)

Name	Description
DAI_EXTD_CLK5	Extended Clock Routing Control Register 5
DAI_EXTD_DAT0	Extended Serial Data Routing Control Register 0
DAI_EXTD_DAT1	Extended Serial Data Routing Control Register 1
DAI_EXTD_DAT2	Extended Serial Data Routing Control Register 2
DAI_EXTD_DAT3	Extended Serial Data Routing Control Register 3
DAI_EXTD_DAT4	Extended Serial Data Routing Control Register 4
DAI_EXTD_DAT5	Extended Serial Data Routing Control Register 5
DAI_EXTD_DAT6	Extended Serial Data Routing Control Register 6
DAI_EXTD_FS0	Extended Frame Sync Routing Control Register 0
DAI_EXTD_FS1	Extended Frame Sync Routing Control Register 1
DAI_EXTD_FS2	Extended Frame Sync Routing Control Register 2
DAI_EXTD_FS4	Extended Frame Sync Routing Control Register 4
DAI_EXTD_MISC0	Extended Miscellaneous Control Register 0
DAI_EXTD_MISC1	Extended Miscellaneous Control Register 1
DAI_EXTD_MISC2	Extended Miscellaneous Control Register 2
DAI_EXTD_PBEN0	Extended Pin Buffer Enable Register 0
DAI_EXTD_PBEN1	Extended Pin Buffer Enable Register 1
DAI_EXTD_PBEN2	Extended Pin Buffer Enable Register 2
DAI_EXTD_PBEN3	Extended Pin Buffer Enable Register 3
DAI_EXTD_PIN0	Extended Pin Buffer Assignment Register 0
DAI_EXTD_PIN1	Extended Pin Buffer Assignment Register 1
DAI_EXTD_PIN2	Extended Pin Buffer Assignment Register 2
DAI_EXTD_PIN3	Extended Pin Buffer Assignment Register 3
DAI_EXTD_PIN4	Extended Pin Buffer Assignment Register 4
DAI_FS0	Frame Sync Routing Control Register 0
DAI_FS1	Frame Sync Routing Control Register 1
DAI_FS2	Frame Sync Routing Control Register 2
DAI_FS4	Frame Sync Routing Control Register 4
DAI_GBL_INT_EN	Global SPORT Interrupt Grouping Register
DAI_GBL_PCG_EN	Global PCG Enable Control Register
DAI_GBL_SP_EN	Global SPORT Enable Register

Table 31-23: ADSP-SC59x DAI Register List (Continued)

Name	Description
DAI_IMSK_FE	Falling-Edge Interrupt Mask Register
DAI_IMSK_PRI	Core Interrupt Priority Assignment Register
DAI_IMSK_RE	Rising-Edge Interrupt Mask Register
DAI_IRPTL_H	High Priority Interrupt Latch Register
DAI_IRPTL_HS	Shadow High Priority Interrupt Latch Register
DAI_IRPTL_L	Low Priority Interrupt Latch Register
DAI_IRPTL_LS	Shadow Low Priority Interrupt Latch Register
DAI_MISC0	Miscellaneous Control Register 0
DAI_MISC1	Miscellaneous Control Register 1
DAI_MISC2	Miscellaneous Control Register 1
DAI_PBEN0	Pin Buffer Enable Register 0
DAI_PBEN1	Pin Buffer Enable Register 1
DAI_PBEN2	Pin Buffer Enable Register 2
DAI_PBEN3	Pin Buffer Enable Register 3
DAI_PIN0	Pin Buffer Assignment Register 0
DAI_PIN1	Pin Buffer Assignment Register 1
DAI_PIN2	Pin Buffer Assignment Register 2
DAI_PIN3	Pin Buffer Assignment Register 3
DAI_PIN4	Pin Buffer Assignment Register 4
DAI_PIN_STAT	Pin Status Register

Clock Routing Control Register 0

The DAI_CLK0 register provides clock routing connections for the serial ports (SPORTs).

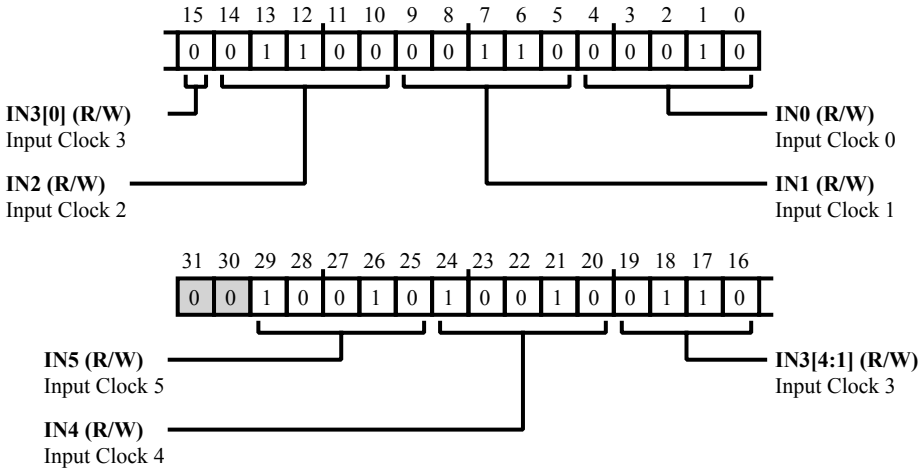


Figure 31-14: DAI_CLK0 Register Diagram

Table 31-24: DAI_CLK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. DAI_CLK0.IN5 holds the Source signal assignment that will be routed to the DAI_CLK0.IN5 Destination. Refer to the Group A Signals table for Source and Destination mappings
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK0.IN4 holds the Source signal assignment that will be routed to the DAI_CLK0.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK0.IN3 holds the Source signal assignment that will be routed to the DAI_CLK0.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings
14:10 (R/W)	IN2	Input Clock 2. DAI_CLK0.IN2 holds the Source signal assignment that will be routed to the DAI_CLK0.IN2 Destination. Refer to the Group A Signals table for Source and Destination mappings
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK0.IN1 holds the Source signal assignment that will be routed to the DAI_CLK0.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings

Table 31-24: DAI_CLK0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Clock 0. `DAI_CLK0.IN0` holds the Source signal assignment that will be routed to the `DAI_CLK0.IN0` Destination. Refer to the Group A Signals table for Source and Destination mappings

Clock Routing Control Register 1

The DAI_CLK1 register provides clock routing connections for the asynchronous sample rate converters (ASRC).

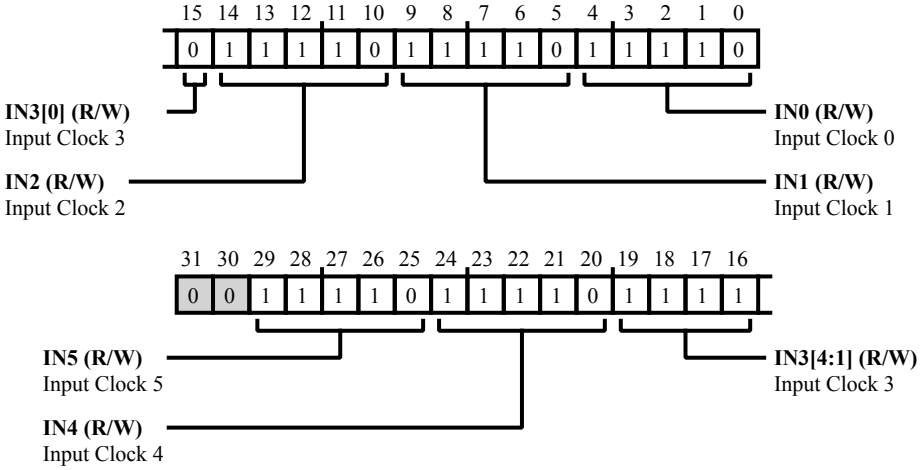


Figure 31-15: DAI_CLK1 Register Diagram

Table 31-25: DAI_CLK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. DAI_CLK1.IN5 holds the Source signal assignment that will be routed to the DAI_CLK1.IN5 Destination. Refer to the Group A Signals table for Source and Destination mappings
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK1.IN4 holds the Source signal assignment that will be routed to the DAI_CLK1.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK1.IN3 holds the Source signal assignment that will be routed to the DAI_CLK1.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings
14:10 (R/W)	IN2	Input Clock 2. DAI_CLK1.IN2 holds the Source signal assignment that will be routed to the DAI_CLK1.IN2 Destination. Refer to the Group A Signals table for Source and Destination mappings
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK1.IN1 holds the Source signal assignment that will be routed to the DAI_CLK1.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings.

Table 31-25: DAI_CLK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Clock 0. `DAI_CLK1.IN0` holds the Source signal assignment that will be routed to the `DAI_CLK1.IN0` Destination. Refer to the Group A Signals table for Source and Destination mappings.

Clock Routing Control Register 2

The DAI_CLK2 register provides clock routing connections for the S/PDIF and ASRC.

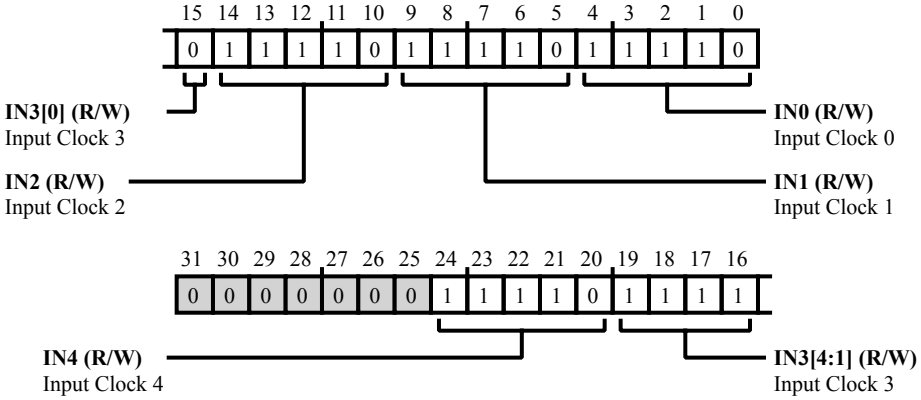


Figure 31-16: DAI_CLK2 Register Diagram

Table 31-26: DAI_CLK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK2.IN4 holds the Source signal assignment that will be routed to the DAI_CLK2.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings.
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK2.IN3 holds the Source signal assignment that will be routed to the DAI_CLK2.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings.
14:10 (R/W)	IN2	Input Clock 2. DAI_CLK2.IN2 holds the Source signal assignment that will be routed to the DAI_CLK2.IN2 Destination. Refer to the Group A Signals table for Source and Destination mappings.
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK2.IN1 holds the Source signal assignment that will be routed to the DAI_CLK2.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings.
4:0 (R/W)	IN0	Input Clock 0. DAI_CLK2.IN0 holds the Source signal assignment that will be routed to the DAI_CLK2.IN0 Destination. Refer to the Group A Signals table for Source and Destination mappings.

Clock Routing Control Register 3

The DAI_CLK3 register provides clock routing connections for the S/PDIF.

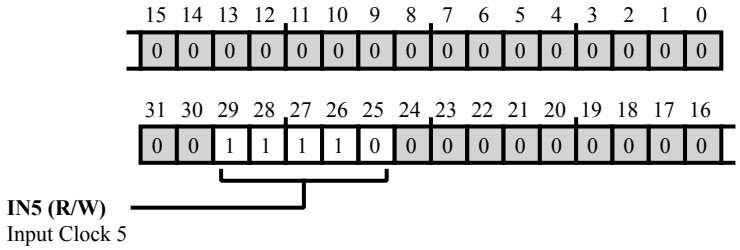


Figure 31-17: DAI_CLK3 Register Diagram

Table 31-27: DAI_CLK3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. The DAI_CLK3.IN5 bit field provides the S/PDIF oversampling clock.

Clock Routing Control Register 4

The DAI_CLK4 register provides clock routing connections for the precision clock generators (PCGs).

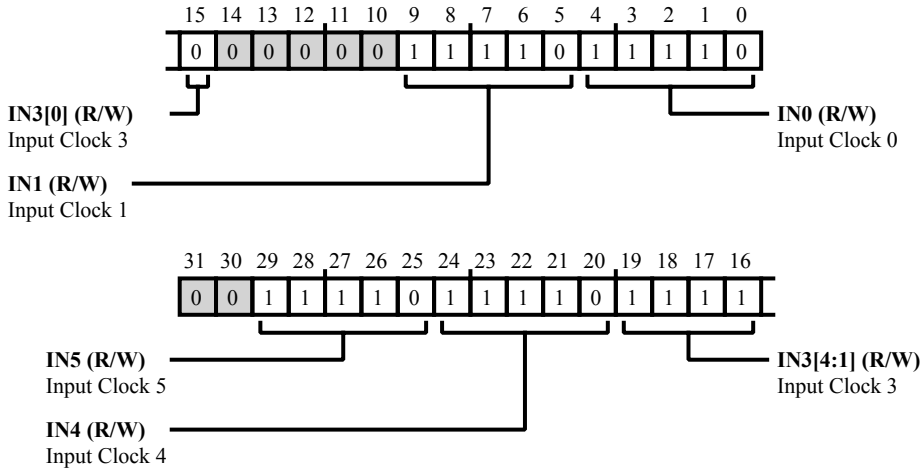


Figure 31-18: DAI_CLK4 Register Diagram

Table 31-28: DAI_CLK4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. DAI_CLK4.IN5 holds the Source signal assignment that will be routed to the DAI_CLK4.IN5 Destination. Refer to the Group A Signals table for Source and Destination mappings.
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK4.IN4 holds the Source signal assignment that will be routed to the DAI_CLK4.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings.
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK4.IN3 holds the Source signal assignment that will be routed to the DAI_CLK4.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings.
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK4.IN1 holds the Source signal assignment that will be routed to the DAI_CLK4.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings.
4:0 (R/W)	IN0	Input Clock 0. DAI_CLK4.IN0 holds the Source signal assignment that will be routed to the DAI_CLK4.IN0 Destination. Refer to the Group A Signals table for Source and Destination mappings.

Clock Routing Control Register 5

The DAI_CLK5 register provides clock routing connections for the serial ports.

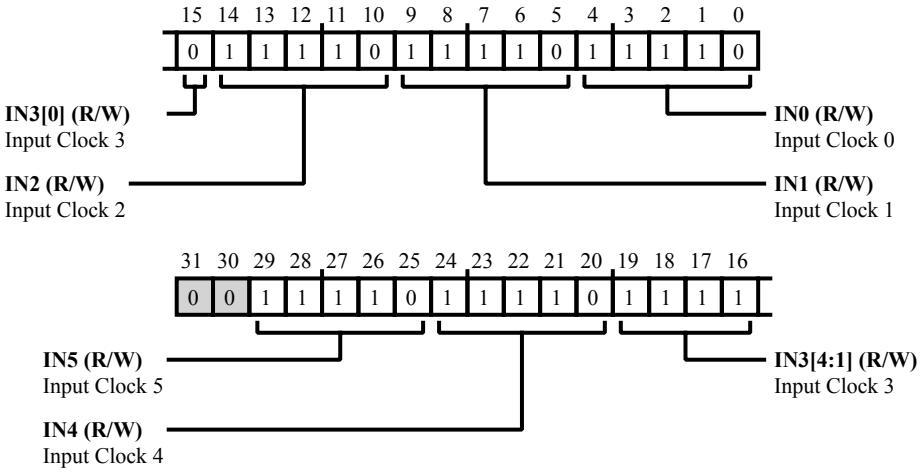


Figure 31-19: DAI_CLK5 Register Diagram

Table 31-29: DAI_CLK5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Clock 5. DAI_CLK5.IN5 holds the Source signal assignment that will be routed to the DAI_CLK5.IN5 Destination. Refer to the Group A Signals table for Source and Destination mappings.
24:20 (R/W)	IN4	Input Clock 4. DAI_CLK5.IN4 holds the Source signal assignment that will be routed to the DAI_CLK5.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings.
19:15 (R/W)	IN3	Input Clock 3. DAI_CLK5.IN3 holds the Source signal assignment that will be routed to the DAI_CLK5.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings.
14:10 (R/W)	IN2	Input Clock 2. DAI_CLK5.IN2 holds the Source signal assignment that will be routed to the DAI_CLK5.IN2 Destination. Refer to the Group A Signals table for Source and Destination mappings.
9:5 (R/W)	IN1	Input Clock 1. DAI_CLK5.IN1 holds the Source signal assignment that will be routed to the DAI_CLK5.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings.

Table 31-29: DAI_CLK5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Clock 0. `DAI_CLK5.IN0` holds the Source signal assignment that will be routed to the `DAI_CLK5.IN0` Destination. Refer to the Group A Signals table for Source and Destination mappings.

Serial Data Routing Control Register 0

The DAI_DAT0 register routes serial data to the serial ports.

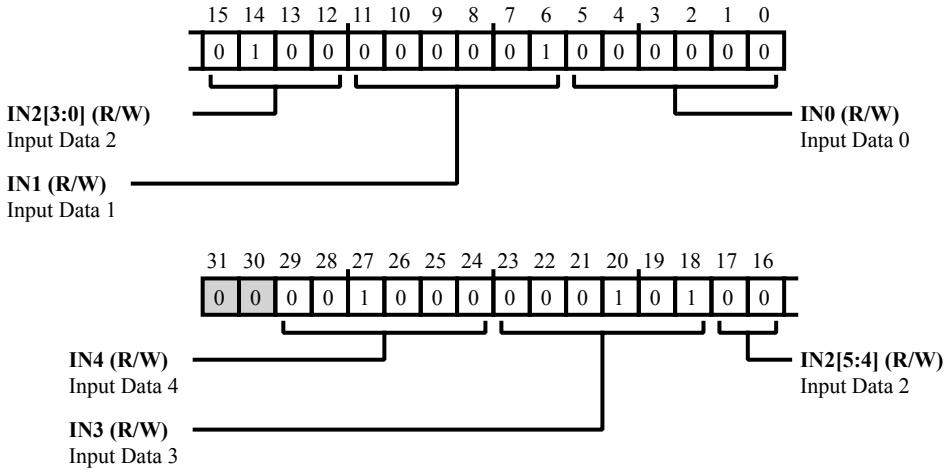


Figure 31-20: DAI_DAT0 Register Diagram

Table 31-30: DAI_DAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. DAI_DAT0.IN4 holds the Source signal assignment that will be routed to the DAI_DAT0.IN4 Destination. Refer to the Group B Signals table for Source and Destination mappings.
23:18 (R/W)	IN3	Input Data 3. DAI_DAT0.IN3 holds the Source signal assignment that will be routed to the DAI_DAT0.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT0.IN2 holds the Source signal assignment that will be routed to the DAI_DAT0.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT0.IN1 holds the Source signal assignment that will be routed to the DAI_DAT0.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT0.IN0 holds the Source signal assignment that will be routed to the DAI_DAT0.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Serial Data Routing Control Register 1

The DAI_DAT1 register routes serial data to the serial ports.

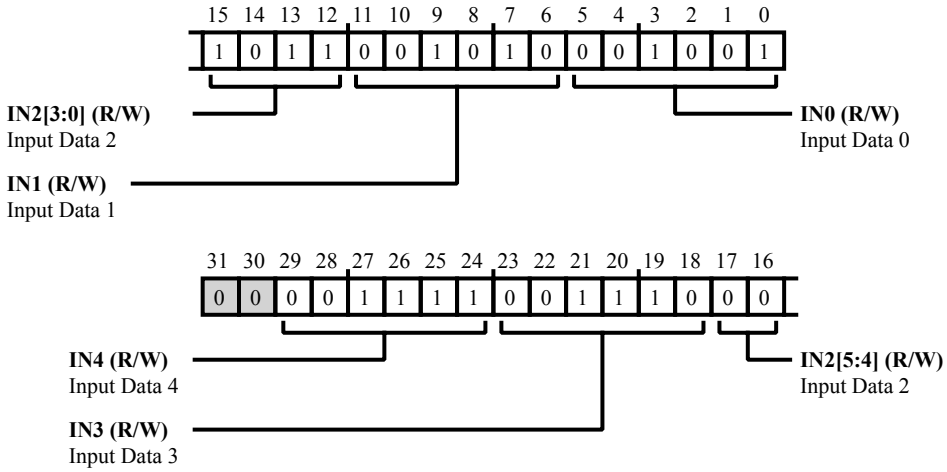


Figure 31-21: DAI_DAT1 Register Diagram

Table 31-31: DAI_DAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. DAI_DAT1.IN4 holds the Source signal assignment that will be routed to the DAI_DAT1.IN4 Destination. Refer to the Group B Signals table for Source and Destination mappings.
23:18 (R/W)	IN3	Input Data 3. DAI_DAT1.IN3 holds the Source signal assignment that will be routed to the DAI_DAT1.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT1.IN2 holds the Source signal assignment that will be routed to the DAI_DAT1.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT1.IN1 holds the Source signal assignment that will be routed to the DAI_DAT1.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT1.IN0 holds the Source signal assignment that will be routed to the DAI_DAT1.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Serial Data Routing Control Register 2

The DAI_DAT2 register routes serial data to the serial ports and the asynchronous sample rate converter (ASRC).

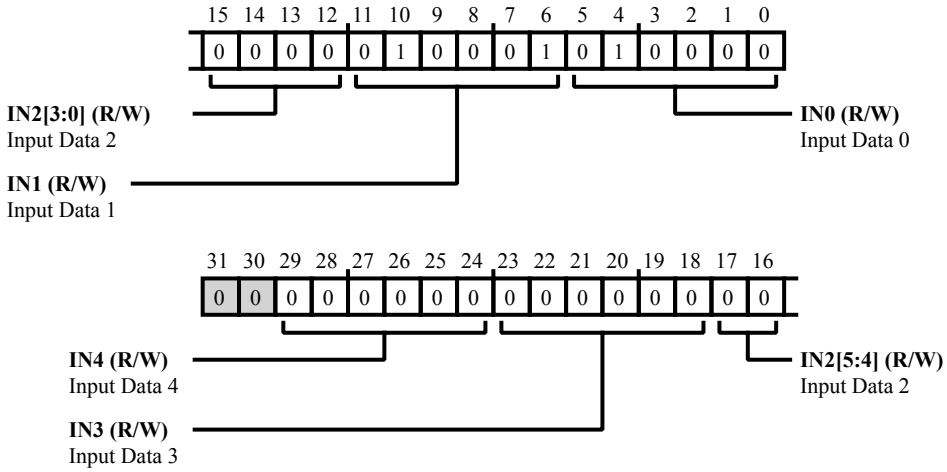


Figure 31-22: DAI_DAT2 Register Diagram

Table 31-32: DAI_DAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. DAI_DAT2.IN4 holds the Source signal assignment that will be routed to the DAI_DAT2.IN4 Destination. Refer to the Group B Signals table for Source and Destination mappings.
23:18 (R/W)	IN3	Input Data 3. DAI_DAT2.IN3 holds the Source signal assignment that will be routed to the DAI_DAT2.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT2.IN2 holds the Source signal assignment that will be routed to the DAI_DAT2.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT2.IN1 holds the Source signal assignment that will be routed to the DAI_DAT2.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT2.IN0 holds the Source signal assignment that will be routed to the DAI_DAT2.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Serial Data Routing Control Register 3

The DAI_DAT3 register routes serial data to the asynchronous sample rate converter (ASRC).

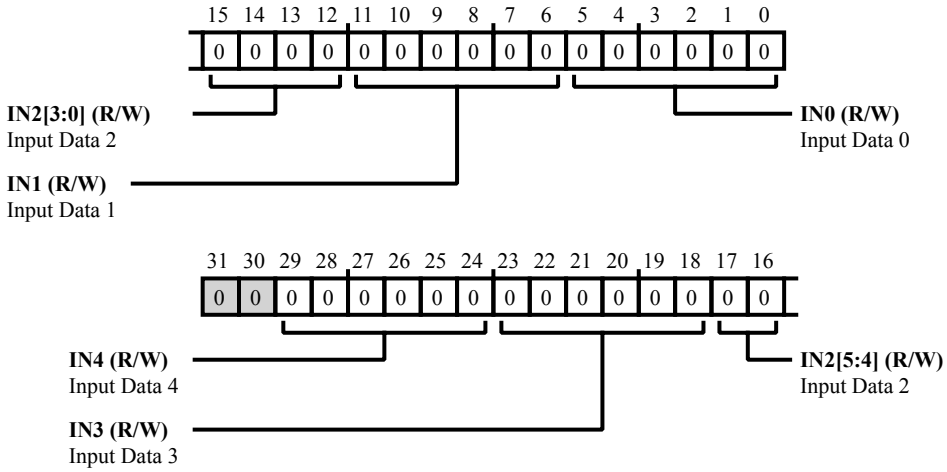


Figure 31-23: DAI_DAT3 Register Diagram

Table 31-33: DAI_DAT3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. DAI_DAT3.IN4 holds the Source signal assignment that will be routed to the DAI_DAT3.IN4 Destination. Refer to the Group B Signals table for Source and Destination mappings.
23:18 (R/W)	IN3	Input Data 3. DAI_DAT3.IN3 holds the Source signal assignment that will be routed to the DAI_DAT3.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT3.IN2 holds the Source signal assignment that will be routed to the DAI_DAT3.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT3.IN1 holds the Source signal assignment that will be routed to the DAI_DAT3.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT3.IN0 holds the Source signal assignment that will be routed to the DAI_DAT3.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Serial Data Routing Control Register 4

The DAI_DAT4 register routes serial data to the S/PDIF.

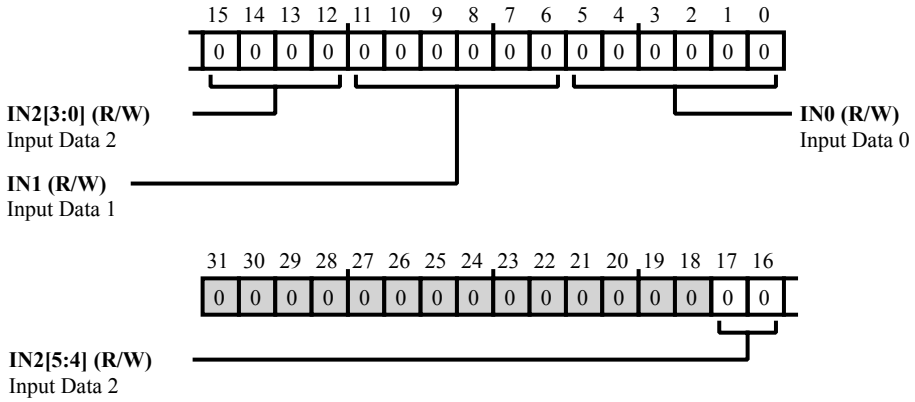


Figure 31-24: DAI_DAT4 Register Diagram

Table 31-34: DAI_DAT4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17:12 (R/W)	IN2	Input Data 2. DAI_DAT4.IN2 holds the Source signal assignment that will be routed to the DAI_DAT4.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT4.IN1 holds the Source signal assignment that will be routed to the DAI_DAT4.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT4.IN0 holds the Source signal assignment that will be routed to the DAI_DAT4.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Serial Data Routing Control Register 5

The DAI_DAT5 register routes serial data to the S/PDIF.

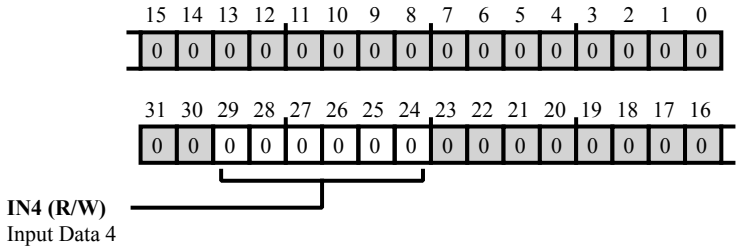


Figure 31-25: DAI_DAT5 Register Diagram

Table 31-35: DAI_DAT5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	IN4	Input Data 4. The DAI_DAT5.IN4 bit field routes input data to the S/PDIF Biphase receiver stream.

Serial Data Routing Control Register 6

The DAI_DAT6 register routes serial data to the serial ports.

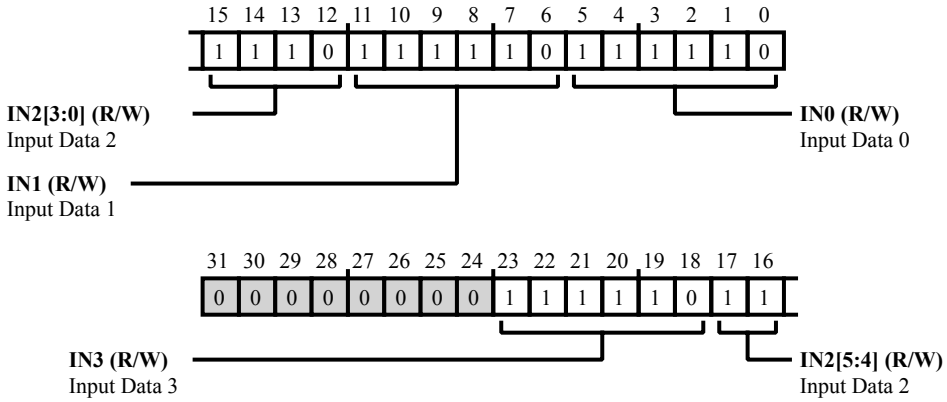


Figure 31-26: DAI_DAT6 Register Diagram

Table 31-36: DAI_DAT6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:18 (R/W)	IN3	Input Data 3. DAI_DAT6.IN3 holds the Source signal assignment that will be routed to the DAI_DAT6.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
17:12 (R/W)	IN2	Input Data 2. DAI_DAT6.IN2 holds the Source signal assignment that will be routed to the DAI_DAT6.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
11:6 (R/W)	IN1	Input Data 1. DAI_DAT6.IN1 holds the Source signal assignment that will be routed to the DAI_DAT6.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5:0 (R/W)	IN0	Input Data 0. DAI_DAT6.IN0 holds the Source signal assignment that will be routed to the DAI_DAT6.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Extended Clock Routing Control Register 0

The DAI_EXTD_CLK0 and DAI_CLK0 registers together provides the extended clock routing connections for the serial ports (SPORTs).

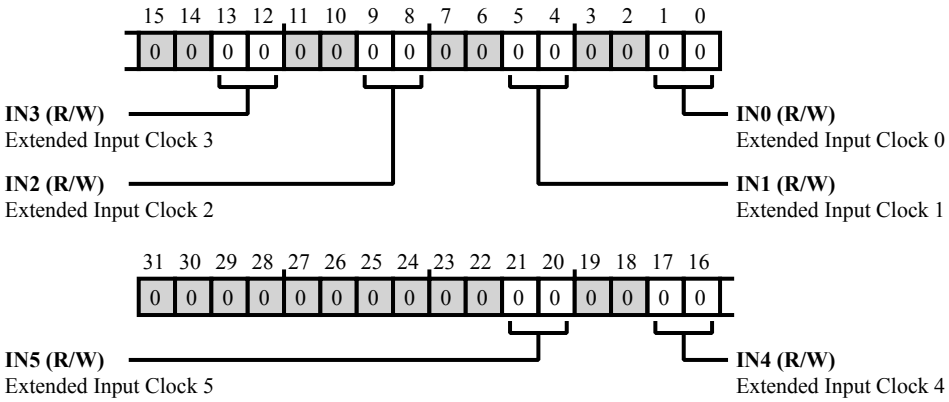


Figure 31-27: DAI_EXTD_CLK0 Register Diagram

Table 31-37: DAI_EXTD_CLK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 5. DAI_EXTD_CLK0.IN5 and DAI_CLK0.IN5 together holds the Extended Source signal assignment that will be routed to the DAI_CLK0.IN5 Destination. Refer to the Group A Signals table for Source and Destination mappings
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK0.IN4 and DAI_CLK0.IN4 together holds the Extended Source signal assignment that will be routed to the DAI_CLK0.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK0.IN3 and DAI_CLK0.IN3 together holds the Extended Source signal assignment that will be routed to the DAI_CLK0.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings
9:8 (R/W)	IN2	Extended Input Clock 2. DAI_EXTD_CLK0.IN2 and DAI_CLK0.IN2 together holds the Extended Source signal assignment that will be routed to the DAI_CLK0.IN2 Destination. Refer to the Group A Signals table for Source and Destination mappings
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK0.IN1 and DAI_CLK0.IN1 together holds the Extended Source signal assignment that will be routed to the DAI_CLK0.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings

Table 31-37: DAI_EXTD_CLK0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Clock 0. `DAI_EXTD_CLK0.IN0` and DAI_CLK0.IN0 together holds the Extended Source signal assignment that will be routed to the DAI_CLK0.IN0 Destination. Refer to the Group A Signals table for Source and Destination mappings

Extended Clock Routing Control Register 1

The DAI_EXTD_CLK1 and DAI_CLK1 registers together provides the extended clock routing connections for the serial ports (SPORTs).

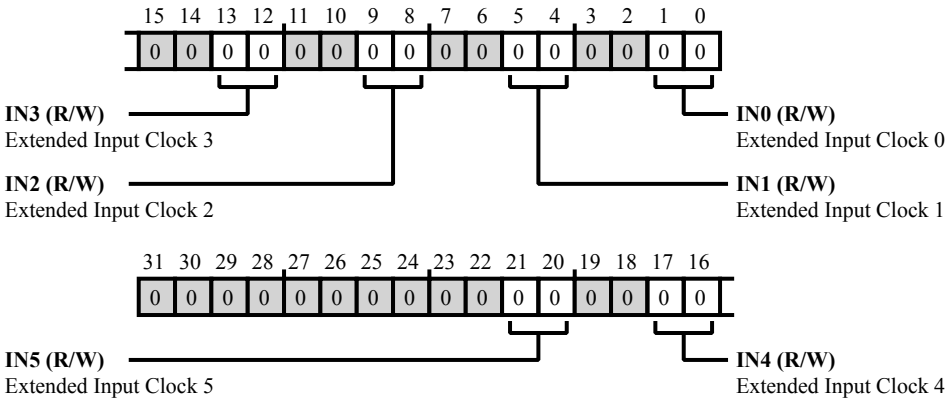


Figure 31-28: DAI_EXTD_CLK1 Register Diagram

Table 31-38: DAI_EXTD_CLK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 5. DAI_EXTD_CLK1.IN5 and DAI_CLK1.IN5 together holds the Extended Source signal assignment that will be routed to the DAI_CLK1.IN5 Destination. Refer to the Group A Signals table for Source and Destination mappings
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK1.IN4 and DAI_CLK1.IN4 together holds the Extended Source signal assignment that will be routed to the DAI_CLK1.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK1.IN3 and DAI_CLK1.IN3 together holds the Extended Source signal assignment that will be routed to the DAI_CLK1.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings
9:8 (R/W)	IN2	Extended Input Clock 2. DAI_EXTD_CLK1.IN2 and DAI_CLK1.IN2 together holds the Extended Source signal assignment that will be routed to the DAI_CLK1.IN2 Destination. Refer to the Group A Signals table for Source and Destination mappings
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK1.IN1 and DAI_CLK1.IN1 together holds the Extended Source signal assignment that will be routed to the DAI_CLK1.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings

Table 31-38: DAI_EXTD_CLK1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Clock 0. `DAI_EXTD_CLK1.IN0` and DAI_CLK1.IN0 together holds the Extended Source signal assignment that will be routed to the DAI_CLK1.IN0 Destination. Refer to the Group A Signals table for Source and Destination mappings

Extended Clock Routing Control Register 2

The DAI_EXTD_CLK2 and DAI_CLK2 registers together provides the extended clock routing connections for the serial ports (SPORTs).

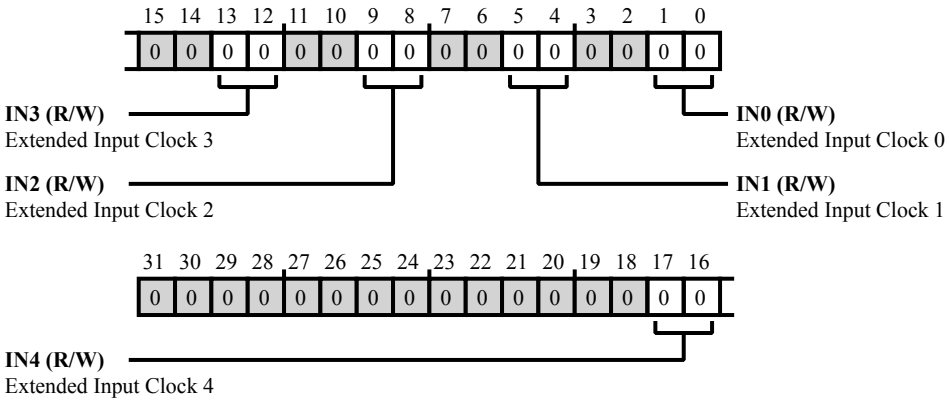


Figure 31-29: DAI_EXTD_CLK2 Register Diagram

Table 31-39: DAI_EXTD_CLK2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK2.IN4 and DAI_CLK2.IN4 together holds the Extended Source signal assignment that will be routed to the DAI_CLK2.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK2.IN3 and DAI_CLK2.IN3 together holds the Extended Source signal assignment that will be routed to the DAI_CLK2.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings
9:8 (R/W)	IN2	Extended Input Clock 2. DAI_EXTD_CLK2.IN2 and DAI_CLK2.IN2 together holds the Extended Source signal assignment that will be routed to the DAI_CLK2.IN2 Destination. Refer to the Group A Signals table for Source and Destination mappings
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK2.IN1 and DAI_CLK2.IN1 together holds the Extended Source signal assignment that will be routed to the DAI_CLK2.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings
1:0 (R/W)	IN0	Extended Input Clock 0. DAI_EXTD_CLK2.IN0 and DAI_CLK2.IN0 together holds the Extended Source signal assignment that will be routed to the DAI_CLK2.IN0 Destination. Refer to the Group A Signals table for Source and Destination mappings

Extended Clock Routing Control Register 3

The DAI_EXTD_CLK3 and DAI_CLK3 registers together provides the extended clock routing connections for the serial ports (SPORTs).

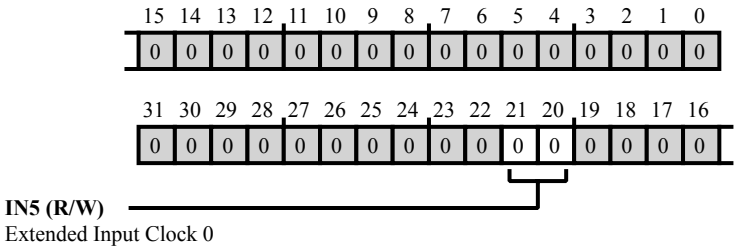


Figure 31-30: DAI_EXTD_CLK3 Register Diagram

Table 31-40: DAI_EXTD_CLK3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 0. DAI_EXTD_CLK3.IN5 and DAI_CLK3.IN0 together holds the Extended Source signal assignment that will be routed to the DAI_CLK3.IN0 Destination. Refer to the Group A Signals table for Source and Destination mappings

Extended Clock Routing Control Register 4

The DAI_EXTD_CLK4 and DAI_CLK4 registers together provides the extended clock routing connections for the serial ports (SPORTs).

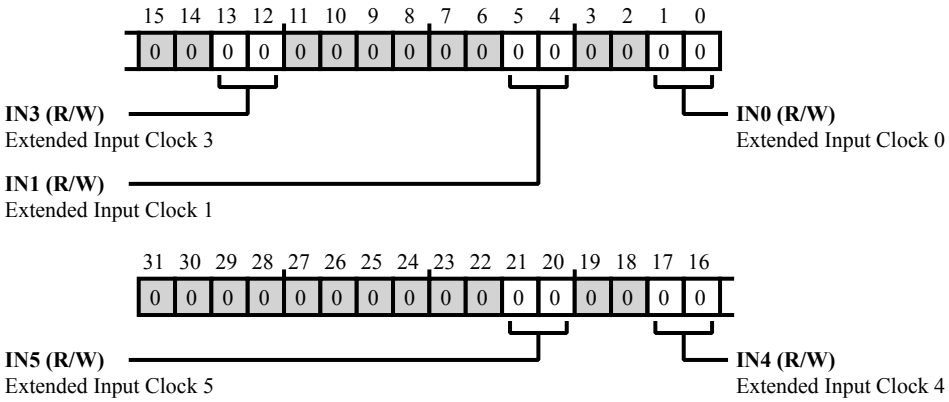


Figure 31-31: DAI_EXTD_CLK4 Register Diagram

Table 31-41: DAI_EXTD_CLK4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 5. DAI_EXTD_CLK4.IN5 and DAI_CLK4.IN5 together holds the Extended Source signal assignment that will be routed to the DAI_CLK4.IN5 Destination. Refer to the Group A Signals table for Source and Destination mappings
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK4.IN4 and DAI_CLK4.IN4 together holds the Extended Source signal assignment that will be routed to the DAI_CLK4.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK4.IN3 and DAI_CLK4.IN3 together holds the Extended Source signal assignment that will be routed to the DAI_CLK4.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK4.IN1 and DAI_CLK4.IN1 together holds the Extended Source signal assignment that will be routed to the DAI_CLK4.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings
1:0 (R/W)	IN0	Extended Input Clock 0. DAI_EXTD_CLK4.IN0 and DAI_CLK4.IN0 together holds the Extended Source signal assignment that will be routed to the DAI_CLK4.IN0 Destination. Refer to the Group A Signals table for Source and Destination mappings

Extended Clock Routing Control Register 5

The DAI_EXTD_CLK5 and DAI_CLK5 registers together provides the extended clock routing connections for the serial ports (SPORTs).

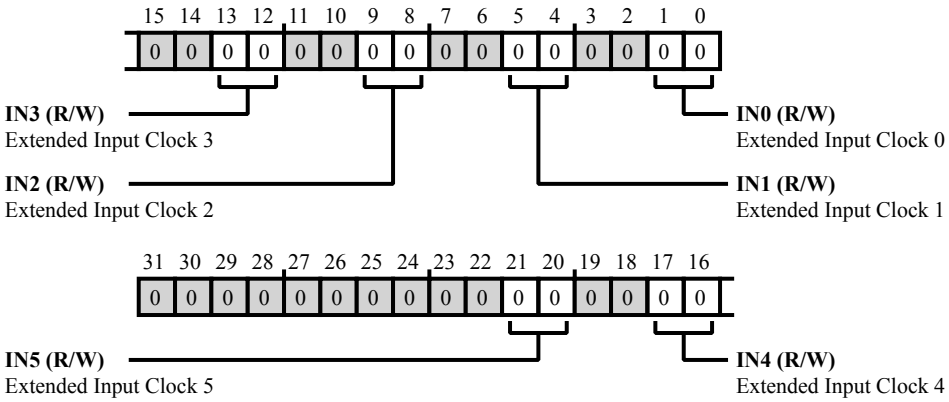


Figure 31-32: DAI_EXTD_CLK5 Register Diagram

Table 31-42: DAI_EXTD_CLK5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Clock 5. DAI_EXTD_CLK5.IN5 and DAI_CLK5.IN5 together holds the Extended Source signal assignment that will be routed to the DAI_CLK5.IN5 Destination. Refer to the Group A Signals table for Source and Destination mappings
17:16 (R/W)	IN4	Extended Input Clock 4. DAI_EXTD_CLK5.IN4 and DAI_CLK5.IN4 together holds the Extended Source signal assignment that will be routed to the DAI_CLK5.IN4 Destination. Refer to the Group A Signals table for Source and Destination mappings
13:12 (R/W)	IN3	Extended Input Clock 3. DAI_EXTD_CLK5.IN3 and DAI_CLK5.IN3 together holds the Extended Source signal assignment that will be routed to the DAI_CLK5.IN3 Destination. Refer to the Group A Signals table for Source and Destination mappings
9:8 (R/W)	IN2	Extended Input Clock 2. DAI_EXTD_CLK5.IN2 and DAI_CLK5.IN2 together holds the Extended Source signal assignment that will be routed to the DAI_CLK5.IN2 Destination. Refer to the Group A Signals table for Source and Destination mappings
5:4 (R/W)	IN1	Extended Input Clock 1. DAI_EXTD_CLK5.IN1 and DAI_CLK5.IN1 together holds the Extended Source signal assignment that will be routed to the DAI_CLK5.IN1 Destination. Refer to the Group A Signals table for Source and Destination mappings

Table 31-42: DAI_EXTD_CLK5 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Clock 0. `DAI_EXTD_CLK5.IN0` and DAI_CLK5.IN0 together holds the Extended Source signal assignment that will be routed to the DAI_CLK5.IN0 Destination. Refer to the Group A Signals table for Source and Destination mappings

Extended Serial Data Routing Control Register 0

The DAI_EXTD_DAT0 and DAI_DAT0 together provides the extended data routing connections to the serial data of serial ports.

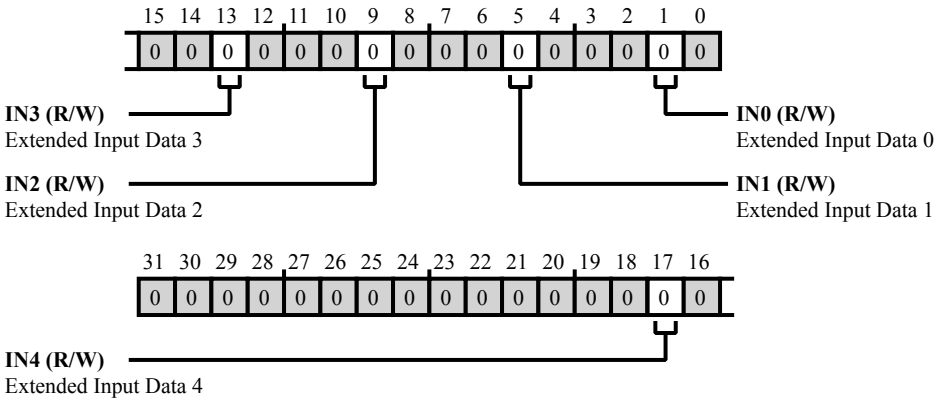


Figure 31-33: DAI_EXTD_DAT0 Register Diagram

Table 31-43: DAI_EXTD_DAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT0.IN4 and DAI_DAT0.IN4 together holds the Source signal assignment that will be routed to the DAI_DAT0.IN4 Destination. Refer to the Group B Signals table for Source and Destination mappings.
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT0.IN3 and DAI_DAT0.IN3 together holds the Source signal assignment that will be routed to the DAI_DAT0.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT0.IN2 and DAI_DAT0.IN2 together holds the Source signal assignment that will be routed to the DAI_DAT0.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT0.IN1 and DAI_DAT0.IN1 together holds the Source signal assignment that will be routed to the DAI_DAT0.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT0.IN0 and DAI_DAT0.IN0 together holds the Source signal assignment that will be routed to the DAI_DAT0.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Extended Serial Data Routing Control Register 1

The DAI_EXTD_DAT1 and DAI_DAT1 together provides the extended data routing connections to the serial data of serial ports.

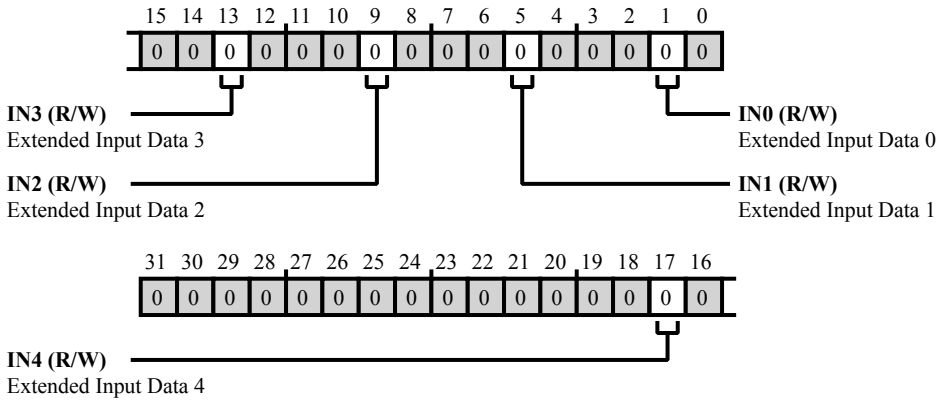


Figure 31-34: DAI_EXTD_DAT1 Register Diagram

Table 31-44: DAI_EXTD_DAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT1.IN4 and DAI_DAT1.IN4 together holds the Source signal assignment that will be routed to the DAI_DAT1.IN4 Destination. Refer to the Group B Signals table for Source and Destination mappings.
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT1.IN3 and DAI_DAT1.IN3 together holds the Source signal assignment that will be routed to the DAI_DAT1.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT1.IN2 and DAI_DAT1.IN2 together holds the Source signal assignment that will be routed to the DAI_DAT1.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT1.IN1 and DAI_DAT1.IN1 together holds the Source signal assignment that will be routed to the DAI_DAT1.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT1.IN0 and DAI_DAT1.IN0 together holds the Source signal assignment that will be routed to the DAI_DAT1.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Extended Serial Data Routing Control Register 2

The DAI_EXTD_DAT2 and DAI_DAT2 together provides the extended data routing connections to the serial data of serial ports.

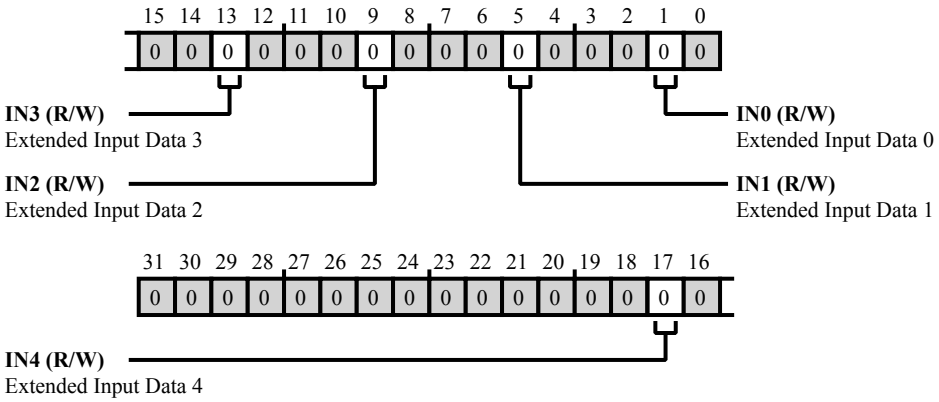


Figure 31-35: DAI_EXTD_DAT2 Register Diagram

Table 31-45: DAI_EXTD_DAT2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT2.IN4 and DAI_DAT2.IN4 together holds the Source signal assignment that will be routed to the DAI_DAT2.IN4 Destination. Refer to the Group B Signals table for Source and Destination mappings.
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT2.IN3 and DAI_DAT2.IN3 together holds the Source signal assignment that will be routed to the DAI_DAT2.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT2.IN2 and DAI_DAT2.IN2 together holds the Source signal assignment that will be routed to the DAI_DAT2.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT2.IN1 and DAI_DAT2.IN1 together holds the Source signal assignment that will be routed to the DAI_DAT2.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT2.IN0 and DAI_DAT2.IN0 together holds the Source signal assignment that will be routed to the DAI_DAT2.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Extended Serial Data Routing Control Register 3

The DAI_EXTD_DAT3 and DAI_DAT3 together provides the extended data routing connections to the serial data of serial ports.

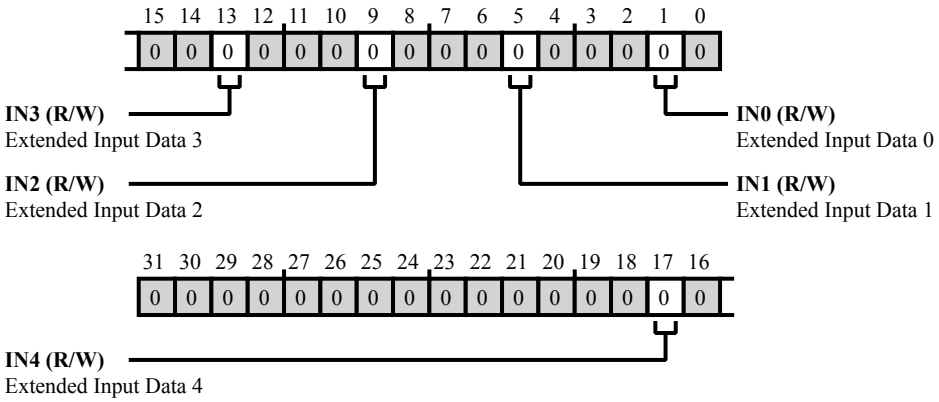


Figure 31-36: DAI_EXTD_DAT3 Register Diagram

Table 31-46: DAI_EXTD_DAT3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT3.IN4 and DAI_DAT3.IN4 together holds the Source signal assignment that will be routed to the DAI_DAT3.IN4 Destination. Refer to the Group B Signals table for Source and Destination mappings.
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT3.IN3 and DAI_DAT3.IN3 together holds the Source signal assignment that will be routed to the DAI_DAT3.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT3.IN2 and DAI_DAT3.IN2 together holds the Source signal assignment that will be routed to the DAI_DAT3.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT3.IN1 and DAI_DAT3.IN1 together holds the Source signal assignment that will be routed to the DAI_DAT3.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT3.IN0 and DAI_DAT3.IN0 together holds the Source signal assignment that will be routed to the DAI_DAT3.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Extended Serial Data Routing Control Register 4

The DAI_EXTD_DAT4 and DAI_DAT4 together provides the extended data routing connections to the serial data of serial ports.

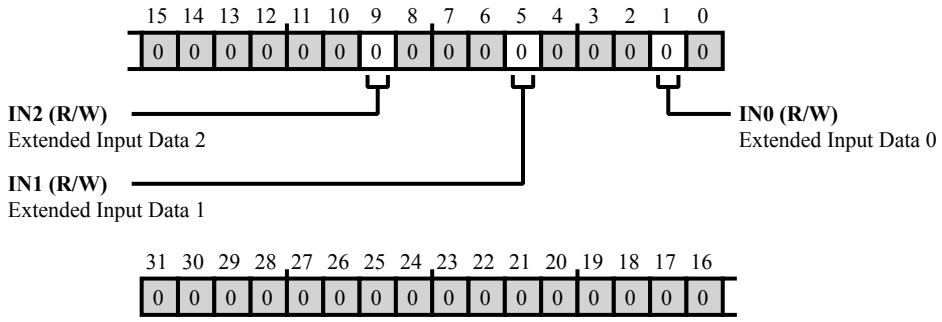


Figure 31-37: DAI_EXTD_DAT4 Register Diagram

Table 31-47: DAI_EXTD_DAT4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT4.IN2 and DAI_DAT4.IN2 together holds the Source signal assignment that will be routed to the DAI_DAT4.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT4.IN1 and DAI_DAT4.IN2 together holds the Source signal assignment that will be routed to the DAI_DAT4.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT4.IN0 and DAI_DAT4.IN0 together holds the Source signal assignment that will be routed to the DAI_DAT4.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Extended Serial Data Routing Control Register 5

The DAI_EXTD_DAT5 and DAI_DAT5 together provides the extended data routing connections to the serial data of serial ports.

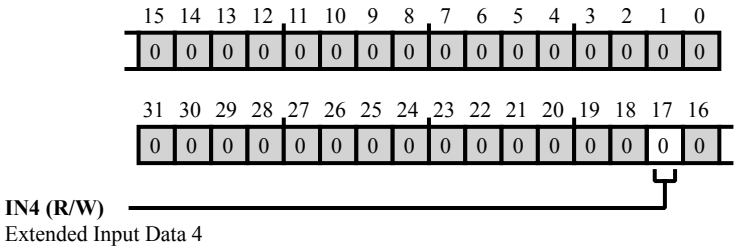


Figure 31-38: DAI_EXTD_DAT5 Register Diagram

Table 31-48: DAI_EXTD_DAT5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	IN4	Extended Input Data 4. DAI_EXTD_DAT5.IN4 and DAI_DAT5.IN4 together holds the Source signal assignment that will be routed to the DAI_DAT5.IN4 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Extended Serial Data Routing Control Register 6

The DAI_EXTD_DAT6 and DAI_DAT6 together provides the extended data routing connections to the serial data of serial ports.

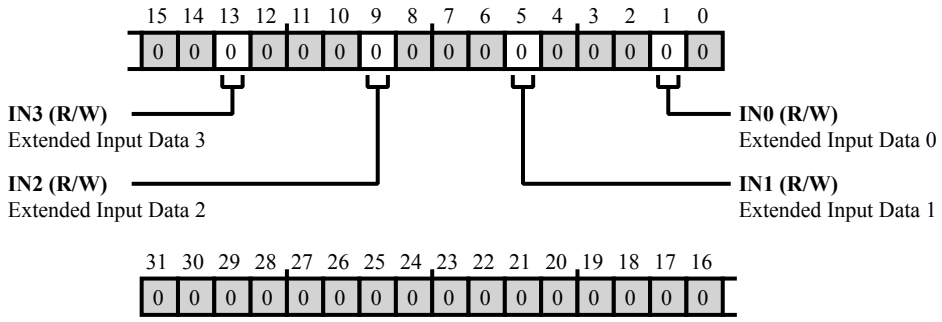


Figure 31-39: DAI_EXTD_DAT6 Register Diagram

Table 31-49: DAI_EXTD_DAT6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	IN3	Extended Input Data 3. DAI_EXTD_DAT6.IN3 and DAI_DAT6.IN3 together holds the Source signal assignment that will be routed to the DAI_DAT6.IN3 Destination. Refer to the Group B Signals table for Source and Destination mappings.
9 (R/W)	IN2	Extended Input Data 2. DAI_EXTD_DAT6.IN2 and DAI_DAT6.IN2 together holds the Source signal assignment that will be routed to the DAI_DAT6.IN2 Destination. Refer to the Group B Signals table for Source and Destination mappings.
5 (R/W)	IN1	Extended Input Data 1. DAI_EXTD_DAT6.IN1 and DAI_DAT6.IN1 together holds the Source signal assignment that will be routed to the DAI_DAT6.IN1 Destination. Refer to the Group B Signals table for Source and Destination mappings.
1 (R/W)	IN0	Extended Input Data 0. DAI_EXTD_DAT6.IN0 and DAI_DAT6.IN0 together holds the Source signal assignment that will be routed to the DAI_DAT6.IN0 Destination. Refer to the Group B Signals table for Source and Destination mappings.

Extended Frame Sync Routing Control Register 0

The DAI_EXTD_FS0 and DAI_FS0 together provides the extended frame sync routing connections to the frame syncs of serial ports.

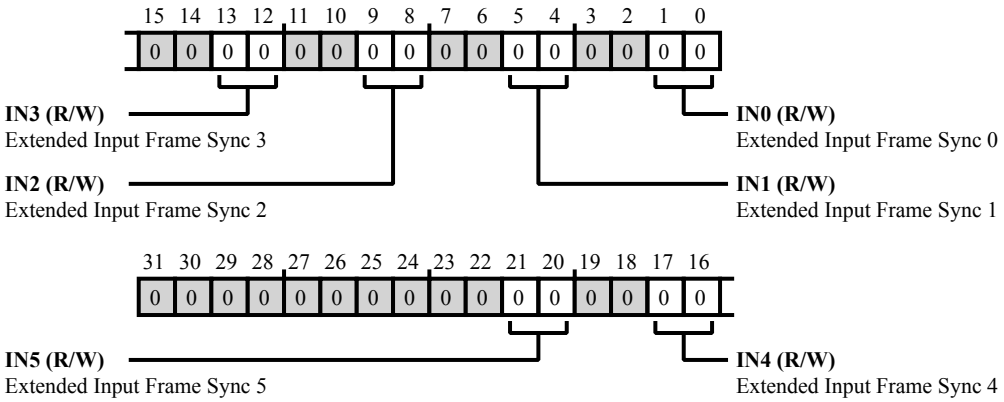


Figure 31-40: DAI_EXTD_FS0 Register Diagram

Table 31-50: DAI_EXTD_FS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Frame Sync 5. DAI_EXTD_FS0.IN5 and DAI_FS0.IN5 together holds the Source signal assignment that will be routed to the DAI_FS0.IN5 Destination. Refer to the Group C Signals table for Source and Destination mappings.
17:16 (R/W)	IN4	Extended Input Frame Sync 4. DAI_EXTD_FS0.IN4 and DAI_FS0.IN4 together holds the Source signal assignment that will be routed to the DAI_FS0.IN4 Destination. Refer to the Group C Signals table for Source and Destination mappings.
13:12 (R/W)	IN3	Extended Input Frame Sync 3. DAI_EXTD_FS0.IN3 and DAI_FS0.IN3 together holds the Source signal assignment that will be routed to the DAI_FS0.IN3 Destination. Refer to the Group C Signals table for Source and Destination mappings.
9:8 (R/W)	IN2	Extended Input Frame Sync 2. DAI_EXTD_FS0.IN2 and DAI_FS0.IN2 together holds the Source signal assignment that will be routed to the DAI_FS0.IN2 Destination. Refer to the Group C Signals table for Source and Destination mappings.
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS0.IN1 and DAI_FS0.IN1 together holds the Source signal assignment that will be routed to the DAI_FS0.IN1 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Table 31-50: DAI_EXTD_FS0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Frame Sync 0. `DAI_EXTD_FS0.IN0` and DAI_FS0.IN0 together holds the Source signal assignment that will be routed to the DAI_FS0.IN0 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Extended Frame Sync Routing Control Register 1

The DAI_EXTD_FS1 and DAI_FS1 together provides the extended frame sync routing connections to the frame syncs of serial ports.

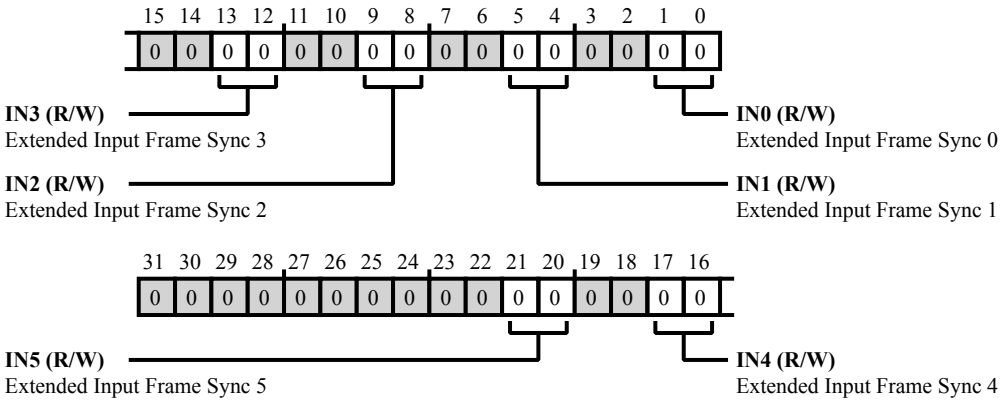


Figure 31-41: DAI_EXTD_FS1 Register Diagram

Table 31-51: DAI_EXTD_FS1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input Frame Sync 5. DAI_EXTD_FS1.IN5 and DAI_FS1.IN5 together holds the Source signal assignment that will be routed to the DAI_FS1.IN5 Destination. Refer to the Group C Signals table for Source and Destination mappings.
17:16 (R/W)	IN4	Extended Input Frame Sync 4. DAI_EXTD_FS1.IN4 and DAI_FS1.IN4 together holds the Source signal assignment that will be routed to the DAI_FS1.IN4 Destination. Refer to the Group C Signals table for Source and Destination mappings.
13:12 (R/W)	IN3	Extended Input Frame Sync 3. DAI_EXTD_FS1.IN3 and DAI_FS1.IN3 together holds the Source signal assignment that will be routed to the DAI_FS1.IN3 Destination. Refer to the Group C Signals table for Source and Destination mappings.
9:8 (R/W)	IN2	Extended Input Frame Sync 2. DAI_EXTD_FS1.IN2 and DAI_FS1.IN2 together holds the Source signal assignment that will be routed to the DAI_FS1.IN2 Destination. Refer to the Group C Signals table for Source and Destination mappings.
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS1.IN1 and DAI_FS1.IN1 together holds the Source signal assignment that will be routed to the DAI_FS1.IN1 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Table 31-51: DAI_EXTD_FS1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN0	Extended Input Frame Sync 0. `DAI_EXTD_FS1.IN0` and DAI_FS1.IN0 together holds the Source signal assignment that will be routed to the DAI_FS1.IN0 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Extended Frame Sync Routing Control Register 2

The DAI_EXTD_FS2 and DAI_FS2 together provides the extended frame sync routing connections to the frame syncs of serial ports.

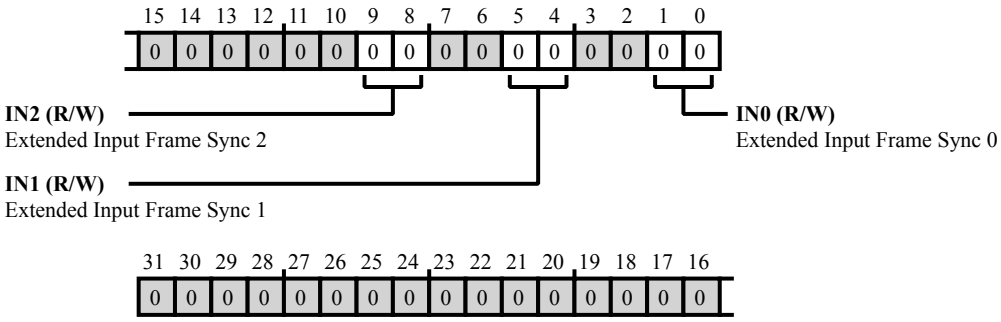


Figure 31-42: DAI_EXTD_FS2 Register Diagram

Table 31-52: DAI_EXTD_FS2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:8 (R/W)	IN2	Extended Input Frame Sync 2. DAI_EXTD_FS2.IN2 and DAI_FS2.IN2 together holds the Source signal assignment that will be routed to the DAI_FS2.IN2 Destination. Refer to the Group C Signals table for Source and Destination mappings.
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS2.IN1 and DAI_FS2.IN1 together holds the Source signal assignment that will be routed to the DAI_FS2.IN1 Destination. Refer to the Group C Signals table for Source and Destination mappings.
1:0 (R/W)	IN0	Extended Input Frame Sync 0. DAI_EXTD_FS2.IN0 and DAI_FS2.IN0 together holds the Source signal assignment that will be routed to the DAI_FS2.IN0 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Extended Frame Sync Routing Control Register 4

The DAI_EXTD_FS4 and DAI_FS4 together provides the extended frame sync routing connections to the frame syncs of serial ports.

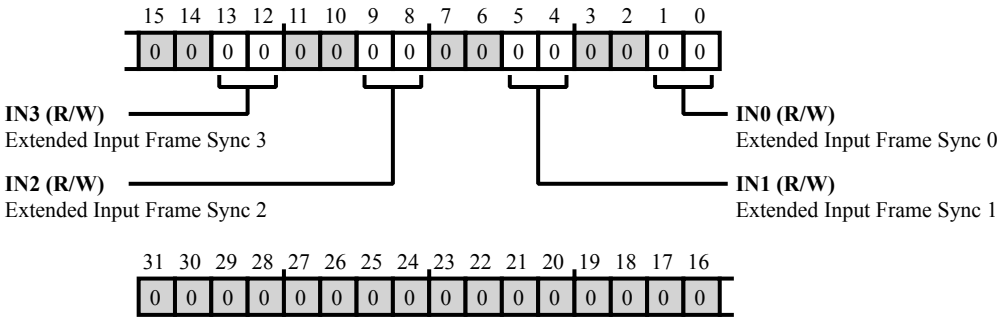


Figure 31-43: DAI_EXTD_FS4 Register Diagram

Table 31-53: DAI_EXTD_FS4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:12 (R/W)	IN3	Extended Input Frame Sync 3. DAI_EXTD_FS4.IN3 and DAI_FS4.IN3 together holds the Source signal assignment that will be routed to the DAI_FS4.IN3 Destination. This register is used to map the frame sync to the Timer ACI inputs. Refer to the Group C Signals table for Source and Destination mappings.
9:8 (R/W)	IN2	Extended Input Frame Sync 2. DAI_EXTD_FS4.IN2 and DAI_FS4.IN2 together holds the Source signal assignment that will be routed to the DAI_FS4.IN2 Destination. This register is used to map the frame sync to the Timer ACI inputs. Refer to the Group C Signals table for Source and Destination mappings.
5:4 (R/W)	IN1	Extended Input Frame Sync 1. DAI_EXTD_FS4.IN1 and DAI_FS4.IN1 together holds the Source signal assignment that will be routed to the DAI_FS4.IN1 Destination. Refer to the Group C Signals table for Source and Destination mappings.
1:0 (R/W)	IN0	Extended Input Frame Sync 0. DAI_EXTD_FS4.IN0 and DAI_FS4.IN0 together holds the Source signal assignment that will be routed to the DAI_FS4.IN0 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Extended Miscellaneous Control Register 0

The DAI_EXTD_MISC0 and DAI_MISC0 register together allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

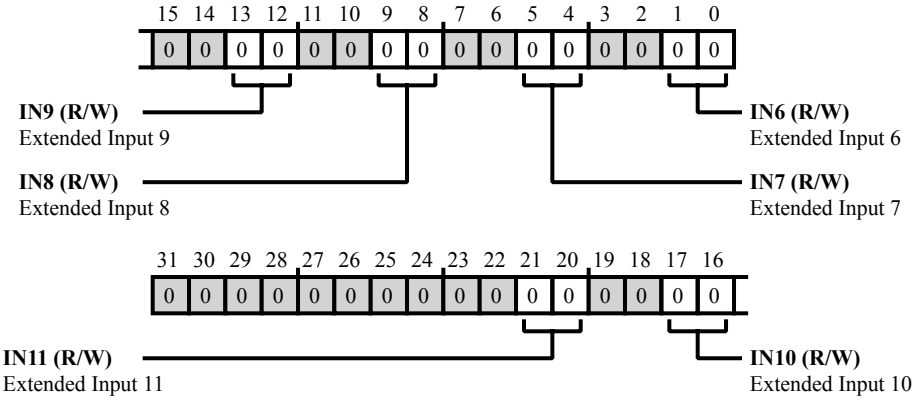


Figure 31-44: DAI_EXTD_MISC0 Register Diagram

Table 31-54: DAI_EXTD_MISC0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN11	Extended Input 11. The DAI_EXTD_MISC0.IN11 bit field and DAI_MISC0.IN11 bit field together holds the source signal assignment for miscellaneous A5 input.
17:16 (R/W)	IN10	Extended Input 10. The DAI_EXTD_MISC0.IN10 bit field and DAI_MISC0.IN10 bit field together holds the source signal assignment for miscellaneous A4 input.
13:12 (R/W)	IN9	Extended Input 9. The DAI_EXTD_MISC0.IN9 bit field and DAI_MISC0.IN9 bit field together holds the source signal assignment for miscellaneous A3 input/miscellaneous Interrupt 9(DAI interrupt 31).
9:8 (R/W)	IN8	Extended Input 8. The DAI_EXTD_MISC0.IN8 bit field and DAI_MISC0.IN8 bit field together holds the source signal assignment for miscellaneous A2 input/miscellaneous Interrupt 8(DAI interrupt 30).
5:4 (R/W)	IN7	Extended Input 7. The DAI_EXTD_MISC0.IN7 bit field and DAI_MISC0.IN7 bit field together holds the source signal assignment for miscellaneous A1 input/miscellaneous Interrupt 7(DAI interrupt 29).

Table 31-54: DAI_EXTD_MISC0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	IN6	Extended Input 6. The `DAI_EXTD_MISC0.IN6` and DAI_MISC0.IN6 bit field together holds the source signal assignment for miscellaneous A0 input/miscellaneous Interrupt 6(DAI interrupt 28).

Extended Miscellaneous Control Register 1

The DAI_EXTD_MISC1 and DAI_MISC1 register together allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

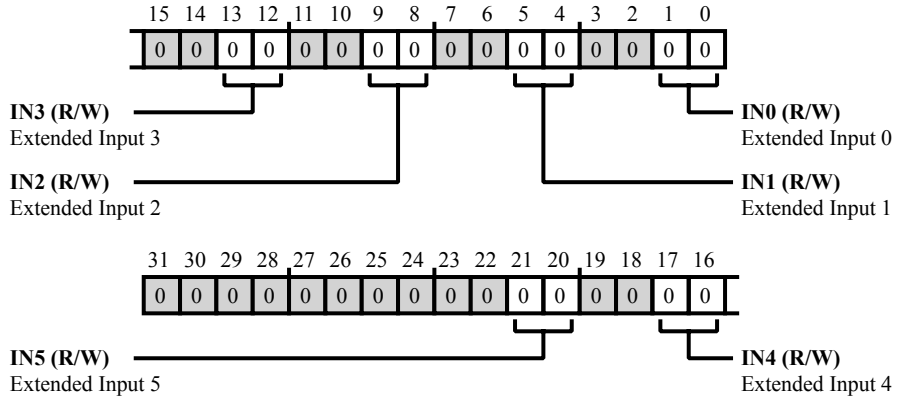


Figure 31-45: DAI_EXTD_MISC1 Register Diagram

Table 31-55: DAI_EXTD_MISC1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
21:20 (R/W)	IN5	Extended Input 5. The DAI_EXTD_MISC1.IN5 bit field and DAI_MISC1.IN5 bit field together holds the source signal assignment for miscellaneous Interrupt 5(DAI interrupt 27).
17:16 (R/W)	IN4	Extended Input 4. The DAI_EXTD_MISC1.IN4 bit field and DAI_MISC1.IN4 bit field together holds the source signal assignment for miscellaneous Interrupt 4(DAI interrupt 26).
13:12 (R/W)	IN3	Extended Input 3. The DAI_EXTD_MISC1.IN3 bit field and DAI_MISC1.IN3 bit field together holds the source signal assignment for miscellaneous Interrupt 3(DAI interrupt 25).
9:8 (R/W)	IN2	Extended Input 2. The DAI_EXTD_MISC1.IN2 bit field and DAI_MISC1.IN2 bit field together holds the source signal assignment for miscellaneous Interrupt 2(DAI interrupt 24).
5:4 (R/W)	IN1	Extended Input 1. The DAI_EXTD_MISC1.IN1 bit field and DAI_MISC1.IN1 bit field together holds the source signal assignment for miscellaneous Interrupt 1(DAI interrupt 23).
1:0 (R/W)	IN0	Extended Input 0. The DAI_EXTD_MISC1.IN0 bit field and DAI_MISC1.IN0 bit field together holds the source signal assignment for miscellaneous Interrupt 0(DAI interrupt 22).

Extended Miscellaneous Control Register 2

The DAI_EXTD_MISC2 and DAI_MISC2 register together allows programs to route to the DAI interrupt latch, PCGs hardware completer trigger, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

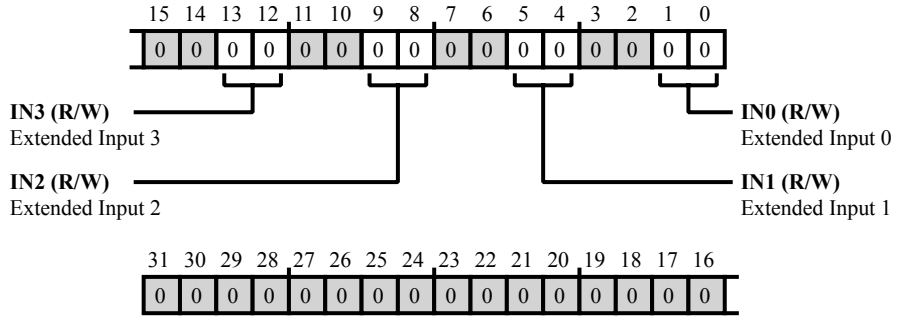


Figure 31-46: DAI_EXTD_MISC2 Register Diagram

Table 31-56: DAI_EXTD_MISC2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13:12 (R/W)	IN3	Extended Input 3. The DAI_EXTD_MISC2.IN3 bit field and DAI_MISC1.IN3 bit field together holds the source signal assignment for PCG-F's hardware completer trigger input.
9:8 (R/W)	IN2	Extended Input 2. The DAI_EXTD_MISC2.IN2 bit field and DAI_MISC2.IN2 bit field together holds the source signal assignment for PCG-E's hardware completer trigger input.
5:4 (R/W)	IN1	Extended Input 1. The DAI_EXTD_MISC2.IN1 bit field and DAI_MISC2.IN1 bit field together holds the source signal assignment for PCG-B's hardware completer trigger input.
1:0 (R/W)	IN0	Extended Input 0. The DAI_EXTD_MISC2.IN0 bit field and DAI_MISC2.IN0 bit field together holds the source signal assignment for PCG-A's hardware completer trigger input.

Extended Pin Buffer Enable Register 0

The DAI_EXTD_PBEN0 and DAI_PBEN0 register together routes extended signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

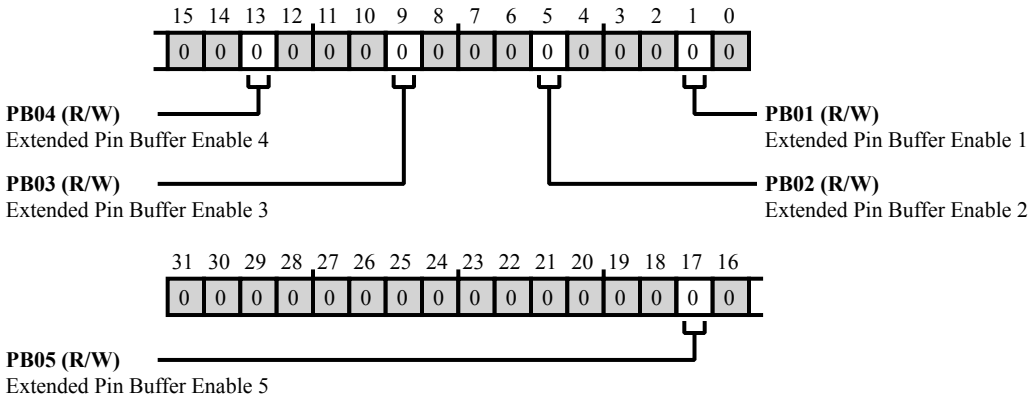


Figure 31-47: DAI_EXTD_PBEN0 Register Diagram

Table 31-57: DAI_EXTD_PBEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	PB05	Extended Pin Buffer Enable 5. The DAI_EXTD_PBEN0.PB05 bit field and DAI_PBEN0.PB05 bit field together holds the extended source signal assignment that will be routed to the pin buffer enable of DAI port 5.
13 (R/W)	PB04	Extended Pin Buffer Enable 4. The DAI_EXTD_PBEN0.PB04 bit field and DAI_PBEN0.PB04 bit field together holds the extended source signal assignment that will be routed to the pin buffer enable of DAI port 4.
9 (R/W)	PB03	Extended Pin Buffer Enable 3. The DAI_EXTD_PBEN0.PB03 bit field and DAI_PBEN0.PB03 bit field together holds the extended source signal assignment that will be routed to the pin buffer enable of DAI port 3.
5 (R/W)	PB02	Extended Pin Buffer Enable 2. The DAI_EXTD_PBEN0.PB02 bit field and DAI_PBEN0.PB02 bit field together holds the extended source signal assignment that will be routed to the pin buffer enable of DAI port 2.
1 (R/W)	PB01	Extended Pin Buffer Enable 1. The DAI_EXTD_PBEN0.PB01 bit field and DAI_PBEN0.PB01 bit field together holds the extended source signal assignment that will be routed to the pin buffer enable of DAI port 1.

Extended Pin Buffer Enable Register 1

The DAI_EXTD_PBEN1 and DAI_PBEN1 register together routes extended signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

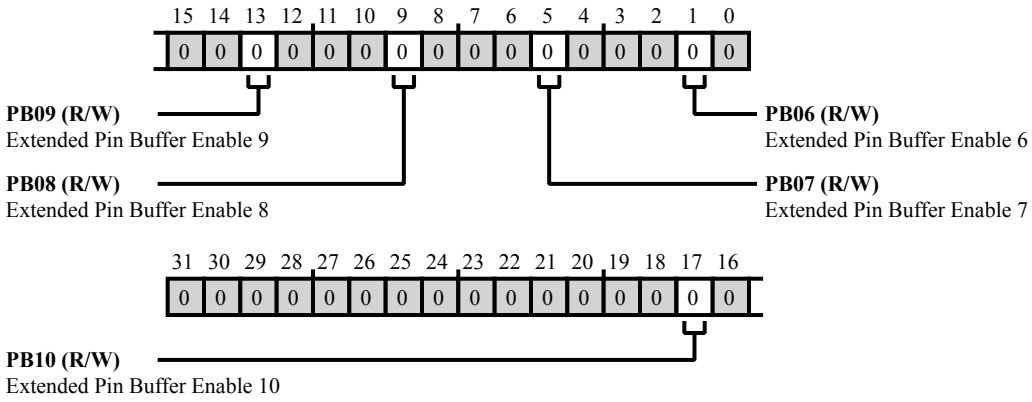


Figure 31-48: DAI_EXTD_PBEN1 Register Diagram

Table 31-58: DAI_EXTD_PBEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	PB10	Extended Pin Buffer Enable 10. The DAI_EXTD_PBEN1.PB10 bit field and DAI_PBEN0.PB10 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 10.
13 (R/W)	PB09	Extended Pin Buffer Enable 9. The DAI_EXTD_PBEN1.PB09 bit field and DAI_PBEN0.PB09 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 9.
9 (R/W)	PB08	Extended Pin Buffer Enable 8. The DAI_EXTD_PBEN1.PB08 bit field and DAI_PBEN0.PB08 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 8.
5 (R/W)	PB07	Extended Pin Buffer Enable 7. The DAI_EXTD_PBEN1.PB07 bit field and DAI_PBEN0.PB07 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 7.
1 (R/W)	PB06	Extended Pin Buffer Enable 6. The DAI_EXTD_PBEN1.PB06 bit field and DAI_PBEN0.PB06 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 6.

Extended Pin Buffer Enable Register 2

The DAI_EXTD_PBEN2 and DAI_PBEN2 register together routes extended signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

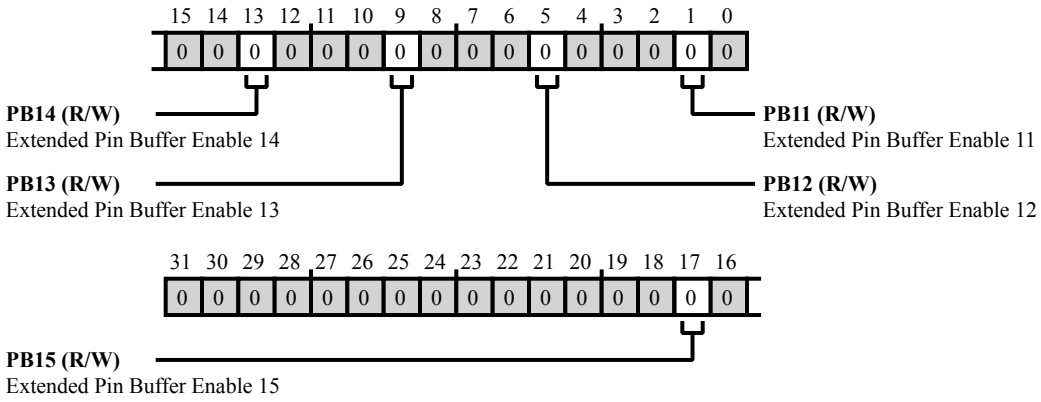


Figure 31-49: DAI_EXTD_PBEN2 Register Diagram

Table 31-59: DAI_EXTD_PBEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	PB15	Extended Pin Buffer Enable 15. The DAI_EXTD_PBEN2.PB15 bit field and DAI_PBEN0.PB15 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 15.
13 (R/W)	PB14	Extended Pin Buffer Enable 14. The DAI_EXTD_PBEN2.PB14 bit field and DAI_PBEN0.PB14 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 14.
9 (R/W)	PB13	Extended Pin Buffer Enable 13. The DAI_EXTD_PBEN2.PB13 bit field and DAI_PBEN0.PB13 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 13.
5 (R/W)	PB12	Extended Pin Buffer Enable 12. The DAI_EXTD_PBEN2.PB12 bit field and DAI_PBEN0.PB12 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 12.
1 (R/W)	PB11	Extended Pin Buffer Enable 11. The DAI_EXTD_PBEN2.PB11 bit field and DAI_PBEN0.PB11 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 11.

Extended Pin Buffer Enable Register 3

The DAI_EXTD_PBEN3 and DAI_PBEN3 register together routes extended signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

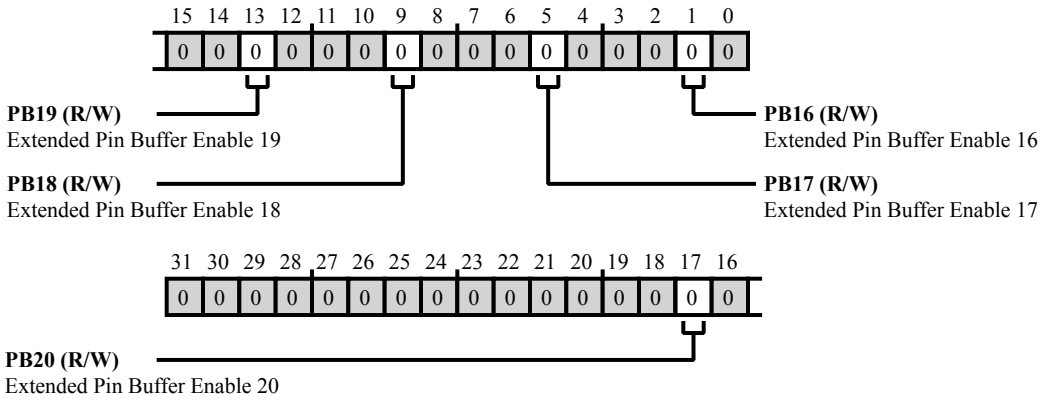


Figure 31-50: DAI_EXTD_PBEN3 Register Diagram

Table 31-60: DAI_EXTD_PBEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	PB20	Extended Pin Buffer Enable 20. The DAI_EXTD_PBEN3.PB20 bit field and DAI_PBEN0.PB20 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 20.
13 (R/W)	PB19	Extended Pin Buffer Enable 19. The DAI_EXTD_PBEN3.PB19 bit field and DAI_PBEN0.PB19 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 19.
9 (R/W)	PB18	Extended Pin Buffer Enable 18. The DAI_EXTD_PBEN3.PB18 bit field and DAI_PBEN0.PB18 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 18.
5 (R/W)	PB17	Extended Pin Buffer Enable 17. The DAI_EXTD_PBEN3.PB17 bit field and DAI_PBEN0.PB17 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 17.
1 (R/W)	PB16	Extended Pin Buffer Enable 16. The DAI_EXTD_PBEN3.PB16 bit field and DAI_PBEN0.PB16 bit field together holds the extended source signal assignment which will be routed to the pin buffer enable of DAI port 16.

Extended Pin Buffer Assignment Register 0

The DAI_EXTD_PIN0 and DAI_PIN0 register routes extended physical pins that are connected to a bonded pad.

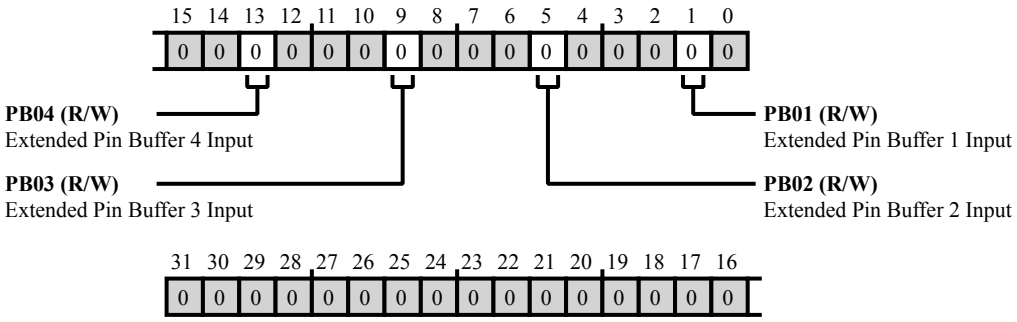


Figure 31-51: DAI_EXTD_PIN0 Register Diagram

Table 31-61: DAI_EXTD_PIN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB04	Extended Pin Buffer 4 Input. DAI_EXTD_PIN0.PB04 and DAI_PIN0.PB04 together holds the Source signal assignment that will be routed to the DAI_PIN0.PB04 Destination. Refer to the Group D Signals table for Source and Destination mappings.
9 (R/W)	PB03	Extended Pin Buffer 3 Input. DAI_EXTD_PIN0.PB03 and DAI_PIN.PB03 together holds the Source signal assignment that will be routed to the DAI_PIN0.PB03 Destination. Refer to the Group D Signals table for Source and Destination mappings.
5 (R/W)	PB02	Extended Pin Buffer 2 Input. DAI_EXTD_PIN0.PB02 and DAI_PIN0.PB02 together holds the Source signal assignment that will be routed to the DAI_PIN0.PB02 Destination. Refer to the Group D Signals table for Source and Destination mappings.
1 (R/W)	PB01	Extended Pin Buffer 1 Input. DAI_EXTD_PIN0.PB01 and DAI_PIN0.PB01 together holds the Source signal assignment that will be routed to the DAI_PIN0.PB01 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Extended Pin Buffer Assignment Register 1

The DAI_EXTD_PIN1 and DAI_PIN1 register routes extended physical pins that are connected to a bonded pad.

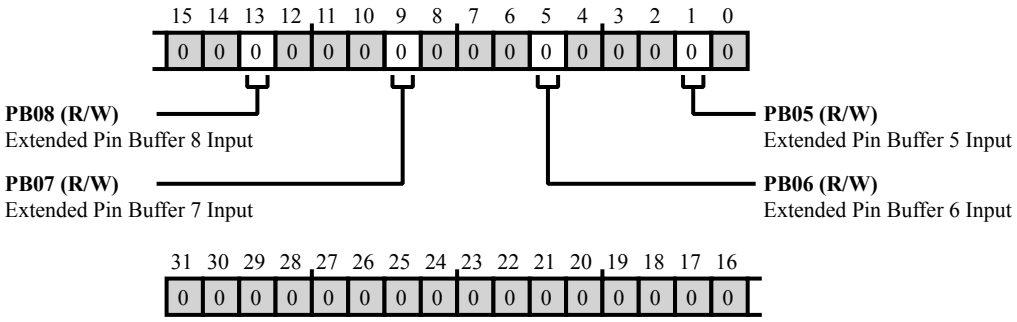


Figure 31-52: DAI_EXTD_PIN1 Register Diagram

Table 31-62: DAI_EXTD_PIN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB08	Extended Pin Buffer 8 Input. DAI_EXTD_PIN1.PB08 and DAI_PIN1.PB08 together holds the Source signal assignment that will be routed to the DAI_PIN1.PB08 Destination. Refer to the Group D Signals table for Source and Destination mappings.
9 (R/W)	PB07	Extended Pin Buffer 7 Input. DAI_EXTD_PIN1.PB07 and DAI_PIN1.PB07 together holds the Source signal assignment that will be routed to the DAI_PIN1.PB07 Destination. Refer to the Group D Signals table for Source and Destination mappings.
5 (R/W)	PB06	Extended Pin Buffer 6 Input. DAI_EXTD_PIN1.PB06 and DAI_PIN1.PB06 together the Source signal assignment that will be routed to the DAI_PIN1.PB06 Destination. Refer to the Group D Signals table for Source and Destination mappings.
1 (R/W)	PB05	Extended Pin Buffer 5 Input. DAI_EXTD_PIN1.PB05 and DAI_PIN1.PB05 together holds the Source signal assignment that will be routed to the DAI_PIN1.PB05 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Extended Pin Buffer Assignment Register 2

The DAI_EXTD_PIN2 and DAI_PIN2 register routes extended physical pins that are connected to a bonded pad.

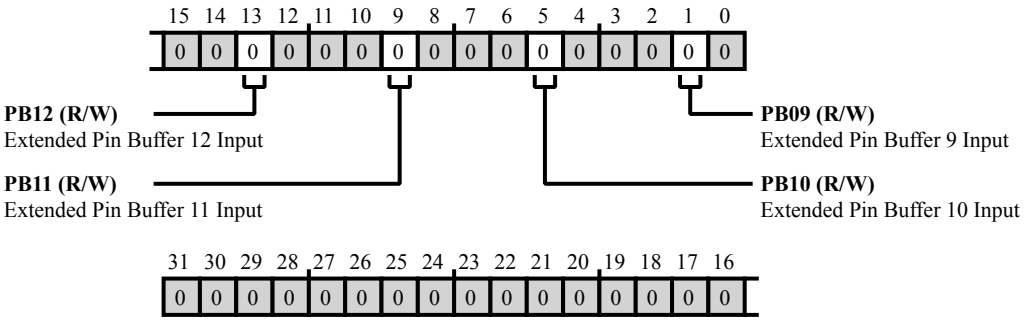


Figure 31-53: DAI_EXTD_PIN2 Register Diagram

Table 31-63: DAI_EXTD_PIN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB12	Extended Pin Buffer 12 Input. DAI_EXTD_PIN2.PB12 and DAI_PIN2.PB12 together holds the Source signal assignment that will be routed to the DAI_PIN2.PB12 Destination. Refer to the Group D Signals table for Source and Destination mappings.
9 (R/W)	PB11	Extended Pin Buffer 11 Input. DAI_EXTD_PIN2.PB11 and DAI_PIN2.PB11 together holds the Source signal assignment that will be routed to the DAI_PIN2.PB11 Destination. Refer to the Group D Signals table for Source and Destination mappings.
5 (R/W)	PB10	Extended Pin Buffer 10 Input. DAI_EXTD_PIN2.PB10 and DAI_PIN2.PB10 together holds the Source signal assignment that will be routed to the DAI_PIN2.PB10 Destination. Refer to the Group D Signals table for Source and Destination mappings.
1 (R/W)	PB09	Extended Pin Buffer 9 Input. DAI_EXTD_PIN2.PB09 and DAI_PIN2.PB09 together holds the Source signal assignment that will be routed to the DAI_PIN2.PB09 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Extended Pin Buffer Assignment Register 3

The DAI_EXTD_PIN3 and DAI_PIN3 register routes extended physical pins that are connected to a bonded pad.

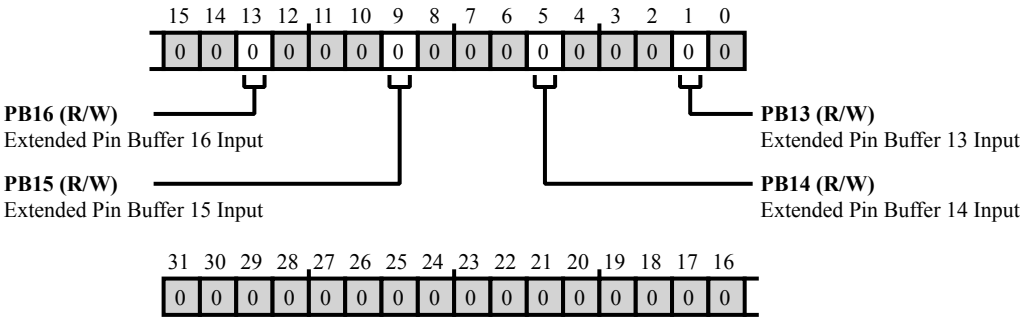


Figure 31-54: DAI_EXTD_PIN3 Register Diagram

Table 31-64: DAI_EXTD_PIN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB16	Extended Pin Buffer 16 Input. DAI_EXTD_PIN3.PB16 and DAI_PIN3.PB16 together holds the Source signal assignment that will be routed to the DAI_PIN3.PB16 Destination. Refer to the Group D Signals table for Source and Destination mappings.
9 (R/W)	PB15	Extended Pin Buffer 15 Input. DAI_EXTD_PIN3.PB15 and DAI_PIN3.PB15 together holds the Source signal assignment that will be routed to the DAI_PIN3.PB15 Destination. Refer to the Group D Signals table for Source and Destination mappings.
5 (R/W)	PB14	Extended Pin Buffer 14 Input. DAI_EXTD_PIN3.PB14 and DAI_PIN3.PB14 together holds the Source signal assignment that will be routed to the DAI_PIN3.PB14 Destination. Refer to the Group D Signals table for Source and Destination mappings.
1 (R/W)	PB13	Extended Pin Buffer 13 Input. DAI_EXTD_PIN3.PB13 and DAI_PIN3.PB13 together holds the Source signal assignment that will be routed to the DAI_PIN3.PB13 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Extended Pin Buffer Assignment Register 4

The DAI_EXTD_PIN4 and DAI_PIN4 register routes extended physical pins that are connected to a bonded pad.

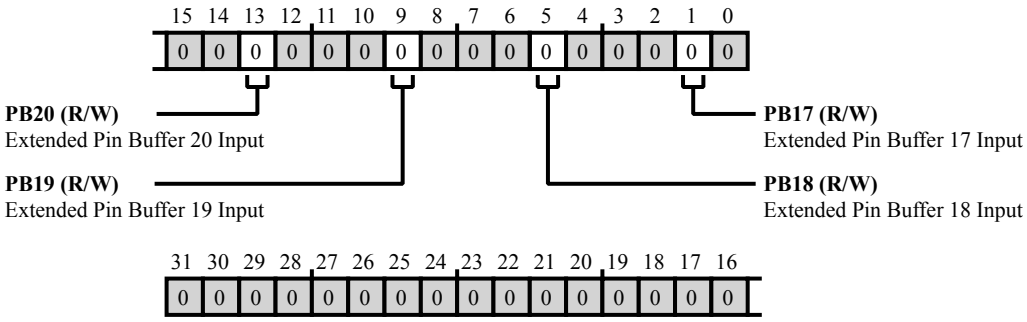


Figure 31-55: DAI_EXTD_PIN4 Register Diagram

Table 31-65: DAI_EXTD_PIN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
13 (R/W)	PB20	Extended Pin Buffer 20 Input. DAI_EXTD_PIN4.PB20 and DAI_PIN4.PB20 together holds the Source signal assignment that will be routed to the DAI_PIN4.PB20 Destination. Refer to the Group D Signals table for Source and Destination mappings.
9 (R/W)	PB19	Extended Pin Buffer 19 Input. DAI_EXTD_PIN4.PB19 and DAI_PIN4.PB19 together holds the Source signal assignment that will be routed to the DAI_PIN4.PB19 Destination. Refer to the Group D Signals table for Source and Destination mappings.
5 (R/W)	PB18	Extended Pin Buffer 18 Input. DAI_EXTD_PIN4.PB18 an DAI_PIN4.PB18 together holds the Source signal assignment that will be routed to the DAI_PIN4.PB18 Destination. Refer to the Group D Signals table for Source and Destination mappings.
1 (R/W)	PB17	Extended Pin Buffer 17 Input. DAI_EXTD_PIN4.PB17 and DAI_PIN4.PB17 together holds the Source signal assignment that will be routed to the DAI_PIN4.PB17 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Frame Sync Routing Control Register 0

The DAI_FS0 register routes frame syncs to the serial ports.

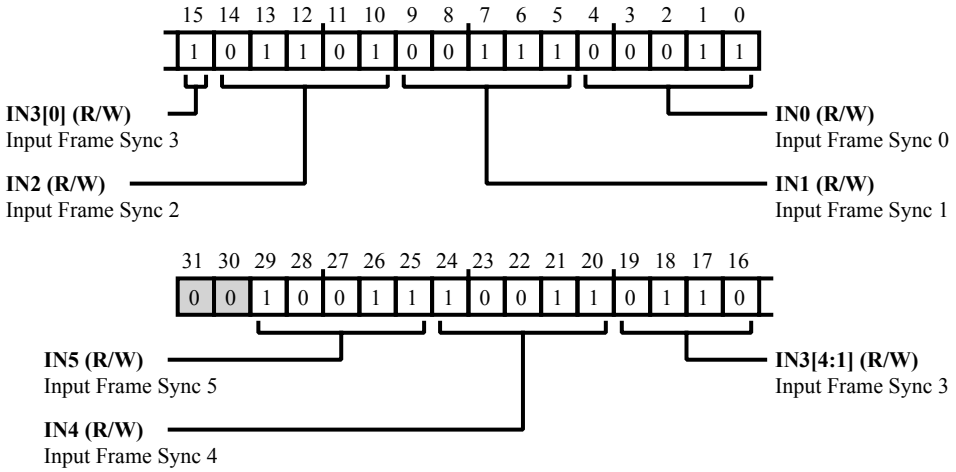


Figure 31-56: DAI_FS0 Register Diagram

Table 31-66: DAI_FS0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Frame Sync 5. DAI_FS0.IN5 holds the Source signal assignment that will be routed to the DAI_FS0.IN5 Destination. Refer to the Group C Signals table for Source and Destination mappings.
24:20 (R/W)	IN4	Input Frame Sync 4. DAI_FS0.IN4 holds the Source signal assignment that will be routed to the DAI_FS0.IN4 Destination. Refer to the Group C Signals table for Source and Destination mappings.
19:15 (R/W)	IN3	Input Frame Sync 3. DAI_FS0.IN3 holds the Source signal assignment that will be routed to the DAI_FS0.IN3 Destination. Refer to the Group C Signals table for Source and Destination mappings.
14:10 (R/W)	IN2	Input Frame Sync 2. DAI_FS0.IN2 holds the Source signal assignment that will be routed to the DAI_FS0.IN2 Destination. Refer to the Group C Signals table for Source and Destination mappings.
9:5 (R/W)	IN1	Input Frame Sync 1. DAI_FS0.IN1 holds the Source signal assignment that will be routed to the DAI_FS0.IN1 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Table 31-66: DAI_FS0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Frame Sync 0. `DAI_FS0.IN0` holds the Source signal assignment that will be routed to the `DAI_FS0.IN0` Destination. Refer to the Group C Signals table for Source and Destination mappings.

Frame Sync Routing Control Register 1

The DAI_FS1 register routes frame syncs to the asynchronous sample rate converter (ASRC).

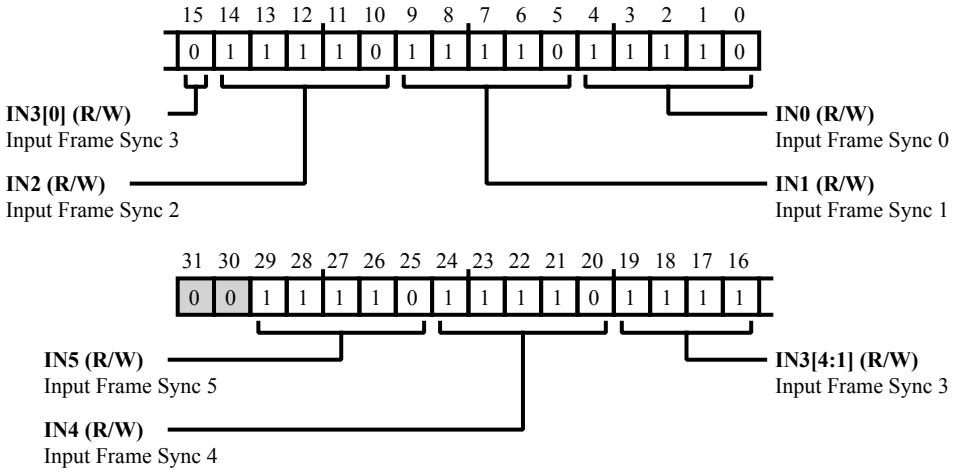


Figure 31-57: DAI_FS1 Register Diagram

Table 31-67: DAI_FS1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input Frame Sync 5. DAI_FS1.IN5 holds the Source signal assignment that will be routed to the DAI_FS1.IN5 Destination. Refer to the Group C Signals table for Source and Destination mappings.
24:20 (R/W)	IN4	Input Frame Sync 4. DAI_FS1.IN4 holds the Source signal assignment that will be routed to the DAI_FS1.IN4 Destination. Refer to the Group C Signals table for Source and Destination mappings.
19:15 (R/W)	IN3	Input Frame Sync 3. DAI_FS1.IN3 holds the Source signal assignment that will be routed to the DAI_FS1.IN3 Destination. Refer to the Group C Signals table for Source and Destination mappings.
14:10 (R/W)	IN2	Input Frame Sync 2. DAI_FS1.IN2 holds the Source signal assignment that will be routed to the DAI_FS1.IN2 Destination. Refer to the Group C Signals table for Source and Destination mappings.
9:5 (R/W)	IN1	Input Frame Sync 1. DAI_FS1.IN1 holds the Source signal assignment that will be routed to the DAI_FS1.IN1 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Table 31-67: DAI_FS1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input Frame Sync 0. `DAI_FS1.IN0` holds the Source signal assignment that will be routed to the `DAI_FS1.IN0` Destination. Refer to the Group C Signals table for Source and Destination mappings.

Frame Sync Routing Control Register 2

The DAI_FS2 register routes frame syncs to the asynchronous sample rate converter (ASRC).

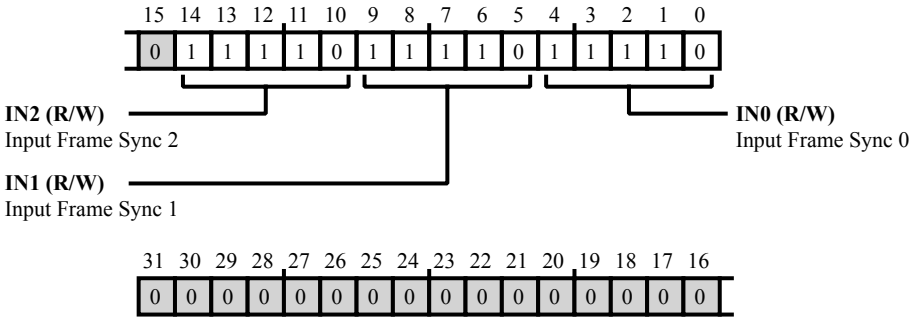


Figure 31-58: DAI_FS2 Register Diagram

Table 31-68: DAI_FS2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14:10 (R/W)	IN2	Input Frame Sync 2. DAI_FS2.IN2 holds the Source signal assignment that will be routed to the DAI_FS2.IN2 Destination. Refer to the Group C Signals table for Source and Destination mappings.
9:5 (R/W)	IN1	Input Frame Sync 1. The DAI_FS2.IN1 bit field routes the frame sync output to the ASRC3 frame sync input port.
4:0 (R/W)	IN0	Input Frame Sync 0. DAI_FS2.IN0 holds the Source signal assignment that will be routed to the DAI_FS2.IN0 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Frame Sync Routing Control Register 4

The DAI_FS4 register routes frame syncs to the asynchronous sample rate converter (ASRC).

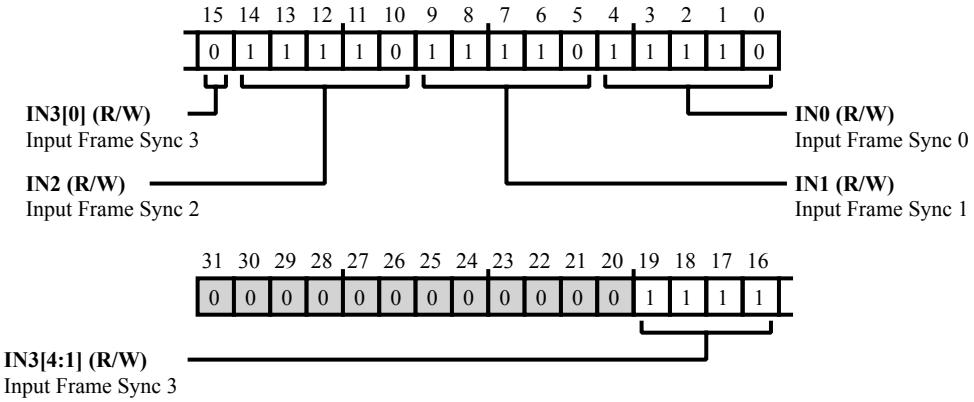


Figure 31-59: DAI_FS4 Register Diagram

Table 31-69: DAI_FS4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:15 (R/W)	IN3	Input Frame Sync 3. DAI_FS4.IN3 holds the Source signal assignment that will be routed to the DAI_FS4.IN3 Destination. This register is used to map the frame sync to the Timer ACI inputs. Refer to the Group C Signals table for Source and Destination mappings.
14:10 (R/W)	IN2	Input Frame Sync 2. DAI_FS4.IN2 holds the Source signal assignment that will be routed to the DAI_FS4.IN2 Destination. This register is used to map the frame sync to the Timer ACI inputs. Refer to the Group C Signals table for Source and Destination mappings.
9:5 (R/W)	IN1	Input Frame Sync 1. DAI_FS4.IN1 holds the Source signal assignment that will be routed to the DAI_FS4.IN1 Destination. Refer to the Group C Signals table for Source and Destination mappings.
4:0 (R/W)	IN0	Input Frame Sync 0. DAI_FS4.IN0 holds the Source signal assignment that will be routed to the DAI_FS4.IN0 Destination. Refer to the Group C Signals table for Source and Destination mappings.

Global SPORT Interrupt Grouping Register

The DAI_GBL_INT_EN register contains bits to configure the group interrupts and triggers of the SPORTs when using the SPORT global enable feature.

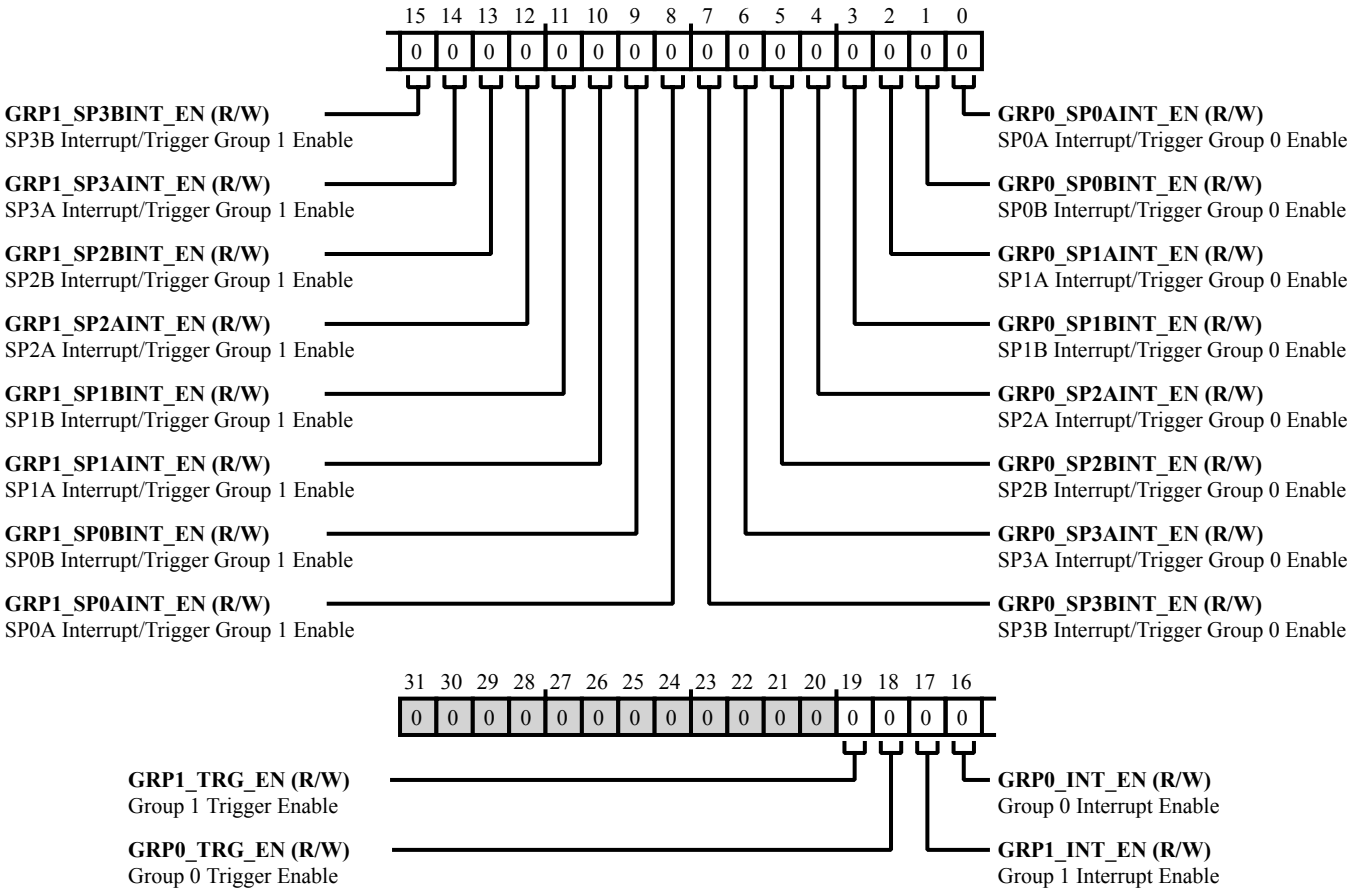


Figure 31-60: DAI_GBL_INT_EN Register Diagram

Table 31-70: DAI_GBL_INT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	GRP1_TRG_EN	Group 1 Trigger Enable. The DAI_GBL_INT_EN.GRP1_TRG_EN bit enables the trigger-out for group 1.
18 (R/W)	GRP0_TRG_EN	Group 0 Trigger Enable. The DAI_GBL_INT_EN.GRP0_TRG_EN bit enables the trigger-out for group 0.
17 (R/W)	GRP1_INT_EN	Group 1 Interrupt Enable. The DAI_GBL_INT_EN.GRP1_INT_EN bit enables the interrupt for group 1.

Table 31-70: DAI_GBL_INT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
16 (R/W)	GRP0_INT_EN	Group 0 Interrupt Enable. The DAI_GBL_INT_EN.GRP0_INT_EN bit enables the interrupt for group 0.
15 (R/W)	GRP1_SP3BINT_EN	SP3B Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP3BINT_EN bit enables the interrupt or trigger-out for SPORT3 B when it is part of group 1.
14 (R/W)	GRP1_SP3AINT_EN	SP3A Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP3AINT_EN bit enables the interrupt or trigger-out for SPORT3 A when it is part of group 1.
13 (R/W)	GRP1_SP2BINT_EN	SP2B Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP2BINT_EN bit enables the interrupt or trigger-out for SPORT2 B when it is part of group 1.
12 (R/W)	GRP1_SP2AINT_EN	SP2A Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP2AINT_EN bit enables the interrupt or trigger-out for SPORT2 A when it is part of group 1.
11 (R/W)	GRP1_SP1BINT_EN	SP1B Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP1BINT_EN bit enables the interrupt or trigger-out for SPORT1 B when it is part of group 1.
10 (R/W)	GRP1_SP1AINT_EN	SP1A Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP1AINT_EN bit enables the interrupt or trigger-out for SPORT1 A when it is part of group 1.
9 (R/W)	GRP1_SP0BINT_EN	SP0B Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP0BINT_EN bit enables the interrupt or trigger-out for SPORT0 B when it is part of group 1.
8 (R/W)	GRP1_SP0AINT_EN	SP0A Interrupt/Trigger Group 1 Enable. The DAI_GBL_INT_EN.GRP1_SP0AINT_EN bit enables the interrupt or trigger-out for SPORT0 A when it is part of group 1.
7 (R/W)	GRP0_SP3BINT_EN	SP3B Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP3BINT_EN bit enables the interrupt or trigger-out for SPORT3 B when it is part of group 0.
6 (R/W)	GRP0_SP3AINT_EN	SP3A Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP3AINT_EN bit enables the interrupt or trigger-out for SPORT3 A when it is part of group 0.
5 (R/W)	GRP0_SP2BINT_EN	SP2B Interrupt/Trigger Group 0 Enable. The DAI_GBL_INT_EN.GRP0_SP2BINT_EN bit enables the interrupt or trigger-out for SPORT2 B when it is part of group 0.

Table 31-70: DAI_GBL_INT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/W)	GRP0_SP2AINT_EN	SP2A Interrupt/Trigger Group 0 Enable. The `DAI_GBL_INT_EN.GRP0_SP2AINT_EN` bit enables the interrupt or trigger-out for SPORT2 A when it is part of group 0.
3 (R/W)	GRP0_SP1BINT_EN	SP1B Interrupt/Trigger Group 0 Enable. The `DAI_GBL_INT_EN.GRP0_SP1BINT_EN` bit enables the interrupt or trigger-out for SPORT1 B when it is part of group 0.
2 (R/W)	GRP0_SP1AINT_EN	SP1A Interrupt/Trigger Group 0 Enable. The `DAI_GBL_INT_EN.GRP0_SP1AINT_EN` bit enables the interrupt or trigger-out for SPORT1 A when it is part of group 0.
1 (R/W)	GRP0_SP0BINT_EN	SP0B Interrupt/Trigger Group 0 Enable. The `DAI_GBL_INT_EN.GRP0_SP0BINT_EN` bit enables the interrupt or trigger-out for SPORT0 B when it is part of group 0.
0 (R/W)	GRP0_SP0AINT_EN	SP0A Interrupt/Trigger Group 0 Enable. The `DAI_GBL_INT_EN.GRP0_SP0AINT_EN` bit enables the interrupt or trigger-out for SPORT0 A when it is part of group 0.

Global PCG Enable Control Register

The DAI_GBL_PCG_EN register bits indicate the status (signal high or low) for each pin. The individual bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

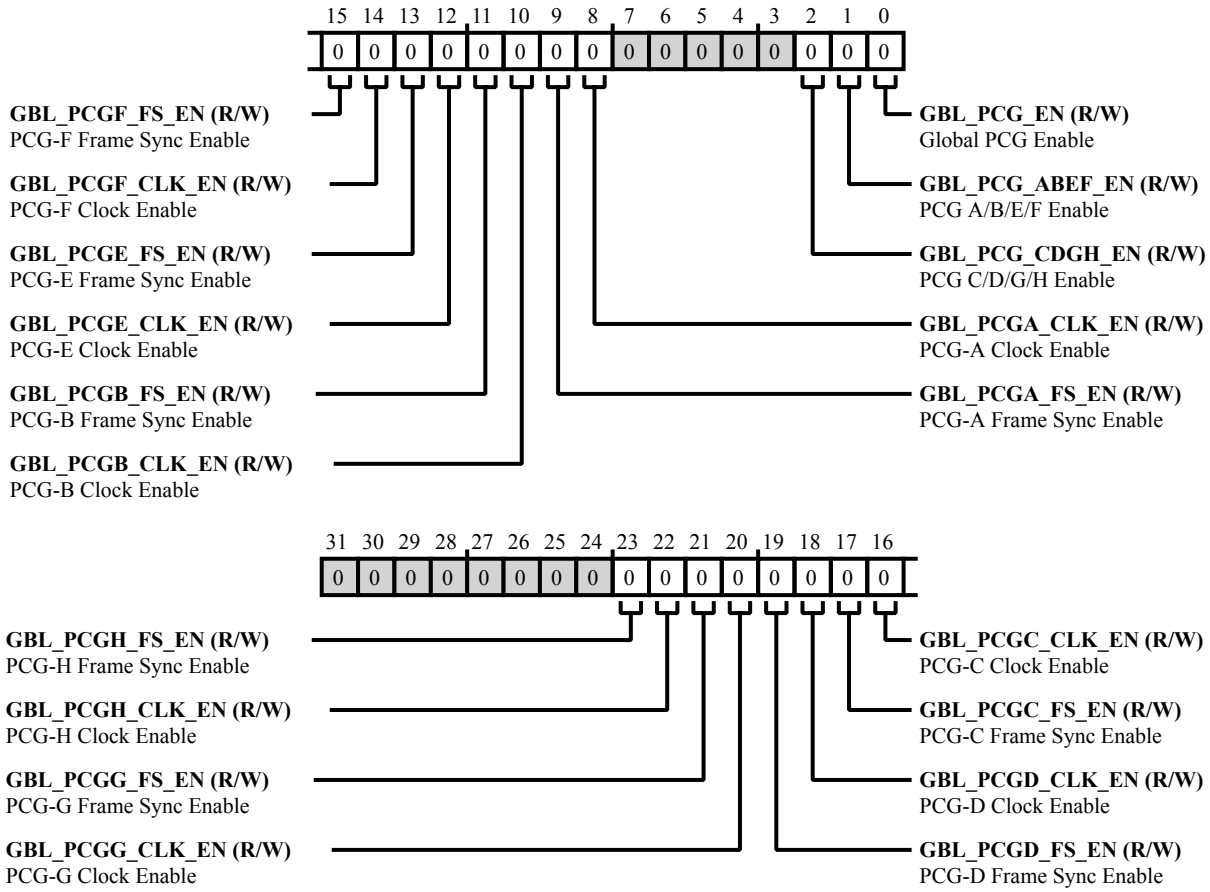


Figure 31-61: DAI_GBL_PCG_EN Register Diagram

Table 31-71: DAI_GBL_PCG_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23 (R/W)	GBL_PCGH_FS_EN	PCG-H Frame Sync Enable. This bit when set to 1 enables the frame sync generation and when set to 0 disables the frame sync generation.
22 (R/W)	GBL_PCGH_CLK_EN	PCG-H Clock Enable. This bit when set to 1 enables the clock generation and when set to 0 disables the clock generation.

Table 31-71: DAI_GBL_PCG_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
21 (R/W)	GBL_PCGG_FS_EN	PCG-G Frame Sync Enable. This bit when set to 1 enables the frame sync generation and when set to 0 disables the frame sync generation.
20 (R/W)	GBL_PCGG_CLK_EN	PCG-G Clock Enable. This bit when set to 1 enables the clock generation and when set to 0 disables the clock generation.
19 (R/W)	GBL_PCGD_FS_EN	PCG-D Frame Sync Enable. This bit when set to 1 enables the frame sync generation and when set to 0 disables the frame sync generation.
18 (R/W)	GBL_PCGD_CLK_EN	PCG-D Clock Enable. This bit when set to 1 enables the clock generation and when set to 0 disables the clock generation.
17 (R/W)	GBL_PCGC_FS_EN	PCG-C Frame Sync Enable. This bit when set to 1 enables the frame sync generation and when set to 0 disables the frame sync generation.
16 (R/W)	GBL_PCGC_CLK_EN	PCG-C Clock Enable. This bit when set to 1 enables the clock generation and when set to 0 disables the clock generation.
15 (R/W)	GBL_PCGF_FS_EN	PCG-F Frame Sync Enable. This bit when set to 1 enables the frame sync generation and when set to 0 disables the frame sync generation.
14 (R/W)	GBL_PCGF_CLK_EN	PCG-F Clock Enable. This bit when set to 1 enables the clock generation and when set to 0 disables the clock generation.
13 (R/W)	GBL_PCGE_FS_EN	PCG-E Frame Sync Enable. This bit when set to 1 enables the frame sync generation and when set to 0 disables the frame sync generation.
12 (R/W)	GBL_PCGE_CLK_EN	PCG-E Clock Enable. This bit when set to 1 enables the clock generation and when set to 0 disables the clock generation.
11 (R/W)	GBL_PCGB_FS_EN	PCG-B Frame Sync Enable. This bit when set to 1 enables the frame sync generation and when set to 0 disables the frame sync generation.
10 (R/W)	GBL_PCGB_CLK_EN	PCG-B Clock Enable. This bit when set to 1 enables the clock generation and when set to 0 disables the clock generation.

Table 31-71: DAI_GBL_PCG_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	GBL_PCGA_FS_EN	PCG-A Frame Sync Enable. This bit when set to 1 enables the frame sync generation and when set to 0 disables the frame sync generation.
8 (R/W)	GBL_PCGA_CLK_EN	PCG-A Clock Enable. This bit when set to 1 enables the clock generation and when set to 0 disables the clock generation.
2 (R/W)	GBL_PCG_CDGH_EN	PCG C/D/G/H Enable. This bit when set to 1 along with bits [23:16] of the same register enables the respective PCGs within a DAI1. This bit when set to 0 disables the PCGs within DAI1.
1 (R/W)	GBL_PCG_ABEF_EN	PCG A/B/E/F Enable. This bit when set to 1 along with bits [15:8] of the same register enables the respective PCGs within a DAI0. This bit when set to 0 disables the PCGs within DAI0.
0 (R/W)	GBL_PCG_EN	Global PCG Enable. This bit Globally enables the PCGs across DAI0 and DAI1. This bit when set to 1 along with bit [2:1] and [23:8] enables all the PCGs across DAI0 and DAI1. This bit when set to 0 disables all the PCGs across DAI0 and DAI1.

Global SPORT Enable Register

The DAI_GBL_SP_EN register indicate the status (signal high or low) for each pin. The individual bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

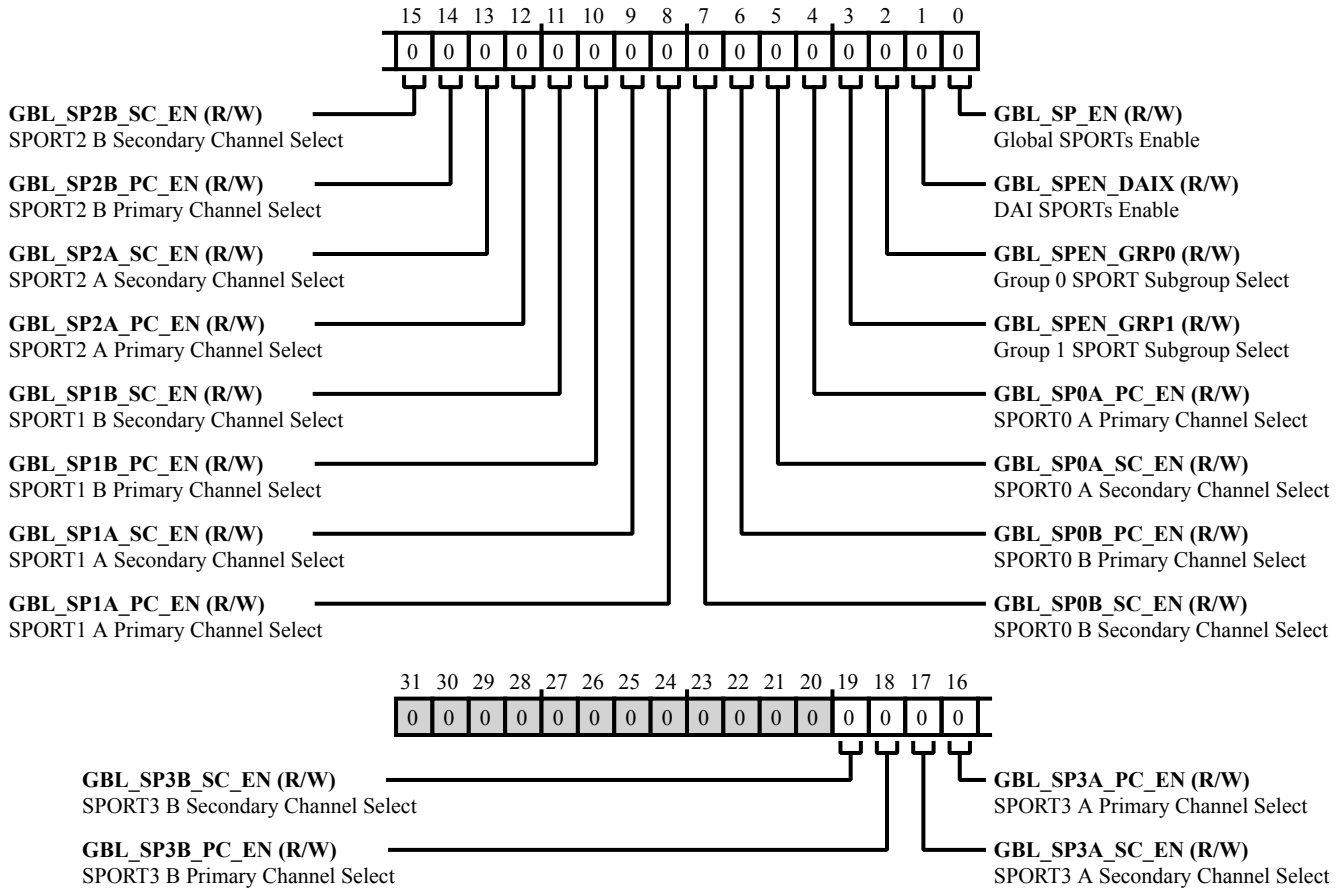


Figure 31-62: DAI_GBL_SP_EN Register Diagram

Table 31-72: DAI_GBL_SP_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/W)	GBL_SP3B_SC_EN	SPORT3 B Secondary Channel Select. The DAI_GBL_SP_EN.GBL_SP3B_SC_EN bit selects SPORT3 B as the secondary channel for global enable.
18 (R/W)	GBL_SP3B_PC_EN	SPORT3 B Primary Channel Select. The DAI_GBL_SP_EN.GBL_SP3B_PC_EN bit selects SPORT3 B as the primary channel for global enable.

Table 31-72: DAI_GBL_SP_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	GBL_SP3A_SC_EN	SPORT3 A Secondary Channel Select. The `DAI_GBL_SP_EN.GBL_SP3A_SC_EN` bit selects SPORT3 A as the secondary channel for global enable.
16 (R/W)	GBL_SP3A_PC_EN	SPORT3 A Primary Channel Select. The `DAI_GBL_SP_EN.GBL_SP3A_PC_EN` bit selects SPORT3 A as the primary channel for global enable.
15 (R/W)	GBL_SP2B_SC_EN	SPORT2 B Secondary Channel Select. The `DAI_GBL_SP_EN.GBL_SP2B_SC_EN` bit selects SPORT2 B as the secondary channel for global enable.
14 (R/W)	GBL_SP2B_PC_EN	SPORT2 B Primary Channel Select. The `DAI_GBL_SP_EN.GBL_SP2B_PC_EN` bit selects SPORT2 B as the primary channel for global enable.
13 (R/W)	GBL_SP2A_SC_EN	SPORT2 A Secondary Channel Select. The `DAI_GBL_SP_EN.GBL_SP2A_SC_EN` bit selects SPORT2 A as the secondary channel for global enable.
12 (R/W)	GBL_SP2A_PC_EN	SPORT2 A Primary Channel Select. The `DAI_GBL_SP_EN.GBL_SP2A_PC_EN` bit selects SPORT2 A as the primary channel for global enable.
11 (R/W)	GBL_SP1B_SC_EN	SPORT1 B Secondary Channel Select. The `DAI_GBL_SP_EN.GBL_SP1B_SC_EN` bit selects SPORT1 B as the secondary channel for global enable.
10 (R/W)	GBL_SP1B_PC_EN	SPORT1 B Primary Channel Select. The `DAI_GBL_SP_EN.GBL_SP1B_PC_EN` bit selects SPORT1 B as the primary channel for global enable.
9 (R/W)	GBL_SP1A_SC_EN	SPORT1 A Secondary Channel Select. The `DAI_GBL_SP_EN.GBL_SP1A_SC_EN` bit selects SPORT1 A as the secondary channel for global enable.
8 (R/W)	GBL_SP1A_PC_EN	SPORT1 A Primary Channel Select. The `DAI_GBL_SP_EN.GBL_SP1A_PC_EN` bit selects SPORT1 A as the primary channel for global enable.
7 (R/W)	GBL_SP0B_SC_EN	SPORT0 B Secondary Channel Select. The `DAI_GBL_SP_EN.GBL_SP0B_SC_EN` bit selects SPORT0 B as the secondary channel for global enable.
6 (R/W)	GBL_SP0B_PC_EN	SPORT0 B Primary Channel Select. Selects SPORT0 B primary channel for global enable.

Table 31-72: DAI_GBL_SP_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/W)	GBL_SP0A_SC_EN	SPORT0 A Secondary Channel Select. The `DAI_GBL_SP_EN.GBL_SP0A_SC_EN` bit selects SPORT0 A as the secondary channel for global enable.
4 (R/W)	GBL_SP0A_PC_EN	SPORT0 A Primary Channel Select. The `DAI_GBL_SP_EN.GBL_SP0A_PC_EN` bit selects SPORT0 A as the primary channel for global enable.
3 (R/W)	GBL_SPEN_GRP1	Group 1 SPORT Subgroup Select. The `DAI_GBL_SP_EN.GBL_SPEN_GRP1` bit creates a subgroup of selected SPORTs.
2 (R/W)	GBL_SPEN_GRP0	Group 0 SPORT Subgroup Select. The `DAI_GBL_SP_EN.GBL_SPEN_GRP0` bit creates a subgroup of selected SPORTs.
1 (R/W)	GBL_SPEN_DAIX	DAI SPORTs Enable. The `DAI_GBL_SP_EN.GBL_SPEN_DAIX` bit enables the SPORTs selected within a DAI.
0 (R/W)	GBL_SP_EN	Global SPORTs Enable. The `DAI_GBL_SP_EN.GBL_SP_EN` bit enables the selected SPORTs.

Falling-Edge Interrupt Mask Register

The DAI_IMSK_FE register masks and unmasks interrupts generated on the falling edge of a waveform. Note that any of the Group E signals can be mapped to any of the Miscellaneous interrupts (9-0).

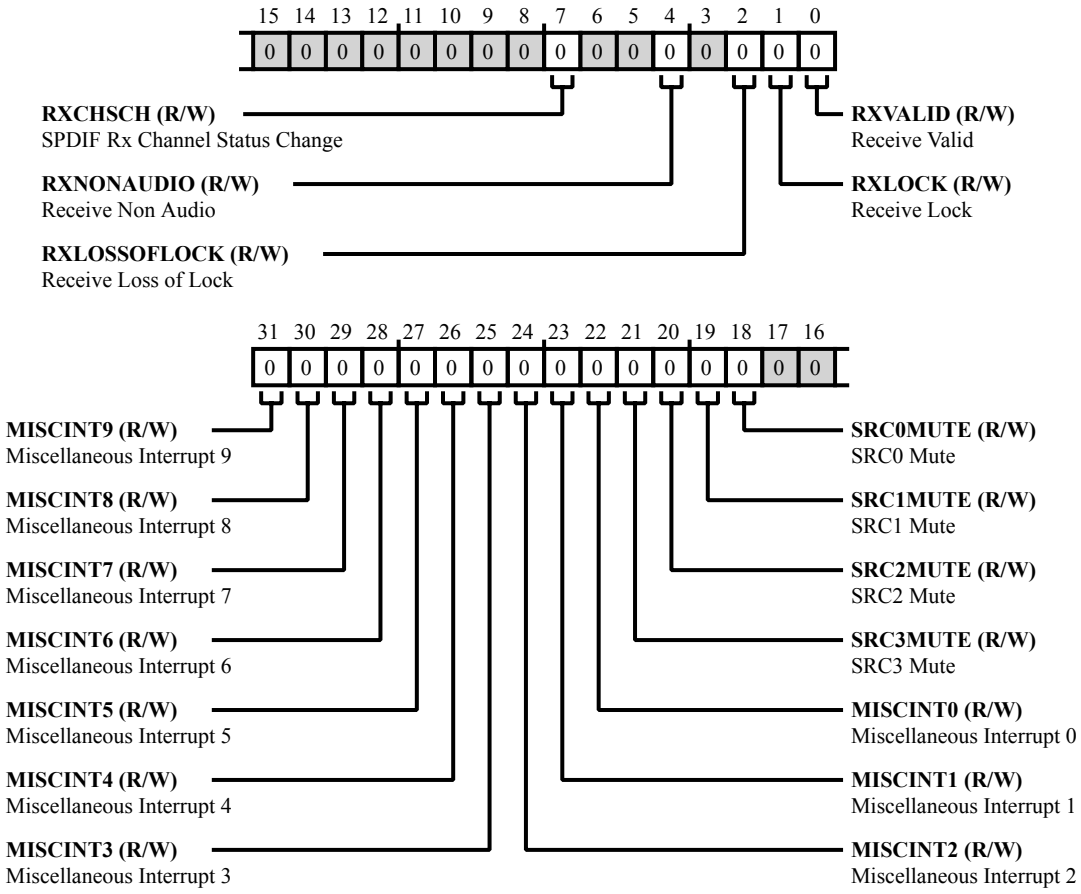


Figure 31-63: DAI_IMSK_FE Register Diagram

Table 31-73: DAI_IMSK_FE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	MISCINT9	Miscellaneous Interrupt 9. Setting the DAI_IMSK_FE.MISCINT9 bit unmasks the interrupt for the falling edge of the routed signal.
30 (R/W)	MISCINT8	Miscellaneous Interrupt 8. Setting the DAI_IMSK_FE.MISCINT8 bit unmasks the interrupt for the falling edge of the routed signal.

Table 31-73: DAI_IMSK_FE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	MISCINT7	Miscellaneous Interrupt 7. Setting the DAI_IMSK_FE.MISCINT7 bit unmasks the interrupt for the falling edge of the routed signal.
28 (R/W)	MISCINT6	Miscellaneous Interrupt 6. Setting the DAI_IMSK_FE.MISCINT6 bit unmasks the interrupt for the falling edge of the routed signal.
27 (R/W)	MISCINT5	Miscellaneous Interrupt 5. Setting the DAI_IMSK_FE.MISCINT5 bit unmasks the interrupt for the falling edge of the routed signal.
26 (R/W)	MISCINT4	Miscellaneous Interrupt 4. Setting the DAI_IMSK_FE.MISCINT4 bit unmasks the interrupt for the falling edge of the routed signal.
25 (R/W)	MISCINT3	Miscellaneous Interrupt 3. Setting the DAI_IMSK_FE.MISCINT3 bit unmasks the interrupt for the falling edge of the routed signal.
24 (R/W)	MISCINT2	Miscellaneous Interrupt 2. Setting the DAI_IMSK_FE.MISCINT2 bit unmasks the interrupt for the falling edge of the routed signal.
23 (R/W)	MISCINT1	Miscellaneous Interrupt 1. Setting the DAI_IMSK_FE.MISCINT1 bit unmasks the interrupt for the falling edge of the routed signal.
22 (R/W)	MISCINT0	Miscellaneous Interrupt 0. Setting the DAI_IMSK_FE.MISCINT0 bit unmasks the interrupt for the falling edge of the routed signal.
21 (R/W)	SRC3MUTE	SRC3 Mute. The DAI_IMSK_FE.SRC3MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
20 (R/W)	SRC2MUTE	SRC2 Mute. The DAI_IMSK_FE.SRC2MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
19 (R/W)	SRC1MUTE	SRC1 Mute. The DAI_IMSK_FE.SRC1MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.

Table 31-73: DAI_IMSK_FE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/W)	SRC0MUTE	SRC0 Mute. The DAI_IMSK_FE.SRC0MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the falling edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
7 (R/W)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IMSK_FE.RXCHSCH bit masks or unmasks RXCHSCH interrupt for the falling edge of this interrupt. This interrupt is set if there is a change in the SPDIF Rx channel status bits (SPDIF_RX_STAT0/1_A/B).
4 (R/W)	RXNONAUDIO	Receive Non Audio. The DAI_IMSK_FE.RXNONAUDIO bit masks or unmasks the non audio frame mode interrupt for the falling edge of this interrupt. If the channel status indicates non-PCM audio, the NONAUDIO bit flag is set.
2 (R/W)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IMSK_FE.RXLOSSOFLOCK bit masks or unmasks the emphasis loss of lock interrupt for the falling edge of this interrupt. The loss of lock status is set by the S/PDIF receiver if receiver loses the lock of biphase stream.
1 (R/W)	RXLOCK	Receive Lock. The DAI_IMSK_FE.RXLOCK bit masks or unmasks the S/PDIF receiver lock for the falling edge of this interrupt. This interrupt occurs when the S/PDIF receiver locks to the S/PDIF stream.
0 (R/W)	RXVALID	Receive Valid. Setting the DAI_IMSK_FE.RXVALID bit unmasks the interrupt for the falling edge of the routed signal. This interrupt is set based on whether data received by the S/PDIF is linear PCM or non-linear audio data.

Core Interrupt Priority Assignment Register

The DAI_IMSK_PRI register masks interrupts for DAI high or DAI low interrupt priority.

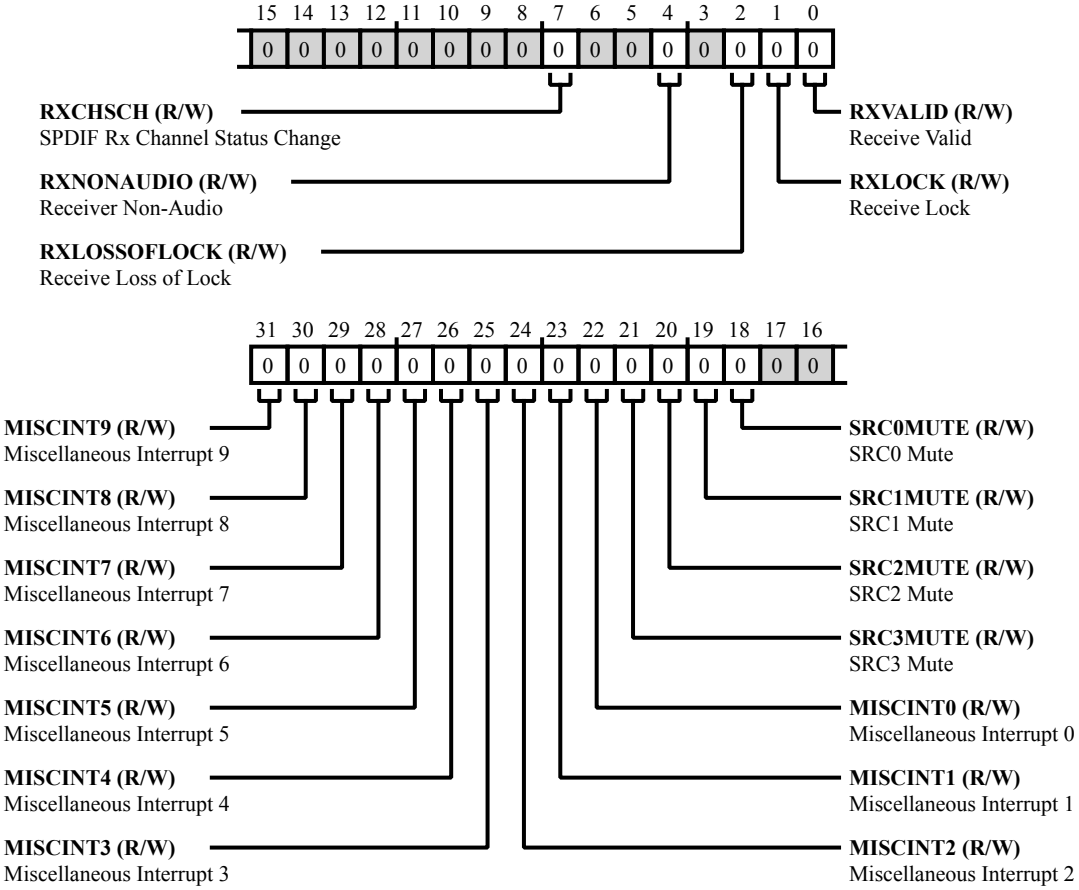


Figure 31-64: DAI_IMSK_PRI Register Diagram

Table 31-74: DAI_IMSK_PRI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	MISCINT9	Miscellaneous Interrupt 9. If the DAI_IMSK_PRI.MISCINT9 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT9 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
30 (R/W)	MISCINT8	Miscellaneous Interrupt 8. If the DAI_IMSK_PRI.MISCINT8 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT8 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.

Table 31-74: DAI_IMSK_PRI Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	MISCINT7	Miscellaneous Interrupt 7. If the DAI_IMSK_PRI.MISCINT7 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT7 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
28 (R/W)	MISCINT6	Miscellaneous Interrupt 6. If the DAI_IMSK_PRI.MISCINT6 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT6 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
27 (R/W)	MISCINT5	Miscellaneous Interrupt 5. If the DAI_IMSK_PRI.MISCINT5 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT5 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
26 (R/W)	MISCINT4	Miscellaneous Interrupt 4. If the DAI_IMSK_PRI.MISCINT4 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT4 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
25 (R/W)	MISCINT3	Miscellaneous Interrupt 3. If the DAI_IMSK_PRI.MISCINT3 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT3 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
24 (R/W)	MISCINT2	Miscellaneous Interrupt 2. If the DAI_IMSK_PRI.MISCINT2 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT2 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
23 (R/W)	MISCINT1	Miscellaneous Interrupt 1. If the DAI_IMSK_PRI.MISCINT1 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT1 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
22 (R/W)	MISCINT0	Miscellaneous Interrupt 0. If the DAI_IMSK_PRI.MISCINT0 bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.MISCINT0 bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
21 (R/W)	SRC3MUTE	SRC3 Mute. If the DAI_IMSK_PRI.SRC3MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC3MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.

Table 31-74: DAI_IMSK_PRI Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
20 (R/W)	SRC2MUTE	SRC2 Mute. If the DAI_IMSK_PRI.SRC2MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC2MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
19 (R/W)	SRC1MUTE	SRC1 Mute. If the DAI_IMSK_PRI.SRC1MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC1MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
18 (R/W)	SRC0MUTE	SRC0 Mute. If the DAI_IMSK_PRI.SRC0MUTE bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.SRC0MUTE bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
7 (R/W)	RXCHSCH	SPDIF Rx Channel Status Change. If the DAI_IMSK_PRI.RXCHSCH bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.RXCHSCH bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
4 (R/W)	RXNONAUDIO	Receiver Non-Audio. If the DAI_IMSK_PRI.RXNONAUDIO bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.RXNONAUDIO bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
2 (R/W)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IMSK_PRI.RXLOSSOFLOCK bit masks or unmasks the emphasis loss of lock interrupt for the falling edge of this interrupt. The loss of lock status is set by the S/PDIF receiver if receiver loses the lock of biphase stream.
1 (R/W)	RXLOCK	Receive Lock. If the DAI_IMSK_PRI.RXLOCK bit is cleared (=0), the interrupt is mapped to INTR_DAI_IRQL signal from the SEC. If the DAI_IMSK_PRI.RXLOCK bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.
0 (R/W)	RXVALID	Receive Valid. If the DAI_IMSK_PRI.RXVALID bit is cleared (=0), the interrupt is mapped to the INTR_DAI_IRQL from the SEC. If the DAI_IMSK_PRI.RXVALID bit is set (=1) the interrupt is mapped to INTR_DAI_IRQH signal from the SEC.

Rising-Edge Interrupt Mask Register

The DAI_IMSK_RE register masks and unmasks interrupts generated on the rising edge of a waveform. Note that any of the Group E signals can be mapped to any of the miscellaneous interrupts (9-0).

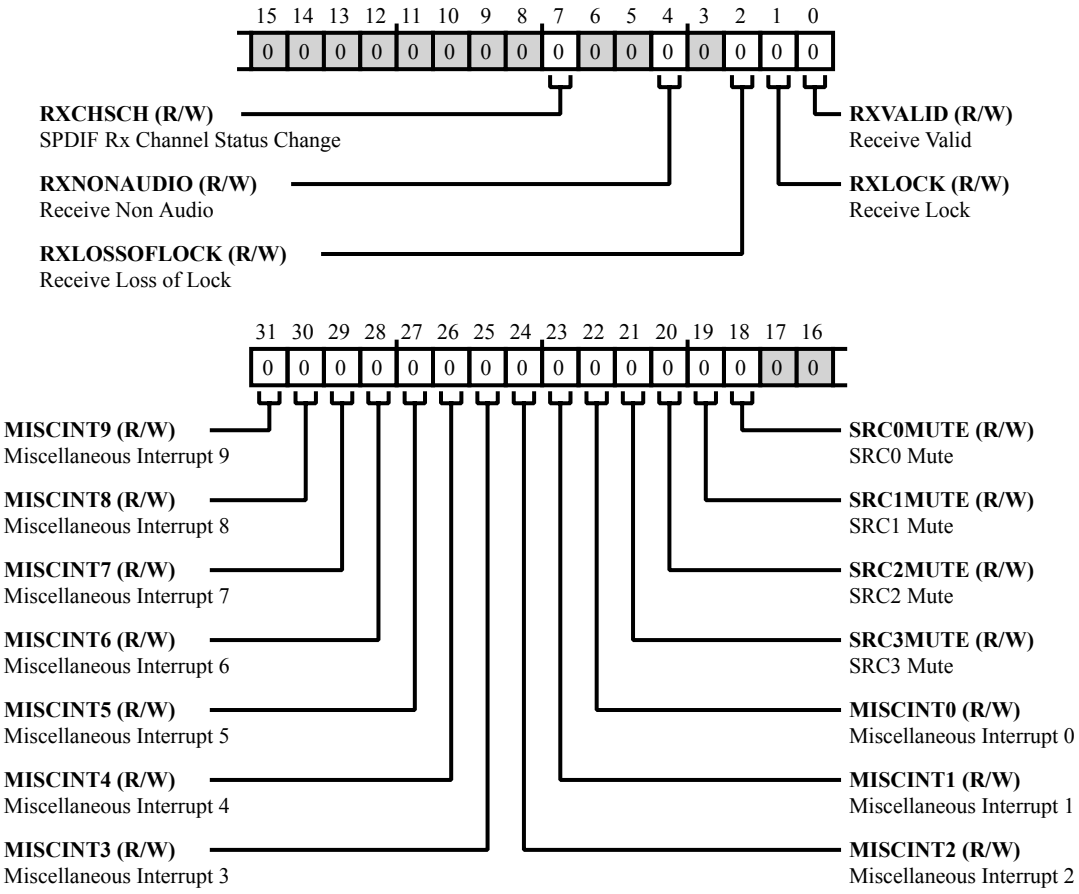


Figure 31-65: DAI_IMSK_RE Register Diagram

Table 31-75: DAI_IMSK_RE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	MISCINT9	Miscellaneous Interrupt 9. Setting the DAI_IMSK_RE.MISCINT9 bit unmasks the interrupt for the rising edge of the routed signal.
30 (R/W)	MISCINT8	Miscellaneous Interrupt 8. Setting the DAI_IMSK_RE.MISCINT8 bit unmasks the interrupt for the rising edge of the routed signal.

Table 31-75: DAI_IMSK_RE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	MISCINT7	Miscellaneous Interrupt 7. Setting the DAI_IMSK_RE.MISCINT7 bit unmasks the interrupt for the rising edge of the routed signal.
28 (R/W)	MISCINT6	Miscellaneous Interrupt 6. Setting the DAI_IMSK_RE.MISCINT6 bit unmasks the interrupt for the rising edge of the routed signal.
27 (R/W)	MISCINT5	Miscellaneous Interrupt 5. Setting the DAI_IMSK_RE.MISCINT5 bit unmasks the interrupt for the rising edge of the routed signal.
26 (R/W)	MISCINT4	Miscellaneous Interrupt 4. Setting the DAI_IMSK_RE.MISCINT4 bit unmasks the interrupt for the rising edge of the routed signal.
25 (R/W)	MISCINT3	Miscellaneous Interrupt 3. Setting the DAI_IMSK_RE.MISCINT3 bit unmasks the interrupt for the rising edge of the routed signal.
24 (R/W)	MISCINT2	Miscellaneous Interrupt 2. Setting the DAI_IMSK_RE.MISCINT2 bit unmasks the interrupt for the rising edge of the routed signal.
23 (R/W)	MISCINT1	Miscellaneous Interrupt 1. Setting the DAI_IMSK_RE.MISCINT1 bit unmasks the interrupt for the rising edge of the routed signal.
22 (R/W)	MISCINT0	Miscellaneous Interrupt 0. Setting the DAI_IMSK_RE.MISCINT0 bit unmasks the interrupt for the rising edge of the routed signal.
21 (R/W)	SRC3MUTE	SRC3 Mute. The DAI_IMSK_RE.SRC3MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
20 (R/W)	SRC2MUTE	SRC2 Mute. The DAI_IMSK_RE.SRC2MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
19 (R/W)	SRC1MUTE	SRC1 Mute. The DAI_IMSK_RE.SRC1MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.

Table 31-75: DAI_IMSK_RE Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/W)	SRC0MUTE	SRC0 Mute. The DAI_IMSK_RE.SRC0MUTE bit masks or unmasks the corresponding SRC mute out interrupt for the rising edge of this interrupt. This interrupt can be generated either entering mute, exiting muting or both.
7 (R/W)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IMSK_RE.RXCHSCH bit masks or unmasks RXCHSCH (Status Change) interrupt for the rising edge of this interrupt. This interrupt is set if there is a change in the SPDIF Rx channel status bits (SPDIF_RX_STAT0/1_A/B).
4 (R/W)	RXNONAUDIO	Receive Non Audio. The DAI_IMSK_RE.RXNONAUDIO bit masks or unmasks the non audio frame mode interrupt for the rising edge of this interrupt. If the channel status indicates non-PCM audio, the NONAUDIO bit flag is set.
2 (R/W)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IMSK_RE.RXLOSSOFLOCK bit masks or unmasks the emphasis loss of lock interrupt for the rising edge of this interrupt. The loss of lock status is set by the S/PDIF receiver if receiver loses the lock of biphase stream.
1 (R/W)	RXLOCK	Receive Lock. The DAI_IMSK_RE.RXLOCK bit masks or unmasks the S/PDIF receiver lock for the rising edge of this interrupt. This interrupt occurs when the S/PDIF receiver locks to the S/PDIF stream.
0 (R/W)	RXVALID	Receive Valid. Setting the DAI_IMSK_RE.RXVALID bit unmasks the interrupt for the rising edge of the routed signal. This interrupt is set based on whether data received by the S/PDIF is linear PCM or non-linear audio data.

High Priority Interrupt Latch Register

The DAI_IRPTL_H register holds the high priority latched interrupt status for interrupt requests that have been unmasked (enabled) by the DAI_IMSK_FE, DAI_IMSK_RE registers and mapped to the INTR_DAI_IRQH signal in the SEC by DAI_IMSK_PRI registers. If a bit in this register is already set and the corresponding interrupt is masked in the DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI registers, the latch holds its old value, leaving the interrupt asserted until its mask registers (DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI) are reset by software with a W1C operation.

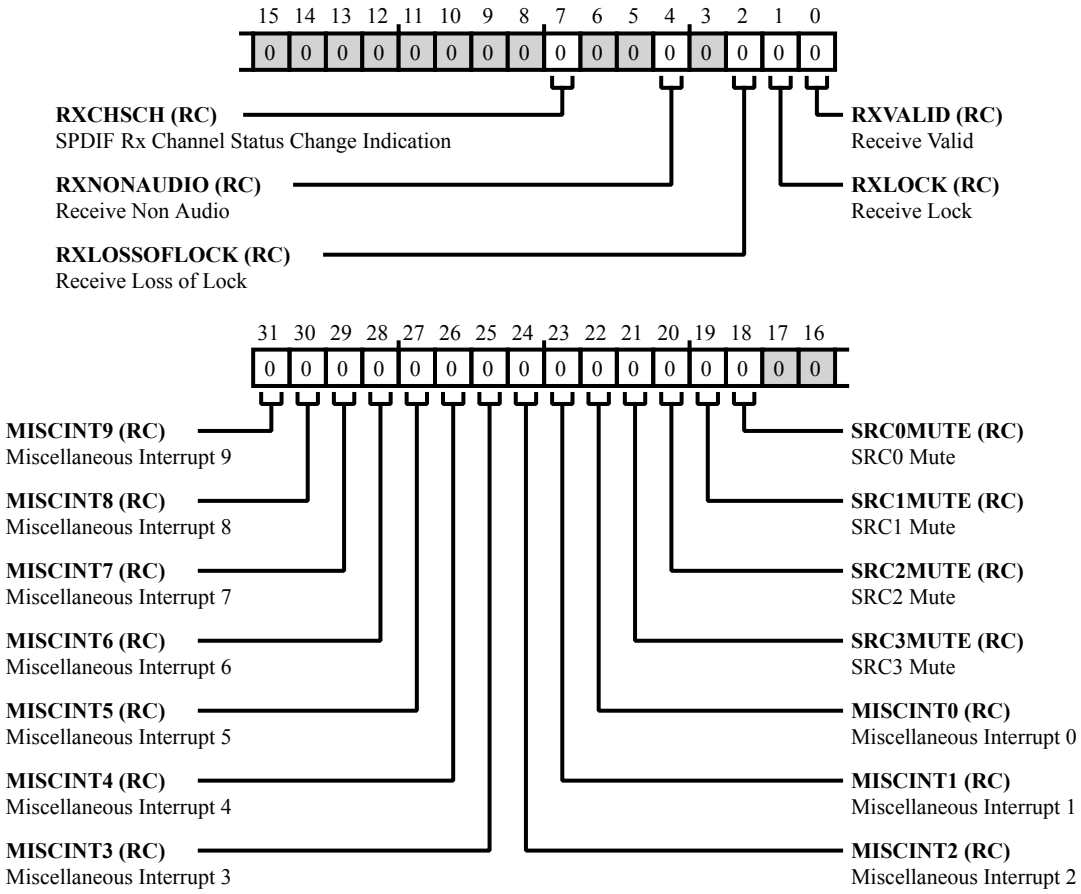


Figure 31-66: DAI_IRPTL_H Register Diagram

Table 31-76: DAI_IRPTL_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (RC/NW)	MISCINT9	Miscellaneous Interrupt 9. The DAI_IRPTL_H.MISCINT9 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.

Table 31-76: DAI_IRPTL_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (RC/NW)	MISCINT8	Miscellaneous Interrupt 8. The DAI_IRPTL_H.MISCINT8 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
29 (RC/NW)	MISCINT7	Miscellaneous Interrupt 7. The DAI_IRPTL_H.MISCINT7 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
28 (RC/NW)	MISCINT6	Miscellaneous Interrupt 6. The DAI_IRPTL_H.MISCINT6 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
27 (RC/NW)	MISCINT5	Miscellaneous Interrupt 5. The DAI_IRPTL_H.MISCINT5 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
26 (RC/NW)	MISCINT4	Miscellaneous Interrupt 4. The DAI_IRPTL_H.MISCINT4 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
25 (RC/NW)	MISCINT3	Miscellaneous Interrupt 3. The DAI_IRPTL_H.MISCINT3 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
24 (RC/NW)	MISCINT2	Miscellaneous Interrupt 2. The DAI_IRPTL_H.MISCINT2 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
23 (RC/NW)	MISCINT1	Miscellaneous Interrupt 1. The DAI_IRPTL_H.MISCINT1 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
22 (RC/NW)	MISCINT0	Miscellaneous Interrupt 0. The DAI_IRPTL_H.MISCINT0 bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
21 (RC/NW)	SRC3MUTE	SRC3 Mute. The DAI_IRPTL_H.SRC3MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
20 (RC/NW)	SRC2MUTE	SRC2 Mute. The DAI_IRPTL_H.SRC2MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.

Table 31-76: DAI_IRPTL_H Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
19 (RC/NW)	SRC1MUTE	SRC1 Mute. The DAI_IRPTL_H.SRC1MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
18 (RC/NW)	SRC0MUTE	SRC0 Mute. The DAI_IRPTL_H.SRC0MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
7 (RC/NW)	RXCHSCH	SPDIF Rx Channel Status Change Indication. The DAI_IRPTL_H.RXCHSCH bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
4 (RC/NW)	RXNONAUDIO	Receive Non Audio. The DAI_IRPTL_H.RXNONAUDIO bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
2 (RC/NW)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IRPTL_H.RXLOSSOFLOCK bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
1 (RC/NW)	RXLOCK	Receive Lock. The DAI_IRPTL_H.RXLOCK bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.
0 (RC/NW)	RXVALID	Receive Valid. The DAI_IRPTL_H.RXVALID bit is set if the interrupt is mapped to the INTR_DAI_IRQH signal in the SEC using the DAI_IMSK_PRI register.

Shadow High Priority Interrupt Latch Register

The DAI_IRPTL_HS register is the shadow register of the DAI_IRPTL_H register. Its content is the same as the DAI_IRPTL_H register. Reading the DAI_IRPTL_HS register does not affect its contents while reading the contents of the DAI_IRPTL_H registers clears it.

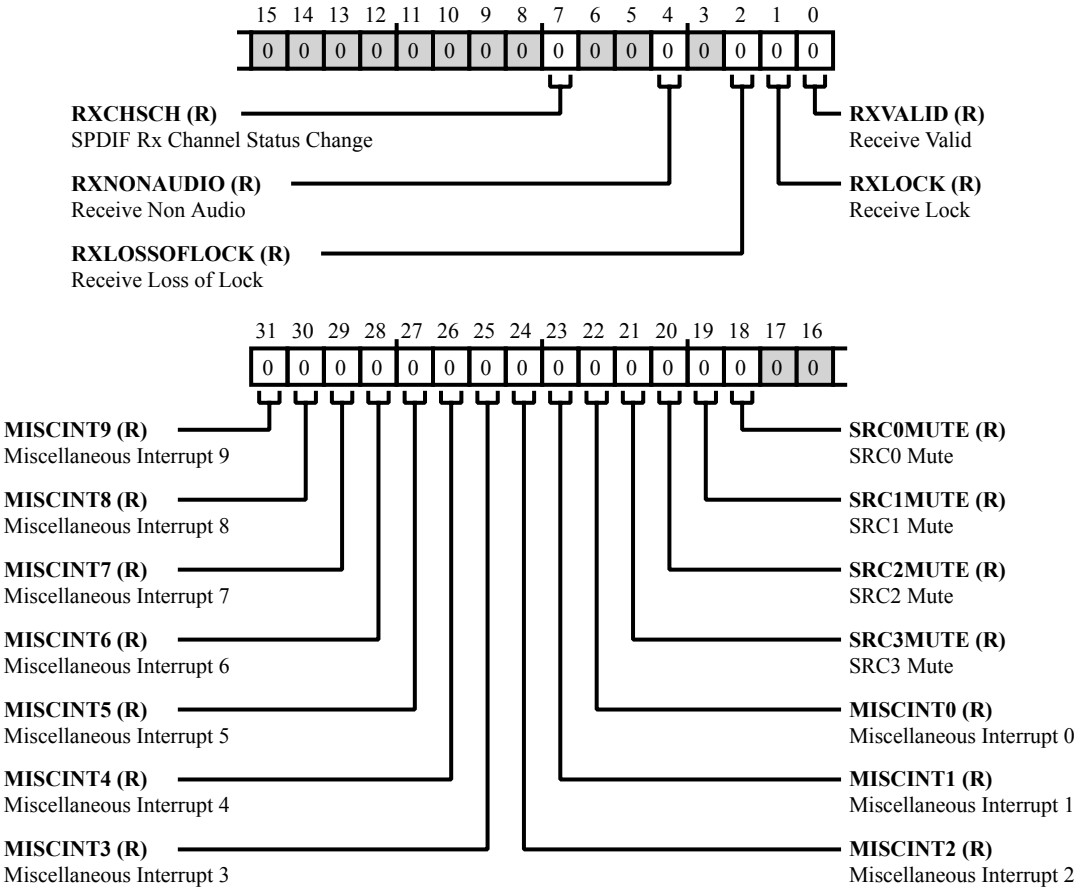


Figure 31-67: DAI_IRPTL_HS Register Diagram

Table 31-77: DAI_IRPTL_HS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MISCINT9	Miscellaneous Interrupt 9. The DAI_IRPTL_HS.MISCINT9 bit is the shadow of the DAI_IRPTL_H.MISCINT9 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT9 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT9 bit clears it.

Table 31-77: DAI_IRPTL_HS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/NW)	MISCINT8	Miscellaneous Interrupt 8. The DAI_IRPTL_HS.MISCINT8 bit is the shadow of the DAI_IRPTL_H.MISCINT8 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT8 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT8 bit clears it.
29 (R/NW)	MISCINT7	Miscellaneous Interrupt 7. The DAI_IRPTL_HS.MISCINT7 bit is the shadow of the DAI_IRPTL_H.MISCINT7 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT7 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT7 bit clears it.
28 (R/NW)	MISCINT6	Miscellaneous Interrupt 6. The DAI_IRPTL_HS.MISCINT6 bit is the shadow of the DAI_IRPTL_H.MISCINT6 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT6 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT6 bit clears it.
27 (R/NW)	MISCINT5	Miscellaneous Interrupt 5. The DAI_IRPTL_HS.MISCINT5 bit is the shadow of the DAI_IRPTL_H.MISCINT5 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT5 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT5 bit clears it.
26 (R/NW)	MISCINT4	Miscellaneous Interrupt 4. The DAI_IRPTL_HS.MISCINT4 bit is the shadow of the DAI_IRPTL_H.MISCINT4 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT4 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT4 bit clears it.
25 (R/NW)	MISCINT3	Miscellaneous Interrupt 3. The DAI_IRPTL_HS.MISCINT3 bit is the shadow of the DAI_IRPTL_H.MISCINT3 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT3 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT3 bit clears it.
24 (R/NW)	MISCINT2	Miscellaneous Interrupt 2. The DAI_IRPTL_HS.MISCINT2 bit is the shadow of the DAI_IRPTL_H.MISCINT2 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT2 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT2 bit clears it.

Table 31-77: DAI_IRPTL_HS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
23 (R/NW)	MISCINT1	Miscellaneous Interrupt 1. The DAI_IRPTL_HS.MISCINT1 bit is the shadow of the DAI_IRPTL_H.MISCINT1 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT1 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT1 bit clears it.
22 (R/NW)	MISCINT0	Miscellaneous Interrupt 0. The DAI_IRPTL_HS.MISCINT0 bit is the shadow of the DAI_IRPTL_H.MISCINT0 bit and contains the same content. Reading the DAI_IRPTL_HS.MISCINT0 bit does not affect its contents while reading the DAI_IRPTL_H.MISCINT0 bit clears it.
21 (R/NW)	SRC3MUTE	SRC3 Mute. The DAI_IRPTL_HS.SRC3MUTE bit is the shadow of the DAI_IRPTL_H.SRC3MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC3MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC3MUTE bit clears it.
20 (R/NW)	SRC2MUTE	SRC2 Mute. The DAI_IRPTL_HS.SRC2MUTE bit is the shadow of the DAI_IRPTL_H.SRC2MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC2MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC2MUTE bit clears it.
19 (R/NW)	SRC1MUTE	SRC1 Mute. The DAI_IRPTL_HS.SRC1MUTE bit is the shadow of the DAI_IRPTL_H.SRC1MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC1MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC1MUTE bit clears it.
18 (R/NW)	SRC0MUTE	SRC0 Mute. The DAI_IRPTL_HS.SRC0MUTE bit is the shadow of the DAI_IRPTL_H.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_HS.SRC0MUTE bit does not affect its contents while reading the DAI_IRPTL_H.SRC0MUTE bit clears it.
7 (R/NW)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IRPTL_HS.RXCHSCH bit is the shadow of the DAI_IRPTL_H.RXCHSCH bit and contains the same content. Reading the DAI_IRPTL_HS.RXCHSCH bit does not affect its contents while reading the DAI_IRPTL_H.RXCHSCH bit clears it.

Table 31-77: DAI_IRPTL_HS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/NW)	RXNONAUDIO	Receive Non Audio. The `DAI_IRPTL_HS.RXNONAUDIO` bit is the shadow of `DAI_IRPTL_H.RXNONAUDIO` bit and contains the same content. Reading this bit does not affect its content.
2 (R/NW)	RXLOSSOFLOCK	Receive Loss of Lock. The `DAI_IRPTL_HS.RXLOSSOFLOCK` bit is the shadow of `DAI_IRPTL_H.RXLOSSOFLOCK` bit and contains the same content. Reading this bit does not affect its content.
1 (R/NW)	RXLOCK	Receive Lock. The `DAI_IRPTL_HS.RXLOCK` bit is the shadow of `DAI_IRPTL_H.RXLOCK` bit and contains the same content. Reading this bit does not affect its content.
0 (R/NW)	RXVALID	Receive Valid. Setting the `DAI_IRPTL_HS.RXVALID` bit is the shadow of `DAI_IRPTL_H.RXVALID` bit and contains the same content. Reading this bit does not affect its content.

Low Priority Interrupt Latch Register

The DAI_IRPTL_L register holds the low priority latched interrupt status for interrupt requests that have been unmasked (enabled) by the DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI registers. If a bit in DAI_IRPTL_L is already set and the corresponding interrupt is masked in DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI registers, the latch holds its old value, leaving the interrupt asserted until its masked registers (DAI_IMSK_FE, DAI_IMSK_RE or DAI_IMSK_PRI) are reset by software with a W1C operation.

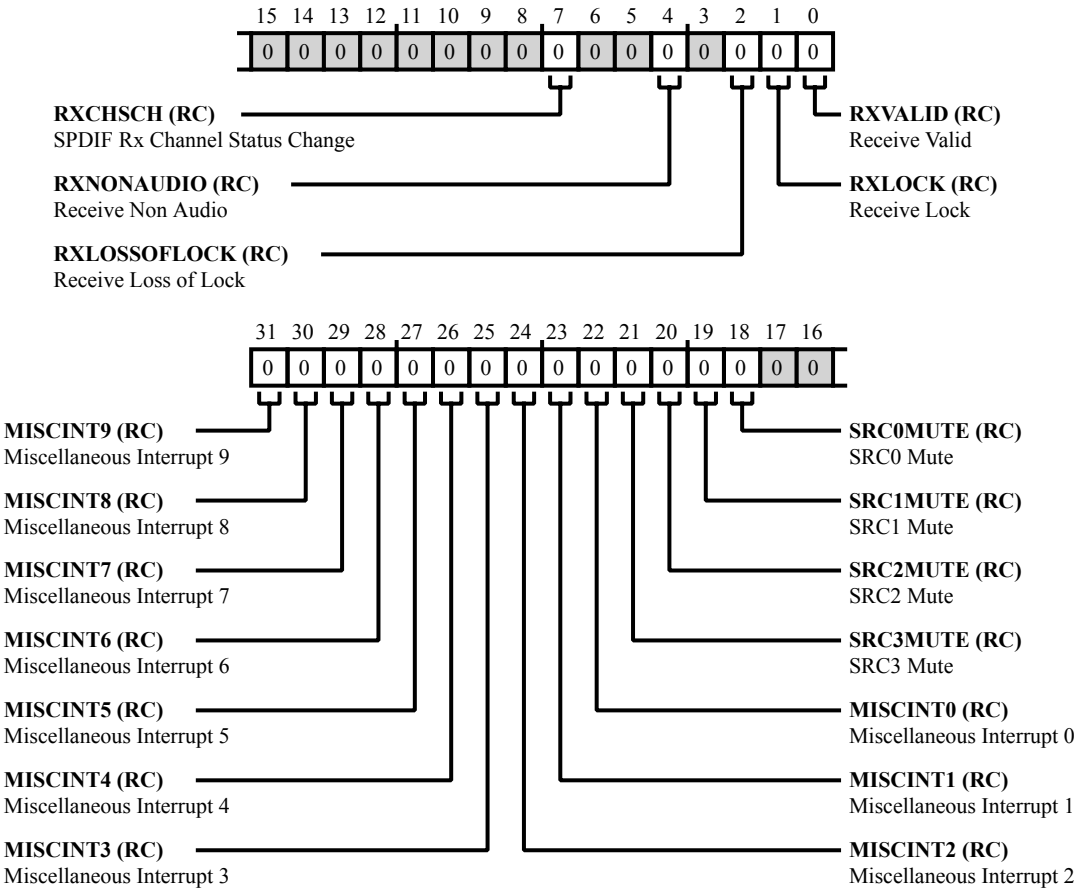


Figure 31-68: DAI_IRPTL_L Register Diagram

Table 31-78: DAI_IRPTL_L Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (RC/NW)	MISCINT9	Miscellaneous Interrupt 9. The DAI_IRPTL_L.MISCINT9 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.

Table 31-78: DAI_IRPTL_L Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (RC/NW)	MISCINT8	Miscellaneous Interrupt 8. The DAI_IRPTL_L.MISCINT8 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
29 (RC/NW)	MISCINT7	Miscellaneous Interrupt 7. The DAI_IRPTL_L.MISCINT7 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
28 (RC/NW)	MISCINT6	Miscellaneous Interrupt 6. The DAI_IRPTL_L.MISCINT6 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
27 (RC/NW)	MISCINT5	Miscellaneous Interrupt 5. The DAI_IRPTL_L.MISCINT5 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
26 (RC/NW)	MISCINT4	Miscellaneous Interrupt 4. The DAI_IRPTL_L.MISCINT4 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
25 (RC/NW)	MISCINT3	Miscellaneous Interrupt 3. The DAI_IRPTL_L.MISCINT3 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
24 (RC/NW)	MISCINT2	Miscellaneous Interrupt 2. The DAI_IRPTL_L.MISCINT2 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
23 (RC/NW)	MISCINT1	Miscellaneous Interrupt 1. The DAI_IRPTL_L.MISCINT1 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
22 (RC/NW)	MISCINT0	Miscellaneous Interrupt 0. The DAI_IRPTL_L.MISCINT0 bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
21 (RC/NW)	SRC3MUTE	SRC3 Mute. The DAI_IRPTL_L.SRC3MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
20 (RC/NW)	SRC2MUTE	SRC2 Mute. The DAI_IRPTL_L.SRC2MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.

Table 31-78: DAI_IRPTL_L Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
19 (RC/NW)	SRC1MUTE	SRC1 Mute. The DAI_IRPTL_L.SRC1MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
18 (RC/NW)	SRC0MUTE	SRC0 Mute. The DAI_IRPTL_L.SRC0MUTE bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
7 (RC/NW)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IRPTL_L.RXCHSCH bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
4 (RC/NW)	RXNONAUDIO	Receive Non Audio. The DAI_IRPTL_L.RXNONAUDIO bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
2 (RC/NW)	RXLOSSOFLOCK	Receive Loss of Lock. The DAI_IRPTL_L.RXLOSSOFLOCK bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
1 (RC/NW)	RXLOCK	Receive Lock. The DAI_IRPTL_L.RXLOCK bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.
0 (RC/NW)	RXVALID	Receive Valid. The DAI_IRPTL_L.RXVALID bit is set if the interrupt is mapped to the INTR_DAI_IRQL signal in the SEC using the DAI_IMSK_PRI register.

Shadow Low Priority Interrupt Latch Register

The DAI_IRPTL_LS register is the shadow register of the DAI_IRPTL_L register. Its content is the same as the DAI_IRPTL_L register. Reading the DAI_IRPTL_LS register does not affect its contents while reading the contents of the DAI_IRPTL_L registers clears it.

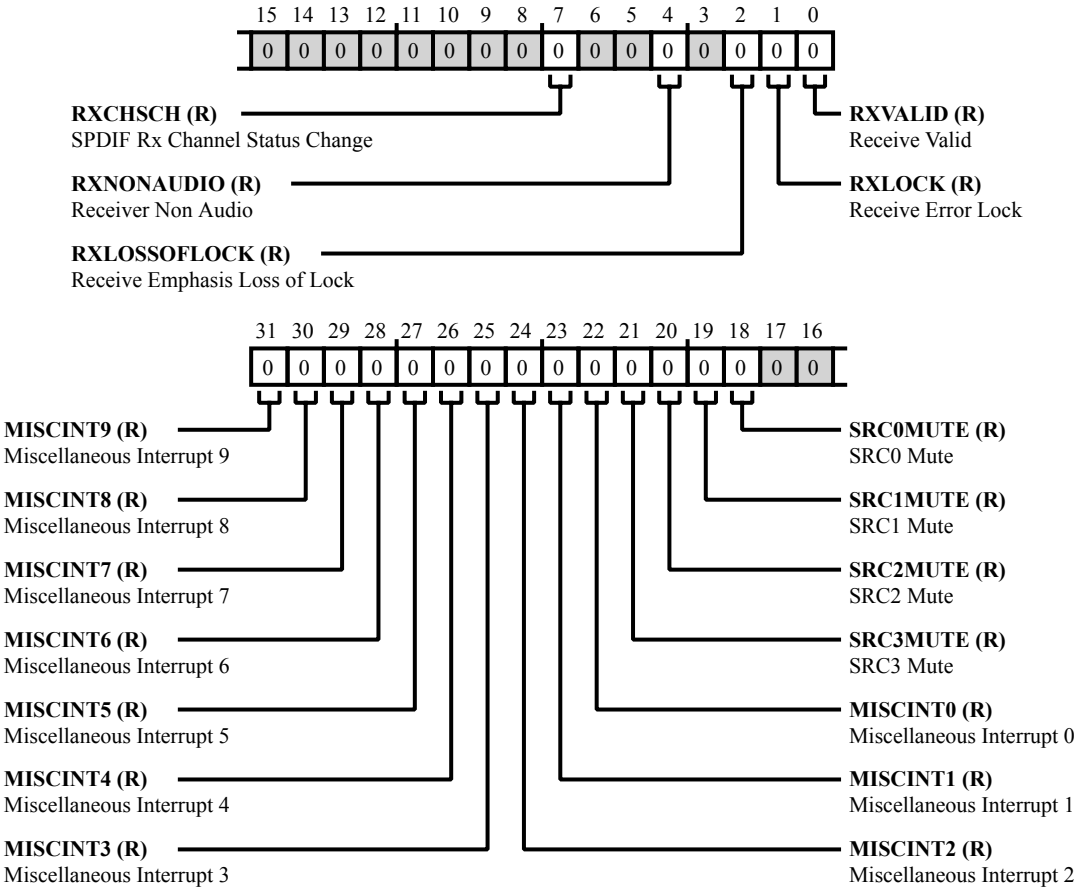


Figure 31-69: DAI_IRPTL_LS Register Diagram

Table 31-79: DAI_IRPTL_LS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MISCINT9	Miscellaneous Interrupt 9. The DAI_IRPTL_LS.MISCINT9 bit is the shadow of the DAI_IRPTL_L.MISCINT9 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT9 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT9 bit clears it.

Table 31-79: DAI_IRPTL_LS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
30 (R/NW)	MISCINT8	Miscellaneous Interrupt 8. The DAI_IRPTL_LS.MISCINT8 bit is the shadow of the DAI_IRPTL_L.MISCINT8 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT8 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT8 bit clears it.
29 (R/NW)	MISCINT7	Miscellaneous Interrupt 7. The DAI_IRPTL_LS.MISCINT7 bit is the shadow of the DAI_IRPTL_L.MISCINT7 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT7 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT7 bit clears it.
28 (R/NW)	MISCINT6	Miscellaneous Interrupt 6. The DAI_IRPTL_LS.MISCINT6 bit is the shadow of the DAI_IRPTL_L.MISCINT6 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT6 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT6 bit clears it.
27 (R/NW)	MISCINT5	Miscellaneous Interrupt 5. The DAI_IRPTL_LS.MISCINT5 bit is the shadow of the DAI_IRPTL_L.MISCINT5 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT5 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT5 bit clears it.
26 (R/NW)	MISCINT4	Miscellaneous Interrupt 4. The DAI_IRPTL_LS.MISCINT4 bit is the shadow of the DAI_IRPTL_L.MISCINT4 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT4 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT4 bit clears it.
25 (R/NW)	MISCINT3	Miscellaneous Interrupt 3. The DAI_IRPTL_LS.MISCINT3 bit is the shadow of the DAI_IRPTL_L.MISCINT3 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT3 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT3 bit clears it.
24 (R/NW)	MISCINT2	Miscellaneous Interrupt 2. The DAI_IRPTL_LS.MISCINT2 bit is the shadow of the DAI_IRPTL_L.MISCINT2 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT2 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT2 bit clears it.

Table 31-79: DAI_IRPTL_LS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
23 (R/NW)	MISCINT1	Miscellaneous Interrupt 1. The DAI_IRPTL_LS.MISCINT1 bit is the shadow of the DAI_IRPTL_L.MISCINT1 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT1 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT1 bit clears it.
22 (R/NW)	MISCINT0	Miscellaneous Interrupt 0. The DAI_IRPTL_LS.MISCINT0 bit is the shadow of the DAI_IRPTL_L.MISCINT0 bit and contains the same content. Reading the DAI_IRPTL_LS.MISCINT0 bit does not affect its contents while reading the DAI_IRPTL_L.MISCINT0 bit clears it.
21 (R/NW)	SRC3MUTE	SRC3 Mute. The DAI_IRPTL_LS.SRC3MUTE bit is the shadow of the DAI_IRPTL_L.SRC3MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC3MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC3MUTE bit clears it.
20 (R/NW)	SRC2MUTE	SRC2 Mute. The DAI_IRPTL_LS.SRC2MUTE bit is the shadow of the DAI_IRPTL_L.SRC2MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC2MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC2MUTE bit clears it.
19 (R/NW)	SRC1MUTE	SRC1 Mute. The DAI_IRPTL_LS.SRC1MUTE bit is the shadow of the DAI_IRPTL_L.SRC1MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC1MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC1MUTE bit clears it.
18 (R/NW)	SRC0MUTE	SRC0 Mute. The DAI_IRPTL_LS.SRC0MUTE bit is the shadow of the DAI_IRPTL_L.SRC0MUTE bit and contains the same content. Reading the DAI_IRPTL_LS.SRC0MUTE bit does not affect its contents while reading the DAI_IRPTL_L.SRC0MUTE bit clears it.
7 (R/NW)	RXCHSCH	SPDIF Rx Channel Status Change. The DAI_IRPTL_LS.RXCHSCH bit is the shadow of the DAI_IRPTL_L.RXCHSCH bit and contains the same content. Reading the DAI_IRPTL_LS.RXCHSCH bit does not affect its contents while reading the DAI_IRPTL_L.RXCHSCH bit clears it.

Table 31-79: DAI_IRPTL_LS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/NW)	RXNONAUDIO	Receiver Non Audio. The DAI_IRPTL_LS.RXNONAUDIO bit is the shadow of the DAI_IRPTL_L.RXNONAUDIO bit and contains the same content. Reading the DAI_IRPTL_LS.RXNONAUDIO bit does not affect its contents while reading the DAI_IRPTL_L.RXNONAUDIO bit clears it.
2 (R/NW)	RXLOSSOFLOCK	Receive Emphasis Loss of Lock. The DAI_IRPTL_LS.RXLOSSOFLOCK bit is the shadow of DAI_IRPTL_L.RXLOSSOFLOCK bit and contains the same content. Reading this bit does not affect its content. Reading the DAI_IRPTL_L.RXLOSSOFLOCK bit clears it.
1 (R/NW)	RXLOCK	Receive Error Lock. The DAI_IRPTL_LS.RXLOCK bit is the shadow of DAI_IRPTL_L.RXLOCK bit and contains the same content. Reading this bit does not affect its content. Reading the DAI_IRPTL_L.RXLOCK bit clears it.
0 (R/NW)	RXVALID	Receive Valid. The DAI_IRPTL_LS.RXVALID bit is the shadow of DAI_IRPTL_L.RXVALID bit and contains the same content. Reading this bit does not affect its content. Reading the DAI_IRPTL_L.RXVALID bit clears it.

Miscellaneous Control Register 0

The DAI_MISC0 register allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

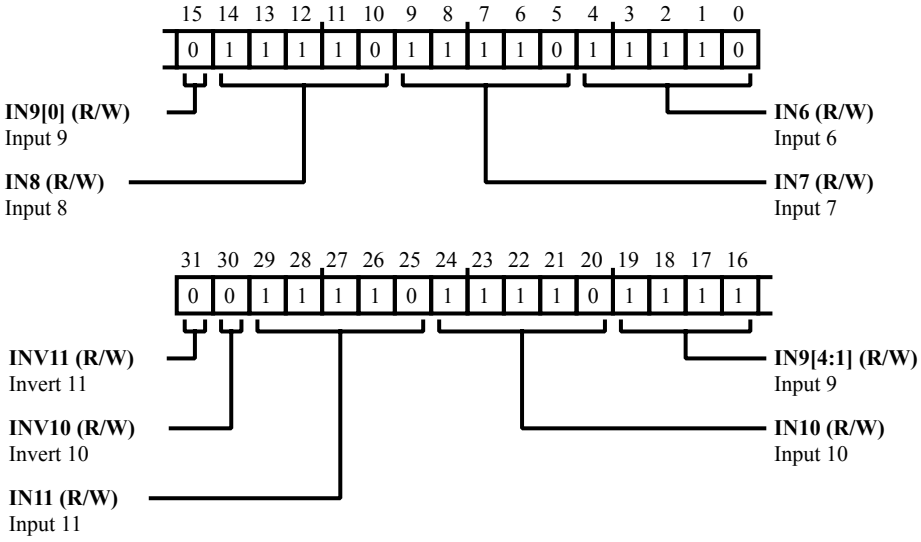


Figure 31-70: DAI_MISC0 Register Diagram

Table 31-80: DAI_MISC0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	INV11	Invert 11. The DAI_MISC0.INV11 bit field inverts miscellaneous A5 input.
30 (R/W)	INV10	Invert 10. The DAI_MISC0.INV10 bit field inverts miscellaneous A4 input.
29:25 (R/W)	IN11	Input 11. The DAI_MISC0.IN11 bit field configures miscellaneous A5 input.
24:20 (R/W)	IN10	Input 10. The DAI_MISC0.IN10 bit field configures miscellaneous A4 input.
19:15 (R/W)	IN9	Input 9. The DAI_MISC0.IN9 bit field configures miscellaneous A3 input/miscellaneous Interrupt 9(DAI interrupt 31).
14:10 (R/W)	IN8	Input 8. The DAI_MISC0.IN8 bit field configures miscellaneous A2 input/miscellaneous Interrupt 8(DAI interrupt 30).

Table 31-80: DAI_MISC0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9:5 (R/W)	IN7	Input 7. The `DAI_MISC0.IN7` bit field configures miscellaneous A1 input/miscellaneous Interrupt 7(DAI interrupt 29).
4:0 (R/W)	IN6	Input 6. The `DAI_MISC0.IN6` bit field configures miscellaneous A0 input/miscellaneous Interrupt 6(DAI interrupt 28).

Miscellaneous Control Register 1

The DAI_MISC1 register allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

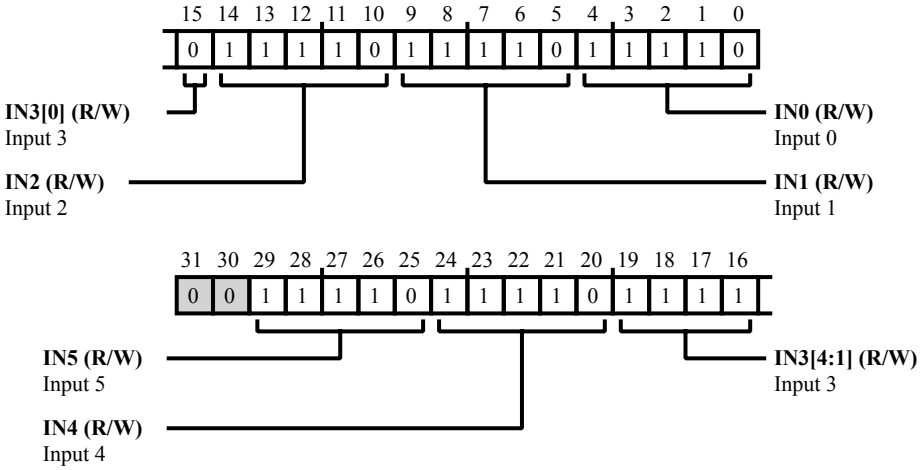


Figure 31-71: DAI_MISC1 Register Diagram

Table 31-81: DAI_MISC1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:25 (R/W)	IN5	Input 5. The DAI_MISC1.IN5 bit field configures miscellaneous Interrupt 5(DAI interrupt 27).
24:20 (R/W)	IN4	Input 4. The DAI_MISC1.IN4 bit field configures miscellaneous Interrupt 4(DAI interrupt 26).
19:15 (R/W)	IN3	Input 3. The DAI_MISC1.IN3 bit field configures miscellaneous Interrupt 3(DAI interrupt 25).
14:10 (R/W)	IN2	Input 2. The DAI_MISC1.IN2 bit field configures miscellaneous Interrupt 2(DAI interrupt 24).
9:5 (R/W)	IN1	Input 1. The DAI_MISC1.IN1 bit field configures miscellaneous Interrupt 1(DAI interrupt 23).

Table 31-81: DAI_MISC1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	IN0	Input 0. The `DAI_MISC1.IN0` bit field configures miscellaneous Interrupt 0(DAI interrupt 22).

Miscellaneous Control Register 1

The DAI_MISC2 register allows programs to route to the DAI interrupt latch, PBEN input routing, or input signal inversion. This register belongs to group E which routes control signals and provides a means of connecting signals between groups.

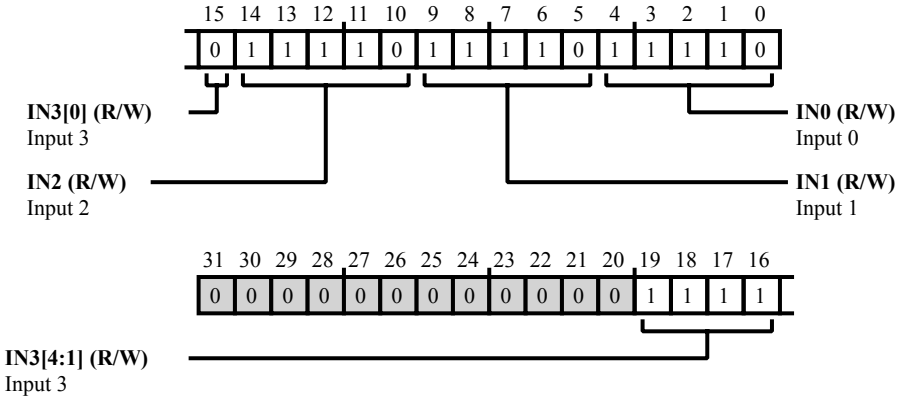


Figure 31-72: DAI_MISC2 Register Diagram

Table 31-82: DAI_MISC2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:15 (R/W)	IN3	Input 3. The DAI_MISC2.IN3 bit field configures PCG-F's hardware completer trigger input.
14:10 (R/W)	IN2	Input 2. The DAI_MISC2.IN2 bit field configures PCG-E's hardware completer trigger input.
9:5 (R/W)	IN1	Input 1. The DAI_MISC2.IN1 bit field configures the PCG-B's hardware completer trigger input.
4:0 (R/W)	IN0	Input 0. The DAI_MISC2.IN0 bit field configures the PCG-A's hardware completer trigger input.

Pin Buffer Enable Register 0

The DAI_PBEN0 register routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

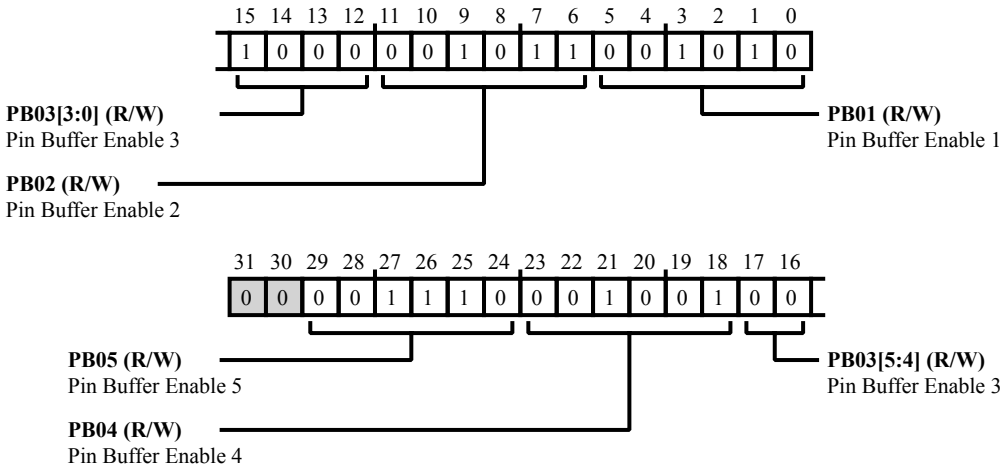


Figure 31-73: DAI_PBEN0 Register Diagram

Table 31-83: DAI_PBEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	PB05	Pin Buffer Enable 5. The DAI_PBEN0.PB05 bit field is the pin buffer enable for DAI port 5.
23:18 (R/W)	PB04	Pin Buffer Enable 4. The DAI_PBEN0.PB04 bit field is the pin buffer enable for DAI port 4.
17:12 (R/W)	PB03	Pin Buffer Enable 3. The DAI_PBEN0.PB03 bit field is the pin buffer enable for DAI port 3.
11:6 (R/W)	PB02	Pin Buffer Enable 2. The DAI_PBEN0.PB02 bit field is the pin buffer enable for DAI port 2.
5:0 (R/W)	PB01	Pin Buffer Enable 1. The DAI_PBEN0.PB01 bit field is the pin buffer enable for DAI port 1.

Pin Buffer Enable Register 1

The DAI_PBEN1 register routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

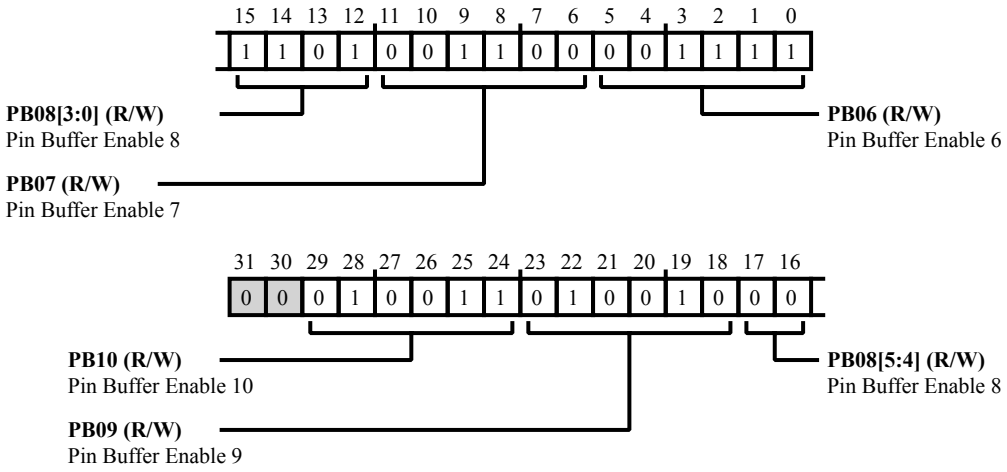


Figure 31-74: DAI_PBEN1 Register Diagram

Table 31-84: DAI_PBEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	PB10	Pin Buffer Enable 10. The DAI_PBEN1.PB10 bit field is the pin buffer enable for DAI port 10.
23:18 (R/W)	PB09	Pin Buffer Enable 9. The DAI_PBEN1.PB09 bit field is the pin buffer enable for DAI port 9.
17:12 (R/W)	PB08	Pin Buffer Enable 8. The DAI_PBEN1.PB08 bit field is the pin buffer enable for DAI port 8.
11:6 (R/W)	PB07	Pin Buffer Enable 7. The DAI_PBEN1.PB07 bit field is the pin buffer enable for DAI port 7.
5:0 (R/W)	PB06	Pin Buffer Enable 6. The DAI_PBEN1.PB06 bit field is the pin buffer enable for DAI port 6.

Pin Buffer Enable Register 2

The DAI_PBEN2 register routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

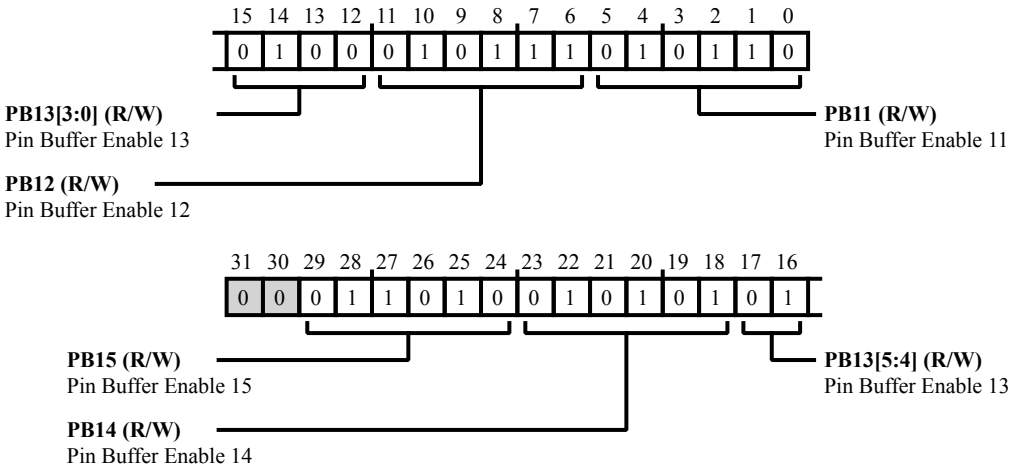


Figure 31-75: DAI_PBEN2 Register Diagram

Table 31-85: DAI_PBEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	PB15	Pin Buffer Enable 15. The DAI_PBEN2.PB15 bit field is the pin buffer enable for DAI port 15.
23:18 (R/W)	PB14	Pin Buffer Enable 14. The DAI_PBEN2.PB14 bit field is the pin buffer enable for DAI port 14.
17:12 (R/W)	PB13	Pin Buffer Enable 13. The DAI_PBEN2.PB13 bit field is the pin buffer enable for DAI port 13.
11:6 (R/W)	PB12	Pin Buffer Enable 12. The DAI_PBEN2.PB12 bit field is the pin buffer enable for DAI port 12.
5:0 (R/W)	PB11	Pin Buffer Enable 11. The DAI_PBEN2.PB11 bit field is the pin buffer enable for DAI port 11.

Pin Buffer Enable Register 3

The DAI_PBEN3 register routes signals to the pin enables, and the value of these signals determines if a DAI pin is used as an output or an input.

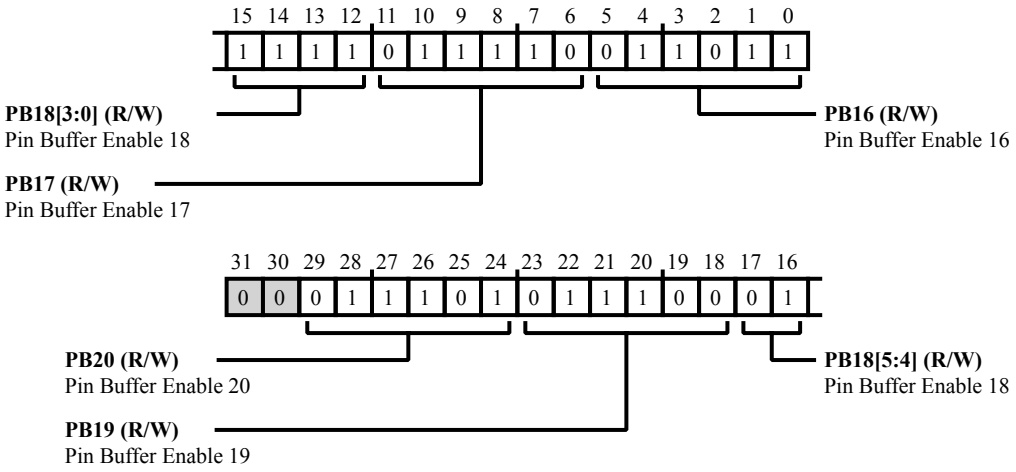


Figure 31-76: DAI_PBEN3 Register Diagram

Table 31-86: DAI_PBEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/W)	PB20	Pin Buffer Enable 20. The DAI_PBEN3.PB20 bit field is the pin buffer enable for DAI port 20.
23:18 (R/W)	PB19	Pin Buffer Enable 19. The DAI_PBEN3.PB19 bit field is the pin buffer enable for DAI port 19.
17:12 (R/W)	PB18	Pin Buffer Enable 18. The DAI_PBEN3.PB18 bit field is the pin buffer enable for DAI port 18.
11:6 (R/W)	PB17	Pin Buffer Enable 17. The DAI_PBEN3.PB17 bit field is the pin buffer enable for DAI port 17.
5:0 (R/W)	PB16	Pin Buffer Enable 16. The DAI_PBEN3.PB16 bit field is the pin buffer enable for DAI port 16.

Pin Buffer Assignment Register 0

The DAI_PIN0 register routes physical pins that are connected to a bonded pad.

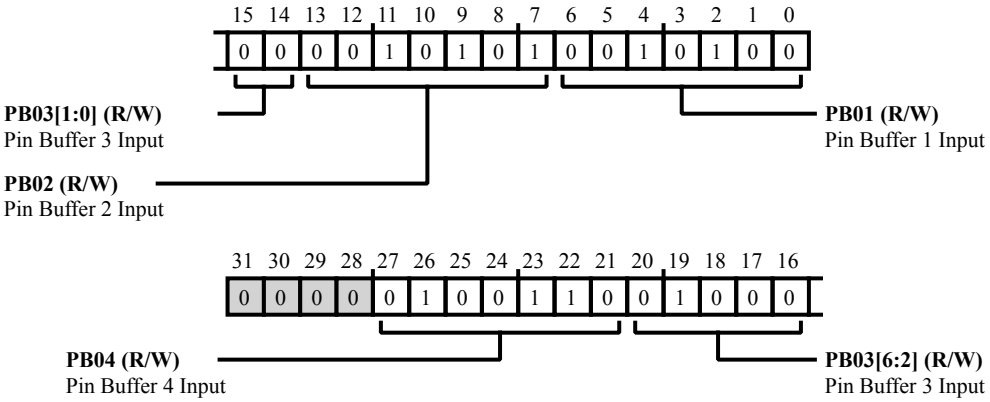


Figure 31-77: DAI_PIN0 Register Diagram

Table 31-87: DAI_PIN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:21 (R/W)	PB04	Pin Buffer 4 Input. DAI_PIN0.PB04 holds the Source signal assignment that will be routed to the DAI_PIN0.PB04 Destination. Refer to the Group D Signals table for Source and Destination mappings.
20:14 (R/W)	PB03	Pin Buffer 3 Input. DAI_PIN0.PB03 holds the Source signal assignment that will be routed to the DAI_PIN0.PB03 Destination. Refer to the Group D Signals table for Source and Destination mappings.
13:7 (R/W)	PB02	Pin Buffer 2 Input. DAI_PIN0.PB02 holds the Source signal assignment that will be routed to the DAI_PIN0.PB02 Destination. Refer to the Group D Signals table for Source and Destination mappings.
6:0 (R/W)	PB01	Pin Buffer 1 Input. DAI_PIN0.PB01 holds the Source signal assignment that will be routed to the DAI_PIN0.PB01 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Pin Buffer Assignment Register 1

The DAI_PIN1 register routes physical pins that are connected to a bonded pad.

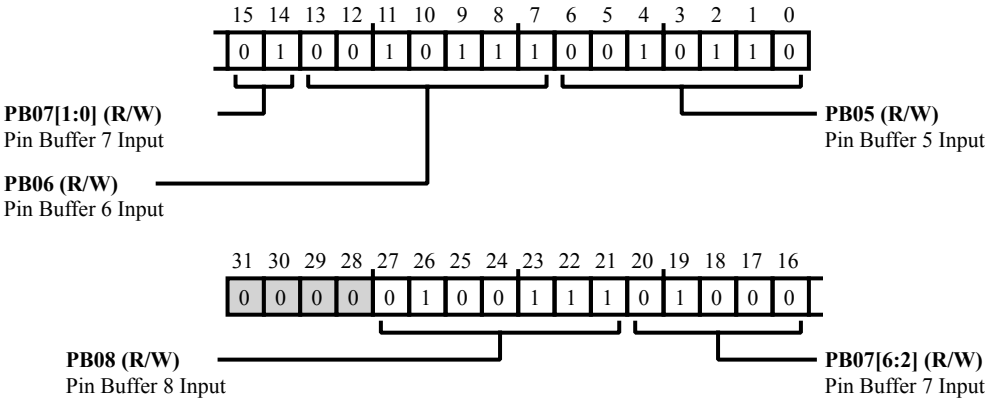


Figure 31-78: DAI_PIN1 Register Diagram

Table 31-88: DAI_PIN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:21 (R/W)	PB08	Pin Buffer 8 Input. DAI_PIN1.PB08 holds the Source signal assignment that will be routed to the DAI_PIN1.PB08 Destination. Refer to the Group D Signals table for Source and Destination mappings.
20:14 (R/W)	PB07	Pin Buffer 7 Input. DAI_PIN1.PB07 holds the Source signal assignment that will be routed to the DAI_PIN1.PB07 Destination. Refer to the Group D Signals table for Source and Destination mappings.
13:7 (R/W)	PB06	Pin Buffer 6 Input. DAI_PIN1.PB06 holds the Source signal assignment that will be routed to the DAI_PIN1.PB06 Destination. Refer to the Group D Signals table for Source and Destination mappings.
6:0 (R/W)	PB05	Pin Buffer 5 Input. DAI_PIN1.PB05 holds the Source signal assignment that will be routed to the DAI_PIN1.PB05 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Pin Buffer Assignment Register 2

The DAI_PIN2 register routes physical pins that are connected to a bonded pad.

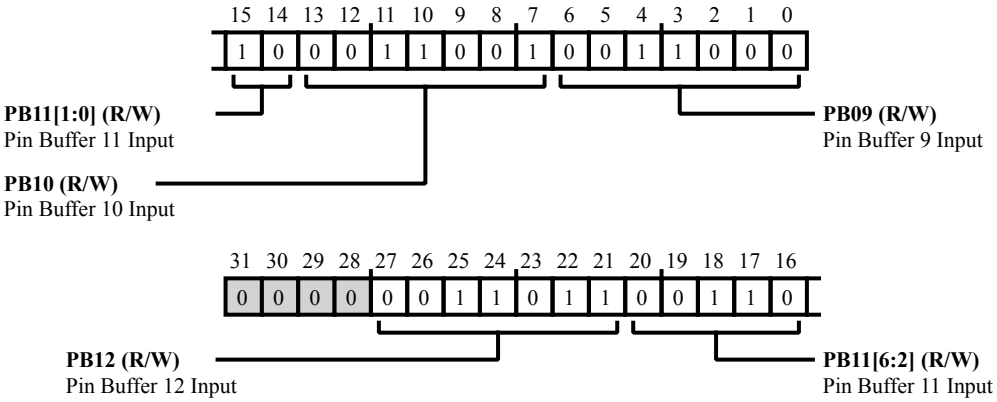


Figure 31-79: DAI_PIN2 Register Diagram

Table 31-89: DAI_PIN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:21 (R/W)	PB12	Pin Buffer 12 Input. DAI_PIN2.PB12 holds the Source signal assignment that will be routed to the DAI_PIN2.PB12 Destination. Refer to the Group D Signals table for Source and Destination mappings.
20:14 (R/W)	PB11	Pin Buffer 11 Input. DAI_PIN2.PB11 holds the Source signal assignment that will be routed to the DAI_PIN2.PB11 Destination. Refer to the Group D Signals table for Source and Destination mappings.
13:7 (R/W)	PB10	Pin Buffer 10 Input. DAI_PIN2.PB10 holds the Source signal assignment that will be routed to the DAI_PIN2.PB10 Destination. Refer to the Group D Signals table for Source and Destination mappings.
6:0 (R/W)	PB09	Pin Buffer 9 Input. DAI_PIN2.PB09 holds the Source signal assignment that will be routed to the DAI_PIN2.PB09 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Pin Buffer Assignment Register 3

The DAI_PIN3 register routes physical pins that are connected to a bonded pad.

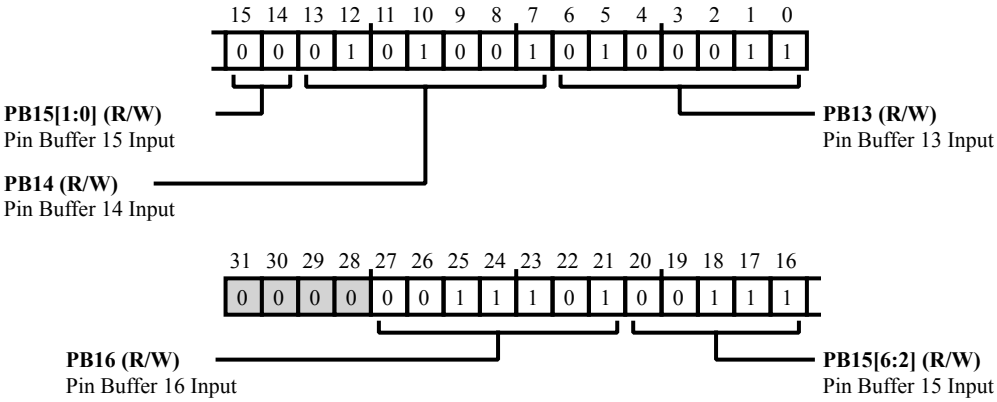


Figure 31-80: DAI_PIN3 Register Diagram

Table 31-90: DAI_PIN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:21 (R/W)	PB16	Pin Buffer 16 Input. The DAI_PIN3.PB16 bit field is the pin buffer 16 input.
20:14 (R/W)	PB15	Pin Buffer 15 Input. DAI_PIN3.PB15 holds the Source signal assignment that will be routed to the DAI_PIN3.PB15 Destination. Refer to the Group D Signals table for Source and Destination mappings.
13:7 (R/W)	PB14	Pin Buffer 14 Input. DAI_PIN3.PB14 holds the Source signal assignment that will be routed to the DAI_PIN3.PB14 Destination. Refer to the Group D Signals table for Source and Destination mappings.
6:0 (R/W)	PB13	Pin Buffer 13 Input. DAI_PIN3.PB13 holds the Source signal assignment that will be routed to the DAI_PIN3.PB13 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Pin Buffer Assignment Register 4

The DAI_PIN4 register routes physical pins that are connected to a bonded pad.

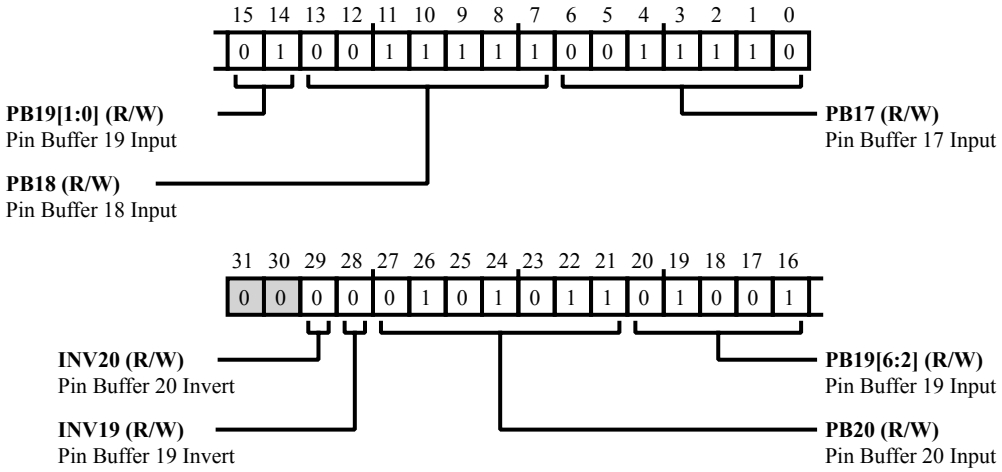


Figure 31-81: DAI_PIN4 Register Diagram

Table 31-91: DAI_PIN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29 (R/W)	INV20	Pin Buffer 20 Invert. DAI_PIN4.INV20 holds the Source signal assignment that will be routed to the DAI_PIN4.INV20 Destination. Refer to the Group D Signals table for Source and Destination mappings.
28 (R/W)	INV19	Pin Buffer 19 Invert. DAI_PIN4.INV19 holds the Source signal assignment that will be routed to the DAI_PIN4.INV19 Destination. Refer to the Group D Signals table for Source and Destination mappings.
27:21 (R/W)	PB20	Pin Buffer 20 Input. DAI_PIN4.PB20 holds the Source signal assignment that will be routed to the DAI_PIN4.PB20 Destination. Refer to the Group D Signals table for Source and Destination mappings.
20:14 (R/W)	PB19	Pin Buffer 19 Input. DAI_PIN4.PB19 holds the Source signal assignment that will be routed to the DAI_PIN4.PB19 Destination. Refer to the Group D Signals table for Source and Destination mappings.
13:7 (R/W)	PB18	Pin Buffer 18 Input. DAI_PIN4.PB18 holds the Source signal assignment that will be routed to the DAI_PIN4.PB18 Destination. Refer to the Group D Signals table for Source and Destination mappings.

Table 31-91: DAI_PIN4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	PB17	Pin Buffer 17 Input. `DAI_PIN4.PB17` holds the Source signal assignment that will be routed to the `DAI_PIN4.PB17` Destination. Refer to the Group D Signals table for Source and Destination mappings.

Pin Status Register

The DAI_PIN_STAT register bits indicate the status (signal high or low) for each pin. The individual bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

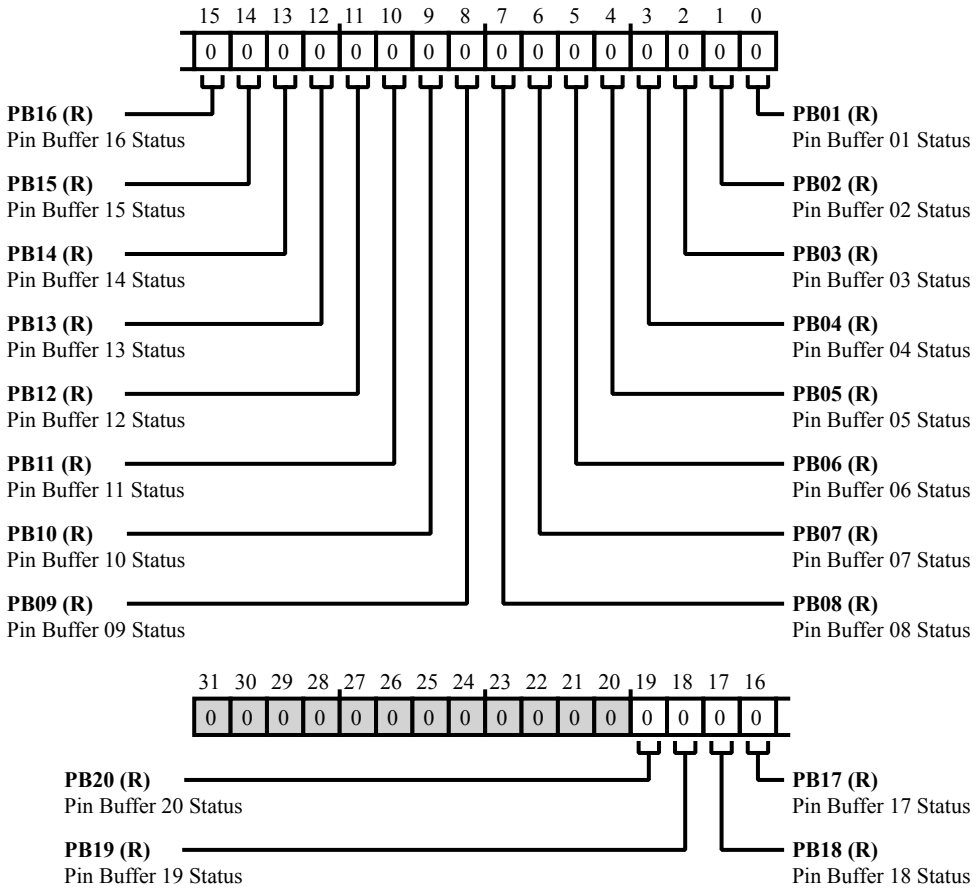


Figure 31-82: DAI_PIN_STAT Register Diagram

Table 31-92: DAI_PIN_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19 (R/NW)	PB20	Pin Buffer 20 Status. The DAI_PIN_STAT.PB20 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
18 (R/NW)	PB19	Pin Buffer 19 Status. The DAI_PIN_STAT.PB19 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

Table 31-92: DAI_PIN_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/NW)	PB18	Pin Buffer 18 Status. The DAI_PIN_STAT.PB18 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
16 (R/NW)	PB17	Pin Buffer 17 Status. The DAI_PIN_STAT.PB17 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
15 (R/NW)	PB16	Pin Buffer 16 Status. The DAI_PIN_STAT.PB16 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
14 (R/NW)	PB15	Pin Buffer 15 Status. The DAI_PIN_STAT.PB15 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
13 (R/NW)	PB14	Pin Buffer 14 Status. The DAI_PIN_STAT.PB14 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
12 (R/NW)	PB13	Pin Buffer 13 Status. The DAI_PIN_STAT.PB13 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
11 (R/NW)	PB12	Pin Buffer 12 Status. The DAI_PIN_STAT.PB12 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
10 (R/NW)	PB11	Pin Buffer 11 Status. The DAI_PIN_STAT.PB11 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
9 (R/NW)	PB10	Pin Buffer 10 Status. The DAI_PIN_STAT.PB10 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
8 (R/NW)	PB09	Pin Buffer 09 Status. The DAI_PIN_STAT.PB09 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
7 (R/NW)	PB08	Pin Buffer 08 Status. The DAI_PIN_STAT.PB08 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

Table 31-92: DAI_PIN_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
6 (R/NW)	PB07	Pin Buffer 07 Status. The DAI_PIN_STAT.PB07 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
5 (R/NW)	PB06	Pin Buffer 06 Status. The DAI_PIN_STAT.PB06 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
4 (R/NW)	PB05	Pin Buffer 05 Status. The DAI_PIN_STAT.PB05 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
3 (R/NW)	PB04	Pin Buffer 04 Status. The DAI_PIN_STAT.PB04 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
2 (R/NW)	PB03	Pin Buffer 03 Status. The DAI_PIN_STAT.PB03 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
1 (R/NW)	PB02	Pin Buffer 02 Status. The DAI_PIN_STAT.PB02 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.
0 (R/NW)	PB01	Pin Buffer 01 Status. The DAI_PIN_STAT.PB01 bit reads 1 if the signal to this pin is high and reads 0 if signal to this pin is low.

32 Serial Port (SPORT)

The programmable serial ports (SPORTs) support various protocols for serial data communication and provide a glueless hardware interface to many industry-standard data converters and codecs. They have high data rates and dual half-duplex datapaths and are ideal for establishing a direct serial connection among two or more processors in a multiprocessor system, as many processors provide compatible serial interfaces.

The SPORT top module consists of two half SPORTs with identical functionality and programming requirements. Each half SPORT can be independently configured as either a transmitter or receiver and can be coupled with the other half SPORT within the same SPORT top module. Further, each half SPORT provides two synchronous half-duplex data lines to double the total supported throughput. As such, a single SPORT top module can be used to provide up to four unidirctional or up to two full-duplex data streams. Further channels are possible as well, but utilization of multiple SPORT top modules is required, thus requiring external connections to provide a common time base.

Features

An individual SPORT top module consists of two independently configurable SPORT halves with identical functionality. These SPORT halves offer the following features:

- Up to two bidirectional data lines - each half SPORT supports up to two transmit or receive channels, thus allowing two unidirectional streams into or out of each half SPORT and providing greater flexibility for serial communications. If full-duplex functionality is desired, two SPORT halves can be combined to enable dual-stream bidirectional communication.
- Six operating modes:
    1. Standard DSP serial mode
    2. I^2S mode
    3. Left-Justified mode
    4. Right-Justified mode
    5. Multichannel (TDM) mode
    6. Packed mode

- Supports internally or externally generated clock.
- Support for both even and odd SCLK0 to SPORT clock (SPORT_CLK) ratios. If both data lines of a half SPORT are active, the maximum throughput is 2 x SPORT_CLK bps.
- Configurable rising or falling edge of the SPORT_CLK for driving and sampling data and frame syncs.
- Gated clock mode support for internally or externally generated clocks in DSP serial mode and stereo modes (left-justified and I^2S mode).
- Supports frameless operation.
- Supports internally or externally generated frame sync signals.
- Programmable frame sync polarity.
- Programmable frame sync timing (synchronous to data or 1 SPORT clock in advance of it).
- Detection of prematurely received external frame syncs (with optional interrupt request generation).
- Programmable level-/edge-sensitivity for external frame syncs.
- Programmable (4–32-bit) data length, either in most significant bit (MSB) first or in least significant bit (LSB) first format, with optional sign-extension on received data.
- Optional 16-bit to 32-bit word packing (as receiver) and 32-bit to 16-bit word unpacking (as transmitter).
- Support for A-law and μ-law compression/decompression hardware companding, according to the G.711 specification, on transmitted/received words in all operating modes.
- Transmit underrun and receive overflow detection (with optional interrupt request generation).
- TDM mode transfers data on 128 contiguous channels from a stream of up to 1024 total channels (useful for H.100/H.110 and other telephony interfaces as a network communication scheme for multiple processors).
- Performs interrupt-driven, single word core transfers to and from on-chip or off-chip memory.
- Dedicated DMA channel for each SPORT half supporting autobuffer (for a repeated, identical range of transfers) and numerous descriptor-based (individual or repeated ranges of transfers with differing DMA parameters) modes.
- Generator and receiver trigger functionality.
- Unique transfer finish interrupt (TFI) signaling when the last transmit word is fully out of the transmit shift register.
- Multiplexer to internally connect critical timing signals between SPORT halves.
- Support for Global Sport Enable functionality, where enabling multiple SPORTs need to be synchronized so that all of them start and/or end at the same time.
- Support for late reception of SPORT data and frame sync.

Signal Descriptions

Each half SPORT module has five dedicated signals, as described in the ***SPORT Signal Descriptions*** table. The actual pin name varies with different SPORT halves. Individual SPORT halves do not share any of its signals across the pair that comprises the SPORT top module; however, it is possible to connect the clock and frame sync signals between the SPORT half pair, as explained in the Multiplexer Logic section.

All of the SPORT signals are multiplexed on the PORT pins, possibly sharing functionality with other peripherals on the device. By default, the PORT pins are in GPIO mode and must be reconfigured for SPORT functionality by setting the appropriate bits in the PORT_FER and PORT_MUX registers. Consult the processor data sheet for details regarding which ports the SPORT signals are available and to ensure to configure the PORT_MUX register before the PORT_FER register.

Table 32-1: SPORT Signal Descriptions

Internal Node	Direction	Description
SPORTx_CLK	I/O	Transmit or receive serial clock. Data and frame syncs are driven or sampled on this clock's edges. This signal can be either internally or externally generated.
SPORTx_FS	I/O	Transmit or receive frame sync. The frame sync pulse initiates shifting of serial data. This signal is either internally or externally generated.
SPORTx_D0	I/O	Primary transmit or receive data channel. This signal can be configured as an output to transmit serial data or as an input to receive serial data.
SPORTx_D1	I/O	Secondary transmit or receive data channel. This signal can be configured as an output to transmit serial data or as an input to receive serial data.
SPORTx_TDV	O	Multichannel transmit data valid. This signal is only active in multichannel transmit mode and is asserted during enabled slots, as defined by the channel selection registers (SPORT_CS0_A through SPORT_CS3_B).

The data channel signals are transmit signals when the serial port is configured in transmit mode (SPORT_CTL_A.SPTRAN = 1). They are receive signals when the serial port is configured in receive mode (SPORT_CTL_A.SPTRAN = 0). The following sections further describe the SPORT signals.

NOTE: These sections explicitly refer to the registers associated with half SPORT A, but the same concepts also apply to half SPORT B.

Serial Clock

The serial port clock (SPORT_ACLK) is either a receive serial clock or a transmit serial clock, depending on the transfer direction (SPORT_CTL_A.SPTRAN), governing when the data bits are serially shifted into or out of the SPORT and when the frame sync signal is driven (in internal frame sync mode) or sampled (in external frame sync mode). It can be internally generated from the processor's system clock (SCLK) or externally provided. If the half SPORT is configured in internal clock mode (SPORT_CTL_A.ICLK= 1), then the SPORT_DIV_A.CLKDIV field specifies the divisor applied to SCLK to generate the SPORT clock. As it is a 16-bit divisor, a wide range of serial clock rates is possible. Use the following equation to calculate the serial clock frequency:

SPORT_ACLK = [SCLK ÷ (SPORT_DIV_A.CLKDIV + 1)]

From this, the following equation can be used to determine the value of SPORT_DIV_A.CLKDIV, given the SCLK frequency and the desired frequency of the SPORT clock:

SPORT_DIV_A.CLKDIV= [(SCLK ÷ SPORT_ACLK) - 1]

The half SPORT also supports a 1:1 SPORT_ACLK to ratio (per the equations above, program the clock divisor field to zero). In this case, the resulting SPORT clock frequency is equal to SCLK.

NOTE: Be careful not to exceed the maximum SPORT_ACLK frequency specified in the processor data sheet.

In certain operating modes, the SPORT can be configured to generate a gated clock, which is active only during valid data. In some applications, a SPORT uses it to generate a general-purpose clock in the system. In this case, enable the SPORT with the appropriate SPORT_DIV_A.CLKDIV divisor field in internal clock mode.

If a SPORT is configured in external clock mode (SPORT_CTL_A.ICLK= 0), the serial clock is an input signal, thus making the SPORT operate in target mode. In this mode, the SPORT_DIV_A.CLKDIV is irrelevant and is ignored. The optional loopback capability provided by the internal SPORT multiplexer (SPMUX) block allows the target SPORT to use the serial clock from the neighboring SPORT half in the same SPORT top module.

An externally-supplied serial clock does not need to be synchronous with the processor clocks. Further, the external clock can be a gated clock, but it must comply with the requirements described in the Gated Clock Mode section.

Refer to the product data sheet for exact AC timing specifications.

Frame Sync

The SPORT frame sync (SPORT_AFS) signal is either a receive frame sync or a transmit frame sync, depending on the transfer direction (SPORT_CTL_A.SPTRAN), which is used to determine the start of a new word or frame. When this signal goes active, the serial port starts serially shifting data into or out of the SPORT. The frame sync signal can be internally generated based on its serial clock (SPORT_ACLK) or externally provided, as configured by the SPORT_CTL_A.IFS bit.

If the half SPORT is configured to generate frame syncs (SPORT_CTL_A.IFS= 1), then the SPORT_DIV_A.FSDIV field specifies the divisor used to generate the periodic SPORT_AFS signal from the SPORT clock. As this is a 16-bit divisor, a wide range of frame sync rates to initiate periodic transfers is possible. Whether the serial clock is internally or externally generated, this divisor is a count of SPORT clock cycles between frame sync pulses, the formula for which is:

Number of serial clocks between frame syncs = (SPORT_DIV_A.FSDIV + 1)

From this, the following equation can be used to determine the value of SPORT_DIV_A.FSDIV, given the serial clock frequency and the desired frame sync frequency:

SPORT_DIV_A.FSDIV = [(SPORT_ACLK ÷ SPORT_AFS) – 1]

The frame sync is continuously active when SPORT_DIV_A.FSDIV= 0. The value of SPORT_DIV_A.FSDIV cannot be less than the serial word length minus one (the value of the SPORT_CTL_A.SLEN bit field). Failure to

adhere to this guideline can cause an external device to abort the current operation or cause other unpredictable results.

NOTE: After enabling the SPORT, the first internal frame sync appears after a delay of SPORT_DIV_A.FSDIV + 3 SPORT clock cycles.

If a SPORT is configured for external frame syncs (SPORT_CTL_A.IFS = 0), then SPORT_AFS is an input signal and the SPORT_DIV_A.FSDIV field of the SPORT_DIV_A register is irrelevant and ignored. By default, this external signal is level-sensitive, but it can be configured as an edge-sensitive signal by setting the SPORT_CTL_A.FSED bit. The frame sync is expected to be synchronous with the serial clock. If not, it must meet the timing requirements that appear in the data sheet.

The SPORT can be used as a counter for dividing an external clock to generate periodic pulses or periodic interrupts. To do so, enable the SPORT with the appropriate SPORT_DIV_A.FSDIV divisor field with the SPORT configured for an external clock and internal data-independent frame syncs.

In some of the operating modes, the SPORT can be programmed to treat the frame sync signal as an optional signal by clearing the SPORT_CTL_A.FSR bit. Even with this bit cleared, the SPORT requires a single frame sync assertion to start the continuous transfers, after which it is ignored (for externally supplied frame syncs) or not generated (for internally-generated frame syncs). Characteristics of the frame sync depend on the settings in the SPORT control registers and the operating mode of the SPORT. For more information, refer to the SPORT_CTL_A register.

Data Signals

Each half SPORT has two bidirectional data lines known as the primary (SPORT_AD0) and secondary (SPORT_AD1) data channels. Both data lines can be configured as either transmitters or receivers using the SPORT_CTL_A.SPTRAN bit, thus permitting dual unidirectional data streams to increase the data throughput of the SPORT.

NOTE: Configuring one transmit data channel and one receive data channel on a single half SPORT is not supported.

The primary and secondary data lines can be individually enabled or disabled using the SPORT_CTL_A.SPENPRI and the SPORT_CTL_A.SPENSEC bits, respectively. However, if using both, enable or disable them concurrently. These data lines operate in a synchronous manner (sharing a clock and frame sync) but have separate datapaths with unique data buffers, shift registers and optional companding logic. All of the SPORT control settings are common for both channels, but the single DMA channel per half SPORT serves both the primary and secondary data channels.

When a SPORT is configured in multichannel transmit mode, the data pins three-state during inactive channel slots, thus allowing multiple transmitters to operate on the same bus with different active channels.

See the Architectural Concepts section for more details about data transfer operation.

Transmit Data Valid Signal

The transmit data valid (SPORT_ATDV) signal is available only in transmit multichannel modes (including packed mode). It is driven active during enabled multichannel slots, and it is driven inactive during the disabled channels. In other words, the SPORT_ATDV signal is active when data is being driven to the data pins and inactive when the data pins are being three-stated. As such, the SPORT_ATDV signal can serve as an output-enable signal for the data transmit pin.

SRU Programming

The SPORT uses the SRU (signal routing unit) to connect the SPORT data, serial clock, frame sync, and external sync (if external synchronization is required). Inputs also must be routed through the SRU. Program the corresponding SRU registers to connect the outputs to the required destinations. For details of the routing, see DAI Routing Capabilities in the *Digital Audio Interface (DAI)* chapter.

SRU SPORT Receive Controller

If the SPORT is receiver, it must feed its controller output clock back to its input clock. This is required to trigger the SPORT's state machine. Using SPORT 0A as an example receiver, programs should route SPT0_ACLK_O to SPT0_ACLK_I. This is not required if the SPORT is operating as a transmitter in controller mode.

Functional Description

The following sections provide general information about the functionality of the processor's serial ports:

- Architectural Concepts
- Data Types and Companding
- Transmit Path
- Receive Path

ADSP-SC59x SPORT Register List

The Serial Port (SPORT) controller, with its range of clock and frame synchronization options, supports a variety of serial communication protocols and provides a glueless hardware interface to many industry-standard data converters and CODECs. Each SPORT has two independent halves (A and B), and each half contains two channels (primary and secondary). A set of registers governs SPORT operations. For more information on SPORT functionality, see the SPORT register descriptions.

Table 32-2: ADSP-SC59x SPORT Register List

Name	Description
SPORT_CS0_A	Half SPORT 'A' Multichannel 0-31 Select Register
SPORT_CS0_B	Half SPORT 'B' Multichannel 0-31 Select Register

Table 32-2: ADSP-SC59x SPORT Register List (Continued)

Name	Description
SPORT_CS1_A	Half SPORT 'A' Multichannel 32-63 Select Register
SPORT_CS1_B	Half SPORT 'B' Multichannel 32-63 Select Register
SPORT_CS2_A	Half SPORT 'A' Multichannel 64-95 Select Register
SPORT_CS2_B	Half SPORT 'B' Multichannel 64-95 Select Register
SPORT_CS3_A	Half SPORT 'A' Multichannel 96-127 Select Register
SPORT_CS3_B	Half SPORT 'B' Multichannel 96-127 Select Register
SPORT_CTL2_A	Half SPORT 'A' Control 2 Register
SPORT_CTL2_B	Half SPORT 'B' Control 2 Register
SPORT_CTL_A	Half SPORT 'A' Control Register
SPORT_CTL_B	Half SPORT 'B' Control Register
SPORT_DIV_A	Half SPORT 'A' Divisor Register
SPORT_DIV_B	Half SPORT 'B' Divisor Register
SPORT_ERR_A	Half SPORT 'A' Error Register
SPORT_ERR_B	Half SPORT 'B' Error Register
SPORT_MCTL_A	Half SPORT 'A' Multichannel Control Register
SPORT_MCTL_B	Half SPORT 'B' Multichannel Control Register
SPORT_MSTAT_A	Half SPORT 'A' Multichannel Status Register
SPORT_MSTAT_B	Half SPORT 'B' Multichannel Status Register
SPORT_RXPRI_A	Half SPORT 'A' Rx Buffer (Primary) Register
SPORT_RXPRI_B	Half SPORT 'B' Rx Buffer (Primary) Register
SPORT_RXSEC_A	Half SPORT 'A' Rx Buffer (Secondary) Register
SPORT_RXSEC_B	Half SPORT 'B' Rx Buffer (Secondary) Register
SPORT_TXPRI_A	Half SPORT 'A' Tx Buffer (Primary) Register
SPORT_TXPRI_B	Half SPORT 'B' Tx Buffer (Primary) Register
SPORT_TXSEC_A	Half SPORT 'A' Tx Buffer (Secondary) Register
SPORT_TXSEC_B	Half SPORT 'B' Tx Buffer (Secondary) Register

ADSP-SC59x SPORT Interrupt List

Table 32-3: ADSP-SC59x SPORT Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
81	SPORT0_A_DMA	SPORT0 Channel A DMA	Level	0
82	SPORT0_A_STAT	SPORT0 Channel A Status	Level	
83	SPORT0_B_DMA	SPORT0 Channel B DMA	Level	1
84	SPORT0_B_STAT	SPORT0 Channel B Status	Level	
85	SPORT1_A_DMA	SPORT1 Channel A DMA	Level	2
86	SPORT1_A_STAT	SPORT1 Channel A Status	Level	
87	SPORT1_B_DMA	SPORT1 Channel B DMA	Level	3
88	SPORT1_B_STAT	SPORT1 Channel B Status	Level	
89	SPORT2_A_DMA	SPORT2 Channel A DMA	Level	4
90	SPORT2_A_STAT	SPORT2 Channel A Status	Level	
91	SPORT2_B_DMA	SPORT2 Channel B DMA	Level	5
92	SPORT2_B_STAT	SPORT2 Channel B Status	Level	
93	SPORT3_A_DMA	SPORT3 Channel A DMA	Level	6
94	SPORT3_A_STAT	SPORT3 Channel A Status	Level	
95	SPORT3_B_DMA	SPORT3 Channel B DMA	Level	7
96	SPORT3_B_STAT	SPORT3 Channel B Status	Level	
97	SPORT4_A_DMA	SPORT4 Channel A DMA	Level	10
98	SPORT4_A_STAT	SPORT4 Channel A Status	Level	
99	SPORT4_B_DMA	SPORT4 Channel B DMA	Level	11
100	SPORT4_B_STAT	SPORT4 Channel B Status	Level	
101	SPORT5_A_DMA	SPORT5 Channel A DMA	Level	12
102	SPORT5_A_STAT	SPORT5 Channel A Status	Level	
103	SPORT5_B_DMA	SPORT5 Channel B DMA	Level	13
104	SPORT5_B_STAT	SPORT5 Channel B Status	Level	
105	SPORT6_A_DMA	SPORT6 Channel A DMA	Level	14
106	SPORT6_A_STAT	SPORT6 Channel A Status	Level	
107	SPORT6_B_DMA	SPORT6 Channel B DMA	Level	15
108	SPORT6_B_STAT	SPORT6 Channel B Status	Level	
109	SPORT7_A_DMA	SPORT7 Channel A DMA	Level	16
110	SPORT7_A_STAT	SPORT7 Channel A Status	Level	
111	SPORT7_B_DMA	SPORT7 Channel B DMA	Level	17

Table 32-3: ADSP-SC59x SPORT Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
112	SPORT7_B_STAT	SPORT7 Channel B Status	Level	
297	SPORT0_A_DMA_ERR	SPORT0 Channel A DMA Error	Level	
298	SPORT0_B_DMA_ERR	SPORT0 Channel B DMA Error	Level	
299	SPORT1_A_DMA_ERR	SPORT1 Channel A DMA Error	Level	
300	SPORT1_B_DMA_ERR	SPORT1 Channel B DMA Error	Level	
301	SPORT2_A_DMA_ERR	SPORT2 Channel A DMA Error	Level	
302	SPORT2_B_DMA_ERR	SPORT2 Channel B DMA Error	Level	
303	SPORT3_A_DMA_ERR	SPORT3 Channel A DMA Error	Level	
304	SPORT3_B_DMA_ERR	SPORT3 Channel B DMA Error	Level	
305	SPORT4_A_DMA_ERR	SPORT4 Channel A DMA Error	Level	
306	SPORT4_B_DMA_ERR	SPORT4 Channel B DMA Error	Level	
307	SPORT5_A_DMA_ERR	SPORT5 Channel A DMA Error	Level	
308	SPORT5_B_DMA_ERR	SPORT5 Channel B DMA Error	Level	
309	SPORT6_A_DMA_ERR	SPORT6 Channel A DMA Error	Level	
310	SPORT6_B_DMA_ERR	SPORT6 Channel B DMA Error	Level	
311	SPORT7_A_DMA_ERR	SPORT7 Channel A DMA Error	Level	
312	SPORT7_B_DMA_ERR	SPORT7 Channel B DMA Error	Level	

ADSP-SC59x SPORT Trigger List

Table 32-4: ADSP-SC59x SPORT Trigger List Generators

Trigger ID	Name	Description	Sensitivity
85	SPORT0_A_DMA	SPORT0 Channel A DMA	Edge
86	SPORT0_B_DMA	SPORT0 Channel B DMA	Edge
87	SPORT1_A_DMA	SPORT1 Channel A DMA	Edge
88	SPORT1_B_DMA	SPORT1 Channel B DMA	Edge
89	SPORT2_A_DMA	SPORT2 Channel A DMA	Edge
90	SPORT2_B_DMA	SPORT2 Channel B DMA	Edge
91	SPORT3_A_DMA	SPORT3 Channel A DMA	Edge
92	SPORT3_B_DMA	SPORT3 Channel B DMA	Edge
93	SPORT4_A_DMA	SPORT4 Channel A DMA	Edge

Table 32-4: ADSP-SC59x SPORT Trigger List Generators (Continued)

Trigger ID	Name	Description	Sensitivity
94	SPORT4_B_DMA	SPORT4 Channel B DMA	Edge
95	SPORT5_A_DMA	SPORT5 Channel A DMA	Edge
96	SPORT5_B_DMA	SPORT5 Channel B DMA	Edge
97	SPORT6_A_DMA	SPORT6 Channel A DMA	Edge
98	SPORT6_B_DMA	SPORT6 Channel B DMA	Edge
99	SPORT7_A_DMA	SPORT7 Channel A DMA	Edge
100	SPORT7_B_DMA	SPORT7 Channel B DMA	Edge

Table 32-5: ADSP-SC59x SPORT Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
60	SPORT0_A_DMA	SPORT0 Channel A DMA	Pulse
61	SPORT0_B_DMA	SPORT0 Channel B DMA	Pulse
62	SPORT1_A_DMA	SPORT1 Channel A DMA	Pulse
63	SPORT1_B_DMA	SPORT1 Channel B DMA	Pulse
64	SPORT2_A_DMA	SPORT2 Channel A DMA	Pulse
65	SPORT2_B_DMA	SPORT2 Channel B DMA	Pulse
66	SPORT3_A_DMA	SPORT3 Channel A DMA	Pulse
67	SPORT3_B_DMA	SPORT3 Channel B DMA	Pulse
68	SPORT4_A_DMA	SPORT4 Channel A DMA	Pulse
69	SPORT4_B_DMA	SPORT4 Channel B DMA	Pulse
70	SPORT5_A_DMA	SPORT5 Channel A DMA	Pulse
71	SPORT5_B_DMA	SPORT5 Channel B DMA	Pulse
72	SPORT6_A_DMA	SPORT6 Channel A DMA	Pulse
73	SPORT6_B_DMA	SPORT6 Channel B DMA	Pulse
74	SPORT7_A_DMA	SPORT7 Channel A DMA	Pulse
75	SPORT7_B_DMA	SPORT7 Channel B DMA	Pulse

ADSP-SC59x SPORT DMA Channel List

Table 32-6: ADSP-SC59x SPORT DMA Channel List

DMA ID	DMA Channel Name	Description
DMA0	SPORT0_A_DMA	SPORT0 Channel A DMA

Table 32-6: ADSP-SC59x SPORT DMA Channel List (Continued)

DMA ID	DMA Channel Name	Description
DMA1	SPORT0_B_DMA	SPORT0 Channel B DMA
DMA2	SPORT1_A_DMA	SPORT1 Channel A DMA
DMA3	SPORT1_B_DMA	SPORT1 Channel B DMA
DMA4	SPORT2_A_DMA	SPORT2 Channel A DMA
DMA5	SPORT2_B_DMA	SPORT2 Channel B DMA
DMA6	SPORT3_A_DMA	SPORT3 Channel A DMA
DMA7	SPORT3_B_DMA	SPORT3 Channel B DMA
DMA10	SPORT4_A_DMA	SPORT4 Channel A DMA
DMA11	SPORT4_B_DMA	SPORT4 Channel B DMA
DMA12	SPORT5_A_DMA	SPORT5 Channel A DMA
DMA13	SPORT5_B_DMA	SPORT5 Channel B DMA
DMA14	SPORT6_A_DMA	SPORT6 Channel A DMA
DMA15	SPORT6_B_DMA	SPORT6 Channel B DMA
DMA16	SPORT7_A_DMA	SPORT7 Channel A DMA
DMA17	SPORT7_B_DMA	SPORT7 Channel B DMA

Block Diagram

Each SPORT top module consists of two separate blocks, known as half SPORTs (HSPORT) A and B, each with identical functionality and programming models. The ***Half Serial Port Block Diagram*** shows a detailed block diagram of a half SPORT.

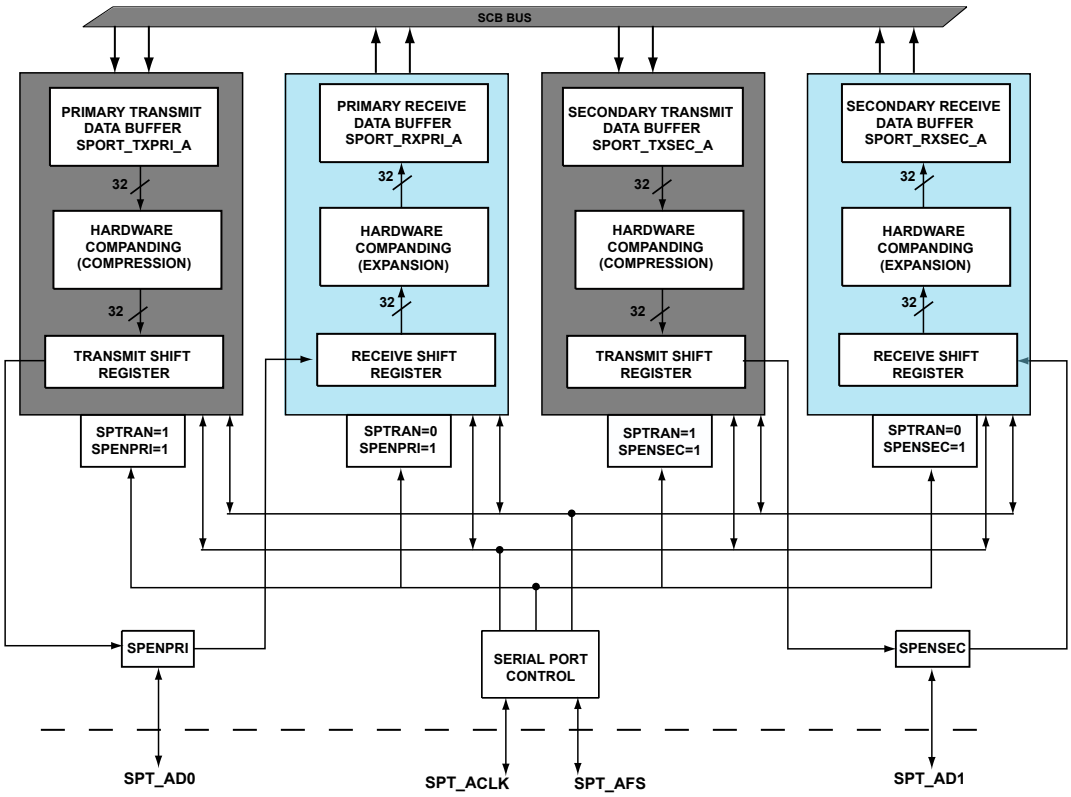


Figure 32-1: Half Serial Port Block Diagram

Architectural Concepts

Each half SPORT (HSPORT) block has its own set of control registers and data buffers, grouped per SPORT module. The HSPORT A and B blocks can be independently configured as either a transmitter or a receiver, with the option to be coupled together internally within the single SPORT top module. Each HSPORT also supports two synchronous bidirectional datapaths, referred to as the primary (`D0`) and secondary (`D1`) data lines, as shown in the *Top-Level SPORT Diagram* figure.

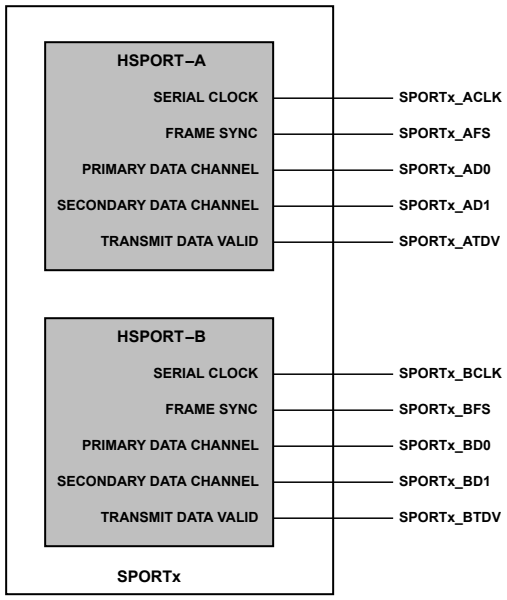


Figure 32-2: Top-Level SPORT Diagram

The SPORT_CTL_A.SPTRAN bit controls the direction for both datapaths of the HSPORT. Depending on whether the HSPORT is a transmitter or a receiver, the pair of data signals respectfully transmit or receive data bits synchronously. The dual data signals of each HSPORT cannot transmit and receive the data simultaneously in support of full-duplex operation, however, two HSPORTs can be combined to achieve this.

Serial communications are synchronized to the serial clock signal, where a valid clock pulse must accompany each data bit. Each HSPORT can take its clock from an external source or internally generate it from the processor's system clock using the SPORT_DIV_A.CLKDIV clock divisor bit field. Both primary and secondary data channels shift data based on the SPORT_CLK rate and the clock polarity defined by the SPORT_CTL_A.CKRE bit.

In addition to the serial clock signal, a frame synchronization signal is used to signify the beginning of an individual data word or a multichannel data stream (block of words). Each SPORT can take the frame sync signal from an external source or generate it (SPORT_FS), depending on the SPORT_CTL_A.IFS bit. An internally generated frame sync is derived from the SPORT clock using the SPORT_DIV_A.FSDIV divisor field. Both primary and secondary datapaths start shifting data either synchronous to or one serial clock in advance of detecting/generating a valid frame sync signal, as determined by the SPORT_CTL_A.LAFS bit. Various communication protocols for serial data can be emulated according to the frame sync format, and all frame sync options are available whether the signal is generated internally or externally.

NOTE: These SPORTs are not UARTs and cannot communicate with an RS-232 device or any other asynchronous communications protocol.

Multiplexer Logic

The SPORT multiplexing block (SPMUX) is situated between the SPORT hardware block and the processor's pin multiplexing logic. It allows the flexibility to route and share the clock and frame sync signals between the HSPORT A and B halves within each SPORT top module, which can double the data throughput (if both SPORT halves are transmitters or both are receivers) or provide full-duplex capabilities (if one HSPORT is a receiver and the other a

transmitter) without the need to allocate pins for the peripheral or make physical connections outside the processor. The SPORT_CTL2_A register is used to configure this loopback feature.

NOTE: Throughout this section, HSPORT A is used as a reference, but all concepts also apply to HSPORT B.

The multiplexing depends on the configuration of the SPORT_CTL_A.IFS and SPORT_CTL_A.ICLK bits, and the SPORT_CTL2_A.CKMUXSEL and SPORT_CTL2_A.FSMUXSEL bit settings control the multiplexing. The ***Frame Sync Combinations*** and ***Clock Combinations*** tables show the valid combinations for the bit settings.

NOTE: All other settings are illegal. However, hardware does not check or prevent the illegal settings. Ensure that programs use only legal combinations.

The column headers in the ***Frame Sync Combinations*** table are defined as follows:

- FS Combination = Frame sync combination, referenced in the notes that follow the ***Clock Combinations*** table.
- HSA_IFS = the setting of the SPORT_CTL_A.IFS configuration bit.
- HSB_IFS = the setting of the SPORT_CTL_B.IFS configuration bit.
- FSAMUX = the setting of the SPORT_CTL2_A.FSMUXSEL configuration bit.
- FSBMUX = the setting of the SPORT_CTL2_B.FSMUXSEL configuration bit.

The Routing column in the ***Frame Sync Combinations*** table defines how the signals are used between the SPORT halves and which pin is used for the frame sync (whether it is an input or an output). Within the column, the inequality characters (≤ and ≥) are used to show the direction of the signal, and the following abbreviations are used (where x = A or B):

- HSx_FI = Frame sync input signal, provided by an external device or the complementing HSPORT.
- HSx_FO = Frame sync output signal.
- SPx_FS = HSPORT's frame sync pin, where the signal is either:
  - Provided by an external source and distributed to both HSPORT frame sync signals.
  - Internally generated by one HSPORT and routed to both the pin and to the complementary HSPORT frame sync signal.

Table 32-7: Frame Sync Combinations

FS Combination	HSA_IFS	HSB_IFS	FSAMUX	FSBMUX	Routing
1	0	0	0	0	Native FS Operation
2	0	1	0	0	Native FS Operation
3	1	0	0	0	Native FS Operation
4	1	1	0	0	Native FS Operation

Table 32-7: Frame Sync Combinations (Continued)

FS Combination	HSA_IFS	HSB_IFS	FSAMUX	FSBMUX	Routing
5	0	0	1	0	HSA_FI ≤ SPB_FS; HSB_FI ≤ SPB_FS
6	0	1	1	0	HSA_FI ≤ HSB_FO ≥ SPB_FS
7	0	0	0	1	HSB_FI ≤ SPA_FS; HSA_FI ≤ SPA_FS
8	1	0	0	1	HSB_FI ≤ HSA_FO ≥ SPA_FS

The column headers in the ***Clock Combinations*** table are defined as follows:

- CLK Combination = Clock combination, referenced in the notes that follow the table.
- HSA_ICLK = the setting of the `SPORT_CTL_A.ICLK` configuration bit.
- HSB_ICLK = the setting of the `SPORT_CTL_B.ICLK` configuration bit.
- CKAMUX = the setting of the `SPORT_CTL2_A.CKMUXSEL` configuration bit.
- CKBMUX = the setting of the `SPORT_CTL2_B.CKMUXSEL` configuration bit.

The Routing column in the ***Clock Combinations*** table defines how the signals are used between the SPORT halves and which pin is used for the serial clock (whether it is an input or an output). Within the column, the inequality characters (≤ and ≥) are used to show the direction of the signal, and the following abbreviations are used (x = A or B):

- HSx_CI = Serial clock input signal, provided by an external device or the complementing HSPORT.
- HSx_CO = Serial clock output signal.
- SPx_CLK = HSPORT's serial clock pin, where the signal is either:
  - provided by an external source and distributed to both HSPORT serial clock signals, or
  - internally generated by one HSPORT and routed to both the pin and to the complementary HSPORT serial clock signal.

Table 32-8: Clock Combinations

CLK Combination	HSA_ICLK	HSB_ICLK	CKAMUX	CKBMUX	Routing
9	0	0	0	0	Native CLK Operation
10	0	1	0	0	Native CLK Operation
11	1	0	0	0	Native CLK Operation
12	1	1	0	0	Native CLK Operation

Table 32-8: Clock Combinations (Continued)

CLK Combination	HSA_ICLK	HSB_ICLK	CKAMUX	CKBMUX	Routing
13	0	0	1	0	HSA_CI ≤ SPB_CLK; HSB_CI ≤ SPB_CLK
14	0	1	1	0	HSA_CI ≤ HSB_CO ≥ SPB_CLK
15	0	0	0	1	HSB_CI ≤ SPA_CLK; HSA_CI ≤ SPA_CLK
16	1	0	0	1	HSB_CI ≤ HSA_CO ≥ SPA_CLK

The following is a comprehensive list of the legal combinations for the above described frame sync and clock multiplexing configurations:

- FS Combinations 1–4 are compatible with all CLK Combinations (9–16)
- CLK Combinations 9–12 are compatible with all FS Combinations (1–8)
- FS Combination 5 is only compatible with CLK Combination 13 (and vice versa)
- FS Combination 6 is only compatible with CLK Combination 14 (and vice versa)
- FS Combination 7 is only compatible with CLK Combination 15 (and vice versa)
- FS Combination 8 is only compatible with CLK Combination 16 (and vice versa)

NOTE: Program only the `SPORT_CTL2` register of the HSPORT that is accepting the signal from the other HSPORT. However, be sure to set the `SPORT_CTL_A.CKRE` and `SPORT_CTL_A.LFS` polarity bits to have identical settings between the HSPORTs when making internal connections via the SPMUX block.

Data Types and Companding

The SPORT uses the data type select field `SPORT_CTL_A.DTYPE` bit to specify one of the four data formats supported by serial ports. These formats apply to any of the operating modes of serial port.

Table 32-9: Data Type Bit Field Settings

DTYPE field	SPORT Receiver	SPORT Transmitter
00	Right-justify, zero-fill unused most significant bits	Normal operation
01	Right-justify, sign-extend unused most significant bits	Reserved
10	Expand using µ-law	Compress using µ-law
11	Expand using A-law	Compress using A-law

These formats apply to data words loaded into the SPORT transmit or receive data buffers. The first two data formats (`00` and `01` values of `SPORT_CTL_A.DTYPE`) are applicable only when SPORT is configured as receiver.

When SPORT is configured as transmitter, only the significant bits are transmitted (per the field defined in control register). Therefore, the transmit data buffers are not actually zero-filled or sign-extended.

The other two data formats enable the companding logic on the transmit or receive path. Companding (compressing or expanding) is the process of logarithmically encoding and decoding data to minimize the number of bits sent. The SPORTs of the processor support the two most widely used companding algorithms, A-law, and µ-law. The algorithms are performed according to the CCITT G.711 specification.

If selected, companding applies to both the enabled data channels. When enabled as SPORT transmitter, writes to transmit buffer make the content compressed to 8 bits according to algorithm selected. (The content is zero filled to the width of the transmit word.) Similarly, if configured in receive mode, the 8 bits in the receive data buffers expand in right-justified, zero fill format per the algorithm selected. If companding is enabled in multichannel mode, it applies to all the active channels.

The compression for transmit data requires a minimum word length of 8 for proper function. If `SPORT_CTL_A.SLEN` is less than 7, then expansion does not work correctly. Also, if the data value is greater than 13-bit A-law or 14-bit µ-law maximum, it automatically compresses to the maximum value.

NOTE: The processor companding logic supports in-place companding feature. So, companding can be used for debug without enabling SPORT.

Companding as a Function

Since the values in the transmit and receive buffers are companded in place, the SPORT can use the companding hardware without transmitting or receiving data, which can be useful during testing or debugging. For companding to execute properly, program the SPORT registers prior to loading data values into the SPORT buffers.

To compress data in place without transmitting, use the following procedure:

1. Set the SPORT as a transmitter (`SPORT_CTL_A.SPTRAN` = 1) with both primary and secondary data channels disabled (`SPORT_CTL_A.SPENPRI` = `SPORT_CTL_A.SPENSEC`= 0).
2. Enable companding in the `SPORT_CTL_A.DTYPE` field.
3. Write a 32-bit data word to the transmit buffer.
4. Wait two system clock cycles to allow the SPORT companding hardware to reload the transmit buffer with the companded value. Any instructions that do not access the transmit buffer can be used to cause this delay.
5. Read the 8-bit compressed value from the transmit buffer.

To expand data in place, use the same sequence of operations with the receive buffer instead of the transmit buffer.

Transmit Path

When the `SPORT_CTL_A.SPTRAN` control bit is set, the HSPORT is in transmit mode. Primary and secondary transmit data paths are then enabled using the `SPORT_CTL_A.SPENPRI` and `SPORT_CTL_A.SPENSEC` bits, respectively. The primary and secondary datapaths are unique and identical, each including its own transmit data buffer, optional companding logic, and transmit shift register.

The data buffer on the primary transmit data path is SPORT_TXPRI_A, and the data buffer on the secondary transmit data path is SPORT_TXSEC_A. The transmit data buffer and output shift register form a FIFO type of structure. When packing is disabled (SPORT_CTL_A.PACK = 0), the SPORT can hold as many as three data words. If packing is enabled (SPORT_CTL_A.PACK = 1), the serial port can hold two packed data words at any time.

The transmit data for primary and secondary channels is written to the SPORT_TXPRI_A and SPORT_TXSEC_A transmit data buffers, respectively. The transmit data buffers can be accessed in core mode through the peripheral bus or in DMA mode through the DMA bus. When a SPORT is configured in transmit mode, the receive paths are deactivated and do not respond to serial clock or frame sync signals. Because the receive data buffers and receive shift registers are also deactivated, reading from an empty and inactive receive data buffer can cause the core to hang indefinitely.

NOTE: Be sure to avoid accesses to inactive data buffers. Such accesses can cause unpredictable SPORT behavior or a hang condition and are not reported in any status register.

This data can optionally be compressed in hardware according to the selected algorithm (μ-law or A-law) and then automatically transferred to the transmit shift register. The shift register, clocked by the SPORT_ACLK signal, then serially outputs this data on the SPORT_AD0 and/or SPORT_AD1 pins (if both are enabled, these output data bits are transmitted synchronously). If the SPORT uses a framing signal, the SPORT_AFS signal indicates the start of the serial word transmission.

When using DMA mode, a single DMA feeds the data buffers of the enabled channels (primary and/or secondary). When using both channels, interleave the data of these channels starting with the primary channel in the transmit buffer.

When the SPORT is configured in non-multichannel mode as a transmitter, the enabled SPORT data pins (SPORT_AD0 and/or SPORT_AD1) are always driven. When a SPORT channel is enabled, data from the transmit data buffer is loaded into the transmit shift register. The shift register then immediately latches the first bit of data (either the LSB or MSB, depending on the SPORT_CTL_A.LSBF configuration bit) and drives it to the respective data pin such that it is ready when the frame sync signal asserts. Similarly, if the frame sync period exceeds the serial word length, then the data pins are driven with the first bit of the next word for transmission immediately after the active word completes, and the outputs are held during the inactive serial clock cycles (clock cycles between frame sync pulses).

When the SPORT is configured in multichannel mode, the data pins are driven only during active transmit channels and are always three-stated during inactive channel slots.

The SPORT provides status of transmit data buffers and also error detection logic for transmit errors such as an underrun condition. See the Error Detection (Status) Interrupt section for more details.

Receive Path

When the SPORT_CTL_A.SPTRAN bit is cleared, the SPORT is in receive mode. Primary and/or secondary receive data paths can be enabled by setting the SPORT_CTL_A.SPENPRI and SPORT_CTL_A.SPENSEC configuration bits, respectively. These data paths are unique but identical, each with a receive shift register, optional companding logic, and a receive data buffer.

The data buffer on the primary receive path is SPORT_RXPRI_A, and the data buffer on the secondary receive path is SPORT_RXSEC_A. The receive data paths act like a three-deep (32-bit words) FIFO because they have two data registers plus an input shift register.

Upon enabling the SPORT data channels, the input shift register shifts in data bits on the SPORT_AD0 and/or SPORT_AD1 pins, synchronous to the SPORT clock signal. If the SPORT uses a framing signal, the SPORT_AFS signal indicates the beginning of the serial word (or frame) to be received. When an entire word is shifted into the primary and secondary channels, the data can be optionally expanded in hardware according to a selected algorithm (µ-law or A-law) and then automatically transferred to the SPORT_RXPRI_A and/or SPORT_RXSEC_A data buffers.

The receive data buffers can be read in core mode through the peripheral bus or in DMA mode through the DMA bus. When the SPORT uses DMA mode, a single DMA reads the data buffers of the enabled channels (primary and/or secondary) and interleaves them in memory beginning with the primary channel when both channels are enabled. When using both channels, software must de-interleave the data of these channels.

The SPORT provides the status of receive data buffers and also error detection logic for receive errors such as overflow. See the Error Detection (Status) Interrupt section for more details.

When a SPORT is configured in receive mode, the transmit paths are deactivated and do not respond to serial clock or frame sync signals. As the transmit data buffers and transmit shift registers in the data paths are also deactivated, programs must not try to access them.

NOTE: Be sure to avoid accesses to inactive data buffers. Such accesses can cause unpredictable SPORT behavior or a hang condition and are not reported in any status register.

Operating Modes and Options

The SPORT has a number of operating modes:

- Standard DSP Serial mode
- I^2S mode
- Left-Justified mode
- Right-Justified mode
- Multichannel (TDM) mode
- Packed I^2S mode

The SPORT halves within a SPORT top module can be independently configured in any of these operating modes unless they are coupled together using SPMUX logic, in which case they must be configured identically. Each half SPORT has its own set of control and data registers and is programmed similarly.

The ***Control Bits for SPORT Operating Modes*** table lists all the programmable configuration bits in the SPORT_CTL_A control register, which combine to determine the overall function and operating mode of the SPORT. The columns are arranged according to the setting of the SPORT_CTL_A.OPMODE bit that selects between standard DSP/multichannel modes and the various I²S modes, and the cell contents are defined as follows:

- Yes – bit is programmable for this mode of operation and may be written
- Reserved – bit is not programmable for this mode of operation and must not be written
- = value – bit must be set to this value to enable this mode of operation
- FUNCTION – indicates alternate function for this bit in this mode

NOTE: When changing operating modes, first clear the SPORT_CTL_A register before again writing the register with the new configuration settings.

Table 32-10: Control Bits for SPORT Operating Modes

Name (Bit #)	Standard DSP Serial Mode	I²S and Left-Justified Mode	Right-Justified Mode	Multichannel (TDM) Mode	Packed I²S Mode
SPENPRI (0)	Valid				
DTYPE (2:1)	Valid	Reserved		Valid	
LSBF (3)	Valid	Reserved		Valid	
SLEN (8:4)	Valid				
PACK (9)	Valid				
ICLK (10)	Valid				
OPMODE (11)	= 0	= 1	= 1	= 0	= 1
CKRE (12)	Valid				
FSR (13)	Valid	Reserved			
IFS (14)	Valid				
DIFS (15)	Valid			Reserved	
LFS (16)	Valid	L_FIRST/PLFS		Valid	L_FIRST/PLFS
LAFS (17)	Valid	OPMODE2	Valid	Reserved	
RJUST (18)	Reserved		= 1	Reserved	
FSED (19)	Valid	Reserved		Valid	Reserved
TFIEN (20)	Valid				
GCLKEN (21)	Valid		Reserved		

Table 32-10: Control Bits for SPORT Operating Modes (Continued)

Name (Bit #)	Standard DSP Serial Mode	I^2S and Left-Justified Mode	Right-Justified Mode	Multichannel (TDM) Mode	Packed I^2S Mode
SPENSEC (24)	Valid				
SPTRAN (25)	Valid				

Serial Word Length

The SPORT uses the `SPORT_CTL_A.SLEN` field to determine the word length of the serial data to transmit or receive. Each half SPORT can independently handle word lengths up to 32 bits, and the value that must be programmed to the `SPORT_CTL_A.SLEN` field is obtained from:

SLEN = Desired SPORT word length - 1

The minimal word length depends on the selected operating mode. Words smaller than 32 bits are right justified in the transmit or receive buffers; however, data can be shifted in or out in MSB or LSB first format, as configured by the `SPORT_CTL_A.LSBF` bit. The received word can also be sign-extended or zero-filled when storing the data to processor memory, as governed by the `SPORT_CTL_A.DTYPE` bit.

The ***Data Lengths for SPORT Operating Modes*** table shows the range of valid word lengths for each of the supported SPORT operating modes.

Table 32-11: Data Lengths for SPORT Operating Modes

Mode	SPORT Word Length (SLEN+1)
Standard DSP Serial	4–32
I^2S	5–32
Left-Justified	5–32
Right-Justified	5–32
Multichannel (TDM)	5–32
Packed I^2S	5–32

NOTE: When the companding feature is enabled on the datapath, it limits the word length settings. See Data Types and Companding for more details about word lengths required for companding. When more than 32 bits per frame sync are required to transmit or receive, use the multichannel mode to spread the data across numerous continuous channels.

Clock Sample and Drive Edges

The SPORT uses two control signals to sample or drive the serial data:

1. Serial clock (`SPORT_ACLK`) - bit clock for the serial data.
2. Frame sync (`SPORT_AFS`) - divides the incoming data stream into frames.

These control signals can be internally generated or externally provided, as determined by the SPORT_CTL_A.ICLK and SPORT_CTL_A.IFS bit settings, respectively.

Data and frame syncs can be sampled on the rising or falling edges of the SPORT clock signal, as determined by the SPORT_CTL_A.CKRE bit. By default, the SPORT_CTL_A.CKRE = 0 setting configures the falling edge of the SPORT_ACLK signal as the sampling edge for receive data and externally supplied frame syncs. The receive data and frame syncs can be sampled on the rising edges of SPORT_ACLK when SPORT_CTL_A.CKRE = 1.

NOTE: The SPORT drives transmit data and internal frame sync signals on the opposite serial clock edge of the sampling edge. Be sure to select the same value for SPORT_CTL_A.CKRE for transmit and receive functions for any two HSPORTs that are connected, and always verify the correct polarity for any external device connected to the SPORT.

The ***Frame Sync and Data Driven on Rising Edge*** figure provides an example of the drive and sample edges when two HSPORTs are connected , each with SPORT_CTL_A.CKRE = 0. In this example, the HSPORT that is configured as the transmitter drives the serial clock and frame sync signals, and both HSPORTs are configured for early, active high frame syncs and a word length of eight bits.

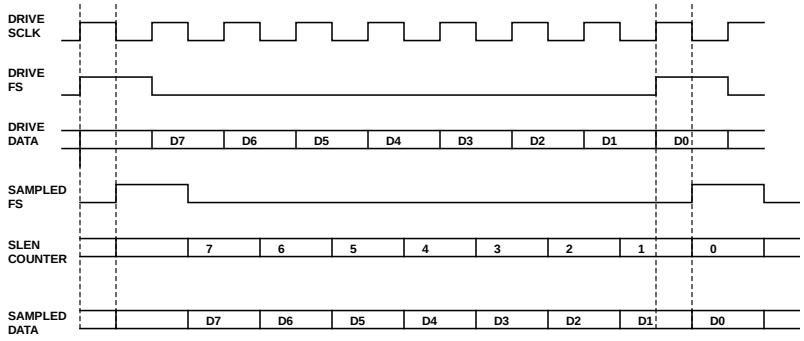


Figure 32-3: Frame Sync and Data Driven on Rising Edge

NOTE: The SCLK in the ***Frame Sync and Data Driven on Rising Edge*** figure is SCLK0.

As shown, the transmitting HSPORT provides the clock and generates the frame sync. Because the HSPORTs are configured for early frame mode, the first bit of data is driven one serial clock later, with subsequent bits being driven on the following rising clock edges in the signal train. When the receiving HSPORT samples the frame sync signal (as indicated in the SAMPLED FS waveform), the SPORT_CTL_A.SLEN bit counter is loaded with the SPORT_CTL_A.SLEN setting, after which each SPORT_ACLK decrements the SPORT_CTL_A.SLEN counter until the full word is received. In this figure, the DRIVE FS and SAMPLED FS waveforms show the frame sync required for the next word in a continuous data stream. Note that it is legal for this frame sync to be sampled synchronous to the last bit of the previous data being sampled, as the early frame mode means that the data lags the frame sync by one serial clock cycle. If the frame sync were sampled as asserted before the D0 bit is sampled, the frame sync error is logged in the receiver's status register.

Since the transmitter drives the internally-generated frame sync and data on the rising edge of the serial clock, the receiver must use the falling edge to sample the externally-supplied frame sync and data.

Frame Sync Options

The following sections provide details regarding the programmable aspects of the SPORT frame sync signal. See the specific operating mode sections for more information regarding frame sync requirements and behavior for each specific operating mode.

Data-Dependent versus Data-Independent Frame Syncs

By default, the generation of a frame sync signal is data-dependent:

- When the SPORT is configured as a transmitter (`SPORT_CTL_A.SPTRAN` = 1), an internally generated transmit frame sync is output when a new data word has been loaded into the channel transmit buffer of the SPORT (by either the core or the DMA engine).
- When the SPORT is configured as a receiver (`SPORT_CTL_A.SPTRAN` = 0), an internally-generated receive frame sync is output only when the receive data buffer is not full.

The data-independent frame sync option, enabled by setting the `SPORT_CTL_A.DIFS` bit, allows for the generation of a periodic framing signal, regardless of the status of the data buffers. When this bit is set, the frame sync output will be continuous and periodic, according to the setting of the `SPORT_DIV_A.FSDIV` field.

Support for Edge-Detected and Level-Sensitive Frame Syncs

The level-sensitive nature of frame sync signals operates well in a noise-free environment. However, if noise corrupts the signals coming into the SPORT, the internal logic can lose synchronization. For example, excessive noise on the frame sync signal may cause the frame sync to be sampled as inactive on the clock edge that it is intended to be synchronous to, but then be sampled at the correct active level one cycle later. Similarly, a noisy clock signal can cause an unintended clock edge, resulting in potential premature sampling of the frame sync signal being applied to the pin.

The ***Level-Sensitive Frame Sync versus Edge Sensitive Frame Sync*** figure describes a scenario where an external frame sync signal is corrupted due to noise, causing the receiving SPORT module to incorrectly sample the signal. If the frame sync is driven on the rising edge of the serial clock at t_A, the SPORT would normally sample the signal on the falling edge of the serial clock at t_B. Due to the noise, however, the SPORT misses the first edge of the frame sync and instead samples it at t_C.

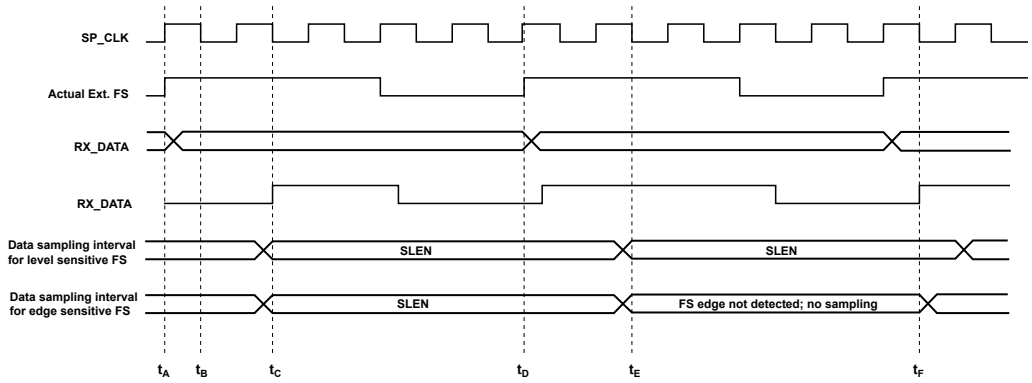


Figure 32-4: Level-Sensitive Frame Sync versus Edge Sensitive Frame Sync

NOTE: SCLK in the ***Level-Sensitive Frame Sync versus Edge Sensitive Frame Sync*** figure is SCLK0.

When the above occurs, the internal word length counter runs for a period equal to the SPORT_CTL_A.SLEN field of the control register, but it erroneously expires at t_E rather than at the appropriate point at t_D, thus receiving incorrect data. Further, if a new level-sensitive frame sync edge arrives at time t_D, the SPORT samples this framing signal again at t_E. As such, the frame sync sampling continues to be misaligned with the external data.

To help address this, the SPORT module provides an option to configure the frame sync signal to instead be edge-sensitive via the SPORT_CTL_A.FSED configuration bit. When this bit is set with active high frame syncs enabled (SPORT_CTL_A.LFS = 0), the rising edge of the frame sync is valid. Conversely, when the frame sync is active low (SPORT_CTL_A.LFS= 1), the falling edge is defined to be valid.

NOTE: SPORT_CTL_A.FSED is valid only in external frame sync mode. In internal frame sync mode, the setting of this bit is irrelevant and ignored.

In the above example, an edge-sensitive frame sync signal is not detected at t_E because the edge of the framing signal already occurred in the previous cycle (t_D) and there is no new edge to detect at t_E. As a result, the internal word length counter remains idle for this frame, thus ignoring the incorrect data, and the counter correctly resumes operation at t_F when a new frame sync edge is detected.

This activity sets the SPORT_ERR_A.FSERRSTAT bit and optionally generates a premature frame sync error interrupt.

Frame sync edge detection is used by default for stereo modes. MCM mode and DSP serial mode choose between edge detection and normal mode of FS detection.

NOTE: When the SPORT is first enabled, an already active externally applied frame sync will not commence operation. The SPORT will wait for a valid change in the frame sync's state from inactive to active before operation begins.

Early versus Late Frame Syncs

Frame sync signals can occur in the same serial clock cycle as the first bit of the data word (late) or one serial clock cycle before the first bit (early), as controlled by the SPORT_CTL_A.LAFS bit.

By default, the frame sync signal is configured to be early (SPORT_CTL_A.LAFS = 0). The first bit of the transmit data word will be driven one serial clock cycle after the frame sync is asserted (whether sensed externally or internally provided), and the first bit of the receive data word is expected to lag the frame sync by one serial clock cycle. The frame sync is not checked again until the entire word has been transferred.

If data transmission is continuous in early framing mode, then an internally-generated frame sync signal will be asserted (pulsed active for one serial clock cycle) synchronous to the last data bit of the current transfer, as the first bit of the next transfer will be immediately driven in the next serial clock cycle (no clocks are wasted). This event is not a premature frame sync error, so the SPORT_ERR_A.FSERRSTAT bit is not set.

The frame sync can alternatively be configured as late (SPORT_CTL_A.LAFS = 1), in which case the first bit of the transmit data word is available in the same serial clock cycle that the frame sync is asserted (whether sensed

externally or internally provided), and the first bit of the receive data word is also latched in the same cycle. Serial clock edges latch the receive data bits, but the frame sync signal is checked only during the first bit of each word. Internally generated frame syncs remain asserted for the entire length of the data word in late framing mode.

The ***Normal Framing (Early Frame Sync) Versus Alternate Framing (Late Frame Sync)*** figure illustrates these concepts.

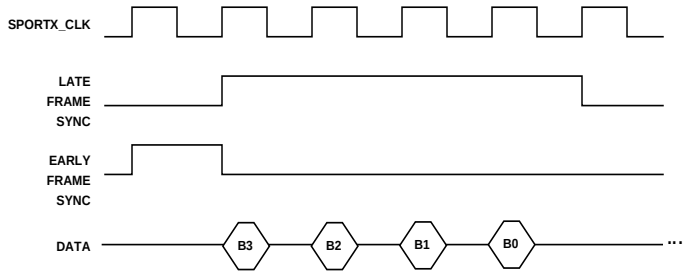


Figure 32-5: Normal Framing (Early Frame Sync) Versus Alternate Framing (Late Frame Sync)

Framed versus Unframed Frame Syncs

The use of a frame sync signal is optional for SPORT operation, as controlled by the `SPORT_CTL_A.FSR` bit. When the frame sync is configured to be required (`SPORT_CTL_A.FSR` = 1), the data is defined to be framed (a frame sync signal must accompany every data word). To allow continuous transmission from the processor, ensure that a new data word is loaded into the transmit buffer before the ongoing transfer is completed (this is automatically cared for when DMA is used to transmit blocks of data).

Data words can be transferred continuously in what is referred to as unframed data mode, which is appropriate for continuous reception, by setting `SPORT_CTL_A.FSR` = 0. In this configuration, a single frame sync is still required to initiate communication, but it is subsequently unrequired once the communication begins. From that point onward, externally provided frame syncs are ignored and internally generated frame syncs are not driven. The ***Framed versus Unframed Data Stream*** figure shows the differences in SPORT operation between framed and unframed data modes with the frame sync configured to be early (`SPORT_CTL_A.LAFS` = 0).

NOTE: When DMA is enabled in a mode where frame syncs are not required, chaining can delay DMA requests. DMA requests are not always serviced frequently enough to guarantee continuous unframed data flow. Monitor status bits or check for a SPORT error interrupt to detect underflow or overflow of data.

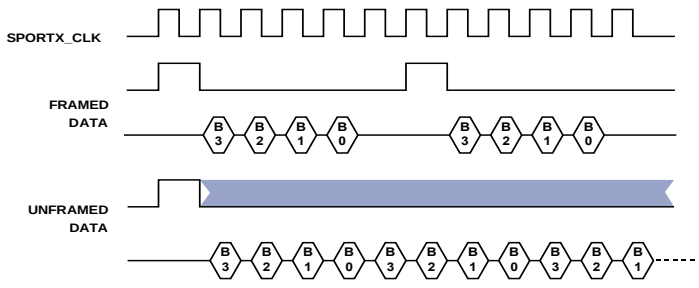


Figure 32-6: Framed versus Unframed Data Stream

Frame Sync Polarity

The framing signals can be active high or active low, as governed by the SPORT_CTL_A.LFS bit:

- When SPORT_CTL_A.LFS = 0, the corresponding frame sync signal is active high.
- When SPORT_CTL_A.LFS = 1, the corresponding frame sync signal is active low.

Active high is the default polarity of the frame sync signal.

Premature Frame Sync Error Detection

A SPORT framing signal is used to synchronize transmit or receive data. In external frame sync mode, any frame sync received during an active frame is premature and invalid. When this occurs, the SPORT_ERR_A.FSERRSTAT bit is set to indicate the framing error, and an optional error interrupt request can be generated for this event by setting the SPORT_ERR_A.FSERRMSK bit.

NOTE: The SPORT_ERR_A.FSERRSTAT bit is not set in the presence of uncleared underflow or overflow errors.

Refer to the ***Frame Sync Error Detection*** figure. The frame sync error bit gets set when an unexpected frame sync occurs during the ongoing data transfer (transmission or reception).

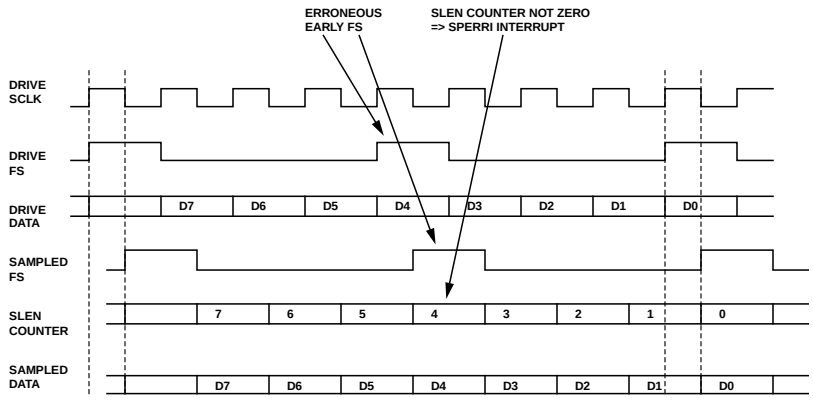


Figure 32-7: Frame Sync Error Detection

NOTE: SCLK in the ***Frame Sync Error Detection*** figure is SCLK0.

Whether a SPORT is receiving or transmitting, its bit count is set to the programmed serial word length when the frame sync is sampled, which is then decremented every subsequent serial clock cycle until the transfer has completed. At this point, the bit count reaches zero and will be reset to the programmed serial length when the next frame sync is sampled. As such, the bit count value is non-zero during an active transfer, and the frame sync error is asserted if a frame sync is sampled when this count is non-zero.

Mode Selection

The SPORT's operating mode is configured in the SPORT_CTL_A and SPORT_MCTL_A registers. The ***SPORT Operating Modes*** table provides specific guidance to properly program these SPORT control registers for the desired mode of operation.

Table 32-12: SPORT Operating Modes

Operating Modes	SPORT_CTL_A.OPMODE	SPORT_CTL_A.LAFS	SPORT_CTL_A.RJUST	SPORT_MCTL_A.MCE
Standard DSP Serial	0	Programmable	Reserved	0
I^2S	1	0	Reserved	0
Left-Justified	1	1	Reserved	0
Right-Justified	1	1	1	0
Multichannel	0	Reserved	Reserved	1
Packed I^2S mode	1	Reserved	Reserved	1

The following sections provide detailed information for each of the supported SPORT modes of operation.

Standard DSP Serial Mode

The SPORT can be configured in standard DSP serial mode by clearing the SPORT_CTL_A.OPMODE and SPORT_MCTL_A.MCE bits. This mode provides great flexibility in terms of programmable options to configure the SPORTs to communicate with various serial devices such as serial data converters and audio codecs. In order to properly connect to such devices, various clocking, framing, and data formatting options are available.

Timing Control Bits

Several bits in the SPORT_CTL_A control register define the configuration of the SPORT in standard DSP serial mode:

- SLEN: serial word length (4–32 bits)
- LSBF: shift LSB or MSB first
- ICLK: internally generated or externally provided serial clock
- CKRE: sample on rising or falling edge of the serial clock
- IFS: internally generated or externally provided frame sync
- FSR: framed or continuous operation
- DIFS: data-dependent or data-independent frame sync
- LFS: active high or active low frame sync
- LAFS: frame sync synchronous to data or one clock cycle before it
- PACK: 16-bit to 32-bit packing option

- GCLKEN: free-running or gated clock

Clocking Options

In standard DSP serial mode, the SPORTs can either accept an external serial clock or generate one internally, as controlled by the SPORT_CTL_A.ICLK bit. For internally generated serial clocks (SPORT_CTL_A.ICLK = 1), the SPORT_DIV_A.CLKDIV field configures the serial clock rate from the system clock.

The SPORT clock can also be gated, where it is only valid during an active transfer, as controlled by the SPORT_CTL_A.GCLKEN bit.

The SPORT clock edge used for driving and sampling of serial data and frame syncs is configured using the SPORT_CTL_A.CKRE bit:

- If SPORT_CTL_A.CKRE = 0, input data and frame sync signals are sampled on the falling edge of the serial clock, and output data and frame sync signals are driven on the rising edge.
- If SPORT_CTL_A.CKRE = 1, input data and frame sync signals are sampled on the rising edge of the serial clock, and output data and frame sync signals are driven on the falling edge.

Stereo Modes

The SPORTs support three widely used stereo modes of operation:

- I^2S mode
- Left-Justified mode
- Right-Justified mode

In these modes, the serial data stream consists of left and right channels. The following sections describe these modes in more detail.

Channel Order

The active low frame sync (SPORT_CTL_A.LFS) bit is used to determine the polarity of the frame sync signal in the non-stereo modes of operation. For the stereo modes of operation, it instead controls whether the right or left channel is first in the data transfer. The ***Channel Order Bit Settings*** table shows which word is transmitted or received first, based on the setting of the SPORT_CTL_A.LFS bit.

Table 32-13: Channel Order Bit Settings

Mode	**SPORT_CTL_A.LFS=0**	**SPORT_CTL_A.LFS=1**
Left-Justified or Right-Justified	Left channel first	Right channel first
I^2S or Packed I^2S	Right channel first	Left channel first

I²S Mode

I²S mode is a commonly used stereo mode, where left and right channel data words are interleaved in the serial data stream and each transition of the frame sync signal is associated with one of the channels. The left channel data is transferred during the low segment of the frame sync signal, and the right channel data is transferred during the high segment of the frame sync signal. As such, the frame sync signal is treated as a left-right (L/R) clock in this mode.

To set up the SPORT in I²S mode, the following configuration is required:

- `SPORT_CTL_A.OPMODE` = 1
- `SPORT_CTL_A.LFS` = 0
- `SPORT_MCTL_A.MCE` = 0

Protocol Configuration Options

Several bits in the `SPORT_CTL_A` control register must be configured to be compliant with the I²S standard, but they can be otherwise configured to support non-standard operations

- SLEN–programmable (allowable word lengths are 5–32 bits)
- LSBF–set to 0 (MSB first)
- ICLK–programmable (serial bit clock can be internally generated or externally provided)
- IFS–programmable (serial L/R clock source must match serial bit clock source)
- LFS–set to 1 (left channel first)
- CKRE–set to 1 (sample L/R clock and data on rising edge of bit clock)

Serial Bit Clock and L/R Clock Rates

When the SPORT is configured to generate the bit clock and the L/R clock (`SPORT_CTL_A.ICLK` = `SPORT_CTL_A.IFS` = 1), set the serial bit clock rate using the `SPORT_DIV_A.CLKDIV` bit field and the L/R clock rate using the `SPORT_DIV_A.FSDIV` bit field.

The ***Word Select Timing in I²S Mode*** figure shows the SPORT timing in I²S mode. The data lags the L/R clock transition by one SCLK0 cycle, and the transfer begins with the left channel data word first.

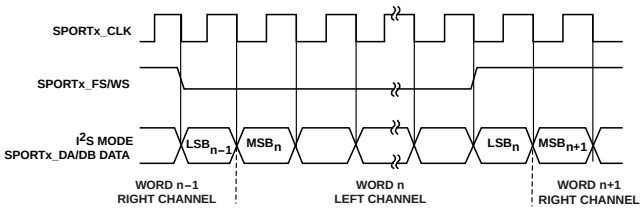


Figure 32-8: Word Select Timing in I²S Mode

Left-Justified Mode

Left-justified mode is a stereo mode subset of the I²S standard. As in I²S mode, the frame sync signal acts as a left-right clock (L/R clock), where left and right data samples are transferred each L/R clock period. The left channel is associated with the high segment of the frame sync, and the right channel aligns with the low segment of the frame sync. The difference between left-justified mode and standard I²S mode is that the channel data is driven in the same bit clock cycle as the L/R clock transition (rather than one bit clock cycle later), such that the MSB is synchronous with the leading edge of the frame sync transition.

To set the SPORT up in left-justified mode, the following configuration is required:

- `SPORT_CTL_A.OPMODE` = 1
- `SPORT_CTL_A.LAFS` = 1
- `SPORT_MCTL_A.MCE` = 0

Protocol Configuration Options

Several bits in the `SPORT_CTL_A` control register must be configured to operate the SPORT in left-justified mode, but they can be otherwise configured as well:

- SLEN: programmable (allowable word lengths are 5–32 bits)
- LSBF: set to 0 (MSB first)
- ICLK: programmable (serial bit clock can be internally generated or externally provided)
- IFS: programmable (serial L/R clock source must match serial bit clock source)
- LFS: set to 0 (left channel first)
- CKRE: set to 1 (sample L/R clock and data on rising edge of bit clock)

Serial Bit Clock and L/R Clock Rates

If the SPORT is configured to generate the bit clock and the L/R clock (`SPORT_CTL_A.ICLK` = `SPORT_CTL_A.IFS` = 1), set the serial bit clock rate using the `SPORT_DIV_A.CLKDIV` bit field and the L/R clock rate using the `SPORT_DIV_A.FSDIV` bit field.

The ***Word Select Timing in Left-Justified Mode*** figure shows the SPORT timing in left-justified mode. The start of a data sample is synchronous to the L/R clock transition, and the transfer begins with the left channel data word first.

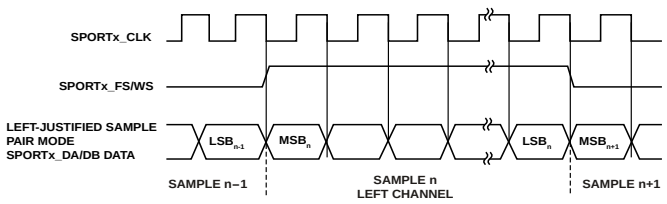


Figure 32-9: Word Select Timing in Left-Justified Mode

Right-Justified Mode

Right-justified mode is a stereo mode subset of the I^2S standard. As in I^2S mode and left-justified mode, the frame sync signal acts as a left-right clock (L/R clock), where left and right data samples are transferred each L/R clock period. The left channel is associated with the high segment of the frame sync, and the right channel aligns with the low segment of the frame sync. The difference between right-justified mode and standard I^2S mode is that the LSB of the channel data ends at the point that the L/R clock transitions to frame the next sample (rather than one bit clock cycle after the L/R clock transition).

To set the SPORT up in right-justified mode, the following configuration is required:

- `SPORT_CTL_A.OPMODE` = 1
- `SPORT_CTL_A.RJUST` = 1
- `SPORT_MCTL_A.MCE` = 0

Timing Control Bits

Several bits in the `SPORT_CTL_A` control register must be configured to operate the SPORT in right-justified mode, but they can be otherwise configured as well:

- SLEN: programmable (allowable word lengths are 5–32 bits)
- LSBF: set to 0 (MSB first)
- ICLK: programmable (serial bit clock can be internally generated or externally provided)
- IFS: programmable (serial L/R clock source must match serial bit clock source)
- LFS: set to 0 (left channel first)
- CKRE: set to 1 (sample L/R clock and data on rising edge of bit clock)

Serial Bit Clock and L/R Clock Rates

If the SPORT is configured to generate the bit clock and the L/R clock (`SPORT_CTL_A.ICLK` = `SPORT_CTL_A.IFS` = 1), set the serial bit clock rate using the `SPORT_DIV_A.CLKDIV` bit field and the L/R clock rate using the `SPORT_DIV_A.FSDIV` bit field.

The ***Word Select Timing in Right-Justified Mode*** figure shows the SPORT timing in right-justified mode. The transmitter aligns the transmit data such that the last bit of the serial word is sent in the last clock cycle of the L/R clock (frame sync) signal marking the channels.

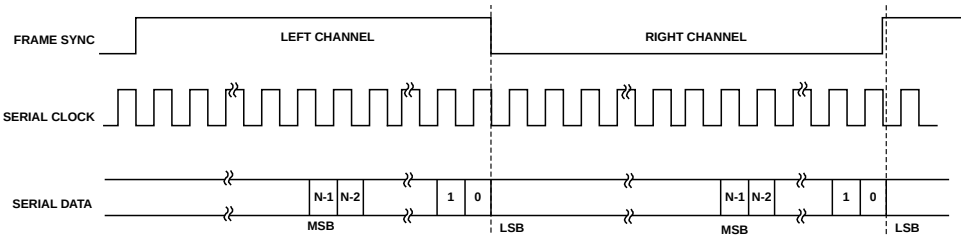


Figure 32-10: Word Select Timing in Right-Justified Mode

NOTE: For some SPORT-compatible ADCs or DACs such as the AD1871, right-justified mode is limited to commonly used ratios such as 64 FS and 128 FS. FS is the sampling frequency of ADCs and DACs, referred to as the SPORT's L/R clock (frame sync) signal.

Consider the SPORT timing for right-justified mode, as shown in the ***Timing Comparison Between Different Stereo Modes*** figure. The frame sync width is limited to 32 SPORT clock periods (or 32 bits per channel) if:

- the SPORT's frame sync (L/R clock) runs at the FS rate, and
- the SPORT's serial bit clock runs at the 64 FS rate

The limitation applies to the frame sync width of either channel. If the data is confined to 24 bits, the SPORT introduces a 32–24=8-bit clock delay before it starts to transmit or capture data.

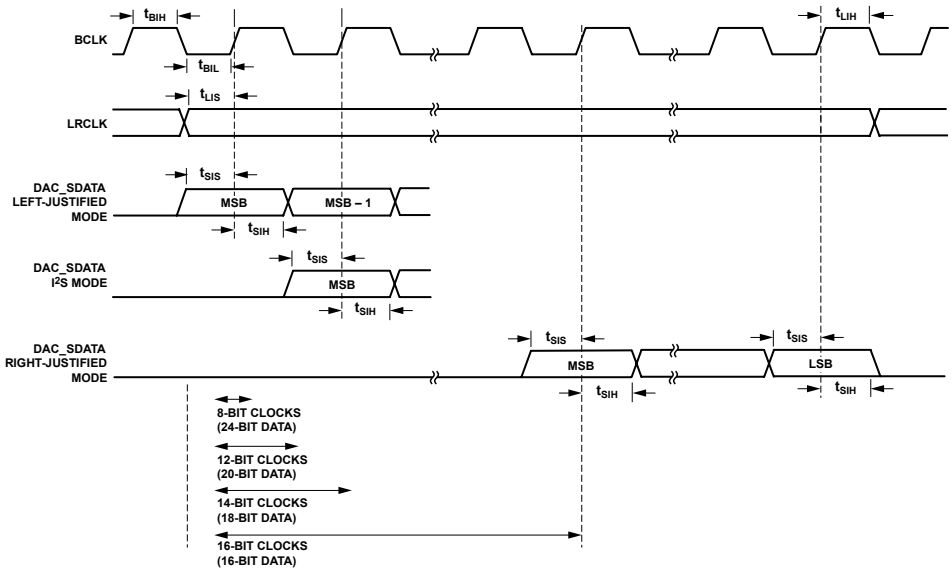


Figure 32-11: Timing Comparison Between Different Stereo Modes

Similarly, to support the 128 FS bit clock frequency, the frame sync width becomes 64 serial bit clock periods per channel. The delay can be a maximum of 59 bit clocks (64 – 5, which is the minimum serial data length in right-justified mode).

The starting point of the first bit is delayed so that the LSB of the serial data aligns properly with the end of the channel. A 6-bit counter is added for this purpose in the stereo mode counter, which is programmed by writing the least significant six bits of the `SPORT_MCTL_A.WOFFSET` field. Though this is a multichannel mode configuration register, the SPORT uses these bits in right-justified mode to configure the offset from the transition of the L/R

clock to where the first data bit must be driven to have the end of the last bit align properly with the next L/R clock transition. The software must program this register with the appropriate delay.

Multichannel (TDM) Mode

The multichannel mode of SPORT operation allows the SPORT to communicate as part of a time division multiplexed (TDM) serial system. In TDM communications, a large frame of streamed serial data words is defined to be a particular length. It consists of a specific number of channels, and each channel contains one serial data word of the defined data length. For example, a 24-word block of 24-bit data can be defined to be a window within a frame, having a duration of 576 bit clocks and comprised of 24 continuous channels. The SPORT is configured to transfer on specific channels within a defined window in this frame.

To set the SPORT up in multichannel mode, the following configuration is required:

- `SPORT_CTL_A.OPMODE` = 0
- `SPORT_MCTL_A.MCE` = 1

In multichannel mode, the SPORT can selectively transfer data on up to a maximum window size of 128 continuous channels out of a maximum 1024-channel frame while ignoring all the disabled channels within the window and all the channels outside the window. The SPORT can do any of the following on each channel:

- Transmit data (`SPORT_CTL_A.SPTRAN` = 1)
- Receive data (`SPORT_CTL_A.SPTRAN` = 0)
- Do nothing (during inactive channels)

Channel selection is configured in the half SPORT multichannel select registers (`SPORT_CS0_A` - `SPORT_CS3_A`) before enabling SPORT operation for multichannel mode. Programming of these registers is especially important in DMA data unpacked mode, since the SPORT data buffers begin operation immediately after the SPORT data lines are enabled. Be sure to enable multichannel operation (set the `SPORT_MCTL_A.MCE` bit) prior to enabling the SPORT itself.

Clocking Options

In multichannel mode, the SPORTs can either accept an external serial clock or generate one internally, as governed by the `SPORT_CTL_A.ICLK` bit. For an internally-generated serial clock (`SPORT_CTL_A.ICLK` = 1), the `SPORT_DIV_A.CLKDIV` bit field is used to configure the serial clock rate, as derived from the system clock.

The serial clock edges used to drive and sample data and frame syncs are also configurable using the `SPORT_CTL_A.CKRE` bit:

- If `SPORT_CTL_A.CKRE`= 0, input data and frame sync signals are sampled on the falling edge of the serial clock, and output data and frame sync signals are driven on the rising edge.
- If `SPORT_CTL_A.CKRE` = 1, input data and frame sync signals are sampled on the rising edge of the serial clock, and output data and frame sync signals are driven on the falling edge.

Frame Sync Options

The frame sync signal synchronizes the channels and restarts each multichannel sequence, starting with the channel 0 data word. For internally-generated frame syncs (SPORT_CTL_A.IFS = 1), the frame sync period in multichannel mode is defined as:

FS period = [(SPORT_CTL_A.SLEN + 1) × number of channels] - 1

The active level for the frame sync signal is also configurable by programming the SPORT_CTL_A.LFS bit. Set this bit to make the frame sync an active low signal, and clear it to make it active high.

In multichannel mode, frame sync timing resembles late framing mode (although the SPORT_CTL_A.LAFS bit is reserved in this mode). The first bit of the transmit data word is driven and the first bit of the receive data word is sampled in the same serial clock cycle as the frame sync, provided there is no programmed frame delay (SPORT_MCTL_A.MFD = 0).

Once the frame sync signal is asserted, word transfers are performed continuously for the duration of the active window, and no further frame syncs are required for different channels within the window. As such, internally-generated frame syncs are always data-independent, and the SPORT_CTL_A.DIFS bit is reserved.

Transmit Data Valid (TDV)

Each SPORT features a transmit data valid signal (SPORT_ATDV), which is driven high during enabled transmit channels. Because the SPORT output data signals are three-stated during inactive channels, the SPORT_ATDV signal signifies when the processor is actively driving the SPORT data outputs, thus serving as an output-enable signal for the data transmit pin(s).

Active Channel Selection Registers

In multichannel mode, the SPORT supports a window size of up to 128 channels for transmitting or receiving data, where it can selectively receive or transmit data in any of these 128 channels. Each channel can be individually enabled or disabled using the multichannel selection registers (SPORT_CS0_A to SPORT_CS3_A) to select the channels in which to transfer data during a multichannel communication stream. Data words associated with enabled channels are transmitted or received in the respective channels, while disabled channels cause a transmit SPORT to three-state the data output pins and a receive SPORT to ignore the data.

The four 32-bit multichannel selection registers combine to form up to a 128-bit meta-register to accommodate the maximum window size of 128 channels. Setting any bit within these registers enables the associated channel. The 128 channels are sequentially numbered from bit 0 in the SPORT_CS0_A register (corresponding to channel 0 of the window) to bit 31 of the SPORT_CS3_A register (corresponding to channel 127 of the window). For example, setting bit 13 of the SPORT_CS1_A register enables channel number 45 (add 32 for the channels in the SPORT_CS0_A). Likewise, setting bit 5 of the SPORT_CS3_A register enables channel number 101 (add 96 for the 32 channels in each of the SPORT_CS0_A, SPORT_CS1_A, and SPORT_CS2_A registers).

Multichannel Frame Delay (MFD)

The multichannel frame delay (SPORT_MCTL_A.MFD) field specifies the delay in serial bit clocks between the frame sync pulse and the first data bit in the frame. This configurability allows the processor to work with different types of telephony interface devices.

As SPORT_MCTL_A.MFD is a 4-bit field, the maximum value allowed for the frame delay is 15 serial clock cycles. When set to 0, the frame sync is concurrent with the first data bit. If SPORT_MCTL_A.MFD>0, a new frame sync can occur during the last channel(s) of a previous frame and still be valid (does not cause a frame sync error).

NOTE: When the required frame delay exceeds 15 serial clocks, use the window offset field (SPORT_MCTL_A.WOFFSET) to delay the start of channel 0 in increments of the serial word length, and then adjust SPORT_MCTL_A.MFD accordingly. For example, when the serial word length is 12 bits and the desired frame delay is 16 serial clock cycles, set the SPORT_MCTL_A.WOFFSET field to 1 to insert a 12-bit delay after the frame sync to where the channel 0 data begins, and then program SPORT_MCTL_A.MFD to four (that is, 16 - 12).

NOTE: When the ASRC module is used with SPORTs, note the following:

- In ASRC TDM mode, the delay between the frame sync (FS) and the data is one clock cycle.
- For the SPORT MFD feature with respect to ASRC TDM mode, select the MFD (SPORT_MCTL_A.MFD / SPORT_MCTL_B.MFD) bit as follows:
  - The ASRC input is expecting one clock cycle between the FS and the data. Configure the MFD transmit bit to 1 (the SPORT which transmits the data to ASRC serial in)
  - The ASRC output has one clock cycle delay between the FS and the data. Configure the MFD receive bit to 1 (for the SPORT which receives the ASRC serial out data).

Window Size (WSIZE)

Select the number of channels used in multichannel operation by programming the 7-bit SPORT_MCTL_A.WSIZE field. This field must be set to the actual number of channels minus one (SPORT_MCTL_A.WSIZE = Number of channels -1).

The 10-bit SPORT_MSTAT_A.CURCHAN field holds the channel number currently being serviced during multi-channel operation.

Window Offset (WOFFSET)

The window offset (SPORT_MCTL_A.WOFFSET) field specifies where in the 1024-channel frame to place the start of the active window (up to 128 channels long). A value of 0 specifies no channel offset from the frame sync (channel 0 immediately follows it). Any non-zero value indicates the number of channels that come between the frame sync and the start of channel 0 of the active frame, with 896 (for example, 1024–128) being the largest value that permits using all 128 channels.

As an example, a program could define an active window comprised of eight channels (SPORT_MCTL_A.WSIZE = 7) with a window offset of 93 (SPORT_MCTL_A.WOFFSET = 93). If configured in this fashion, the 8-channel window that the SPORT will transfer within resides in the channel range from 93 to 100 in the up-to-1024-channel frame.

Do not change the window offset or the number of multichannel slots (SPORT_MCTL_A.WSIZE) while the SPORT is enabled. If the combination of the window size and offset place any portion of the window out-of-range relative to the channel counter, none of the channels are enabled.

Companding Selection

Like the other operating modes, companding logic can optionally be applied to serial data (compression logic for transmit mode or expansion logic for receive mode). The two widely used companding algorithms, A-law and μ-law, are selectable using the SPORT_CTL_A.DTYPE field.

If companding is enabled, the companding algorithm is applied to both the primary and secondary datapaths. In multichannel mode, companding can be applied to either all or none of the enabled channels (companding cannot be selected on a per-channel basis).

Multichannel DMA Data Packing (MCPDE)

Multichannel DMA data packing and unpacking are enabled using the SPORT_MCTL_A.MCPDE bit.

When set, data is packed, and the SPORT expects the data in the DMA buffer to correspond only with enabled SPORT channels. For example, if only channels 1 and 9 are enabled in a 10-channel window (SPORT_MCTL_A.WSIZE = 9), the SPORT expects the buffer to be exactly two words in length, where channel 1 is associated with the first element in the buffer and channel 9 is associated with the second.

When cleared, data is unpacked, and the SPORT expects the DMA buffer to have a word for each of the channels in the active window, whether the channel is enabled or not. As such, the DMA buffer size must be exactly the size of the window. Using the same example as the packed case above, if only channels 1 and 9 are enabled in a 10-channel window (SPORT_MCTL_A.WSIZE = 9), then the DMA buffer size is ten words. The data at offsets 1 and 9 within the buffer are associated with the data transfers of channels 1 and 9, respectively. The rest of the words in the buffer are unused.

Packed I^2S Mode

The SPORT supports a packed I^2S mode, which can be used for audio codec communications using multiple channels. This mode allows applications to send more than the standard 32 bits per channel available through standard I^2S mode. Packed mode is implemented using standard multichannel mode (and is therefore programmed similarly to multichannel mode).

To set the SPORT up in packed I^2S mode, the following configuration is required:

- SPORT_CTL_A.OPMODE = 1
- SPORT_MCTL_A.MCE = 1

Like multichannel mode, packed I²S mode also supports a maximum of 128 channels, where up to 128 channels of data can be transferred for every transition of the frame sync signal acting as an L/R clock (for example, up to 128 left-channel words transfer during the high portion of the L/R clock, and up to 128 right-channel words transfer during the low portion).

As shown in the ***Packed I²S Mode 128 Operation*** figure, the packed waveforms are the same as those waveforms used in multichannel mode, except the frame sync is toggled for every frame and emulates I²S mode.

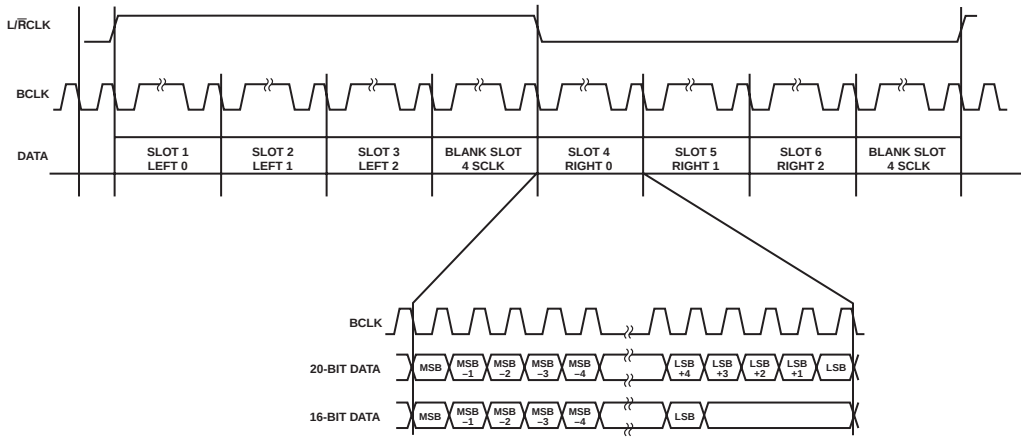


Figure 32-12: Packed I²S Mode 128 Operation

Serial Bit Clock Options

In packed I²S mode, the SPORTs can either accept an external serial bit clock or generate one internally, as governed by the `SPORT_CTL_A.ICLK` configuration bit. For an internally-generated serial bit clock (`SPORT_CTL_A.ICLK` = 1), use the `SPORT_DIV_A.CLKDIV` bit field to configure the serial bit clock rate from the system clock.

The serial bit clock edge that is used for sampling or driving data and frame syncs is programmable using the `SPORT_CTL_A.CKRE` bit.

L/R Clock (Frame Sync) Options

The frame sync period in packed I²S mode is defined as:

FS period = [(`SPORT_CTL_A.SLEN` + 1) × number of channels] - 1.

The L/R clock can be supplied externally or internally generated depending on the `SPORT_CTL_A.IFS` bit setting. The logic level of the L/R clock associated with the left and right channel data can be changed using the `SPORT_CTL_A.LFS` configuration bit.

Gated Clock Mode

Some system components such as ADCs and DACs utilize a SPI-compatible protocol for the interface. To communicate with such devices, the SPORT must support a gated clock, where the data valid information is embedded in

the clock (for example, the clock only toggles when data is valid). This gated clock feature is enabled using the SPORT_CTL_A.GCLKEN bit.

To enable the gated clock mode of operation, program the SPORT to comply with the following requirements.

- Do not enable gated clock functionality in right-justified or multichannel mode
- Gated clock mode has the following requirements for other control bits:
  - The serial clock and frame sync signals must have the same source (SPORT_CTL_A.ICLK = SPORT_CTL_A.IFS)
  - Unframed mode is not supported (SPORT_CTL_A.FSR must be set)
  - Clear the SPORT_CTL_A.DIFS bit in transmit mode; set it in receive mode
- Satisfy the following necessary conditions when gated clock mode is enabled:
  - Seven serial clock cycles are required between enabling the SPORT and the first frame sync. If this requirement is not met, the SPORT can drop the first data (for subsequent data, this requirement is not applicable)
  - For externally-provided clock and frame sync, the frame sync must be inactive during clock synchronization after the SPORT has been enabled
  - For an edge-detected frame sync (SPORT_CTL_A.FSED = 1), the frame sync must transition back to the inactive state before the current word transfer is complete (or when the clock is still running). If this requirement is not met, the SPORT does not recognize the next valid frame sync and skips the channel. The SPORT continues to skip the frame syncs until the frame sync transitions back to an inactive state while the clock is active.

Data Transfers and Interrupts

SPORT data can be transferred to or from internal or external memory by two methods:

- Core-driven, single-word transfers
- DMA-driven, multiple-word transfers (optionally with multiple work units)

Core-driven transfers use SPORT interrupts to signal the processor core to perform MMR-based single-word transfers to or from the SPORT data buffers. DMA can be set up to automatically transfer a configurable number of serial words between the SPORT transmit/receive data buffers and memory, and then generate a data completion interrupt request when a work unit or a series of work units completes, thus signaling by the SEC to the processor core that a block of data has been transferred.

The following sections provide information on core-driven and DMA-driven data transfers.

Data Buffers

When programming the serial port data channels (primary or secondary) as a transmitter by setting SPORT_CTL_A.SPTRAN =1, only the corresponding transmit data buffers (SPORT_TXPRI_A and SPORT_TXSEC_A) become active. The receive data buffers (SPORT_RXPRI_A and SPORT_RXSEC_A) remain inactive. Similarly, when the SPORT data channels are programmed for receive operation (SPORT_CTL_A.SPTRAN =0), then only corresponding receive data buffers (SPORT_RXPRI_A and SPORT_RXSEC_A) are active. Do not attempt to read or write inactive data buffers. When the processor operates on the inactive transmit or receive buffers while the SPORT is enabled, unpredictable results can occur.

Each of these buffers is 32-bit wide (corresponds to maximum serial data word length). When using word lengths less than 32 bits for SPORT operation, the data in these buffers is automatically right justified. (The LSB bit of data is at the bit 0 location of the buffer). The upper unused bits can be zero-filled or sign-extended depending on SPORT_CTL_A.DTYPE field.

Transmit Data Buffers (SPORT_TXPRI_A and SPORT_TXSEC_A)

When enabled as a transmitter (SPORT_CTL_A.SPTRAN =1), each SPORT half has its own set of transmit data buffers. The primary (0) and secondary (1) datapaths of each SPORT half have separate data buffers, referred to as SPORT_TXPRI_A and SPORT_TXSEC_A, respectively.

These transmit data buffers are 32 bits wide. Load these buffers with the data for transmission on the primary and secondary data channels. The DMA controller loads the data automatically. Or, the program running on the processor core loads the data manually.

Together with the output shift register, transmit data buffers act like a two-location FIFO. If data packing is disabled (SPORT_CTL_A.PACK =0), the transmit path can hold as many as three data words. If data packing is enabled (SPORT_CTL_A.PACK =1), it can hold two packed data words at any given time.

When the transmit shift register becomes empty (transfer out all the bits of previous word), data in the transmit data buffer is automatically loaded into it. An interrupt occurs when the output transmit shift register has been loaded, signifying that the transmit data buffer is empty and ready to accept the next word. This interrupt does not occur when serial port is operating in DMA mode or when the corresponding interrupt enable mask bit is set.

If only the primary datapath of a SPORT half is enabled, programs must not write to the inactive secondary transmit data buffer and conversely. If the core keeps writing to the inactive buffer, the status of that transmit buffer becomes full. This state can cause the core to hang indefinitely, since data is never transmitted to the output shift register.

Receive Data Buffers (SPORT_RXPRI_A and SPORT_RXSEC_A)

When enabled as receiver (SPORT_CTL_A.SPTRAN =0), each SPORT half has its own set of receive data buffers. The primary (0) and secondary (1) datapaths of each SPORT half have separate data buffers, referred as SPORT_RXPRI_A and SPORT_RXSEC_A, respectively. Together with input shift register, the receive data buffers act like a three-location FIFO, as the receive path has two data registers.

These receive data buffers are the 32 bits wide. These buffers are automatically loaded from the receive shift register when a complete word has been received into it. An interrupt occurs when the receive data buffer is loaded, signifying that new data is available in the receive data buffer and is ready to read. This interrupt does not occur when the serial port is operating in DMA mode or when the corresponding interrupt enable mask bit is set.

If only the primary datapath of a SPORT half is enabled, programs must not read from the inactive secondary receive data buffer and conversely. If the core keeps reading from the inactive buffer, the status of that receive buffer becomes empty. This state can cause the core to hang indefinitely since new data is never received through the input shift register.

Data Buffer Status

The SPORT provides status information about its primary and secondary data buffers through the SPORT_CTL_A.DXSPRI and SPORT_CTL_A.DXSSEC bits, respectively. It also provides error status information through the corresponding SPORT_CTL_A.DERRPRI and SPORT_CTL_A.DERRSEC bits, respectively. Depending on the SPORT_CTL_A.SPTRAN bit setting, these bits reflect the status of either the pair of transmit (SPORT_TXPRI_A and SPORT_TXSEC_A) or receive (SPORT_RXPRI_A and SPORT_RXSEC_A) buffers, indicating whether the buffer is full, partially full, or empty.

When attempting to read from an empty receive buffer or write to a full transmit buffer, the SPORT delays access until the buffer is ready, potentially resulting in excessive MMR bus response times. To avoid this when doing core-driven transfers, always check the buffer status to determine if the access can be made. The SPORT updates the status bits in the SPORT_CTL_A register during reads and writes by the core processor.

NOTE: These status bits are updated during reads and writes from the core processor even when the SPORT is disabled.

Two complete 32-bit words can be stored in the receive buffer while a third word shifts in. Therefore, almost three complete words can be received without the receive buffer being read before an overflow occurs. After receiving the third word completely, the shift register contents overwrite the second word, which will occur if the first word has not yet been read by the processor core or the DMA controller. This receive overflow condition is flagged through the error status bits of the SPORT_CTL_A register on the last bit of the third word.

Data Buffer Packing

When the SPORT is configured as a receiver with a serial data word length of 16 or less, the received data words can be packed into a 32-bit word. Similarly, if the SPORT is configured as a transmitter with a serial data word length of 16 or less, then 32-bit words being transmitted can be unpacked into 16-bit words. The SPORT_CTL_A.PACK bit is used to select this packing or unpacking feature.

When SPORT_CTL_A.PACK = 1, two consecutive received words are packed into a single 32-bit word, or each 32-bit word is unpacked and transmitted as two 16-bit words. The first 16-bit (or smaller) word is right-justified in bits 15–0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31–16. This packing method applies to both receive (packing) and transmit (unpacking) operations. In this case, the transmit and receive interrupt requests are generated for the 32-bit packed words, not for each 16-bit word.

NOTE: When 16-bit received data is packed into 32-bit words and stored in normal word space in the processor's internal memory, the 16-bit words can be read or written using short word space addressing.

Single-Word (Core) Transfers

The SPORTs can transmit or receive individual data words with interrupt requests occurring as each data word is transferred. When a SPORT is enabled with the corresponding DMA channel disabled, interrupt requests are generated when:

- a complete word has been received in the receive data buffer or
- the transmit data buffer is not full

When performing core transfers, be sure to access only those buffers that are associated with enabled datapaths, as governed by the transfer direction (`SPORT_CTL_A.SPTRAN`) bit and the primary/secondary (`SPORT_CTL_A.SPENPRI`/`SPORT_CTL_A.SPENSEC`) data enable bits. If inactive SPORT data buffers are read from or written to by the core while the SPORT is enabled, the core can hang. For example, if a half SPORT is programmed to be a transmitter and the core reads from one of the receive buffers associated with that half SPORT, the core can hang as if it were reading an empty buffer that is active and awaiting new data to arrive. Because this is a transmitting HSPORT, that data will never arrive, thus locking the core up until the SPORT is reset. To avoid such a situation, be sure to check the status of the appropriate data buffer before attempting a core access to it by interrogating the `SPORT_CTL_A.DXSPRI` or `SPORT_CTL_A.DXSSEC` status bits.

DMA Transfers

Direct memory access (DMA) provides a mechanism for transferring an entire block of serial data before an interrupt is generated. The processor's on-chip DMA controller automatically handles the DMA transfer, thus allowing the processor core to run in parallel until the entire block of data is transferred. When the interrupt request occurs, a service routine can then process the entire block of data (rather than react to single words), thus significantly reducing overhead.

Each half SPORT has a dedicated DMA channel that serves both the primary and secondary datapaths. When configured as a transmitter (`SPORT_CTL_A.SPTRAN` = 1) with both the primary and secondary datapaths enabled (`SPORT_CTL_A.SPENPRI` = `SPORT_CTL_A.SPENSEC` = 1), the DMA channel requires that the source DMA buffer interleave the data beginning with the primary channel, as it will alternately load to the primary and secondary transmit data buffers once it is enabled. The complementary operation is true in receive mode (`SPORT_CTL_A.SPTRAN` = 0) when both datapaths are enabled, as the DMA channel alternately reads from the primary and secondary receive data buffers and interleaves them in the destination DMA buffer. As such, software must de-interleave the data corresponding to the primary and secondary channels from the receive DMA buffer.

If the SPORT is configured in stereo mode, the same DMA channel handles both the left and right channels of both datapaths (primary and/or secondary). Therefore, for a transmit DMA with only one datapath enabled, the source buffer must be populated such that the left- and right-channel data is interleaved. If both datapaths are enabled, the DMA channel alternately loads to the primary and secondary transmit data buffers once it is enabled. As such, the interleaving requirement is for the primary left-channel data to be followed by the secondary left-channel data, then the primary right-channel data, and finally the secondary right-channel data. The complementary operation is true

in receive mode, where the DMA channel alternately reads from the primary and secondary receive data buffers and interleave them in the destination DMA buffer. For the stereo modes of operation, the destination DMA buffer is interleaved as left-right data for a single data input. If both datapaths are enabled, the destination DMA buffer is written with the primary and secondary left-channel data followed by the primary and secondary right-channel data. As such, software must de-interleave the primary and secondary left- and right-channel data from the receive DMA buffer, as defined by this scheme.

Since both the primary and secondary datapaths share the single DMA channel, each half SPORT has a single interrupt request for data completion, as well as an error interrupt request. The DMA controller can generate an interrupt request at the end of a chain of DMA work units (when using multiple descriptors) or at the end of individual DMA work unit.

The SPORT DMA channels are assigned a higher priority than all the other DMA channels (for example, the SPI port). Having higher priority causes the SPORT DMA transfers to execute first when multiple DMA requests occur in the same cycle. The SPORT DMA channels are numbered and prioritized in the DMA channel list table in the DMA chapter.

Although the most efficient DMA transfers execute with 32-bit words, the SPORTs can handle word sizes from 4 to 32 bits (as defined by `SPORT_CTL_A.SLEN` field). If the serial data length is 16 bits or smaller, two pieces of data can be packed into 32-bit words for each DMA transfer, as selected by setting the `SPORT_CTL_A.PACK` bit. When this bit is set, the SPORT generates the transmit and receive interrupts for the 32-bit packed words, not for each 16-bit word. For more information, see the Data Buffer Status section.

NOTE: The SPORT DMA channel can access both internal memory and external memory of the processor without any core overhead.

Data Transfer Interrupt

Each half SPORT features a data transfer interrupt request that is shared by both the primary and secondary data channels in both transmit and receive modes. To determine the source of the data transfer interrupt request, applications can check the primary and secondary data buffer status bits (`SPORT_CTL_A.DXSPRI` and , `SPORT_CTL_A.DXSSEC` , respectively).

When using core-driven transfers, this interrupt's meaning depends on the direction of the SPORT:

- As transmitter (`SPORT_CTL_A.SPTRAN` = 1) - the transmit data buffer is empty
- As receiver (`SPORT_CTL_A.SPTRAN` = 0) - new data is available in the receive data buffer

NOTE: When data packing is enabled (`SPORT_CTL_A.PACK` = 1), the core-driven transmit and receive interrupt requests are generated for 32-bit packed words, not for each 16-bit word.

In both cases, the interrupt request can be used to signal the core that an individual transfer has completed. For transmit operations, it indicates that the transmit data buffer can be safely loaded (either the buffer is already empty, or the last data has moved from the data buffer to the shift register). For receive operations, it indicates that new data has arrived and can be read (or must be read before a subsequent word overwrites it).

When the SPORT is configured to use DMA to move data between memory and the peripheral (the most generic way to use dedicated DMA for sport data transfers), the same data transfer interrupt request instead indicates the completion of the transfer of a block of serial data (rather than a single word). When DMA is used, the DMA count register must be initialized to specify the number of words to transfer. This count decrements after each DMA transfer on the channel, and the data transfer interrupt request signal is asserted when the word count reaches zero (for example, a DMA work unit has finished).

For transmit DMA, the interrupt request is raised when the last word in the DMA work unit is loaded from the source memory to the HSPORT FIFO. This interrupt request can signal to the core that a new DMA work unit can be configured or that other software threads can now run. The transmit interrupt request can optionally be deferred until the last word of the work unit has fully shifted out of the shift register (see the Transfer Finish Interrupt (TFI) section for details).

For receive DMA, the interrupt request is raised when the last word is loaded to the destination memory. In addition to that described for transmit DMA, this interrupt request also serves as an indication to the core that there is a buffer of newly acquired data that is ready to be processed.

See the DMA chapter for further details regarding enabling of the DMA interrupt requests associated with the various modes of DMA operation.

NOTE: As a single DMA channel services both the primary and secondary datapaths associated with the SPORT, there is a single DMA completion interrupt request.

Transfer Finish Interrupt (TFI)

When configured for transmit DMA (`SPORT_CTL_A.SPTRAN` =1), the data transfer interrupt request gets generated by the DMA engine itself when it decrements its count register upon loading the last element from memory to the HSPORT hardware. Alternately, the SPORT can use a Transmit Finish Interrupt (TFI) to signal the actual end of the transmission (for example, when the last bit of the last data word of the buffer has shifted out of the SPORT to the system) by setting the `SPORT_CTL_A.TFIEN` bit. When this bit is set, then DMA signal that would normally assert the data transfer interrupt request instead signals the SPORT that the DMA work unit is complete. The SPORT then waits until all the data in the FIFO is shifted out (including the transmit shift register) and asserts the TFI interrupt request upon completion.

NOTE: To enable this functionality in the DMA engine, be sure to configure the interrupt type field in the DMA configuration register for Peripheral interrupt. See the DMA chapter for further details.

Error Detection (Status) Interrupt

In addition to the dedicated data transfer interrupt request, each half SPORT also features an optional error status interrupt request that can be triggered when error conditions occur relative to data or frame syncs associated with the half SPORT.

Data-related errors depend on the direction of the SPORT and reflect overflow or underflow conditions, which are depicted in the `SPORT_CTL_A` control register as read-only sticky bits `SPORT_CTL_A.DERRPRI` and `SPORT_CTL_A.DERRSEC` (for the primary and secondary channels, respectively).

- When the SPORT is configured as a transmitter, these bits provide transmit data buffer underflow status. When the frame sync signal occurs when the transmit data buffer is empty, the underflow bit corresponding with the offending transmit data buffer is set, as the SPORT will transmit data whenever it detects a valid frame sync signal, whether new data is present or not.
- When the SPORT is configured as a receiver, these bits provide receive overflow status. When a channel receives new data while the receive buffer is already full, the new data overwrites the existing data, thus causing an overflow. When this occurs, the overflow bit corresponding with the offending receive data buffer is set, as the SPORT receives data whenever it detects a valid frame sync signal, whether there is room in the receive buffer or not.

Each half SPORT also features an error register (SPORT_ERR_A), which is the source for the assertion of the described data-related error status bits. When a data-related error occurs on the primary or secondary datapaths, the error is logged in the SPORT_ERR_A.DERRPSTAT or SPORT_ERR_A.DERRSSTAT bits, respectively. To enable these status bits to generate the HSPORT status interrupt request in the SEC, the corresponding SPORT_ERR_A.DERRPMSK and SPORT_ERR_A.DERRSMSK bits must be set (for the primary and secondary datapaths, respectively).

The SPORT_CTL_A.DERRPRI and SPORT_CTL_A.DERRSEC channel error status bits are sticky read-only bits that can be cleared in two ways:

- Reset the error detection logic by disabling the channel associated with the error condition (clear the SPORT_CTL_A.SPENPRI or SPORT_CTL_A.SPENSEC control bit).
- Clear the source of the interrupt by writing-1-to-clear the SPORT_ERR_A.FSERRSTAT, SPORT_ERR_A.DERRPSTAT, or SPORT_ERR_A.DERRSSTAT status bits.

In addition to data-related errors, SPORT_ERR_A also tracks frame sync errors in the SPORT_ERR_A.FSERRSTAT status bit. Similar to the data-related errors, the frame sync error can be enabled as a source for raising the error status interrupt request via the SEC by setting the SPORT_ERR_A.FSERRMSK bit. A frame sync error occurs when the frame sync is detected prematurely, as explained in the Premature Frame Sync Error Detection section.

A frame sync error is not detected in the following cases:

- When there is no active transmit or receive data, and the frame sync pulse occurs due to noise on the input signal–when there is no active transfer, a noise-induced frame sync pulse is valid.
- When there is an active underflow or overflow error–frame sync errors cannot be detected because the SPORT error logic does not run after one of the data errors has occurred and remains un-serviced.
- When the frame sync pulse does not the meet minimum timing requirements–when the frame sync pulse is shorter than a SPORT clock period, there is no guarantee that it gets sampled and may go unnoticed.

Grouping of SPORTs

For some applications, enabling multiple SPORTs need to be synchronized so that all of them start and/or end at the same time. It may also be required to generate/receive a single interrupt/trigger for the DMA transfer of all the SPORTs in the same group.

The DAI_GBL_SP_EN register is used to control enabling/disabling. The DAI_GBL_INT_EN register can be used to control interrupt/trigger generation/reception by multiple SPORTs in a group.

NOTE: All references to SPORT 0-3 in the context of DAI1 must be read as SPORT4-7.

SPORT Enable Grouping

The figure shows the block diagram of possible group enabling options for the SPORTs in DAI0 and DA1.

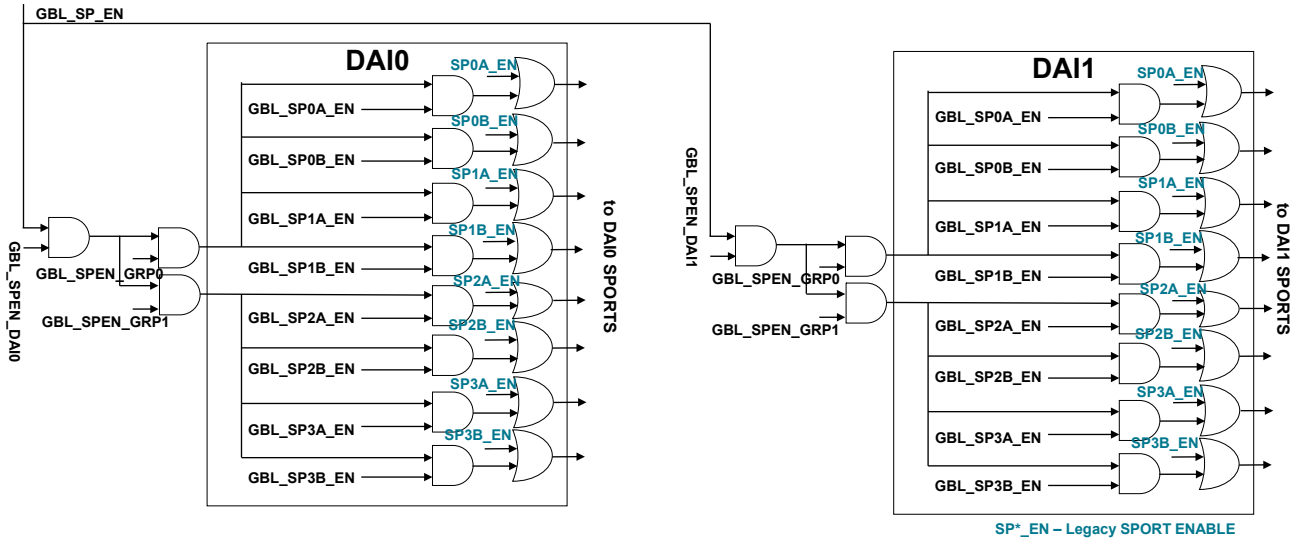


Figure 32-13: Group Enabling Options in DAI0 and DA1

It shows one of the channels in each half sport, same is applicable for other channel as well. For example, GBL_SP0A_EN means GBL_SP0A_PC_EN or GBL_SP0A_SC_EN.

Each DAI has a DAI_GBL_SP_EN register. To enable the SPORTs grouped within the same DAI at the same time, write to the corresponding DAI_GBL_SP_EN register is required. When a group of SPORTs in different DAIs must be enabled simultaneously, writing to the DAI_GBL_SP_EN.GBL_SP_EN field is required. This bit is reserved for the DAI1.

The following group enable options are possible:

- Up to 16 channels in 8 half SPORTs in DAI0
- Up to 16 channels in 8 half SPORTs in DAI1
- Up to 8 channels in 4 half SPORTs in each DAI
- Up to 32 channels in 16 half SPORTs (8 half SPORTs in DAI0 and 8 half SPORTs in DAI1)

NOTE: While grouping SPORTs across DAIs, two interrupts (one per DAI) are generated and managed by software. Grouping can be done only once as the DAI_GBL_SP_EN.GBL_SP_EN field is used.

NOTE: The grouping of SPORTs and its interrupts/triggers are two separate entities. When the interrupts and triggers must be grouped similar to the SPORT grouping, use the group interrupt/trigger enable register.

SPORT Interrupt/Trigger Grouping

Each DAI has a DAI_GBL_INT_EN register. Each of these registers has two group interrupt enable bits DAI_GBL_INT_EN.GRP0_INT_EN and DAI_GBL_INT_EN.GRP1_INT_EN, two group trigger (controller) enable bits DAI_GBL_INT_EN.GRP0_TRG_EN and DAI_GBL_INT_EN.GRP1_TRG_EN. There are individual GRPx_SPyINT_EN bits for each group to enable both interrupt/trigger for up to 8 channels (4 half SPORTs). Enable these bits as per the grouping in the DAI_GBL_SP_EN register.

NOTE: To enable or to disable the SPORT interrupt/trigger grouping feature, the individual sport interrupt/trigger enable bits must be programmed along with the respective group interrupt/trigger enable bits in the DAI_GBL_INT_EN register.

Regarding grouping of SPORT interrupts/triggers:

- All the interrupts/triggers in a single group are ANDed and a single interrupt/trigger goes to SEC/TRU.
- To generate group interrupt/trigger, all the SPORTs in the corresponding group should be configured to generate interrupt (DMA_CFG.INT)/trigger (DMA_CFG.TRIG).
- The interrupts/triggers generated by the SPORTs in a group can either be used as a single group interrupt/trigger (enabled by the DAI_GBL_INT_EN register) or the individual SPORT DMA done interrupts/triggers.
- As there are no separate status registers for global interrupts, to clear the global interrupts, clear (W1C) the individual SPORT DMA_STAT.IRQDONE bits. Theoretically, the global interrupt gets cleared by clearing the DMA_STAT.IRQDONE bit of at least one SPORT in the group. However, it is recommended to clear the DMA_STAT.IRQDONE bits of all the SPORTs in the group to achieve a clean state for the next global interrupt generation.
- The DAI_GBL_INT_EN.GRP1_TRG_EN/DAI_GBL_INT_EN.GRP0_TRG_EN bits enables group controller trigger functionality for SPORTs in a group.

SPORT Programming Model

The following sections provide programming guidance for setting up the SPORTs for use in an application:

- Initializing Core-Driven (Non-MCM) Transfers
- Initializing Multichannel Transfers
- Using DMA for SPORT Transfers
- Using Companding as a Function

Initializing Core-Driven (Non-MCM) Transfers

The following programming model applies to all of Standard DSP Serial Mode, I²S Mode, Left-Justified Mode, and Right-Justified Mode for core-driven transfers. More steps are required to properly initialize the SEC to service the SPORT interrupts (see the SEC chapter for details).

NOTE: This example uses half SPORT A registers. With appropriate changes to register names, this example also applies to half SPORT B.

1. Clear the `SPORT_CTL_A` and `SPORT_MCTL_A` configuration registers.

   ADDITIONAL INFORMATION: Clearing these registers ensures that the SPORT logic (including the multi-channel logic) is fully reset before attempting to reprogram it.

2. Optionally program the `SPORT_DIV_A` clock divisor register.

   ADDITIONAL INFORMATION: This step is only required for internally-generated timing signals. Configure the serial bit clock and/or frame sync (or L/R clock, for stereo modes) rates according to the guidance in the Serial Clock and Frame Sync sections.

3. Program the `SPORT_CTL_A` primary configuration register.

   ADDITIONAL INFORMATION: Set the SPORT operating mode along with the configurable clock, frame sync, word length, direction, and data format options (see the Operating Modes and Options section for details). Do not set the `SPORT_CTL_A.SPENPRI` and/or `SPORT_CTL_A.SPENSEC` buffer enable bits in this step.

4. Optionally program the `SPORT_CTL2_A` secondary configuration register.

   ADDITIONAL INFORMATION: This step is required only if internal multiplexing logic must be enabled to share clock and frame sync signals between a top SPORT module's A and B halves (see the Multiplexer Logic section for details).

5. Optionally program the `SPORT_ERR_A` error register.

   ADDITIONAL INFORMATION: This step is required only if a separate SPORT error interrupt is desired (see the Error Detection (Status) Interrupt section for details).

6. For Right-Justified mode only, program the `SPORT_MCTL_A.WOFFSET` field.

   ADDITIONAL INFORMATION: In Right-Justified mode, this field serves as the delay count (`DCNT`) required to align the LSB of each stereo channel with the L/R clock transition and must be programmed manually (see the Right-Justified Mode section for details).

7. Enable the primary/secondary datapath(s) in the `SPORT_CTL_A` register.

   ADDITIONAL INFORMATION: This should be performed in a read-modify-write operation setting the `SPORT_CTL_A.SPENPRI` and/or `SPORT_CTL_A.SPENSEC` bits, as appropriate.

8. Write data to be transmitted to the transmit buffer (SPORT_TXPRI_A and/or SPORT_TXSEC_A) *or* read data that has been received from the receive buffer (SPORT_RXPRI_A and/or SPORT_RXSEC_A).

   ADDITIONAL INFORMATION: These accesses are typically performed in the context of an interrupt service routine. See the SEC chapter for further information. Do not attempt to read or write inactive data buffers. If the core attempts to access inactive transmit or receive buffers while the SPORT is enabled, unpredictable results may occur.

Initializing Multichannel Transfers

When in Multichannel (TDM) Mode or Packed I^{2}S Mode, the SPORT is in a multichannel operational mode. Follow the steps below to properly initialize the SPORT for multichannel modes of operation. More steps are required to properly (see the SEC chapter for details).

NOTE: This example uses half SPORT A registers. With appropriate register changes, this example also applies to half SPORT B.

1. Clear the SPORT_CTL_A and SPORT_MCTL_A registers.

   ADDITIONAL INFORMATION: Clearing these registers ensures that the SPORT logic (including the multichannel logic) is fully reset before attempting to reprogram it.

2. Optionally program the SPORT_DIV_A clock divisor register.

   ADDITIONAL INFORMATION: This step is only required for internally-generated timing signals. Configure the serial bit clock and/or frame sync (or L/R clock, for stereo modes) rates according to the guidance in the Serial Clock and SPORT_CTL2_A sections.

3. Program the SPORT_CS0_A - SPORT_CS3_A channel select registers.

4. Program the SPORT_MCTL_A multichannel configuration register.

   ADDITIONAL INFORMATION: The SPORT supports many multichannel options. For more information, see the Multichannel (TDM) Mode section. Do not set the SPORT_MCTL_A.MCE enable bit in this step.

5. Program the SPORT_CTL_A primary configuration register.

   ADDITIONAL INFORMATION: Set the SPORT operating mode along with the configurable clock, frame sync, word length, direction, and data format options (see the Operating Modes and Options section for details). Do not set the SPORT_CTL_A.SPENPRI and/or SPORT_CTL_A.SPENSEC buffer enable bits in this step.

6. Optionally program the SPORT_CTL2_A secondary configuration register.

   ADDITIONAL INFORMATION: This step is required only if internal multiplexing logic must be enabled to share clock and frame sync signals between a top SPORT module's A and B halves (see the Multiplexer Logic section for details).

7. Optionally program the SPORT_ERR_A error register.

ADDITIONAL INFORMATION: This step is required only if a separate SPORT error interrupt request is desired (see the Error Detection (Status) Interrupt section for details).

8. Set the SPORT_MCTL_A.MCE bit to enable multichannel mode.
9. Enable the primary/secondary datapath(s) in the SPORT_CTL_A register.

   ADDITIONAL INFORMATION: This should be performed in a read-modify-write operation setting the SPORT_CTL_A.SPENPRI and/or SPORT_CTL_A.SPENSEC bits, as appropriate. DMA mode is recommended for multichannel modes of operation. For more information, see the Using DMA for SPORT Transfers programming model.

Using DMA for SPORT Transfers

DMA is supported in all SPORT operating modes (Standard DSP Serial Mode, I^2S Mode, Left-Justified Mode, Right-Justified Mode, Multichannel (TDM) Mode or Packed I^2S Mode). To enable DMA operation with the SPORT, execute the steps described in this section after initializing and enabling the SPORT. Instead of using the single word read or write operations described in the referenced programming models, the DMA engine automates accesses to the enabled SPORT data buffers.

NOTE: This example uses half SPORT A registers. With appropriate changes to register names, it also applies to half SPORT B.

1. Follow the guidance in the multichannel (Initializing Multichannel Transfers) or non-multichannel (Initializing Core-Driven (Non-MCM) Transfers) programming models to properly initialize and enable the SPORT hardware.
2. Prepare the data buffers in memory.

   ADDITIONAL INFORMATION: Ensure that the DMA buffer is defined according to the DMA Transfers section. For the multichannel modes of operation, be sure to also consider the setting of the SPORT_MCTL_A.MCPDE bit, as described in the Multichannel DMA Data Packing (MCPDE) section.
3. Initialize and enable the DMA channel allocated for the SPORT, as described in the Direct Memory Access (DMA) chapter.

Using Companding as a Function

The data in the transmit and receive buffers are companded in place. The following programming model can be used to exercise the companding hardware without transferring data, which is useful for test and debug purposes.

NOTE: This example uses half SPORT A registers. With appropriate changes to register names, this example also applies to half SPORT B.

1. Configure the SPORT as a transmitter (SPORT_CTL_A.SPTRAN=1) with both the primary and secondary data channels disabled (SPORT_CTL_A.SPENPRI=0 and SPORT_CTL_A.SPENSEC=0).

2. Enable the desired companding scheme in the SPORT_CTL_A.DTYPE field.
3. Write a 32-bit word to one of the transmit buffers.
4. Wait two system clock cycles.

   ADDITIONAL INFORMATION: This delay is required to allow the SPORT companding hardware to reload the transmit buffer with the companded result. Any instructions that do not access the transmit buffer can be used to cause this delay.
5. Read the 8-bit compressed value from the transmit buffer written above.

To expand data in place, use the same sequence of operations with the receive buffer instead of the transmit buffer. When expanding data in this way, set the appropriate serial word length (SPORT_CTL_A.SLEN).

Programming Global SPORT Groups

Complete the following steps to program global SPORT groups.

NOTE: To complete all of the SPORT group transfers simultaneously, all of the SPORTs in the group must have a similar data/clock/frame sync configuration.

1. For each SPORT, program the following registers for the desired SPORT mode of operation.
   - SPORT_DIV_A/SPORT_DIV_B
   - SPORT_CTL2_A/SPORT_CTL2_B
   - SPORT_CTL_A/SPORT_CTL_B
   - SPORT_MCTL_A/SPORT_MCTL_B
   - SPORT_CS0_A/SPORT_CS0_B
   - SPORT_CS1_ASPORT_CS1_B
   - SPORT_CS2_ASPORT_CS2_B
   - SPORT_CS3_ASPORT_CS3_B

   ADDITIONAL INFORMATION: The sport enable control bits (in SPORT_CTL2_A/SPORT_CTL2_B registers) must be cleared (disabled).
2. For a global group trigger as the controller, program each SPORT DMA in the group as the controller trigger.
3. Program the sport enable and group enable bits in the DAI_GBL_SP_EN register for the desired global sport grouping.

   ADDITIONAL INFORMATION: The grouping of SPORTs (enable and disable) and the associated interrupts/triggers are separate activities. If the interrupts and triggers must be grouped similar to the SPORTs (enable and disable) grouping, use the DAI_GBL_INT_EN register.

4. Program the DAI_GBL_INT_EN register to enable the grouped interrupt or trigger generation for the SPORT group.

   ADDITIONAL INFORMATION: To enable the SPORT grouped interrupt and trigger feature, each sport interrupt or trigger enable bit must be programmed with the respective group interrupt and trigger enable bits in the DAI_GBL_INT_EN register.

5. Program the DAI_GBL_SP_EN.GBL_SP_EN bit to enable the global sport groups at the same instance.

Disabling Global SPORT Groups

Complete the following steps to disable global SPORT groups.

1. Use the individual SPORT DMA status register to clear the group interrupt. There is no specific status register for global interrupts.
2. Disable the global SPORT group interrupt and trigger bits for the group in the DAI_GBL_INT_EN register.

   ADDITIONAL INFORMATION: To disable the SPORT grouped interrupt and trigger feature, each sport interrupt and trigger enable bit must be programmed with the respective group interrupt and trigger enable bits in the DAI_GBL_INT_EN register.

3. Disable the global individual SPORT enable and the global SPORT group enable bits in the DAI_GBL_SP_EN register.
4. Disable the DMAs for the SPORTs when the transfer is complete.

ADSP-SC59x SPORT Register Descriptions

Serial Port (SPORT) contains the following registers.

Table 32-14: ADSP-SC59x SPORT Register List

Name	Description
SPORT_CS0_A	Half SPORT 'A' Multichannel 0-31 Select Register
SPORT_CS0_B	Half SPORT 'B' Multichannel 0-31 Select Register
SPORT_CS1_A	Half SPORT 'A' Multichannel 32-63 Select Register
SPORT_CS1_B	Half SPORT 'B' Multichannel 32-63 Select Register
SPORT_CS2_A	Half SPORT 'A' Multichannel 64-95 Select Register
SPORT_CS2_B	Half SPORT 'B' Multichannel 64-95 Select Register
SPORT_CS3_A	Half SPORT 'A' Multichannel 96-127 Select Register
SPORT_CS3_B	Half SPORT 'B' Multichannel 96-127 Select Register
SPORT_CTL2_A	Half SPORT 'A' Control 2 Register
SPORT_CTL2_B	Half SPORT 'B' Control 2 Register

Table 32-14: ADSP-SC59x SPORT Register List (Continued)

Name	Description
SPORT_CTL_A	Half SPORT 'A' Control Register
SPORT_CTL_B	Half SPORT 'B' Control Register
SPORT_DIV_A	Half SPORT 'A' Divisor Register
SPORT_DIV_B	Half SPORT 'B' Divisor Register
SPORT_ERR_A	Half SPORT 'A' Error Register
SPORT_ERR_B	Half SPORT 'B' Error Register
SPORT_MCTL_A	Half SPORT 'A' Multichannel Control Register
SPORT_MCTL_B	Half SPORT 'B' Multichannel Control Register
SPORT_MSTAT_A	Half SPORT 'A' Multichannel Status Register
SPORT_MSTAT_B	Half SPORT 'B' Multichannel Status Register
SPORT_RXPRI_A	Half SPORT 'A' Rx Buffer (Primary) Register
SPORT_RXPRI_B	Half SPORT 'B' Rx Buffer (Primary) Register
SPORT_RXSEC_A	Half SPORT 'A' Rx Buffer (Secondary) Register
SPORT_RXSEC_B	Half SPORT 'B' Rx Buffer (Secondary) Register
SPORT_TXPRI_A	Half SPORT 'A' Tx Buffer (Primary) Register
SPORT_TXPRI_B	Half SPORT 'B' Tx Buffer (Primary) Register
SPORT_TXSEC_A	Half SPORT 'A' Tx Buffer (Secondary) Register
SPORT_TXSEC_B	Half SPORT 'B' Tx Buffer (Secondary) Register

Half SPORT 'A' Multichannel 0-31 Select Register

Each of the bits (when set, =1) of the SPORT_CS0_A register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

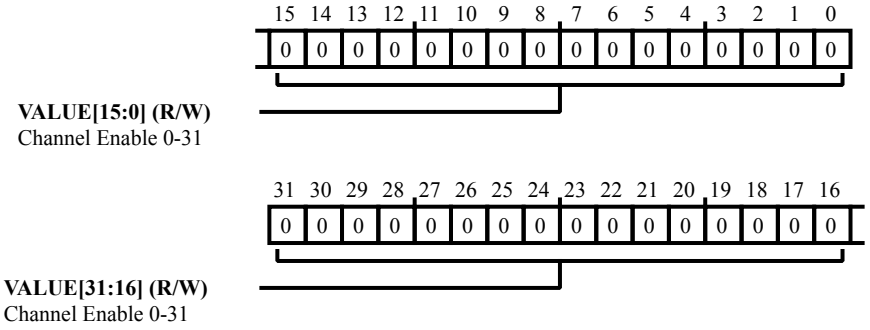


Figure 32-14: SPORT_CS0_A Register Diagram

Table 32-15: SPORT_CS0_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 0-31.

Half SPORT 'B' Multichannel 0-31 Select Register

Each of the bits (when set, =1) of the SPORT_CS0_B register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

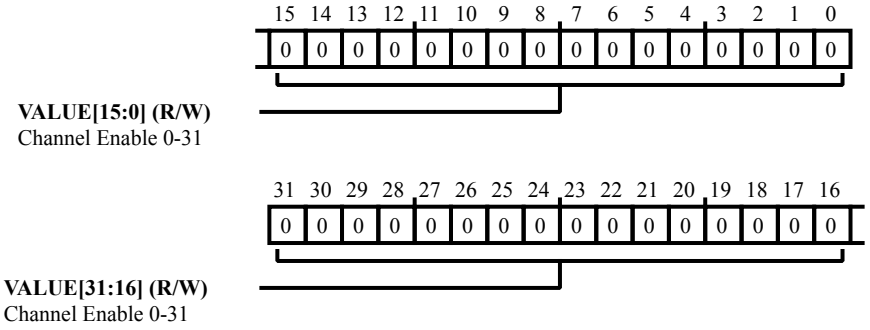


Figure 32-15: SPORT_CS0_B Register Diagram

Table 32-16: SPORT_CS0_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 0-31.

Half SPORT 'A' Multichannel 32-63 Select Register

Each of the bits (when set, =1) of the SPORT_CS1_A register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

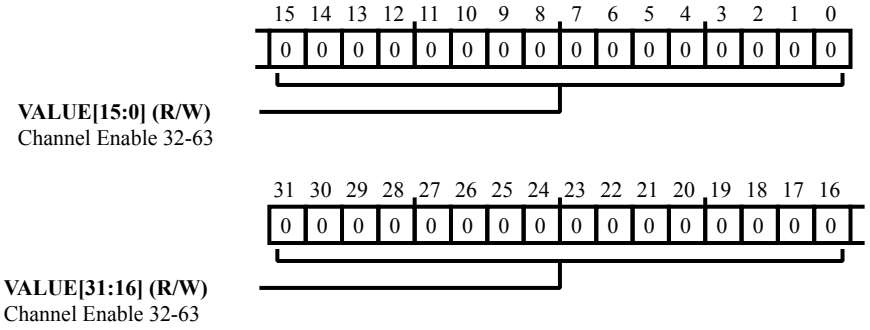


Figure 32-16: SPORT_CS1_A Register Diagram

Table 32-17: SPORT_CS1_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 32-63.

Half SPORT 'B' Multichannel 32-63 Select Register

Each of the bits (when set, =1) of the SPORT_CS1_B register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

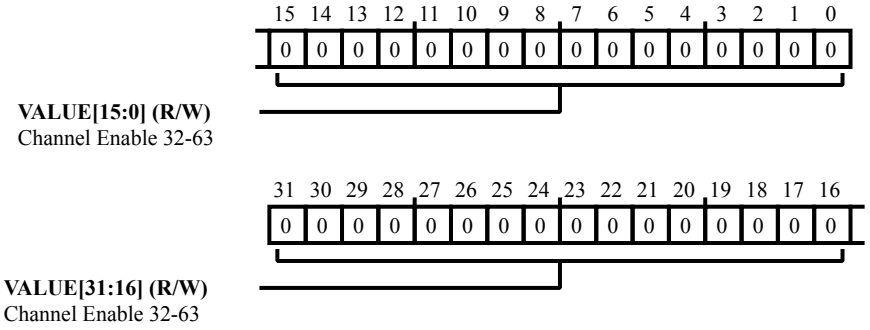


Figure 32-17: SPORT_CS1_B Register Diagram

Table 32-18: SPORT_CS1_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 32-63.

Half SPORT 'A' Multichannel 64-95 Select Register

Each of the bits (when set, =1) of the SPORT_CS2_A register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

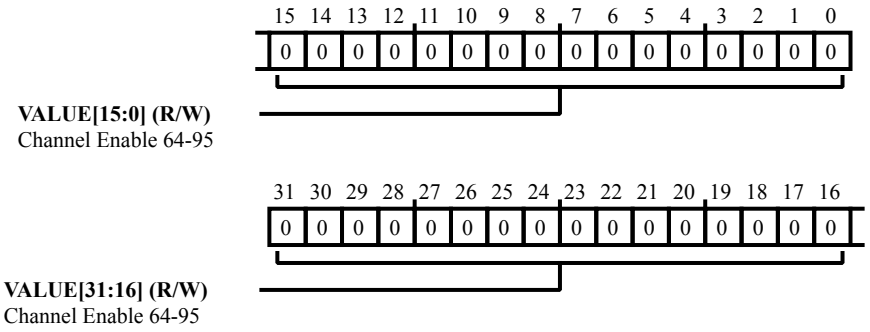


Figure 32-18: SPORT_CS2_A Register Diagram

Table 32-19: SPORT_CS2_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 64-95.

Half SPORT 'B' Multichannel 64-95 Select Register

Each of the bits (when set, =1) of the SPORT_CS2_B register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

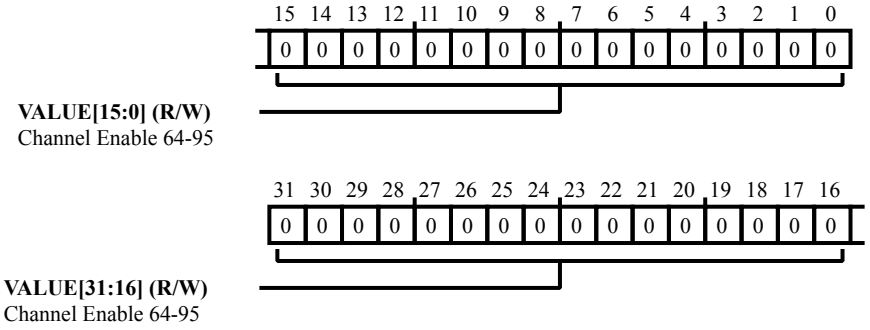


Figure 32-19: SPORT_CS2_B Register Diagram

Table 32-20: SPORT_CS2_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 64-95.

Half SPORT 'A' Multichannel 96-127 Select Register

Each of the bits (when set, =1) of the SPORT_CS3_A register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

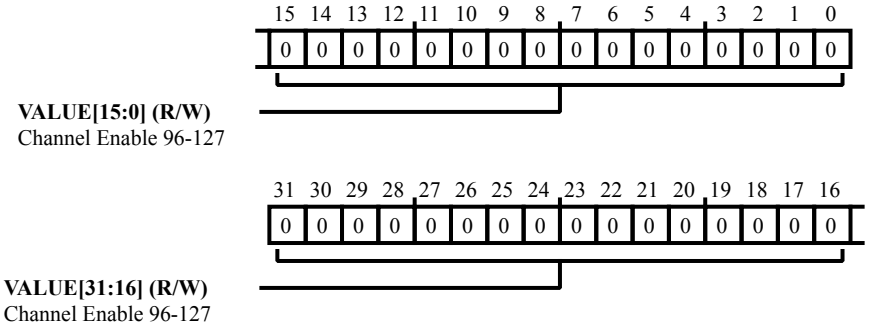


Figure 32-20: SPORT_CS3_A Register Diagram

Table 32-21: SPORT_CS3_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 96-127.

Half SPORT 'B' Multichannel 96-127 Select Register

Each of the bits (when set, =1) of the SPORT_CS3_B register correspond to an active channel for the half SPORT in multichannel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register deactivates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

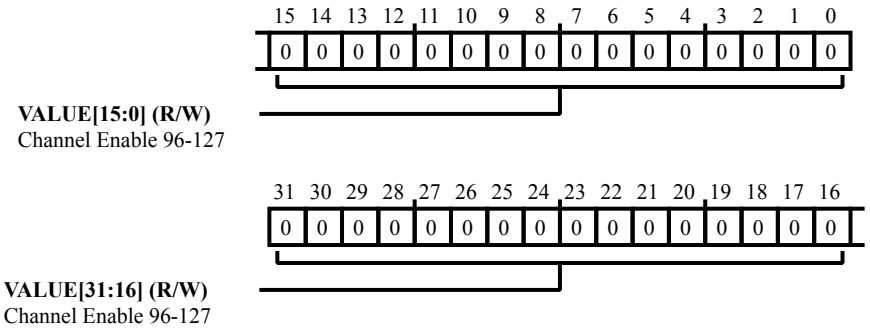


Figure 32-21: SPORT_CS3_B Register Diagram

Table 32-22: SPORT_CS3_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Channel Enable 96-127.

Half SPORT 'A' Control 2 Register

The SPORT_CTL2_A register controls multiplexing options for sharing serial clock and frame sync signals across the related half SPORTs.

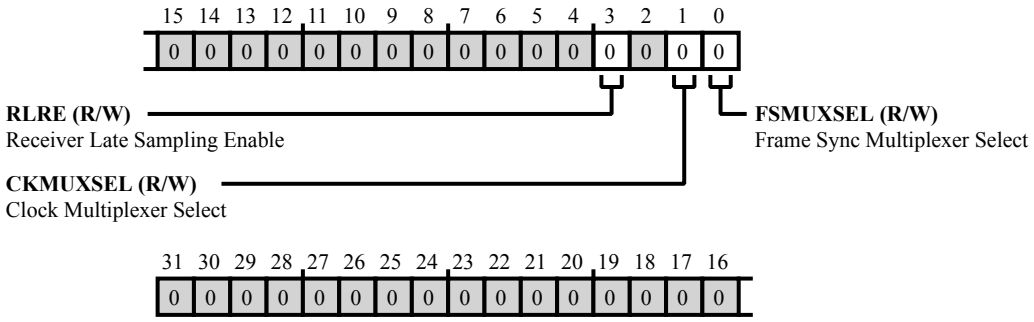


Figure 32-22: SPORT_CTL2_A Register Diagram

Table 32-23: SPORT_CTL2_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RLRE	Receiver Late Sampling Enable. The SPORT_CTL2_A.RLRE bit enables the half SPORT's late sampling of received data and frame sync by half of the operating clock cycle. Note: This mode of operation is applicable when Clock/FS are internally generated, and data is externally generated (i.e data and control are in opposite direction).	
1 (R/W)	CKMUXSEL	Clock Multiplexer Select. The SPORT_CTL2_A.CKMUXSEL bit enables multiplexing of the half SPORT's serial clock. In this mode, the serial clock of the related half SPORT is used instead of the half SPORT's own serial clock. For example, if the SPORT_CTL2_A.CKMUXSEL bit is enabled, half SPORT 'A' uses SPORT_BCLK instead of SPORT_ACLK.	
		0	Disable serial clock multiplexing
		1	Enable serial clock multiplexing
0 (R/W)	FSMUXSEL	Frame Sync Multiplexer Select. The SPORT_CTL2_A.FSMUXSEL bit enables multiplexing of the half SPORT's frame sync. In this mode, the frame sync of the related half SPORT is used instead of the half SPORT's own frame sync. For example, if the SPORT_CTL2_A.FSMUXSEL bit is enabled, half SPORT 'A' uses SPORT_BFS instead of SPORT_AFS.	
		0	Disable frame sync multiplexing
		1	Enable frame sync multiplexing

Half SPORT 'B' Control 2 Register

The SPORT_CTL2_B register controls multiplexing options for sharing serial clock and frame sync signals across the related half SPORTs.

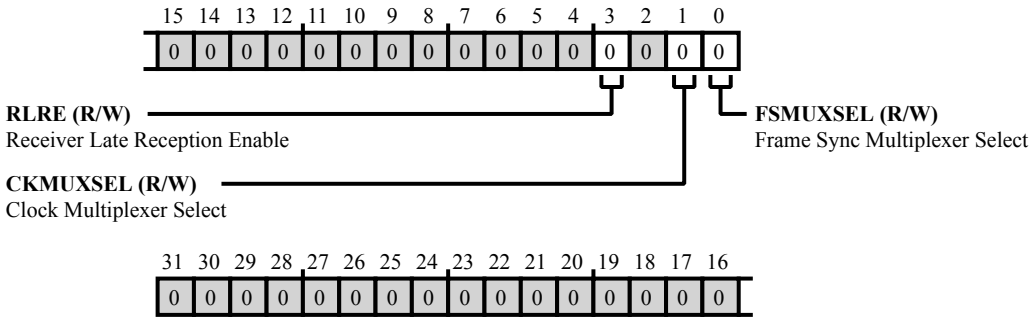


Figure 32-23: SPORT_CTL2_B Register Diagram

Table 32-24: SPORT_CTL2_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	RLRE	Receiver Late Reception Enable. The SPORT_CTL2_B.RLRE bit enables the half SPORT's late sampling of received data and frame sync by half of the operating clock cycle.	
1 (R/W)	CKMUXSEL	Clock Multiplexer Select. The SPORT_CTL2_B.CKMUXSEL bit enables multiplexing of the half SPORT's serial clock. In this mode, the serial clock of the related half SPORT is used instead of the half SPORT's own serial clock. For example, if the SPORT_CTL2_B.CKMUXSEL bit is enabled, half SPORT 'B' uses SPORT_ACLK instead of SPORT_BCLK.	
		0	Disable serial clock multiplexing
		1	Enable serial clock multiplexing
0 (R/W)	FSMUXSEL	Frame Sync Multiplexer Select. The SPORT_CTL2_B.FSMUXSEL bit enables multiplexing of the half SPORT's frame sync. In this mode, the frame sync of the related half SPORT is used instead of the half SPORT's own frame sync. For example, if the SPORT_CTL2_B.FSMUXSEL bit is enabled, half SPORT 'B' uses SPORT_AFS instead of SPORT_BFS.	
		0	Disable frame sync multiplexing
		1	Enable frame sync multiplexing

Half SPORT 'A' Control Register

The SPORT_CTL_A register contains transmit and receive control bits for SPORT half 'A', including serial port mode selection for the half SPORT's primary and secondary channels. The function of some bits in the SPORT_CTL_A register vary depending on the SPORT's operating mode. For more information, see the SPORT operating modes description. If reading reserved bits, the read value is the last written value to these bits or is the reset value of these bits.

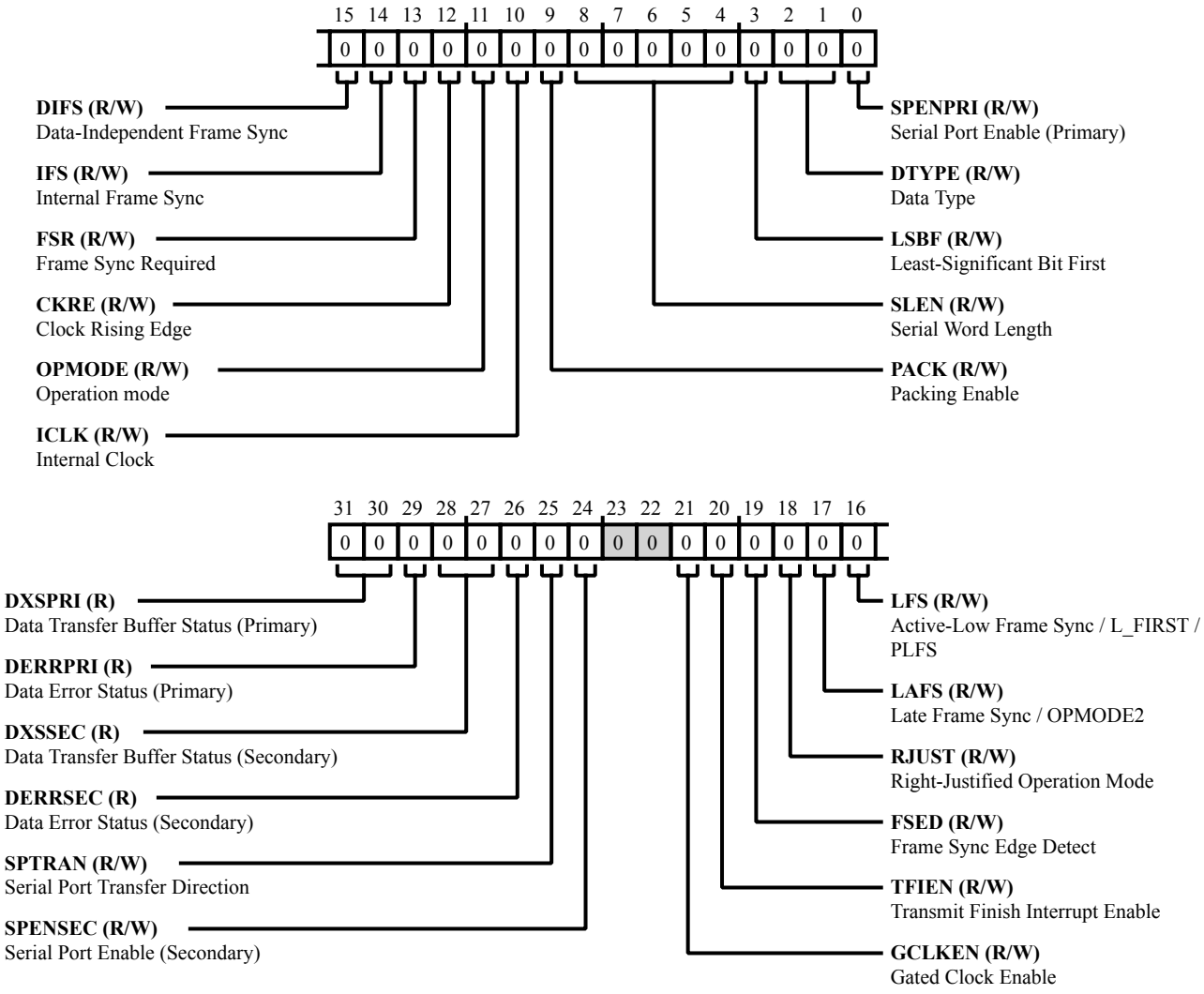


Figure 32-24: SPORT_CTL_A Register Diagram

Table 32-25: SPORT_CTL_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	DXSPRI	Data Transfer Buffer Status (Primary). The SPORT_CTL_A.DXSPRI bit field indicates the status of the half SPORT's primary channel data buffer.	
		0	Empty
		1	Reserved
		2	Partially full
		3	Full
29 (R/NW)	DERRPRI	Data Error Status (Primary). The SPORT_CTL_A.DERRPRI bit reports the half SPORT's primary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction. If the SPORT_CTL_A.FSR bit =1, the SPORT_CTL_A.DERRPRI bit indicates whether the SPORT_AFS signal (from an internal or external source) occurred while the SPORT_TXPRI_A data buffer was empty (during transmit) or the SPORT_RXPRI_A data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_AFS signal. It is important to note that, as a receiver, the SPORT_CTL_A.DERRPRI bit indicates when the channel has received new data while the SPORT_RXPRI_A receive buffer is full. This new data overwrites existing data. If the SPORT_CTL_A.FSR bit =0, the SPORT_CTL_A.DERRPRI bit is set whenever the SPORT is required to transmit while the SPORT_TXPRI_A transmit buffer is empty. It is also set whenever the SPORT is required to receive while the SPORT_RXPRI_A receive buffer is full. The SPORT clears the SPORT_CTL_A.DERRPRI bit if the SPORT_ERR_A.DERRPSTAT bit is cleared.	
		0	No error
		1	Error (Tx underflow or Rx overflow)
28:27 (R/NW)	DXSSEC	Data Transfer Buffer Status (Secondary). The SPORT_CTL_A.DXSSEC bit field indicates the status of the half SPORT's secondary channel data buffer.	
		0	Empty
		1	Reserved
		2	Partially full
		3	Full

Table 32-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/NW)	DERRSEC	Data Error Status (Secondary). The SPORT_CTL_A.DERRSEC bit reports the half SPORT's secondary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction. If the SPORT_CTL_A.FSR bit =1, the SPORT_CTL_A.DERRSEC bit indicates whether the SPORT_AFS signal (from an internal or external source) occurred while the SPORT_TXSEC_A data buffer was empty (during transmit) or the SPORT_RXSEC_A data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_AFS signal. It is important to note that, as a receiver, the SPORT_CTL_A.DERRSEC bit indicates when the channel has received new data while the SPORT_RXSEC_A receive buffer is full. This new data overwrites existing data. If the SPORT_CTL_A.FSR bit =0, the SPORT_CTL_A.DERRSEC bit is set whenever the SPORT is required to transmit while the SPORT_TXSEC_A transmit buffer is empty. It is also set whenever the SPORT is required to receive while the SPORT_RXSEC_A receive buffer is full. The SPORT clears the SPORT_CTL_A.DERRSEC bit if the SPORT_ERR_A.DERRSSTAT bit is cleared.	
		0	No error
		1	Error (Tx underflow or Rx overflow)
25 (R/W)	SPTRAN	Serial Port Transfer Direction. The SPORT_CTL_A.SPTRAN bit selects the transfer direction (receive or transmit) for the half SPORT's primary and secondary channels. When the direction is receive, the half SPORT activates the receive buffers, and the SPORT_ACLK and SPORT_AFS pins control the receive buffers. The transmit buffers are inactive when the half SPORT's transfer direction is receive. When the direction is transmit, the half SPORT activates the transmit buffers, and the SPORT_ACLK and SPORT_AFS pins control the transmit shift registers. The receive buffers are inactive when the half SPORT's transfer direction is transmit.	
		0	Receive
		1	Transmit
24 (R/W)	SPENSEC	Serial Port Enable (Secondary). The SPORT_CTL_A.SPENSEC bit enables the half SPORT's secondary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers.	
		0	Disable
		1	Enable

Table 32-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	GCLKEN	Gated Clock Enable. The SPORT_CTL_A.GCLKEN bit enables gated clock operation for the half SPORT when in DSP serial mode or left-justified stereo modes (SPORT_CTL_A.OPMODE = 0 or 1). This bit is ignored when the half SPORT is in right-justified mode (SPORT_CTL_A.RJUST =1) or multichannel mode (SPORT_MCTL_A.MCE =1). When the SPORT_CTL_A.GCLKEN bit is enabled, the SPORT clock is active when the SPORT is transferring data or when the frame sync changes (transitions to active state).	
		0	Disable
		1	Enable
20 (R/W)	TFIEN	Transmit Finish Interrupt Enable. The SPORT_CTL_A.TFIEN bit selects when the half SPORT issues its transmission complete interrupt request, if a DMA complete interrupt request is enabled by the DMA_CFG.INT configuration. When enabled (SPORT_CTL_A.TFIEN =1), the DMA complete peripheral interrupt request is generated when the last bit of last word in the DMA is shifted out. When disabled (SPORT_CTL_A.TFIEN =0), the DMA interrupt request is generated when the DMA counter expires (the last word goes to the transmit buffer).	
		0	Last word sent (DMA count done) interrupt
		1	Last bit sent (Tx buffer done) interrupt
19 (R/W)	FSED	Frame Sync Edge Detect. The SPORT_CTL_A.FSED bit enables the half SPORT to start transmitting or receiving after detecting an active edge of an external frame sync. The SPORT_CTL_A.FSED bit may be enabled even during an active frame sync, and the half SPORT starts the transfer on the next valid rising or falling edge of external frame sync. If disabled (SPORT_CTL_A.FSED =0), the half SPORT operates in the standard level-sensitive detection mode for external frame sync.	
		0	Level detect frame sync
		1	Edge detect frame sync

Table 32-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	RJUST	Right-Justified Operation Mode. The SPORT_CTL_A.RJUST bit enables the half SPORT (if SPORT_CTL_A.OPMODE =1) to transfer data in right-justified operation mode. In this mode, the half SPORT aligns data to the end of the frame sync, rather than the start of the frame sync. When using right-justified mode, systems should program an appropriate delay count to introduce a clock delay before the half SPORT state machine starts to capture data. This value is set in the DCNT field (right-justified mode usage of the SPORT_MCTL_A.WOFFSET field). For information about appropriate delay selections, see the SPORT operating modes section.	
		0	Disable
		1	Enable
17 (R/W)	LAFS	Late Frame Sync / OPMODE2. When the half SPORT is in DSP standard mode (SPORT_CTL_A.OPMODE =0) or in right-justified mode (SPORT_CTL_A.RJUST =1), the SPORT_CTL_A.LAFS bit selects whether the half SPORT generates a late frame sync (SPORT_AFS during first data bit) or generates an early frame sync signal (SPORT_AFS during serial clock cycle before first data bit). When the half SPORT is in I^2S / left-justified mode (SPORT_CTL_A.OPMODE =1), the SPORT_CTL_A.LAFS bit acts as OPMODE2, selecting whether the half SPORT is in left-justified mode or I^2S mode. When the half SPORT is in multichannel mode (SPORT_MCTL_A.MCE =1), the SPORT_CTL_A.LAFS bit is reserved.	
		0	Early frame sync (or I^2S mode)
		1	Late frame sync (or left-justified mode)
16 (R/W)	LFS	Active-Low Frame Sync / L_FIRST / PLFS. When the half SPORT is in DSP standard mode and multichannel mode (SPORT_CTL_A.OPMODE =0), the SPORT_CTL_A.LFS bit selects whether the half SPORT uses active low or active high frame sync. When the half SPORT is in I^2S / packed / left-justified mode (SPORT_CTL_A.OPMODE =1), the SPORT_CTL_A.LFS bit acts as L_FIRST, selecting whether the half SPORT transfers data first for the left or right channel.	
		0	Active high frame sync (DSP standard mode) or rising edge frame sync (multichannel mode) or right channel first (I^2S/packed mode) or left channel first (left-justified mode)
		1	Active low frame sync (DSP standard mode) or falling edge frame sync (multichannel mode) or left channel first (I^2S/packed mode) or right channel first (left-justified mode)

Table 32-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	DIFS	Data-Independent Frame Sync. The SPORT_CTL_A.DIFS bit selects whether the half SPORT uses a data-independent or data-dependent frame sync. When using a data-independent frame sync, the half SPORT generates the sync at the interval selected by the SPORT_DIV_A.FSDIV bit. When using a data-dependent frame sync, the half SPORT generates the sync on the selected interval when the transmit buffer is not empty or when the receive buffer is not full. Note that the SPORT_CTL_A.DIFS bit is automatically set when the half SPORT is in packed or multichannel modes.	
		0	Data-dependent frame sync
		1	Data-independent frame sync
14 (R/W)	IFS	Internal Frame Sync. The SPORT_CTL_A.IFS bit selects whether the half SPORT uses an internal frame sync or uses an external frame sync. Note that the externally-generated frame sync does not need to be synchronous with the processor's system clock.	
		0	External frame sync
		1	Internal frame sync
13 (R/W)	FSR	Frame Sync Required. The SPORT_CTL_A.FSR bit selects whether or not the half SPORT requires frame sync for data transfer. This bit is automatically set when the half SPORT is in I^2S / packed / left-justified mode (SPORT_CTL_A.OPMODE =1) or is in multichannel mode (SPORT_MCTL_A.MCE =1).	
		0	No frame sync required
		1	Frame sync required
12 (R/W)	CKRE	Clock Rising Edge. The SPORT_CTL_A.CKRE bit selects the rising or falling edge of the SPORT_ACLK clock for the half SPORT to sample receive data and frame sync. Note that the half SPORT changes the state of transmit data and frame sync signals on the non-selected edge of the SPORT_ACLK. Also, note that the transmit and receive related SPORT halves (A and B) should be programmed with the same value for the SPORT_CTL_A.CKRE bit. This programming drives the internally-generated signals on one edge of SPORT_ACLK and samples the received signals on the opposite edge.	
		0	Clock falling edge
		1	Clock rising edge

Table 32-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	OPMODE	Operation mode. The SPORT_CTL_A.OPMODE bit selects whether the half SPORT operates in DSP standard/multichannel mode or operates in I^2S/packed/left-justified mode. The mode selection affects the operation of the SPORT_CTL_A.LAFS and SPORT_CTL_A.LFS bits. Also, the SPORT_CTL_A.OPMODE bit enables or disables operation of the SPORT_CTL_A.GCLKEN, SPORT_CTL_A.FSED, SPORT_CTL_A.RJUST, SPORT_CTL_A.DIFS, SPORT_CTL_A.FSR, and SPORT_CTL_A.CKRE bits.	
		0	DSP standard/multichannel mode
		1	I^2S/packed/left-justified mode
10 (R/W)	ICLK	Internal Clock. When the half SPORT is in DSP standard mode (SPORT_CTL_A.OPMODE =0), the SPORT_CTL_A.ICLK bit selects whether the half SPORT uses an internal or external clock. For internal clock enabled, the half SPORT generates the SPORT_ACLK clock signal, and SPORT_ACLK is an output. The SPORT_DIV_A.CLKDIV serial clock divisor value determines the clock frequency. For internal clock disabled, the SPORT_ACLK clock signal is an input, and the serial clock divisor is ignored. Note that the externally-generated serial clock does not need to be synchronous with the processor's system clock.	
		0	External clock
		1	Internal clock
9 (R/W)	PACK	Packing Enable. The SPORT_CTL_A.PACK bit enables the half SPORT to perform 16- to 32-bit packing on received data and to perform 32- to 16-bit unpacking on transmitted data. The receive packing operation packs two successive received words into a single 32-bit word. The transmit unpacking operation unpacks each 32-bit word and transmits it as two 16-bit words. The first 16-bit (or smaller) word is right-justified in bits 15:0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31:16. This format applies to both receive (packing) and transmit (unpacking) operations. Companding may be used with word packing or unpacking. The half SPORT generates data transfer related interrupts when packing is enabled. The transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word.	
		0	Disable
		1	Enable

Table 32-25: SPORT_CTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8:4 (R/W)	SLEN	Serial Word Length. The SPORT_CTL_A.SLEN bits selects word length in bits for the half SPORT's data transfers. Word may be from 4- to 32-bits in length. The formula for selecting the word length in bits is: SPORT_CTL_A.SLEN = (serial word length in bits) - 1 For DSP standard mode (SPORT_CTL_A.OPMODE =0), use SPORT_CTL_A.SLEN of 3 to 31 bits. For I^2S / packed / left-justified mode (SPORT_CTL_A.OPMODE =1), use SPORT_CTL_A.SLEN of 4 to 31 bits.	
3 (R/W)	LSBF	Least-Significant Bit First. The SPORT_CTL_A.LSBF bit selects whether the half SPORT transmits or receives data LSB first or MSB first.	
		0	MSB first sent/received (big endian)
		1	LSB first sent/received (little endian)
2:1 (R/W)	DTYPE	Data Type. The SPORT_CTL_A.DTYPE bits selects the data type formatting for the half SPORT's data transfers in DSP standard mode (SPORT_CTL_A.OPMODE =0).	
		0	Right-justify data, zero-fill unused MSBs
		1	Right-justify data, sign-extend unused MSBs
		2	u-law compand data
		3	A-law compand data
0 (R/W)	SPENPRI	Serial Port Enable (Primary). The SPORT_CTL_A.SPENPRI bit enables the half SPORT's primary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers.	
		0	Disable
		1	Enable

Half SPORT 'B' Control Register

The SPORT_CTL_B register contains transmit and receive control bits for SPORT half 'B', including serial port mode selection for the half SPORT's primary and secondary channels. The function of some bits in the SPORT_CTL_B register vary, depending on the SPORT's operating mode. For more information, see the SPORT operating modes description. If reading reserved bits, the read value is the last written value to these bits or is the reset value of these bits.

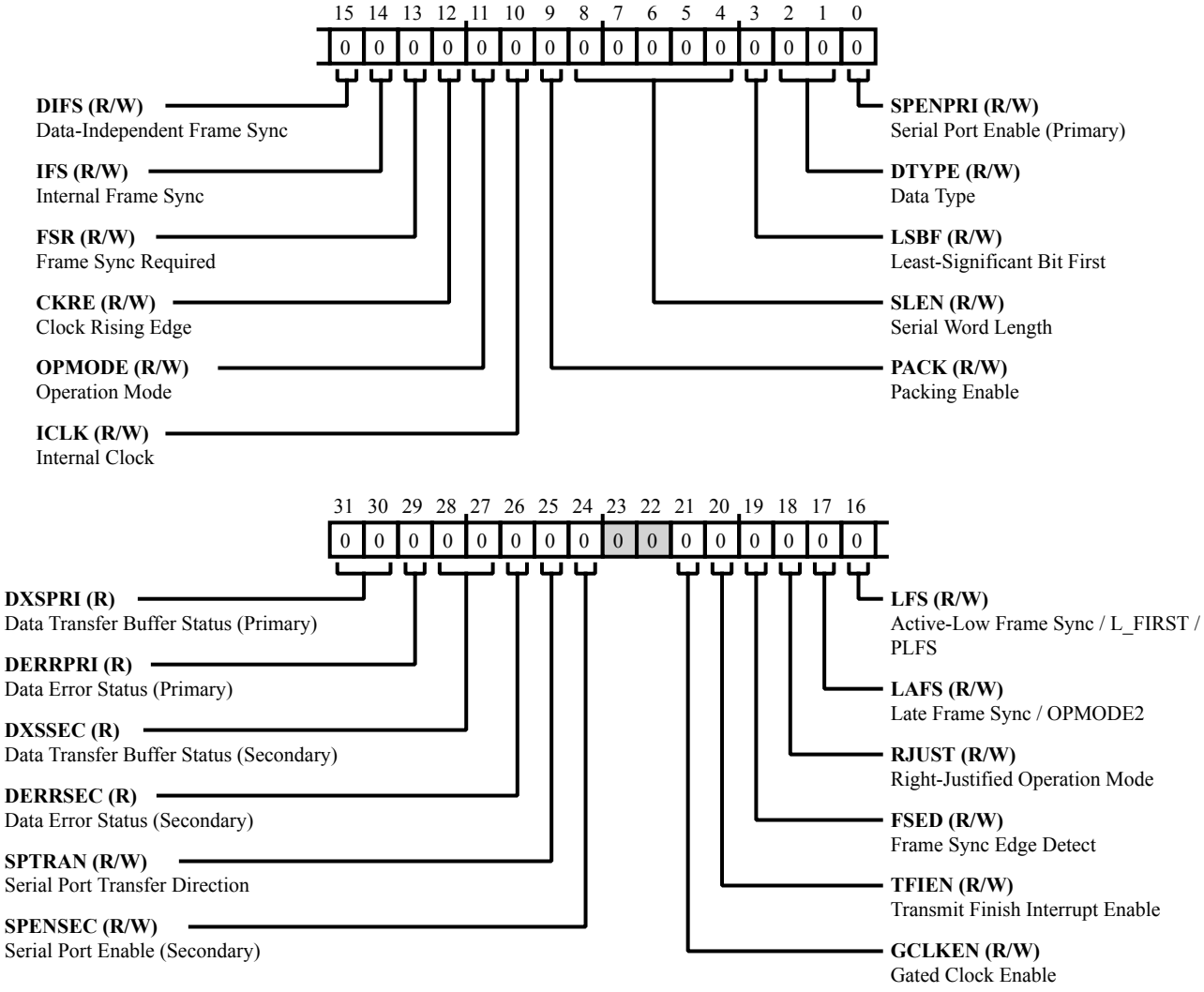


Figure 32-25: SPORT_CTL_B Register Diagram

Table 32-26: SPORT_CTL_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:30 (R/NW)	DXSPRI	Data Transfer Buffer Status (Primary). The SPORT_CTL_B.DXSPRI bit field indicates the status of the half SPORT's primary channel data buffer.	
		0	Empty
		1	Reserved
		2	Partially full
		3	Full
29 (R/NW)	DERRPRI	Data Error Status (Primary). The SPORT_CTL_B.DERRPRI bit reports the half SPORT's primary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction. If the SPORT_CTL_B.FSR bit =1, the SPORT_CTL_B.DERRPRI bit indicates whether the SPORT_BFS signal (from an internal or external source) occurred while the SPORT_TXPRI_B data buffer was empty (during transmit) or the SPORT_RXPRI_B data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_BFS signal. It is important to note that, as a receiver, the SPORT_CTL_B.DERRPRI bit indicates when the channel has received new data while the SPORT_RXPRI_B receive buffer is full. This new data overwrites existing data. If the SPORT_CTL_B.FSR bit =0, the SPORT_CTL_B.DERRPRI bit is set whenever the SPORT is required to transmit while the SPORT_TXPRI_B transmit buffer is empty and is set whenever the SPORT is required to receive while the SPORT_RXPRI_B receive buffer is full. The SPORT clears the SPORT_CTL_B.DERRPRI bit if the SPORT_ERR_B.DERRPSTAT bit is cleared.	
		0	No error
		1	Error (Tx underflow or Rx overflow)
28:27 (R/NW)	DXSSEC	Data Transfer Buffer Status (Secondary). The SPORT_CTL_B.DXSSEC bit field indicates the status of the half SPORT's secondary channel data buffer.	
		0	Empty
		1	Reserved
		2	Partially full
		3	Full

Table 32-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/NW)	DERRSEC	Data Error Status (Secondary). The SPORT_CTL_B.DERRSEC bit reports the half SPORT's secondary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction. If the SPORT_CTL_B.FSR bit =1, the SPORT_CTL_B.DERRSEC bit indicates whether the SPORT_BFS signal (from an internal or external source) occurred while the SPORT_TXSEC_B data buffer was empty (during transmit) or the SPORT_RXSEC_B data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_BFS signal. It is important to note that, as a receiver, the SPORT_CTL_B.DERRSEC bit indicates when the channel has received new data while the SPORT_RXSEC_B receive buffer is full. This new data overwrites existing data. If the SPORT_CTL_B.FSR bit =0, the SPORT_CTL_B.DERRSEC bit is set whenever the SPORT is required to transmit while the SPORT_TXSEC_B transmit buffer is empty. It is also set whenever the SPORT is required to receive while the SPORT_RXSEC_B receive buffer is full. The SPORT clears the SPORT_CTL_B.DERRSEC bit if the SPORT_ERR_B.DERRSSTAT bit is cleared.	
		0	No error
		1	Error (Tx underflow or Rx overflow)
25 (R/W)	SPTRAN	Serial Port Transfer Direction. The SPORT_CTL_B.SPTRAN bit selects the transfer direction (receive or transmit) for the half SPORT's primary and secondary channels. When the direction is receive, the half SPORT activates the receive buffers, and the SPORT_BCLK and SPORT_BFS pins control the receive buffers. The transmit buffers are inactive when the half SPORT's transfer direction is receive. When the direction is transmit, the half SPORT activates the transmit buffers, and the SPORT_BCLK and SPORT_BFS pins control the transmit shift registers. The receive buffers are inactive when the half SPORT's transfer direction is transmit.	
		0	Receive
		1	Transmit
24 (R/W)	SPENSEC	Serial Port Enable (Secondary). The SPORT_CTL_B.SPENSEC bit enables the half SPORT's secondary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers.	
		0	Disable
		1	Enable

Table 32-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21 (R/W)	GCLKEN	Gated Clock Enable. The SPORT_CTL_B.GCLKEN bit enables gated clock operation for the half SPORT when in DSP serial mode or left-justified stereo modes (SPORT_CTL_B.OPMODE = 0 or 1). This bit is ignored when the half SPORT is in right-justified mode (SPORT_CTL_B.RJUST =1) or multichannel mode (SPORT_MCTL_B.MCE =1). When SPORT_CTL_B.GCLKEN is enabled, the SPORT clock is active when the SPORT is transferring data or when the frame sync changes (transitions to active state).	
		0	Disable
		1	Enable
20 (R/W)	TFIEN	Transmit Finish Interrupt Enable. The SPORT_CTL_B.TFIEN bit selects when the half SPORT issues its transmission complete interrupt request, if a DMA complete interrupt request is enabled by the DMA_CFG.INT configuration. When enabled (SPORT_CTL_B.TFIEN =1), the DMA complete peripheral interrupt request is generated when the last bit of last word in the DMA is shifted out. When disabled (SPORT_CTL_B.TFIEN =0), the DMA interrupt request is generated when the DMA counter expires (the last word goes to the transmit buffer).	
		0	Last word sent (DMA count done) interrupt
		1	Last bit sent (Tx buffer done) interrupt
19 (R/W)	FSED	Frame Sync Edge Detect. The SPORT_CTL_B.FSED bit enables the half SPORT to start transmitting or receiving after detecting an active edge of an external frame sync. The SPORT_CTL_B.FSED may be enabled even during an active frame sync, and the half SPORT starts the transfer on the next valid rising or falling edge of external frame sync. If disabled (SPORT_CTL_B.FSED =0), the half SPORT operates in the standard level-sensitive detection mode for external frame sync.	
		0	Level detect frame sync
		1	Edge detect frame sync

Table 32-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	RJUST	Right-Justified Operation Mode. The SPORT_CTL_B.RJUST bit enables the half SPORT (if SPORT_CTL_B.OPMODE =1) to transfer data in right-justified operation mode. In this mode, the half SPORT aligns data to the end of the frame sync, rather than the start of the frame sync. When using right-justified mode, systems should program an appropriate delay count to introduce a clock delay before the half SPORT state machine starts to capture data. This value is set in the DCNT field (right-justified mode usage of the SPORT_MCTL_B.WOFFSET field). For information about appropriate delay selections, see the SPORT operating modes section.	
		0	Disable
		1	Enable
17 (R/W)	LAFS	Late Frame Sync / OPMODE2. When the half SPORT is in DSP standard mode (SPORT_CTL_B.OPMODE =0) or in right-justified mode (SPORT_CTL_B.RJUST =1), the SPORT_CTL_B.LAFS bit selects whether the half SPORT generates a late frame sync (SPORT_BFS during first data bit) or generates an early frame sync signal (SPORT_BFS during serial clock cycle before first data bit). When the half SPORT is in I^2S / left-justified mode (SPORT_CTL_B.OPMODE =1), the SPORT_CTL_B.LAFS bit acts as OPMODE2, selecting whether the half SPORT is in left-justified mode or I^2S mode. When the half SPORT is in multichannel mode (SPORT_MCTL_B.MCE =1), the SPORT_CTL_B.LAFS bit is reserved.	
		0	Early frame sync (or I^2S mode)
		1	Late frame sync (or left-justified mode)

Table 32-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	LFS	Active-Low Frame Sync / L_FIRST / PLFS. When the half SPORT is in DSP standard mode and multichannel mode (SPORT_CTL_B.OPMODE =0), the SPORT_CTL_B.LFS bit selects whether the half SPORT uses active low or active high frame sync. When the half SPORT is in I^2S / packed / left-justified mode (SPORT_CTL_B.OPMODE =1), the SPORT_CTL_B.LFS bit acts as L_FIRST, selecting whether the half SPORT transfers data first for the left or right channel.	
		0	Active high frame sync (DSP standard mode) or rising edge frame sync (multichannel mode) or right channel first (I^2S/packed mode) or left channel first (left-justified mode)
		1	Active low frame sync (DSP standard mode) or falling edge frame sync (multichannel mode) or left channel first (I^2S/packed mode) or right channel first (left-justified mode)
15 (R/W)	DIFS	Data-Independent Frame Sync. The SPORT_CTL_B.DIFS bit selects whether the half SPORT uses a data-independent or data-dependent frame sync. When using a data-independent frame sync, the half SPORT generates the sync at the interval selected by SPORT_DIV_B.FSDIV. When using a data-dependent frame sync, the half SPORT generates the sync on the selected interval when the transmit buffer is not empty or when the receive buffer is not full. Note that the SPORT_CTL_B.DIFS bit is automatically set when the half SPORT is in packed or multichannel modes.	
		0	Data-dependent frame sync
		1	Data-independent frame sync
14 (R/W)	IFS	Internal Frame Sync. The SPORT_CTL_B.IFS bit selects whether the half SPORT uses an internal frame sync or uses an external frame sync. Note that the externally-generated frame sync does not need to be synchronous with the processor's system clock.	
		0	External frame sync
		1	Internal frame sync

Table 32-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	FSR	Frame Sync Required. The SPORT_CTL_B.FSR selects whether or not the half SPORT requires frame sync for data transfer. This bit is automatically set when the half SPORT is in I^2S / packed / left-justified mode (SPORT_CTL_B.OPMODE =1) or is in multichannel mode (SPORT_MCTL_B.MCE =1).	
		0	No frame sync required
		1	Frame sync required
12 (R/W)	CKRE	Clock Rising Edge. The SPORT_CTL_B.CKRE selects the rising or falling edge of the SPORT_BCLK clock for the half SPORT to sample receive data and frame sync. Note that the half SPORT changes the state of transmit data and frame sync signals on the non-selected edge of the SPORT_BCLK. Also note that the transmit and receive related SPORT halves (A and B) should be programmed with the same value for SPORT_CTL_B.CKRE. This programming drives the internally-generated signals on one edge of SPORT_BCLK and samples the received signals on the opposite edge.	
		0	Clock falling edge
		1	Clock rising edge
11 (R/W)	OPMODE	Operation Mode. The SPORT_CTL_B.OPMODE bit selects whether the half SPORT operates in DSP standard / multichannel mode or operates in I^2S / packed / left-justified mode. The mode selection affects the operation of the SPORT_CTL_B.LAFS and SPORT_CTL_B.LFS bits. Also, the SPORT_CTL_B.OPMODE bit enables or disables operation of the SPORT_CTL_B.GCLKEN, SPORT_CTL_B.FSED, SPORT_CTL_B.RJUST, SPORT_CTL_B.DIFS, SPORT_CTL_B.FSR, and SPORT_CTL_B.CKRE bits.	
		0	DSP standard/multichannel mode
		1	I^2S/packed/left-justified mode
10 (R/W)	ICLK	Internal Clock. When the half SPORT is in DSP standard mode (SPORT_CTL_B.OPMODE =0), the SPORT_CTL_B.ICLK bit selects whether the half SPORT uses an internal or external clock. For internal clock enabled, the half SPORT generates the SPORT_BCLK clock signal, and the SPORT_BCLK is an output. The SPORT_DIV_B.CLKDIV serial clock divisor value determines the clock frequency. For internal clock disabled, the SPORT_BCLK clock signal is an input, and the serial clock divisor is ignored. Note that the externally-generated serial clock does not need to be synchronous with the processor's system clock.	
		0	External clock
		1	Internal clock

Table 32-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9 (R/W)	PACK	Packing Enable. The SPORT_CTL_B.PACK bit enables the half SPORT to perform 16- to 32-bit packing on received data and to perform 32- to 16-bit unpacking on transmitted data. The receive packing operation packs two successive received words into a single 32-bit word. The transmit unpacking operation unpacks each 32-bit word and transmits it as two 16-bit words. The first 16-bit (or smaller) word is right-justified in bits 15:0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31:16. This format applies to both receive (packing) and transmit (unpacking) operations. Companding may be used with word packing or unpacking. The half SPORT generates data transfer related interrupts when packing is enabled. The transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word.	
		0	Disable
		1	Enable
8:4 (R/W)	SLEN	Serial Word Length. The SPORT_CTL_B.SLEN bits selects word length in bits for the half SPORT's data transfers. Word may be from 4- to 32-bits in length. The formula for selecting the word length in bits is: SPORT_CTL_B.SLEN = (serial word length in bits) - 1 For DSP standard mode (SPORT_CTL_B.OPMODE =0), use SPORT_CTL_B.SLEN of 3 to 31 bits. For I^2S / packed / left-justified mode (SPORT_CTL_B.OPMODE =1), use SPORT_CTL_B.SLEN of 4 to 31 bits.	
3 (R/W)	LSBF	Least-Significant Bit First. The SPORT_CTL_B.LSBF bit selects whether the half SPORT transmits or receives data LSB first or MSB first.	
		0	MSB first sent/received (big endian)
		1	LSB first sent/received (little endian)
2:1 (R/W)	DTYPE	Data Type. The SPORT_CTL_B.DTYPE bits selects the data type formatting for the half SPORT's data transfers in DSP standard mode (SPORT_CTL_B.OPMODE =0).	
		0	Right-justify data, zero-fill unused MSBs
		1	Right-justify data, sign-extend unused MSBs
		2	u-law compand data
		3	A-law compand data

Table 32-26: SPORT_CTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	SPENPRI	Serial Port Enable (Primary). The `SPORT_CTL_B.SPENPRI` bit enables the half SPORT's primary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers.	
		0	Disable
		1	Enable

Half SPORT 'A' Divisor Register

The SPORT_DIV_A register contains divisor values that determine frequencies of internally-generated clocks and frame syncs for half SPORT 'A'.

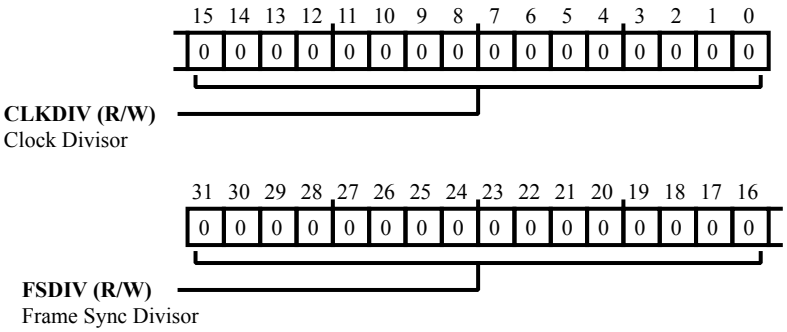


Figure 32-26: SPORT_DIV_A Register Diagram

Table 32-27: SPORT_DIV_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	FSDIV	Frame Sync Divisor. The SPORT_DIV_A.FSDIV bits select the number of transmit or receive clock cycles that the half SPORT counts before generating a frame sync pulse. The half SPORT counts serial clock cycles whether these are from an internally- or an externally-generated serial clock. The formula relating SPORT_DIV_A.FSDIV to the number of cycles between frame sync pulses is: SPORT_DIV_A.FSDIV = (number of serial clocks between frame syncs) - 1 Use the following equation to determine the value of SPORT_DIV_A.FSDIV, given the serial clock frequency and desired frame sync frequency: FSDIV = (SPORT_ACLK / SPORT_AFS) - 1 Note that the frame sync is continuously active when SPORT_DIV_A.FSDIV = 0. The value of SPORT_DIV_A.FSDIV should not be less than the serial word length (SPORT_CTL_A.SLEN), as this may cause an external device to abort the current operation or cause other unpredictable results.
15:0 (R/W)	CLKDIV	Clock Divisor. The SPORT_DIV_A.CLKDIV bits select the divisor that the half SPORT uses to calculate the serial clock (SPORT_ACLK) from the processor system clock (SCLK0). The divisor is a 16-bit value, allowing a wide range of serial clock rates. When configured for internal clock (SPORT_CTL_A.ICLK =1), legal SPORT_DIV_A.CLKDIV values are 0 to 65535. Given the processor system clock frequency and desired serial clock frequency, use the following formula to calculate the value of SPORT_DIV_A.CLKDIV: CLKDIV = (SCLK0 / SPORT_ACLK) - 1 For the maximum serial clock frequency, see the processor data sheet.

Half SPORT 'B' Divisor Register

The SPORT_DIV_B contains divisor values that determine frequencies of internally-generated clocks and frame syncs for SPORT half 'B'.

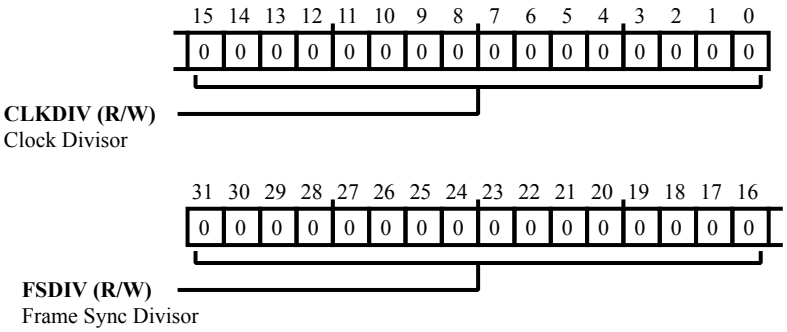


Figure 32-27: SPORT_DIV_B Register Diagram

Table 32-28: SPORT_DIV_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	FSDIV	Frame Sync Divisor. The SPORT_DIV_B.FSDIV bits select the number of transmit or receive clock cycles that the half SPORT counts before generating a frame sync pulse. The half SPORT counts serial clock cycles whether these are from an internally- or an externally-generated serial clock. The formula relating SPORT_DIV_B.FSDIV to the number of cycles between frame sync pulses is: SPORT_DIV_B.FSDIV = (number of serial clocks between frame syncs) - 1 Use the following equation to determine the value of SPORT_DIV_B.FSDIV, given the serial clock frequency and desired frame sync frequency: FSDIV = (SPORT_BCLK / SPORT_BFS) - 1 Note that the frame sync is continuously active when SPORT_DIV_B.FSDIV = 0. The value of SPORT_DIV_B.FSDIV should not be less than the serial word length (SPORT_CTL_B.SLEN), as this may cause an external device to abort the current operation or cause other unpredictable results.
15:0 (R/W)	CLKDIV	Clock Divisor. The SPORT_DIV_B.CLKDIV bits select the divisor that the half SPORT uses to calculate the serial clock (SPORT_BCLK) from the processor system clock (SCLK0). The divisor is a 16-bit value, allowing a wide range of serial clock rates. When configured for internal clock (SPORT_CTL_B.ICLK =1), legal SPORT_DIV_B.CLKDIV values are 0 to 65535. Given the processor system clock frequency and desired serial clock frequency, use the following formula to calculate the value of SPORT_DIV_B.CLKDIV: CLKDIV = (SCLK0 SPORT_BCLK) - 1 For the maximum serial clock frequency, see the processor data sheet.

Half SPORT 'A' Error Register

The SPORT_ERR_A register contains error status and error interrupt mask bits for SPORT half 'A', including error handling bits for the half SPORT's primary and secondary channels and frame sync. Detected errors are frame sync violations or buffer over/underflow conditions.

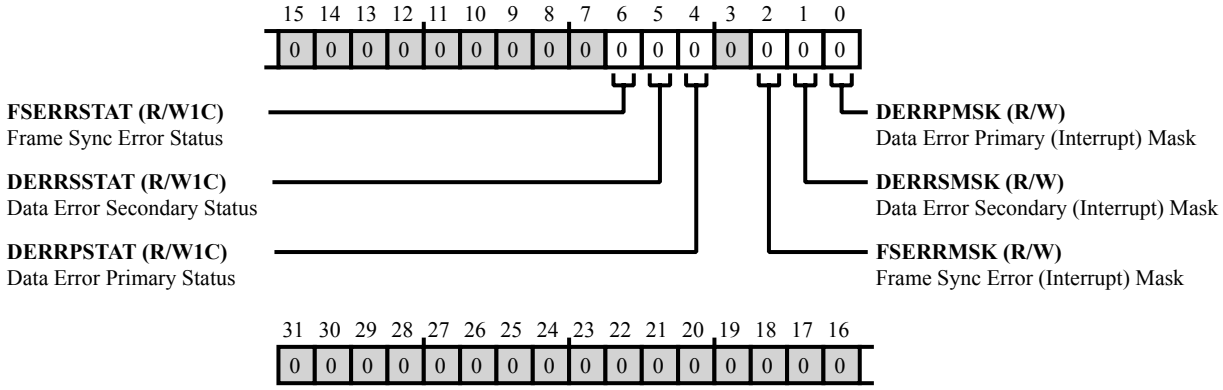


Figure 32-28: SPORT_ERR_A Register Diagram

Table 32-29: SPORT_ERR_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1C)	FSERRSTAT	Frame Sync Error Status. The SPORT_ERR_A.FSERRSTAT bit indicates that the half SPORT has detected a frame sync when the bit count (bits remaining in the frame) is non-zero. When a half SPORT is receiving or transmitting, its bit count is set to a word length (for example, SPORT_CTL_A.SLEN = 31). After each serial clock edge, the half SPORT decrements the transfer's bit count. After the word is received or transmitted, the transfer's bit count reaches zero, and the half SPORT resets it (for example, to 32) on next frame sync. Normal SPORT data transfers always have a non-zero bit count value when active transmission or reception is occurring. Normal SPORT frame syncs occur after the bit count becomes zero.	
		0	No error
		1	Error (non-zero bit count at frame sync)

Table 32-29: SPORT_ERR_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W1C)	DERRSSTAT	Data Error Secondary Status. The SPORT_ERR_A.DERRSSTAT bit indicates the error status for the half SPORT's secondary channel data buffers. During transmit (SPORT_CTL_A.SPTRAN =1), the SPORT_ERR_A.DERRSSTAT bit indicates the transmit underflow status. During receive (SPORT_CTL_A.SPTRAN =0), the SPORT_ERR_A.DERRSSTAT bit indicates the receive overflow status. This bit is used to clear the latch of SPORT status interrupt request when triggered by a secondary data error. This bit can also be used to clear the read-only SPORT_CTL_A.DERRSEC status bit.	
		0	No error
		1	Error (transmit underflow or receive overflow)
4 (R/W1C)	DERRPSTAT	Data Error Primary Status. The SPORT_ERR_A.DERRPSTAT bit indicates the error status for the half SPORT's primary channel data buffers. During transmit (SPORT_CTL_A.SPTRAN =1), the SPORT_ERR_A.DERRPSTAT bit indicates the transmit underflow status. During receive (SPORT_CTL_A.SPTRAN =0), the SPORT_ERR_A.DERRPSTAT bit indicates the receive overflow status. This bit is used to clear the latch of SPORT status interrupt request when triggered by a primary data error. This bit can also be used to clear the read-only SPORT_CTL_A.DERRPRI status bit.	
		0	No error
		1	Error (transmit underflow or receive overflow)
2 (R/W)	FSERRMSK	Frame Sync Error (Interrupt) Mask. The SPORT_ERR_A.FSERRMSK unmasks (enables) the half SPORT to generate the frame sync error interrupt request.	
		0	Mask (disable)
		1	Unmask (enable)
1 (R/W)	DERRSMSK	Data Error Secondary (Interrupt) Mask. The SPORT_ERR_A.DERRSMSK unmasks (enables) the half SPORT to generate the data error interrupt request for the secondary channel.	
		0	Mask (disable)
		1	Unmask (enable)
0 (R/W)	DERRPMSK	Data Error Primary (Interrupt) Mask. The SPORT_ERR_A.DERRPMSK unmasks (enables) the half SPORT to generate the data error interrupt request for the primary channel.	
		0	Mask (disable)
		1	Unmask (enable)

Half SPORT 'B' Error Register

The SPORT_ERR_B register contains error status and error interrupt mask bits for SPORT half 'B', including error handling bits for the half SPORT's primary and secondary channels and frame sync. Detected errors are frame sync violations or buffer over/underflow conditions.

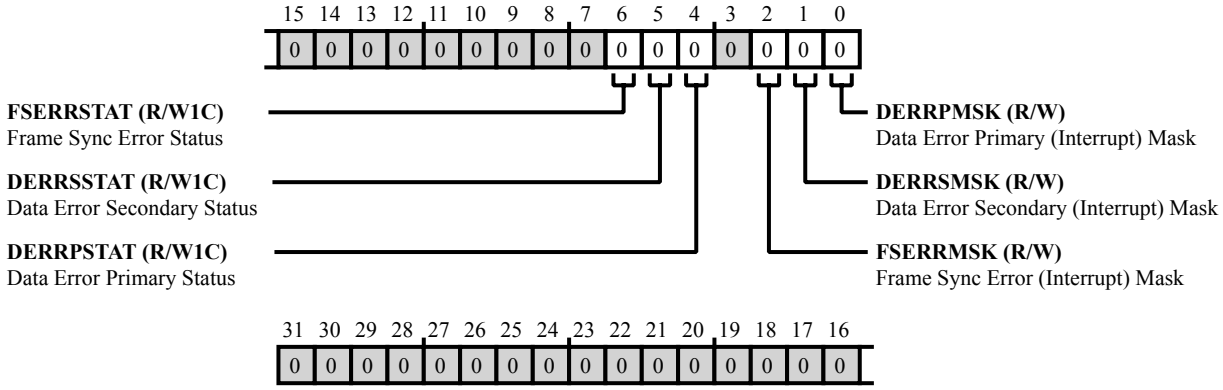


Figure 32-29: SPORT_ERR_B Register Diagram

Table 32-30: SPORT_ERR_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
6 (R/W1C)	FSERRSTAT	Frame Sync Error Status. The SPORT_ERR_B.FSERRSTAT bit indicates that the half SPORT has detected a frame sync when the bit count (bits remaining in the frame) is non-zero. When a half SPORT is receiving or transmitting, its bit count is set to a word length (for example, SPORT_CTL_B.SLEN = 31). After each serial clock edge, the half SPORT decrements the transfer's bit count. After the word is received or transmitted, the transfer's bit count reaches zero, and the half SPORT resets it (for example, to 32) on next frame sync. Normal SPORT data transfers always have a non-zero bit count value when active transmission or reception is occurring. Normal SPORT frame syncs occur after the bit count becomes zero.	
		0	No error
		1	Error (non-zero bit count at frame sync)

Table 32-30: SPORT_ERR_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W1C)	DERRSSTAT	Data Error Secondary Status. The SPORT_ERR_B.DERRSSTAT bit indicates the error status for the half SPORT's secondary channel data buffers. During transmit (SPORT_CTL_B.SPTRAN =1), SPORT_ERR_B.DERRSSTAT indicates the transmit underflow status. During receive (SPORT_CTL_B.SPTRAN =0), SPORT_ERR_B.DERRSSTAT indicates the receive overflow status. This bit is used to clear the latch of SPORT status interrupt request when triggered by a secondary data error. This bit can also be used to clear the read-only SPORT_CTL_B.DERRSEC status bit.	
		0	No error
		1	Error (transmit underflow or receive overflow)
4 (R/W1C)	DERRPSTAT	Data Error Primary Status. The SPORT_ERR_B.DERRPSTAT bit indicates the error status for the half SPORT's primary channel data buffers. During transmit (SPORT_CTL_B.SPTRAN =1), the SPORT_ERR_B.DERRPSTAT bit indicates the transmit underflow status. During receive (SPORT_CTL_B.SPTRAN =0), the SPORT_ERR_B.DERRPSTAT bit indicates the receive overflow status. This bit is used to clear the latch of SPORT status interrupt request when triggered by a primary data error. This bit can also be used to clear the read-only SPORT_CTL_B.DERRPRI status bit.	
		0	No error
		1	Error (transmit underflow or receive overflow)
2 (R/W)	FSERRMSK	Frame Sync Error (Interrupt) Mask. The SPORT_ERR_B.FSERRMSK unmasks (enables) the half SPORT to generate the frame sync error interrupt request.	
		0	Mask (disable)
		1	Unmask (enable)
1 (R/W)	DERRSMSK	Data Error Secondary (Interrupt) Mask. The SPORT_ERR_B.DERRSMSK unmasks (enables) the half SPORT to generate the data error interrupt request for the secondary channel.	
		0	Mask (disable)
		1	Unmask (enable)
0 (R/W)	DERRPMSK	Data Error Primary (Interrupt) Mask. The SPORT_ERR_B.DERRPMSK unmasks (enables) the half SPORT to generate the data error interrupt request for the primary channel.	
		0	Mask (disable)
		1	Unmask (enable)

Half SPORT 'A' Multichannel Control Register

The SPORT_MCTL_A register controls the half SPORT's multichannel operations. This register enables multichannel operation, enables multichannel data packing, selects the multichannel frame delay, selects the number of multichannel slots, and selects the multichannel window offset size.

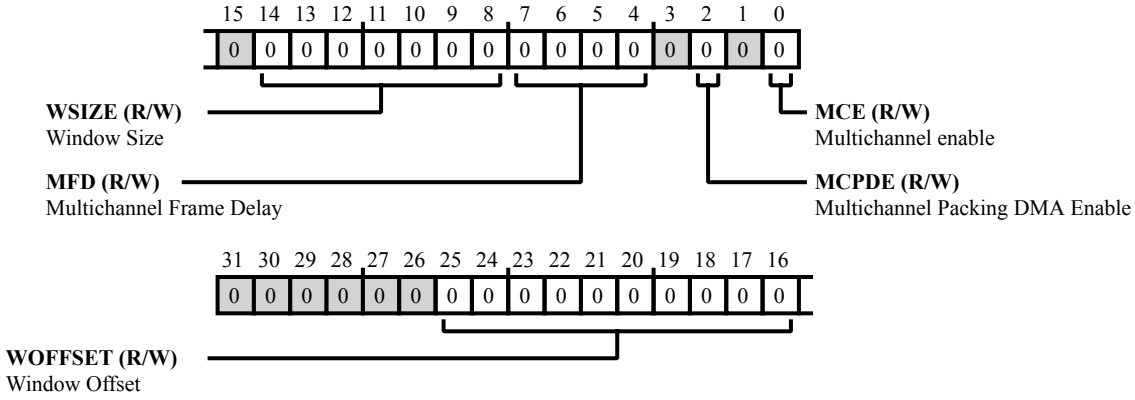


Figure 32-30: SPORT_MCTL_A Register Diagram

Table 32-31: SPORT_MCTL_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:16 (R/W)	WOFFSET	Window Offset. The SPORT_MCTL_A.WOFFSET bits select the start location for the half SPORT's active window of channels within the 1024-channel range. A value of 0 specifies no offset and 896 is the largest value that permits using all 128 channels. When multichannel mode is disabled (SPORT_MCTL_A.MCE =0) and the right-justified mode is enabled (SPORT_CTL_A.RJUST =1), the least significant 6 bits of SPORT_MCTL_A.WOFFSET serve as the delay count (DCNT) field. These bits introduce a clock delay before the half SPORT state machine starts to capture data. For information about appropriate delay selections, see the SPORT operating modes section.
14:8 (R/W)	WSIZE	Window Size. The SPORT_MCTL_A.WSIZE bits select the window size for the half SPORT's active window of channels. Use the following formula to calculate the window size value: SPORT_MCTL_A.WSIZE = (number of channel slots) -1
7:4 (R/W)	MFD	Multichannel Frame Delay. The SPORT_MCTL_A.MFD bits select the delay (in serial clock cycles) between the half SPORT's multichannel frame sync pulse and channel 0. The 4-bit field allows selecting multichannel frame delay of 0-15 serial clocks.

Table 32-31: SPORT_MCTL_A Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	MCPDE	Multichannel Packing DMA Enable. The SPORT_MCTL_A.MCPDE bit enables DMA data packing for transmit and enables DMA data unpacking for the half SPORT's multichannel data transfers.	
		0	Disable
		1	Enable
0 (R/W)	MCE	Multichannel enable. The SPORT_MCTL_A.MCE bit enables multichannel operations for the half SPORT. The half SPORT is configured in normal multichannel mode if SPORT_CTL_A.OPMODE=0; while it is configured in packed mode if SPORT_CTL_A.OPMODE=1. When configuring in these modes, the multichannel enable bit (SPORT_MCTL_A.MCE) should be set before enabling the SPORT data channel enable bits (SPORT_CTL_A.SPENPRI and/or SPORT_CTL_A.SPENSEC). When these channel bits transition from 1 to 0, note that the half SPORT's data transfer buffers are cleared, and the SPORT_CTL_A.DERRPRI and SPORT_CTL_A.DERRSEC bits are cleared.	
		0	Disable
		1	Enable

Half SPORT 'B' Multichannel Control Register

The SPORT_MCTL_B register controls the half SPORT's multichannel operations. This register enables multichannel operation, enables multichannel data packing, selects the multichannel frame delay, selects the number of multichannel slots, and selects the multichannel window offset size.

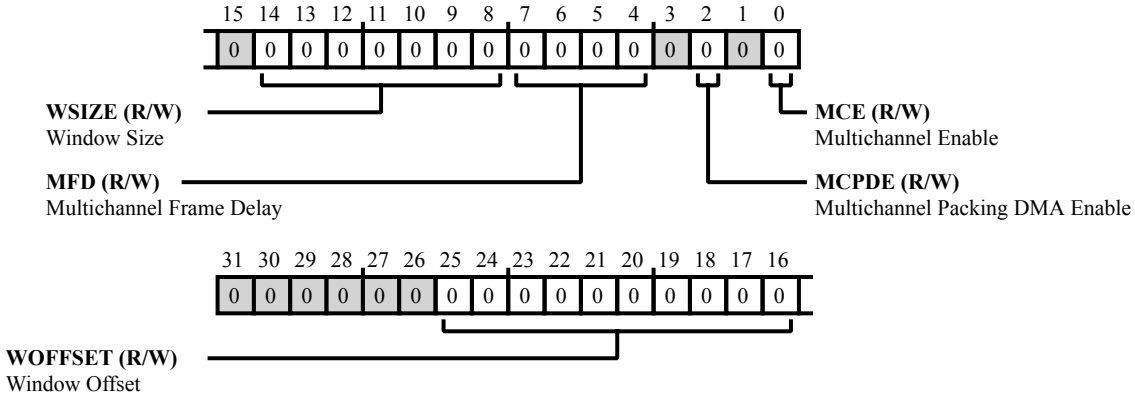


Figure 32-31: SPORT_MCTL_B Register Diagram

Table 32-32: SPORT_MCTL_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:16 (R/W)	WOFFSET	Window Offset. The SPORT_MCTL_B.WOFFSET bits select the start location for the half SPORT's active window of channels within the 1024-channel range. A value of 0 specifies no offset and 896 is the largest value that permits using all 128 channels. When multichannel mode is disabled (SPORT_MCTL_B.MCE =0) and right-justified mode is enabled (SPORT_CTL_B.RJUST =1), the least significant 6 bits of SPORT_MCTL_B.WOFFSET serve as the delay count (DCNT) field. These bits introduce a clock delay before the half SPORT state machine starts to capture data. For information about appropriate delay selections, see the SPORT operating modes section.
14:8 (R/W)	WSIZE	Window Size. The SPORT_MCTL_B.WSIZE bits select the window size for the half SPORT's active window of channels. Use the following formula to calculate the window size value: SPORT_MCTL_B.WSIZE = (number of channel slots) -1
7:4 (R/W)	MFD	Multichannel Frame Delay. The SPORT_MCTL_B.MFD bits select the delay (in serial clock cycles) between the half SPORT's multichannel frame sync pulse and channel 0. The 4-bit field allows selecting a multichannel frame delay of 0-15 serial clocks.

Table 32-32: SPORT_MCTL_B Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	MCPDE	Multichannel Packing DMA Enable. The SPORT_MCTL_B.MCPDE bit enables DMA data packing for transmit and enables DMA data unpacking for the half SPORT's multichannel data transfers.	
		0	Disable
		1	Enable
0 (R/W)	MCE	Multichannel Enable. The SPORT_MCTL_B.MCE bit enables multichannel operations for the half SPORT. The half SPORT is configured in normal multichannel mode if SPORT_CTL_B.OPMODE=0; while it is configured in packed mode if SPORT_CTL_B.OPMODE=1. When configuring in these modes, the multichannel enable bit (SPORT_MCTL_B.MCE) should be set before enabling SPORT data channel enable bits (SPORT_CTL_B.SPENPRI and/or SPORT_CTL_B.SPENSEC). When these channel bits transition from 1 to 0, note that the half SPORT's data transfer buffers are cleared, and the SPORT_CTL_B.DERRPRI and SPORT_CTL_B.DERRSEC bits are cleared.	
		0	Disable
		1	Enable

Half SPORT 'A' Multichannel Status Register

The SPORT_MSTAT_A register indicates the current multichannel being serviced among the half SPORT's active channels in multichannel mode. The half SPORT increments the value by one in this register as each channel is serviced. The value in the SPORT_MSTAT_A register restarts at 0 at each frame sync.

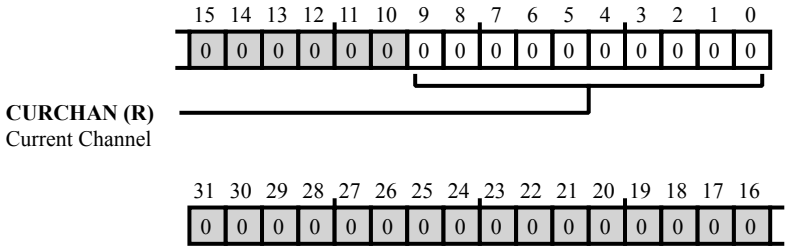


Figure 32-32: SPORT_MSTAT_A Register Diagram

Table 32-33: SPORT_MSTAT_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	CURCHAN	Current Channel. The SPORT_MSTAT_A.CURCHAN bits indicate the half SPORT's current channel being serviced in multichannel mode.

Half SPORT 'B' Multichannel Status Register

The SPORT_MSTAT_B register indicates the current multichannel being serviced among the half SPORT's active channels in multichannel mode. The half SPORT increments the value by one in this register as each channel is serviced. The value in the SPORT_MSTAT_B register restarts at 0 at each frame sync.

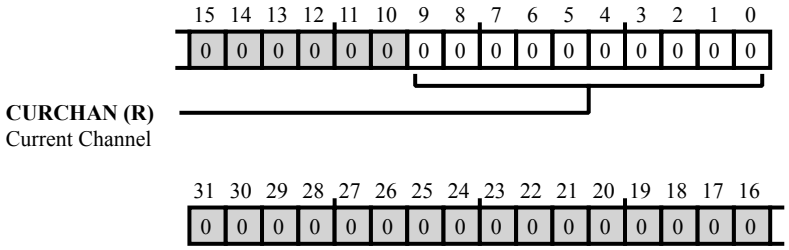


Figure 32-33: SPORT_MSTAT_B Register Diagram

Table 32-34: SPORT_MSTAT_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
9:0 (R/NW)	CURCHAN	Current Channel. The SPORT_MSTAT_B.CURCHAN bits indicate the half SPORT's current channel being serviced in multichannel mode.

Half SPORT 'A' Rx Buffer (Primary) Register

The SPORT_RXPRI_A register buffers the half SPORT's primary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the primary channel. After a complete word has been received in the receive shifter, it is placed into the SPORT_RXPRI_A register. This data can be read in core mode (interrupt-based or polling-based mechanism) or directly transferred into processor memory using the DMA controller. With a data buffer and an input shift register, the SPORT_RXPRI_A register acts as a two-location buffer. So, the SPORT can keep a maximum of two 32-bit received words at any given time (independent of the SPORT_CTL_A.PACK bit setting).

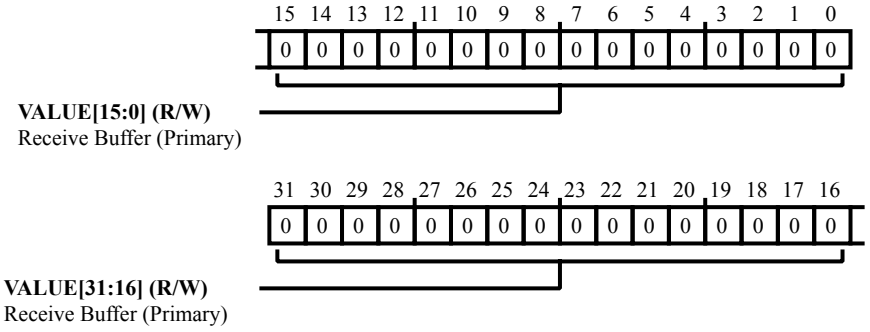


Figure 32-34: SPORT_RXPRI_A Register Diagram

Table 32-35: SPORT_RXPRI_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Receive Buffer (Primary). The SPORT_RXPRI_A.VALUE bits hold the half SPORT's primary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions.

Half SPORT 'B' Rx Buffer (Primary) Register

The SPORT_RXPRI_B register buffers the half SPORT's primary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the primary channel. After a complete word has been received in the receive shifter, it is placed into the SPORT_RXPRI_B register. This data can be read in core mode (interrupt-based or polling-based mechanism) or directly transferred into processor memory using the DMA controller. With a data buffer and an input shift register, the SPORT_RXPRI_B register acts as a two-location buffer. So, the SPORT can keep a maximum of two 32-bit received words at any given time (independent of the SPORT_CTL_A.PACK bit setting).

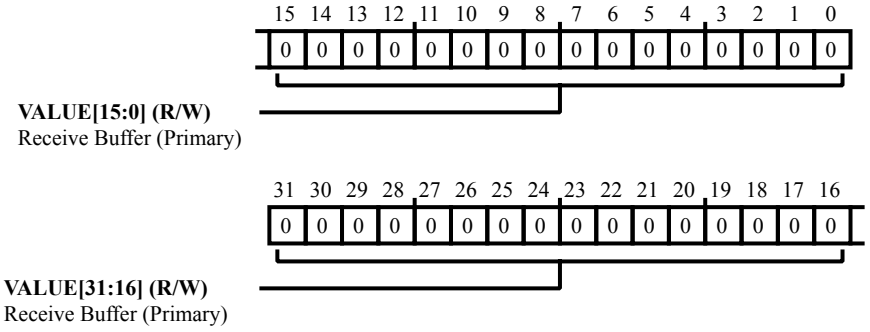


Figure 32-35: SPORT_RXPRI_B Register Diagram

Table 32-36: SPORT_RXPRI_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Receive Buffer (Primary). The SPORT_RXPRI_B.VALUE bits hold the half SPORT's primary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions.

Half SPORT 'A' Rx Buffer (Secondary) Register

The SPORT_RXSEC_A register buffers the half SPORT's secondary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the secondary channel. After a complete word has been received in the receive shifter, it is placed into the SPORT_RXSEC_A register. This data can be read in core mode (in interrupt-based or polling-based mechanism) or directly transferred into processor memory using the DMA controller. With a data buffer and an input shift register, the SPORT_RXSEC_A register acts as a two-location buffer. So, the SPORT can keep a maximum of two 32-bit received words at any given time (independent of the SPORT_CTL_A.PACK bit setting).

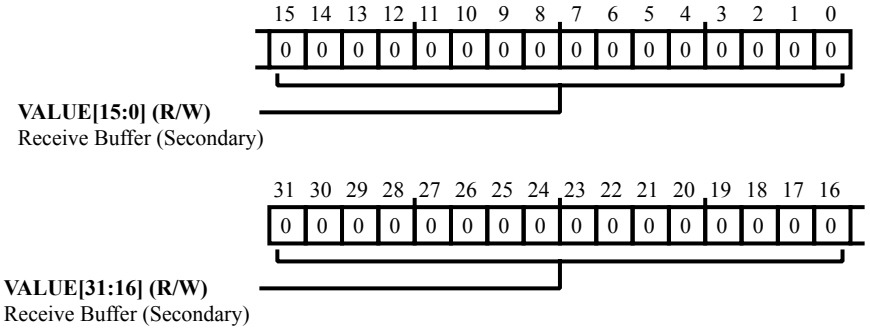


Figure 32-36: SPORT_RXSEC_A Register Diagram

Table 32-37: SPORT_RXSEC_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Receive Buffer (Secondary). The SPORT_RXSEC_A.VALUE bits hold the half SPORT's secondary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions.

Half SPORT 'B' Rx Buffer (Secondary) Register

The SPORT_RXSEC_B register buffers the half SPORT's secondary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the secondary channel. After a complete word has been received in the receive shifter, it is placed into the SPORT_RXSEC_B register. This data can be read in core mode (in interrupt-based or polling-based mechanism) or directly transferred into processor memory using the DMA controller. With a data buffer and an input shift register, the SPORT_RXSEC_B register acts as a two-location buffer. So, the SPORT can keep a maximum of two 32-bit received words at any given time (independent of the SPORT_CTL_A.PACK bit setting).

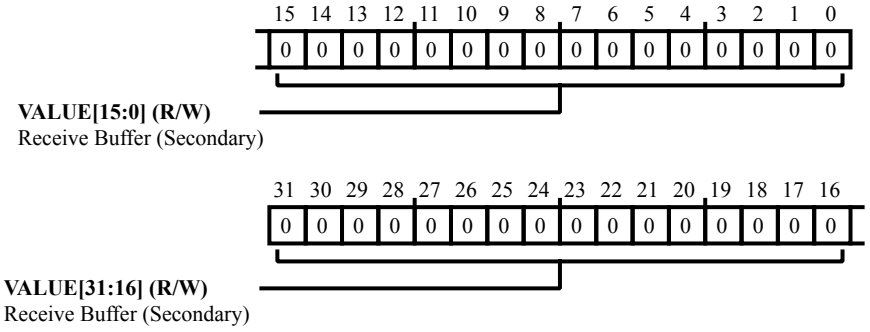


Figure 32-37: SPORT_RXSEC_B Register Diagram

Table 32-38: SPORT_RXSEC_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Receive Buffer (Secondary). The SPORT_RXSEC_B.VALUE bits hold the half SPORT's secondary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions.

Half SPORT 'A' Tx Buffer (Primary) Register

The SPORT_TXPRI_A register buffers the half SPORT's primary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the primary channel. Either a program running on the processor core loads the data into the buffer (word-by-word process) or the DMA controller automatically loads the data into the buffer (DMA process).

The SPORT_TXPRI_A register acts as a three-location buffer if SPORT data packing is disabled (SPORT_CTL_A.PACK =0); while it acts as a two-location buffer when packing is enabled (SPORT_CTL_A.PACK =1). So, depending on the PACK bit setting, two 32-bit words or three 32-bit words can be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT_TXPRI_A register contents are automatically loaded into the output shifter. The half SPORT can issue an interrupt request (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt request does not occur when the half SPORT is executing a DMA-based transfer.

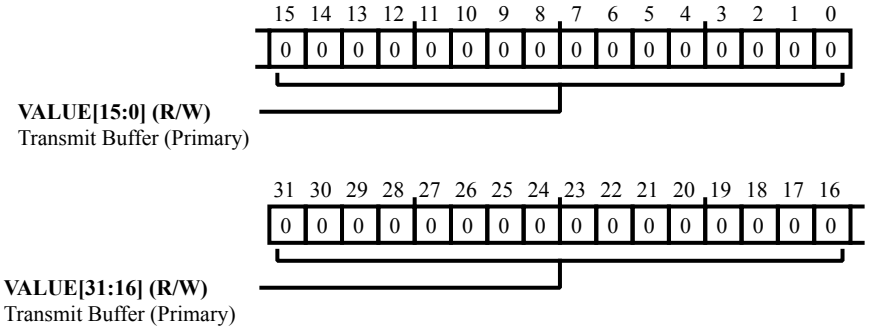


Figure 32-38: SPORT_TXPRI_A Register Diagram

Table 32-39: SPORT_TXPRI_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Transmit Buffer (Primary). The SPORT_TXPRI_A.VALUE bits hold the half SPORT's primary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions.

Half SPORT 'B' Tx Buffer (Primary) Register

The SPORT_TXPRI_B register buffers the half SPORT's primary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the primary channel. Either a program running on the processor core loads the data into the buffer (word-by-word process) or the DMA controller automatically loads the data into the buffer (DMA process).

The SPORT_TXPRI_B register acts as a three-location buffer if SPORT data packing is disabled (SPORT_CTL_B.PACK =0); while it acts as a two-location buffer when packing is enabled (SPORT_CTL_B.PACK =1). So, depending on the PACK bit setting, two 32-bit words or three 32-bit words can be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT_TXPRI_B register contents are automatically loaded into the output shifter. The half SPORT can issue an interrupt request (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt request does not occur when the half SPORT is executing a DMA-based transfer.

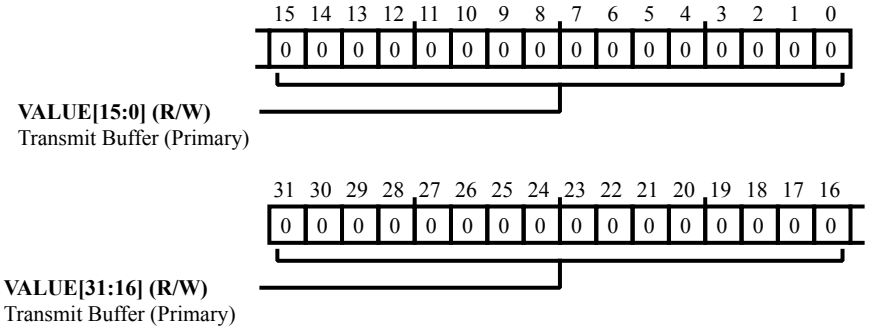


Figure 32-39: SPORT_TXPRI_B Register Diagram

Table 32-40: SPORT_TXPRI_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Transmit Buffer (Primary). The SPORT_TXPRI_B.VALUE bits hold the half SPORT's primary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions.

Half SPORT 'A' Tx Buffer (Secondary) Register

The SPORT_TXSEC_A register buffers the half SPORT's secondary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the secondary channel. Either a program running on the processor core loads the data into the buffer (word-by-word process) or the DMA controller automatically loads the data into the buffer (DMA process).

The SPORT_TXSEC_A register acts as a three-location buffer if SPORT data packing is disabled (SPORT_CTL_A.PACK =0); while it acts as a two-location buffer when packing is enabled (SPORT_CTL_A.PACK =1). So, depending on the PACK bit setting, two 32-bit words or three 32-bit words can be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT_TXSEC_A register contents are automatically loaded into the output shifter. The half SPORT can issue an interrupt request (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt request does not occur when the half SPORT is executing a DMA-based transfer.

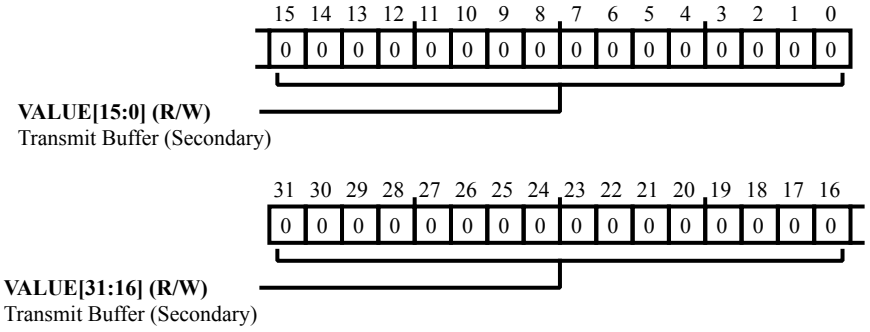


Figure 32-40: SPORT_TXSEC_A Register Diagram

Table 32-41: SPORT_TXSEC_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Transmit Buffer (Secondary). The SPORT_TXSEC_A.VALUE bits hold the half SPORT's secondary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions.

Half SPORT 'B' Tx Buffer (Secondary) Register

The SPORT_TXSEC_B register buffers the half SPORT's secondary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the secondary channel. Either a program running on the processor core loads the data into the buffer (word-by-word process) or the DMA controller automatically loads the data into the buffer (DMA process).

The SPORT_TXSEC_B register acts as a three-location buffer if SPORT data packing is disabled (SPORT_CTL_B.PACK =0); while it acts as two-location buffer when packing is enabled (SPORT_CTL_B.PACK =1). So, depending on the PACK bit setting, two 32-bit words or three 32-bit words can be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT_TXSEC_B register contents are automatically loaded into the output shifter. The half SPORT can issue an interrupt request (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt request does not occur when the half SPORT is executing a DMA-based transfer.

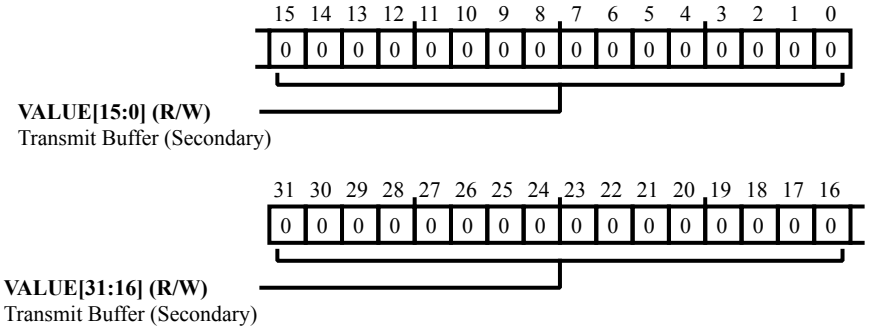


Figure 32-41: SPORT_TXSEC_B Register Diagram

Table 32-42: SPORT_TXSEC_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Transmit Buffer (Secondary). The SPORT_TXSEC_B.VALUE bits hold the half SPORT's secondary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions.

33 Precision Clock Generators (PCG)

The precision clock generators are used to produce a pair of signals from a clock input signal. The two signals generated are normally used as a serial bit clock and frame sync pair. The PCG is part of the DAI. There are four PCG units per DAI. *PCGA*, *PCGB*, *PCGE*, and *PCGF* belong to DAI0; *PCGC*, *PCGD*, *PCGG*, and *PCGH* belong to DAI1.

Each of these four units (A, B, E, and F) generates one clock (CLKA_O, CLKB_O, CLKE_O, and CLKF_O) and one frame sync (FSA_O, FSB_O, FSE_O, and FSF_O) output. In addition to clock and frame sync outputs, each of these four units generates one inverted version of clock output (INV_CLKA_O, INV_CLKB_O, INV_CLKE_O, and INV_CLKF_O) and one inverted version of frame sync output (INV_FSA_O, INV_FSB_O, INV_FSE_O, and INV_FSF_O).

Features

The following list describes the features of the precision clock generators.

- SRU allows the routing of all of the PCG signals in one DAI (two PCG units in DAI)
- Input clock selection: SYS_CLKIN0 and SYS_CLKIN1 can be used as the input clock for PCG, when configured to use CLKIN as clock source
- Provides four different clock dividers for serial clock, frame sync, phase (20-bit), and pulse width (16-bit)
- Phase shift allows adjustment of the frame sync relative to the serial clock and can be shifted the full period and wrapped around
- Provides pulse width control for arbitrary frame sync signal generation
- Bypass mode for external frame sync manipulation
- External trigger mode starts PCG operation
- No additional jitter is introduced when using off-chip clocks
- Global enable. (See the DAI chapter for details)

Functional Description

The *PCG Block Diagram* shows the blocks within the module and its connection to the DAI. The following sections provide information on the function of these blocks.

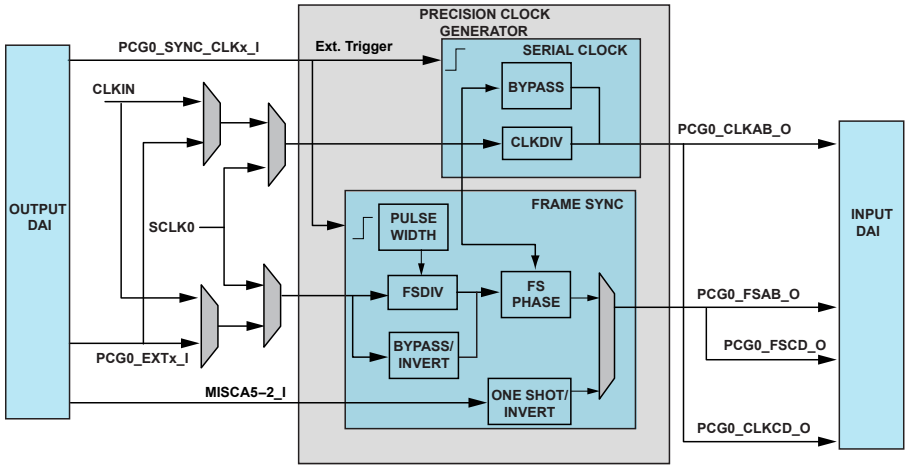


Figure 33-1: PCG Block Diagram

NOTE: In the *PCG Block Diagram*, the CLKINSEL bit field determines whether SYS_CLKIN0 or SYS_CLKIN1 is input to DAI.

ADSP-SC59x PCG Register List

Precision Clock Generator

Table 33-1: ADSP-SC59x PCG Register List

Name	Description
PCG_CTLA0	Precision Clock A Control 0 Register
PCG_CTLA1	Precision Clock A Control 1 Register
PCG_CTLB0	Precision Clock B Control 0 Register
PCG_CTLB1	Precision Clock B Control 1 Register
PCG_CTLC0	Precision Clock C Control 0 Register
PCG_CTLC1	Precision Clock C Control 1 Register
PCG_CTLD0	Precision Clock D Control 0 Register
PCG_CTLD1	Precision Clock D Control 1 Register
PCG_CTLE0	Precision Clock E Control 0 Register
PCG_CTLE1	Precision Clock E Control 1 Register
PCG_CTLF0	Precision Clock F Control 0 Register

Table 33-1: ADSP-SC59x PCG Register List (Continued)

Name	Description
PCG_CTLF1	Precision Clock F Control 1 Register
PCG_CTLG0	Precision Clock G Control 0 Register
PCG_CTLG1	Precision Clock G Control 1 Register
PCG_CTLH0	Precision Clock H Control 0 Register
PCG_CTLH1	Precision Clock H Control 1 Register
PCG_PW1	Precision Clock Pulse Width Control 1 Register
PCG_PW2	Precision Clock Pulse Width Control 2 Register
PCG_PW3	Precision Clock Pulse Width Control 3 Register
PCG_PW4	Precision Clock Pulse Width Control 4 Register
PCG_SYNC1	Precision Clock Frame Sync Synchronization 1 Register
PCG_SYNC2	Precision Clock Frame Sync Synchronization 2 Register
PCG_SYNC3	Precision Clock Frame Sync Synchronization 3 Register
PCG_SYNC4	Precision Clock Frame Sync Synchronization 4 Register

ADSP-SC59x PCG Trigger List

Table 33-2: ADSP-SC59x PCG Trigger List Generators

Trigger ID	Name	Description	Sensitivity
172	PCG0_HWA	PCG0 PCG-A Hardware trigger control	None
173	PCG0_HWB	PCG0 PCG-B Hardware trigger control	None
174	PCG0_HWC	PCG0 PCG-C Hardware trigger control	None
175	PCG0_HWD	PCG0 PCG-D Hardware trigger control	None
176	PCG0_HWE	PCG0 PCG-E Hardware trigger control	None
177	PCG0_HWF	PCG0 PCG-F Hardware trigger control	None
178	PCG0_HWG	PCG0 PCG-G Hardware trigger control	None
179	PCG0_HWH	PCG0 PCG-H Hardware trigger control	None

Table 33-3: ADSP-SC59x PCG Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
177	PCG0_HWA	PCG0 PCG-A Hardware trigger control	Pulse
178	PCG0_HWB	PCG0 PCG-B Hardware trigger control	Pulse
179	PCG0_HWC	PCG0 PCG-C Hardware trigger control	Pulse

Table 33-3: ADSP-SC59x PCG Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
180	PCG0_HWD	PCG0 PCG-D Hardware trigger control	Pulse
181	PCG0_HWE	PCG0 PCG-E Hardware trigger control	Pulse
182	PCG0_HWF	PCG0 PCG-F Hardware trigger control	Pulse
183	PCG0_HWG	PCG0 PCG-G Hardware trigger control	Pulse
184	PCG0_HWH	PCG0 PCG-H Hardware trigger control	Pulse

Internal Interface

The fundamental clock of the PCG is *SCLK0*. The clock to this module can be shut off for power savings.

Serial Clock

Each of the four units (A, B, C, and D) produces a clock output. Serial clock generation from a unit is independently enabled and controlled. Sources for the serial clock generation can be either from the CLKIN, SCLK0, or a DAI pin source.

When CLKIN is chosen as input clock in PCG, the clock source bits (PCG_SYNC1.CLKA_CLKINSEL, PCG_SYNC1.CLKB_CLKINSEL) determine whether the clock source is SYS_CLKIN0 or SYS_CLKIN1.

Note that the divider is working in normal mode for PCG0_CTLx1.CLKDIV > 1. For PCG_CTLA1.CLKDIV/PCG_CTLB1.CLKDIV = 0 or 1, the divider operates in bypass mode (input clock is fed directly to its output). In bypass mode, the clock at the output can theoretically run at up to the SCLK0 frequency. Check the data sheet for the specified maximum operation speed of the DAI pin buffers.

Note that the clock output is always set (as closely as possible) to a 50% duty cycle. If the clock divisor is even, the duty cycle of the clock output is exactly 50%. If the clock divisor is odd, then the duty cycle is slightly less than 50%. The low period of the output clock is one input clock period more than the high period of the output clock. For higher values of an odd divisor, the duty cycle is close to 50%.

NOTE: A PCG clock output cannot be fed to its own input.

Frame Sync

The following sections describe the use of frame syncs in the PCGs.

Frame Sync Output

Each of the four units (A through D) produces a synchronization signal for framing serial data. The frame sync outputs are much more flexible since they must accommodate the wide variety of serial protocols used by peripherals.

Frame sync generation from a unit is independently enabled and controlled. Sources for the frame sync generation can be either from the crystal buffer output, SCLK0, or an external pin source. There is only one external source pin for both frame sync and clock output for a unit.

If an external source is selected for both frame sync and clock output for a unit, then they operate on the same input signal. Apart from enable and source select control bits, a 20-bit divisor controls frame sync generation.

Divider Mode Selection

If a frame sync divisor is greater than 1, the PCG frame sync output frequency is equal to the input clock frequency, divided by a 20-bit integer. This integer is specified in the `PCG_CTLA0.FSDIV`/`PCG_CTLB0.FSDIV` bit field (bits 19:0).

However, if the frame sync divisor is 0 or 1, the PCG's frame sync clock generation unit is bypassed, and the frame sync input is connected directly to the frame sync output. For `PCG_CTLB0.FSDIV` =0, 1 the `PCG_PW1` - `PCG_PW4` registers function differently than in normal mode.

Phase Shift

Phase shift is a frame sync parameter that defines the phase shift of the frame sync relative to the input clock of the same unit. This feature allows the shifting of the frame sync signal in time relative to the clock input signal. Frame sync phase shifting is often required by peripherals that need a frame sync signal to lead or lag a clock signal.

For example, the I^2S protocol specifies that the frame sync transition from high-to-low occurs one clock cycle before the beginning of a frame. Since an I^2S frame is 64 clock cycles long, delaying the frame sync by 63 cycles produces the required framing.

Phase shifting is represented as a full 20-bit value. Even when the frame sync is divided by the maximum amount, the phase can be shifted to the full range, from zero to one input clock short of the period.

NOTE: Phase shifting is specified as a 2 x 10-bit divider value in the `PCG_CTLA0.FSPHASEHI` bit field (bits 29:20) and in the `PCG_CTLA1.FSPHASELO` bit field (bits 29:20).

A single 20-bit value spans these two-bit fields. The upper half of the word (bits 19:10) is in the `PCG_CTLA0` register, and the lower half (bits 9:0) is in the `PCG_CTLA1` register.

The phase shift between clock and frame sync outputs can be programmed using the `PCG_PW1` - `PCG_PW4` registers function registers and all of the control registers under these conditions:

- The input clock source for the clock generator output and the frame sync generator output is the same.
- The clock and frame sync are enabled at the same time using a single atomic instruction.
- The frame sync divisor is an integral multiple of the clock divisor.

NOTE: When using a clock and frame sync as a synchronous pair, the units must be enabled in a single atomic instruction before their parameters are modified. Both units must also be disabled in a single atomic instruction.

NOTE: If the phase shift is 0 (see the ***Phase and Pulse Width Settings*** figure), the clock and frame sync outputs rise at the same time.

If the phase shift is 1, the frame sync output transitions one input clock period ahead of the clock transition.

If the phase shift is *divisor* – 1, the frame sync transitions *divisor* – 1 input clock periods ahead of the clock transitions.

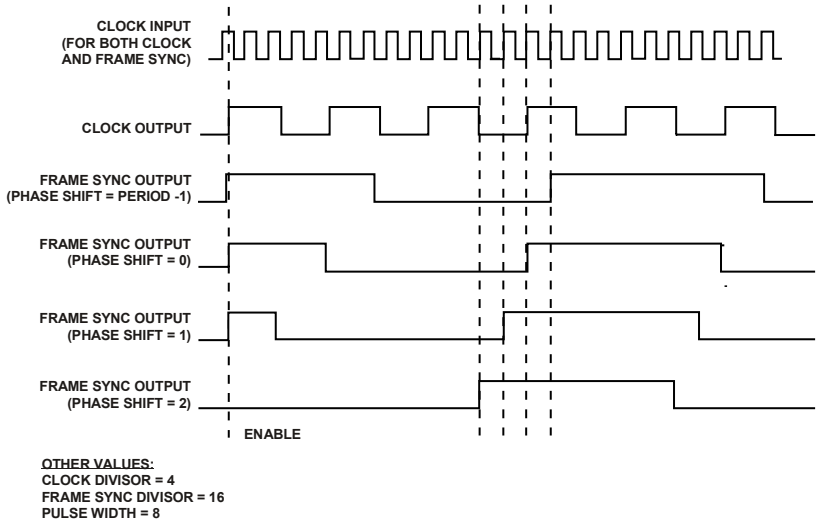


Figure 33-2: Phase and Pulse Width Settings

NOTE: When generating single frame sync pulses (the length of one SPORT clock cycle), take care with respect to the drive and sampling edges. If the rules are violated, for example if the SPORT is not driving data, the module cannot detect a valid sample edge.

Pulse Width

Pulse width is the number of input clock periods for which the frame sync output is high.

A 16-bit value determines the width of the framing pulse. Settings for pulse width can range from zero to `DIV` - 1. The pulse width should be less than the divisor of the frame sync. The pulse width of frame sync is specified in the `PCG_PW1.FSA`, `PCG_PW1.FSB`, `PCG_PW2.FSC`, and `PCG_PW2.FSD` bits (15-0) and (31-16). .

Default Pulse Width

If the pulse width count is equal to 0 and if the `PCG_CTLA0.FSDIV`/`PCG_CTLB0.FSDIV` bit field is even, then the actual pulse width of the frame sync output is equal to:

For even divisors: frame sync divisor/2

If the pulse width count is equal to 0 and if the `PCG_CTLA0.FSDIV`/`PCG_CTLB0.FSDIV` bit field is odd, then the actual pulse width of the frame sync output is equal to:

For odd divisors: frame sync divisor – 1/2

Input Clock Source Considerations

The core Phase-Locked Loop (PLL) has been designed to provide clocking for the processor core. The performance specifications of this PLL are appropriate for the core. But they have not been optimized or specified for precision data converters where jitter directly translates into time quantization errors and distortion.

The PCG allows the routing of external clock sources which are independent of the core PLL.

Timing Example for I²S Mode

For I²S mode, the frame sync should be driven at the falling edge of SPORT clock. In other words, the frame sync edge must coincide with the falling edge of the SPORT clock. To satisfy this requirement, program the phase of the frame sync accordingly in the PCG control registers.

For example, assume that the input clock source for both clock and frame sync are the same and both the clock and frame sync are enabled at the same time. Also, assume that the clock divisor value for generating the required SPORT clock is `PCG_CTLA1.CLKDIV` = 4. Then, for a 32-bit word length, the frame sync divisor value is:

`PCG_CTLA0.FSDIV` = 64, `PCG_CTLA1.CLKDIV` = 4.

By default, for phase = 0, the rising edge of both SPORT clock and frame sync coincide. To make sure that the frame sync edges coincide with the falling edge of the SPORT clock, program the phase value as:

`PCG_CTLA1.CLKDIV`/2 = 2.

Cross Mode Connections

The symmetric dual DAI architecture allows cross connections between both PCGs (A,B) and (C,D) to the other DAI. Each PCG (A through D) supports an alternative input clock (PCG0_EXTx_I)(see Figure 33-1 PCG Block Diagram) which can be sourced via a DAI pin buffer from the other DAI. Note however if routing a source (clock or FS) only DAI pin buffer 2 to 20 can be used (DAI pin buffer 1 is no longer available and is replaced by the DAI CRS buffer for the other DAI).

See DAI Routing Capabilities for more information.

Operating Modes

The following sections provide information on the operating modes of the precision clock generator.

Normal Mode

When the frame sync divisor is set to any value other than zero or one, the PCGs operate in normal mode. In normal mode, the divisor determines the frequency of the frame sync output where:

Frequency of Frame Sync Output = Input Frequency/Divisor

The value of the pulse width control determines the high period of the frame sync output. The value of the pulse width control must be less than the value of the divisor.

The value of the phase control determines the phase of the frame sync output. When the phase is zero, then the positive edges of the clock and frame sync coincide when:

- the clock and frame sync dividers are enabled at the same time using an atomic instruction
- the divisors of the clock and frame sync are the same
- the source for the clock and frame sync is the same

The number of input clock cycles that have already elapsed before the frame sync is enabled is equal to the difference between the divisor and the phase values. When the phase is a small fraction of the divisor, then the frame sync leads the clock. When the phase is only slightly less than the frame sync divisor, then the frame sync lags the clock. The frame sync phase must not be greater than the divisor.

Bypass Mode

When the frame sync divisor for the frame sync has a value of zero or one, the frame sync is in bypass mode, and the `PCG_PW1` - `PCG_PW4` registers are registers have different functionality than in normal mode.

NOTE: In normal mode, bits 15:0 and 31:16 of the `PCG_PW1` - `PCG_PW4` registers are used to program the pulse width count. In bypass mode, bits 15:2 and 31:18 are ignored. Bits 1:0 and 17:16 are renamed to `PCG_PW1.STROBEA` and `PCG_PW1.INVFSA`, respectively. This functionality is described in more detail as follows.

If the `PCG_PW1.STROBEA` register is cleared, then the input is directly passed (see the ***Bypass and Inverted Bypass*** figure) to the frame sync output either inverted or not inverted, depending on the `PCG_PW1.INVFSA` bits.

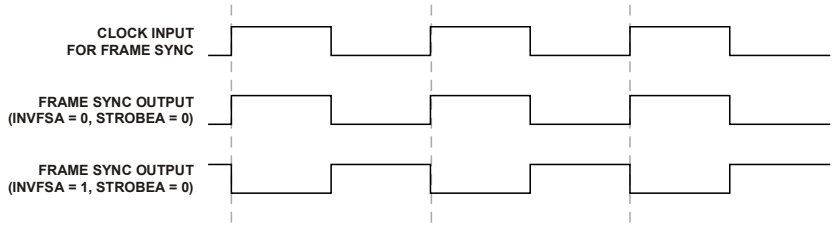


Figure 33-3: Bypass and Inverted Bypass

One-Shot Mode

In the one-shot mode operation shown in the ***One Shot Mode PCG A (MISCA2_I Input)*** figure, the PCG produces a series of periods but does not run continuously.

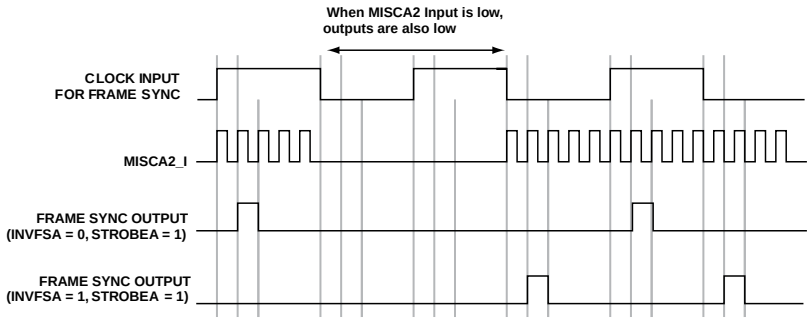


Figure 33-4: One Shot Mode PCG A (MISCA2_I Input)

Bypass mode also enables the generation of a strobe pulse (one-shot frame sync). Strobe usage ignores the divider counters and looks to the SRU to provide the input signal.

In the bypass mode, if the `PCG_PW1.STROBEA` bit =1, then a one-shot pulse is generated. This one-shot pulse has the duration equal to the period of `DAI_MISCAx_I` for the PCGx unit. This pulse is generated either at the falling or rising edge of the input clock, depending on the value of the `PCG_PW1.INVFSA` bit. The output pulse width is equal to the period of the SRU source signal `DAI_MISCAx_I`. The pulse begins at the second rising edge of `MISCAx_I` following a rising edge of the clock input. When the `PCG_PW1.INVFSA` bit is set, the pulse begins at the second rising edge of `DAI_MISCAx_I` coinciding with or following a falling edge of the clock input.

NOTE: A strobe period is defined to be the period of the FS input clock signal as specified by the `PCG_CTLA1.FSSRC` bit.

Audio System Example

The ***PCG Setup for I2S or Left-Justified DAI*** figure shows an example of the interconnections between the S/PDIF receiver, ASRC, and the PCGs. The interconnections are made by programming the signal routing unit. It shows how to set up two precision clock generators using the S/PDIF receiver and an asynchronous sample rate converter (ASRC) to interface to an external audio DAC. The PCG is configured to provide a fixed ASRC/DAC output sample rate of 65.098 kHz. The input to the S/PDIF receiver is typically 44.1 kHz if supplied by a CD player, but can also be from other source at any nominal sample rates.

Similarly, the phase shift for frame syncs B, C, and D is specified in the corresponding control registers (`PCG_CTLA0` through `PCG_CTLB1`).

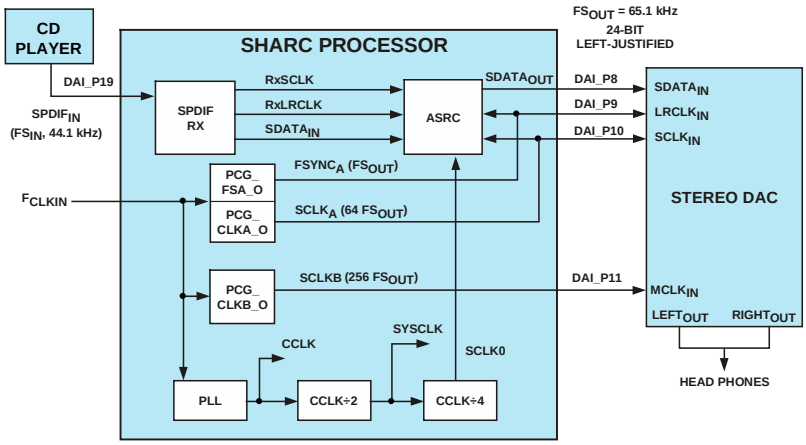


Figure 33-5: PCG Setup for I²S or Left-Justified DAI

Three synchronous clocks are required in audio systems:

1. Frame sync (FS)
2. Serial bit clock (64 FS)
3. Controller DAC clock (256 FS)

Since each PCG has only two outputs, this example requires two PCGs. Furthermore, because the digital audio interface requires a fixed-phase relation between serial clock and FS, these two outputs should come from one PCG (PCG A). The controller clock comes from the second (PCG B).

The combined PCGs can provide a selection of synchronous clock frequencies to support alternate sample rates for the ASRCs and external DACs. However, the range of choices is limited by CLKIN and the ratio of `PCG_CLKx_O`:serial clock:*FS*. The ratio is normally fixed at 256:64:1 to support digital audio left-justified, I²S and right-justified interface modes.

Many DACs also support 384, 512, and 786x FS for `PCG_CLKx_O`, which allows some additional flexibility in choosing serial clock.

Note that the falling edge of serial clock must always be synchronous with both edges of FS. This condition requires that the phase of the serial clock and FS signals for a common PCG (PCG A) be adjustable.

While the frequency of the controller DAC clock (`PCG_CLKx_O`) must be synchronous with the sample rate supplied to the external DAC, there is no fixed-phase requirement.

Set the clock divisor and source and low-phase word first, followed by the control register enable bits, which are set together. When the `PCG_PW1` - `PCG_PW4` registers are set to zero (default), the FS pulse width is (divisor /2) for even divisors and (divisor-1)/2 for odd divisors. Alternatively, the `PCG_PW1`-`PCG_PW4` registers can be set high for exactly one-half the period of CLKIN cycles for a 50% duty cycle, provided the FS divisor is an even number.

Hardware Trigger Control

In addition to the existing enable control to generate the clock and frame sync, this feature provides a flexibility to start the clock and frame sync generation upon receiving a hardware trigger (internal completer trigger). This feature applies to all modes of operation.

For example, when the PCG_SYNC1.HWB_TRIGEN bit is set, it enables the synchronization of clock A and frame sync A to the external LRCLK sync (in external LRCLK synchronization mode) upon receiving a hardware trigger. Refer to the PCG_SYNC1.HWA_TRIGEN/ PCG_SYNC1.HWB_TRIGEN and PCG_SYNC2.HWC_TRIGEN/ PCG_SYNC2.HWD_TRIGEN bit descriptions.

NOTE: To exercise the hardware trigger based synchronization, program the destination registers (DAI_MISC2.IN0, DAI_MISC2.IN1 DAI_MISC2.IN2, and DAI_MISC2.IN3) to route the trigger signals to the PCGs . For example, if PCGA must be enabled for the hardware trigger based synchronization, program the DAI_MISC2.IN0 and DAI_EXTD_MISC2.IN0 bit fields to 5'b00000 and 2'b01, respectively. Each PCG has a dedicated trigger control from the TRU unit, there are four trigger controls for DAI0 and four trigger controls for DAI1.

Refer to TRU Completer Trigger list table for the completer trigger information.

Clock Configuration Examples

For a CLKIN = 33.330 MHz, the two PCGs provide the three synchronous clocks PCGx_CLK, serial clock and *FS* for the SRCs and external DAC. These divisors are stored in the PCG_CTLA1.CLKDIV/PCG_CTLB1.CLKDIV bit fields.

The integer divisors for several sample rates based on 33.330 MHz *CLKIN* are shown in the ***Precision Clock Generator Division Ratios*** table.

Table 33-4: Precision Clock Generator Division Ratios (33.330 CLKIN)

Sample Rate kHz)	PCG Divisors		FSDIV A*1
	CLKDIV B	CLKDIV A	
130.195	1	4	256
65.098	***2***	***8***	***512***
43.398	3	12	768
32.549	4	16	1024
26.039	5	20	1280
21.699	6	24	1536
18.599	7	28	1792

*1 The frame sync divisor must be an even integer to produce a 50% duty cycle waveform. See Frame Sync.

Global PCG Enable

For some applications, enabling multiple PCGs (CLK and FS) must be synchronized. This feature provides the flexibility to enable multiple PCGs (CLK and FS) globally. The global PCG enable feature works independently, while the legacy PCG enable (CLK and FS) remains the same. The legacy PCG enable can be used when the PCGs need to be enabled individually.

The DAI_GBL_PCG_EN register controls the enabling and disabling of PCGs globally. The global enable feature can be used in the following scenarios:

- to enable all the eight PCGs (includes both DAI0 and DAI1) at the same time
- to enable all four PCGs within each DAIx at the same time
- to enable multiple PCGs (across DAIs) at the same time

NOTE: See the DAI_GBL_PCG_EN register description for details.

Programming Guidelines

Complete the following steps to globally enable all of the PCGs :

1. Clear the individual (local) PCG enable bits from PCG_CTLx registers. (for example, PCG_CTLA0.CLKEN and PCG_CTLA0.FSEN bits) without modifying any other settings.
2. Wait for n CCLK cycles; n = PCG source clock period/processor clock period.
3. Program the PCG registers (PCG_CTLx/PCG_SYNC1) with the desired configuration without setting the PCG enable bits.
4. Enable all the required PCGs in the DAI_GBL_PCG_EN register that need to be enabled globally.

Complete the following steps to globally disable the PCGs using the DAI_GBL_PCG_EN register:

1. Disable individual PCG bits to disable a particular PCG.
2. Disable DAI0/1 enable bits to disable PCGs within the DAI.
3. Disable the global PCG enable bit in cases where all PCGs must be disabled.

PCG Event Control

The following sections describe the generation and control of PCG events.

External Event Trigger

The trigger with the external clock is enabled by setting bits 0 and 16 of the PCG_SYNC1/PCG_SYNC2/PCG_SYNC3/PCG_SYNC4 registers .

Refer to the ***FS Output Synchronization With External Trigger Input*** figure. Since the rising edge of the external clock is used to synchronize with the frame sync, the frame sync output is not generated until a rising edge of the external clock is sensed.

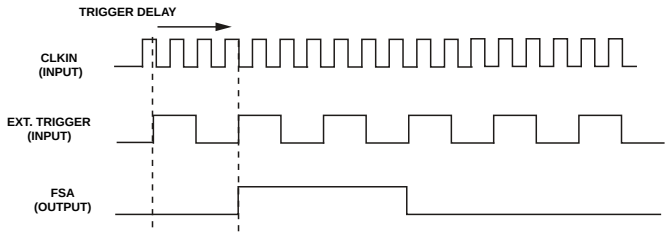


Figure 33-6: FS Output Synchronization With External Trigger Input

External Event Trigger Delay

The time delay between the rising trigger edge and the start of serial clock and frame sync varies between 2.5 to 3.5 input clock periods. If the input clock and the trigger signal are synchronous, the delay is 3 input clock periods. Consider the following cases:

- SCLK0 is the input source. In this case, if the given trigger event is synchronous to SCLK0, the delay is 3 SCLK0 periods. If the trigger signal is asynchronous with SCLK0, the delay varies from 2.5 SCLK0 periods to 3.5 SCLK0 periods. (It depends on whether the trigger edge occurs in the positive half cycle or negative half cycle of SCLK0.)
- The CLKIN source bits of the `PCG_SYNC1` register select the input source for PCG. In this case, if the given trigger signal is synchronous to CLKIN, the delay is 3 CLKIN periods. But if they are asynchronous to CLKIN, the delay can vary between 2.5 CLKIN periods to 3.5 CLKIN periods.
- SRU DAI0 is the input source for PCG A, B. If the input clock and trigger signal are synchronous, the delay is exactly 3 input clock periods. If asynchronous, it varies between 2.5 to 3.5 input clock periods depending on the phase difference between the input clock and trigger signal.

Programming Model

This section describes the sequence of software steps required for successful PCG operation.

If the PCG is disabled to reprogram a parameter, use a delay after writing to the disable bit. This delay in core clock (*CCLK*) cycles = (PCG source clock period/*CCLK* period). In summary, use the following general procedure:

1. Clear the PCG enable bits without modifying any other settings.
2. Wait for N *CCLK* cycles (N = PCG source clock period/processor clock period).
3. Program all new parameters without setting the PCG enable bit.
4. Enable the PCG.

Frame Sync Phase Setting

The phase unit requires that the clock and FS are enabled simultaneously in an atomic instruction.

1. Write the clock divider/low 10-bit phase divider to the PCG_CTLA1/PCG_CTLB1 registers.
2. Program the FS divider/high 10-bit phase divider, enabling both the PCG_CTLA0.CLKEN/PCG_CTLA0.FSEN and the PCG_CTLB0.CLKEN/PCG_CTLB0.FSEN bits.

Note that both units must be disabled in the same way.

External Event Trigger

The trigger with the external clock is is enabled by setting bits 0 and 16 of the PCG_SYNC register. Program the phase to 3, so that the rising edge of the external clock is in-sync with the frame sync (***FS Output Synchronization With External Trigger Input***).

Use the following steps.

1. Program the PCG_SYNC1 - PCG_SYNC4 registers and the PCG_CTLA0 through PCG_CTLB1 registers appropriately.
2. Enable the clock or frame sync, or both.

Since the rising edge of the external clock is used to synchronize with the frame sync, the frame sync output is not generated until a rising edge of the external clock is sensed.

Debug Features

Take care in cases where any input to the phase unit is modified. Any individual change of the PCG_CTLA1.CLKDIV or PCG_CTLA0.FSDIV dividers can cause a failure in the PCG sync operation between the serial clock and the frame sync. Only the programming model ensures a correct setup for phase settings.

ADSP-SC59x PCG Register Descriptions

Precision Clock Generator (PCG) contains the following registers.

Table 33-5: ADSP-SC59x PCG Register List

Name	Description
PCG_CTLA0	Precision Clock A Control 0 Register
PCG_CTLA1	Precision Clock A Control 1 Register
PCG_CTLB0	Precision Clock B Control 0 Register
PCG_CTLB1	Precision Clock B Control 1 Register
PCG_CTLC0	Precision Clock C Control 0 Register
PCG_CTLC1	Precision Clock C Control 1 Register

Table 33-5: ADSP-SC59x PCG Register List (Continued)

Name	Description
PCG_CTLD0	Precision Clock D Control 0 Register
PCG_CTLD1	Precision Clock D Control 1 Register
PCG_CTLE0	Precision Clock E Control 0 Register
PCG_CTLE1	Precision Clock E Control 1 Register
PCG_CTLF0	Precision Clock F Control 0 Register
PCG_CTLF1	Precision Clock F Control 1 Register
PCG_CTLG0	Precision Clock G Control 0 Register
PCG_CTLG1	Precision Clock G Control 1 Register
PCG_CTLH0	Precision Clock H Control 0 Register
PCG_CTLH1	Precision Clock H Control 1 Register
PCG_PW1	Precision Clock Pulse Width Control 1 Register
PCG_PW2	Precision Clock Pulse Width Control 2 Register
PCG_PW3	Precision Clock Pulse Width Control 3 Register
PCG_PW4	Precision Clock Pulse Width Control 4 Register
PCG_SYNC1	Precision Clock Frame Sync Synchronization 1 Register
PCG_SYNC2	Precision Clock Frame Sync Synchronization 2 Register
PCG_SYNC3	Precision Clock Frame Sync Synchronization 3 Register
PCG_SYNC4	Precision Clock Frame Sync Synchronization 4 Register

Precision Clock A Control 0 Register

The PCG_CTLA0 register enables the clock, frame sync, and select divisor for the PCG0 clock A signal.

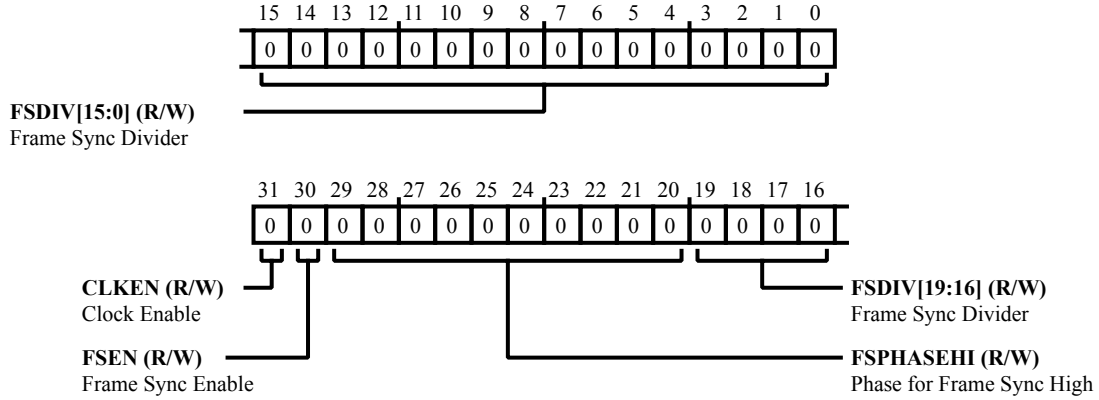


Figure 33-7: PCG_CTLA0 Register Diagram

Table 33-6: PCG_CTLA0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLA0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLA0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLA0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLA0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLA0.FSDIV >1 PCGx is in normal mode, PCG_CTLA0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	PCG is in normal mode

Precision Clock A Control 1 Register

The PCG_CTLA1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock A signal.

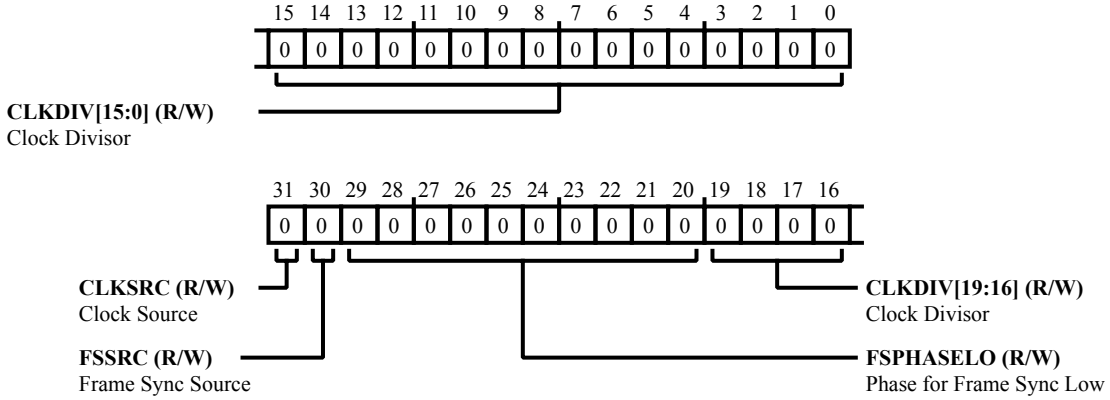


Figure 33-8: PCG_CTLA1 Register Diagram

Table 33-7: PCG_CTLA1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLA1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI0 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLA1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXTX_DAI0 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLA1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLA1.CLKDIV bit field contains the clock divisor value.	

Precision Clock B Control 0 Register

The PCG_CTLB0 register enables the clock, frame sync, and select divisor for the PCG0 clock B signal.

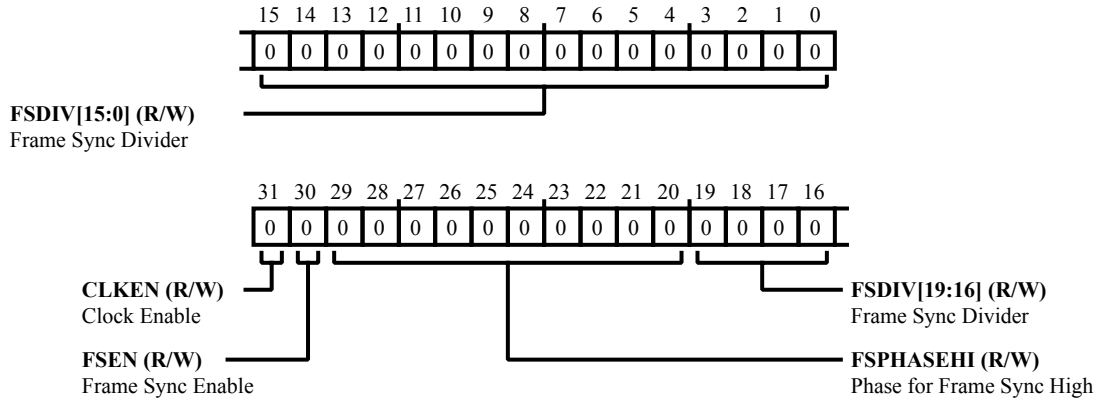


Figure 33-9: PCG_CTLB0 Register Diagram

Table 33-8: PCG_CTLB0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLB0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLB0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLB0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLB0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLB0.FSDIV >1 PCGx is in normal mode, PCG_CTLB0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock B Control 1 Register

The PCG_CTLB1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock B signal.

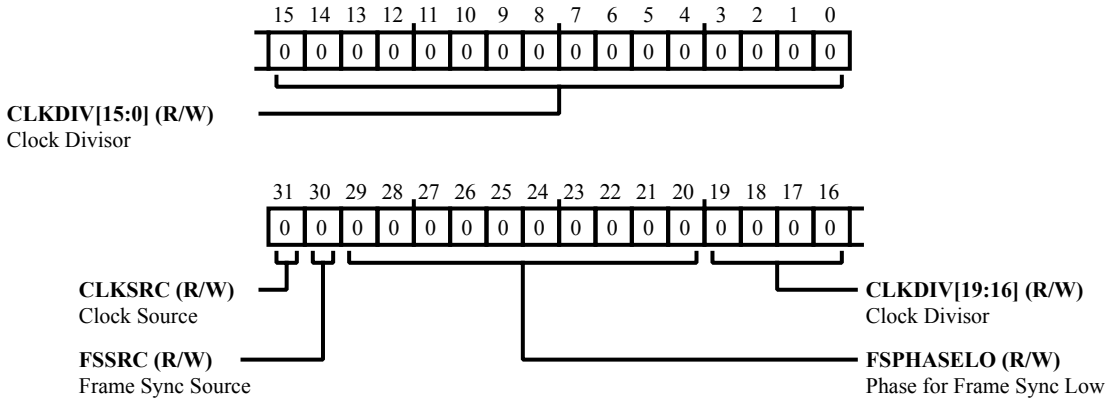


Figure 33-10: PCG_CTLB1 Register Diagram

Table 33-9: PCG_CTLB1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLB1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI0 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLB1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI0 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLB1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLB1.CLKDIV bit field contains the clock divisor value.	

Precision Clock C Control 0 Register

The PCG_CTLC0 register enables the clock, frame sync, and select divisor for the PCG0 clock C signal.

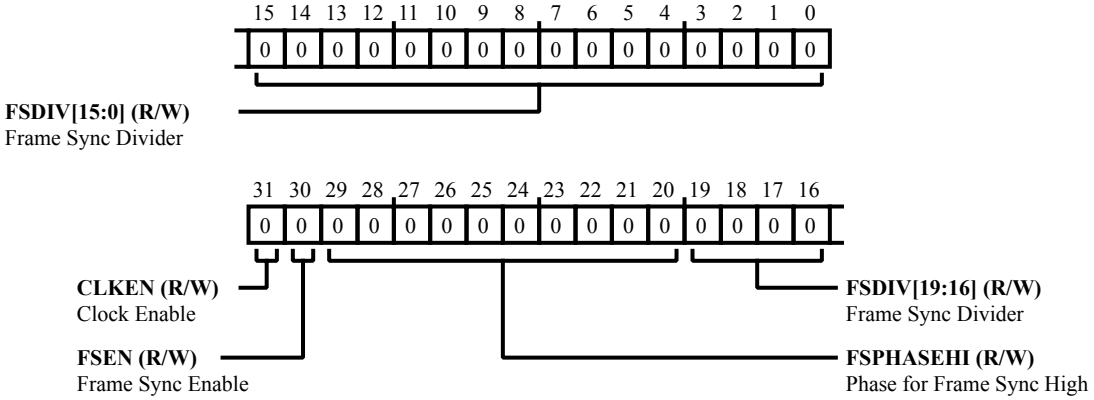


Figure 33-11: PCG_CTLC0 Register Diagram

Table 33-10: PCG_CTLC0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLC0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLC0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLC0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLC0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLC0.FSDIV >1 PCGx is in normal mode, PCG_CTLC0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock C Control 1 Register

The PCG_CTLC1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock C signal.

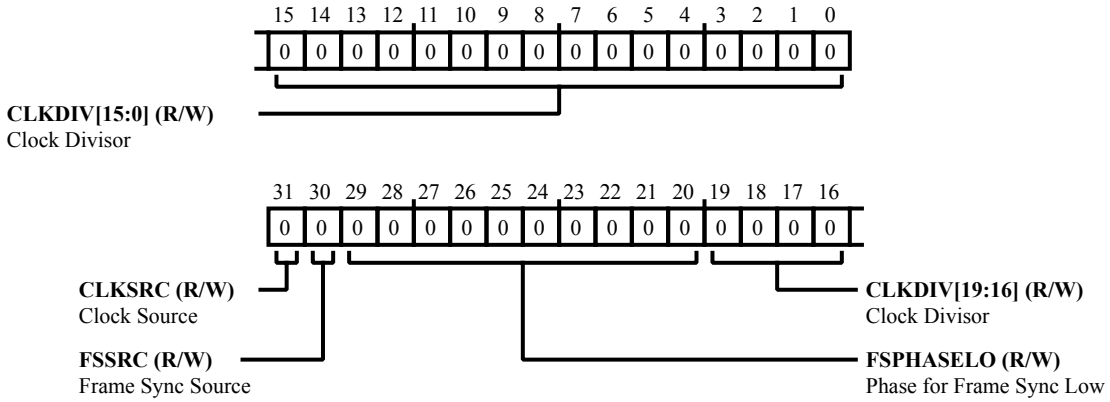


Figure 33-12: PCG_CTLC1 Register Diagram

Table 33-11: PCG_CTLC1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLC1.CLKSRC bit specifies the clock source.	
		0	CLKIN1 pin selected for clock
		1	PCG_EXT_DAI1 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLC1.FSSRC bit specifies the frame sync source.	
		0	CLKIN1 pin selected for frame sync
		1	PCG_EXT_DAI1 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLC1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLC1.CLKDIV bit field contains the clock divisor value.	

Precision Clock D Control 0 Register

The PCG_CTLD0 register enables the clock, frame sync, and select divisor for the PCG0 clock D signal.

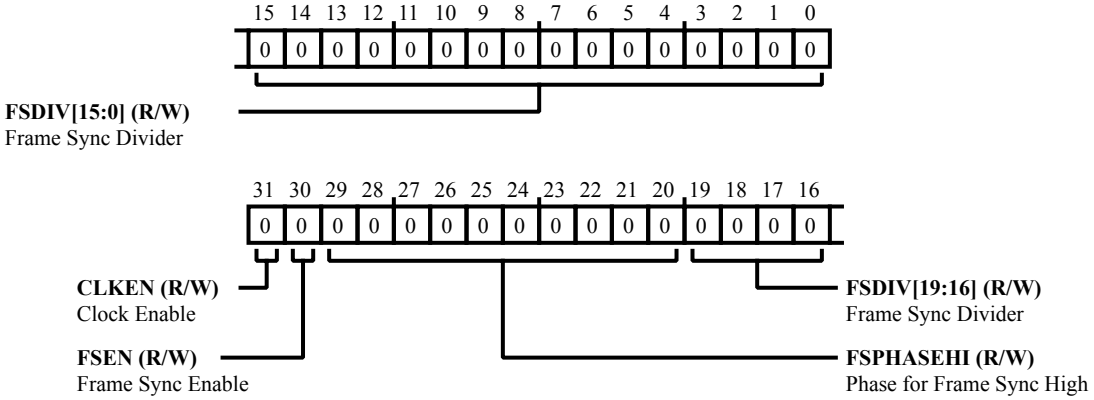


Figure 33-13: PCG_CTLD0 Register Diagram

Table 33-12: PCG_CTLD0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLD0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLD0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLD0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLD0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLD0.FSDIV >1 PCGx is in normal mode, PCG_CTLD0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock D Control 1 Register

The PCG_CTLD1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock D signal.

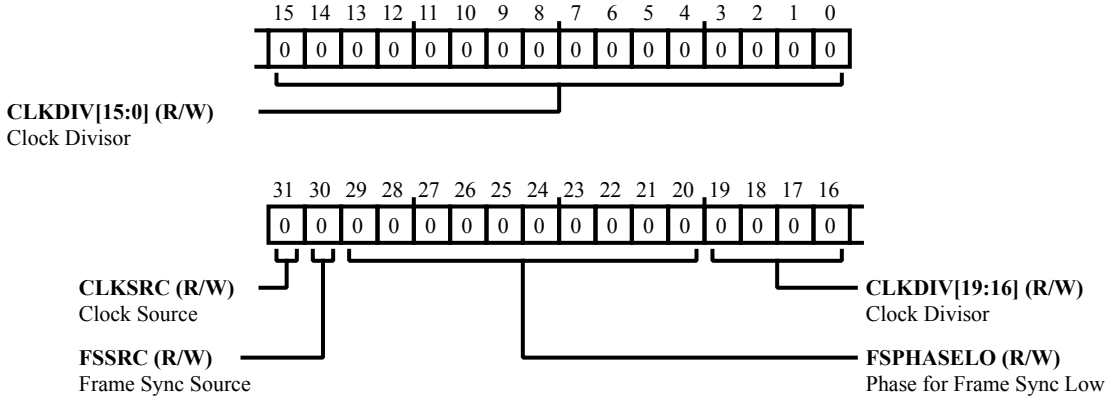


Figure 33-14: PCG_CTLD1 Register Diagram

Table 33-13: PCG_CTLD1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLD1.CLKSRC bit specifies the clock source.	
		0	CLKIN1 pin selected for clock
		1	PCG_EXT_DAI1 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLD1.FSSRC bit specifies the frame sync source.	
		0	CLKIN1 pin selected for frame sync
		1	PCG_EXT_DAI1 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLD1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLD1.CLKDIV bit field contains the clock divisor value.	

Precision Clock E Control 0 Register

The PCG_CTLE0 register enables the clock, frame sync, and select divisor for the PCG0 clock E signal.

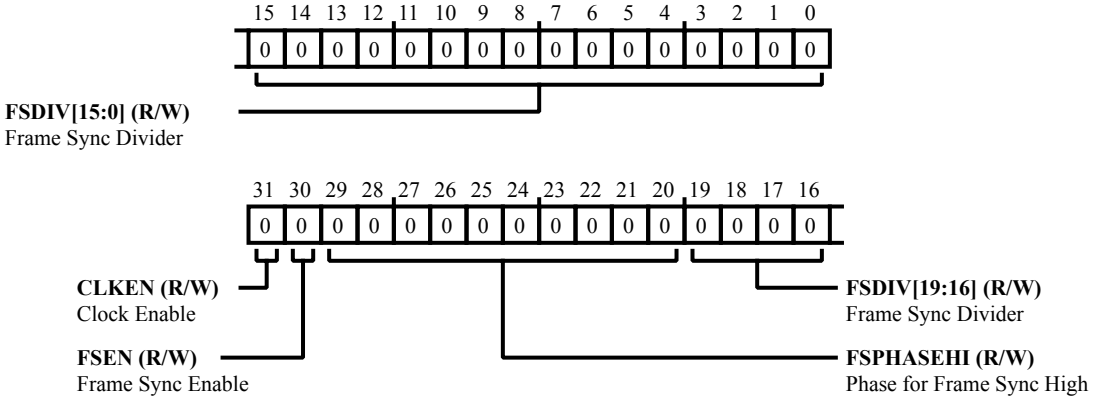


Figure 33-15: PCG_CTLE0 Register Diagram

Table 33-14: PCG_CTLE0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLE0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLE0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLE0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLE0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLE0.FSDIV >1 PCGx is in normal mode, PCG_CTLE0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock E Control 1 Register

The PCG_CTLE1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock E signal.

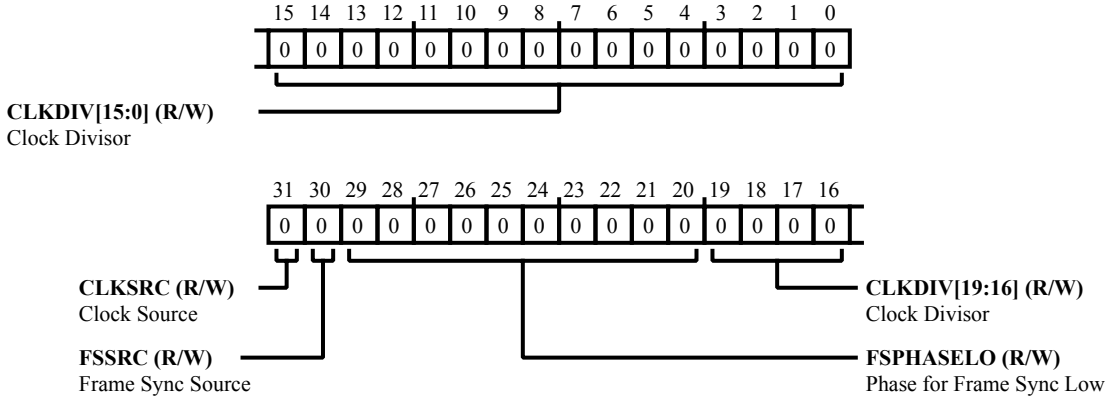


Figure 33-16: PCG_CTLE1 Register Diagram

Table 33-15: PCG_CTLE1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLE1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLE1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLE1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLE1.CLKDIV bit field contains the clock divisor value.	

Precision Clock F Control 0 Register

The PCG_CTLF0 register enables the clock, frame sync, and select divisor for the PCG0 clock F signal.

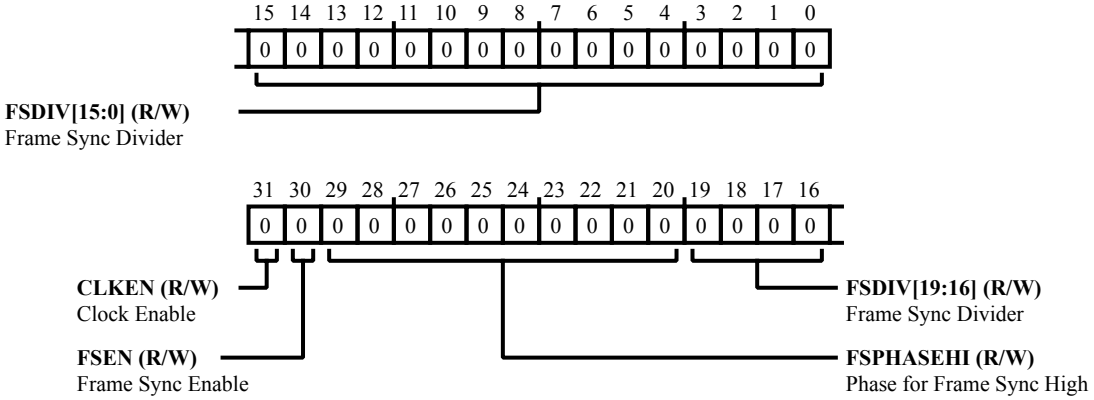


Figure 33-17: PCG_CTLF0 Register Diagram

Table 33-16: PCG_CTLF0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLF0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLF0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLF0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLF0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLF0.FSDIV >1 PCGx is in normal mode, PCG_CTLF0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock F Control 1 Register

The PCG_CTLF1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock F signal.

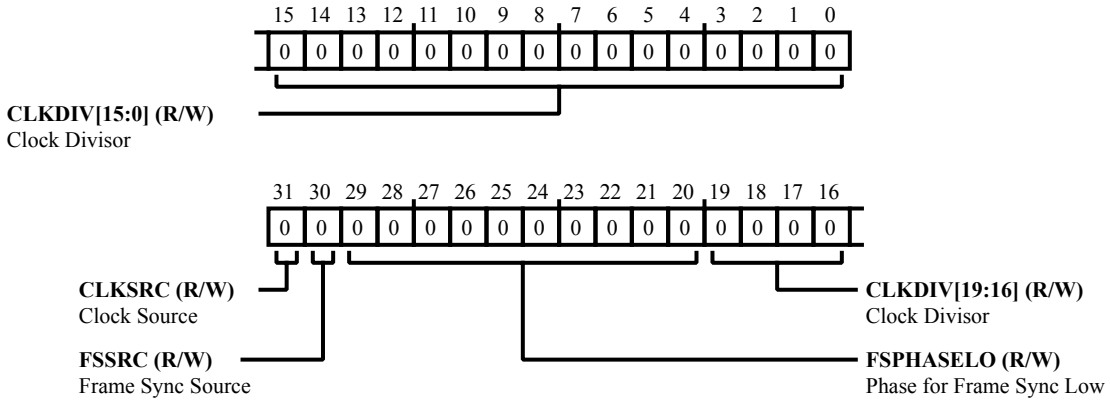


Figure 33-18: PCG_CTLF1 Register Diagram

Table 33-17: PCG_CTLF1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLF1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI0 selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLF1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI0 selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLF1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLF1.CLKDIV bit field contains the clock divisor value.	

Precision Clock G Control 0 Register

The PCG_CTLG0 register enables the clock, frame sync, and select divisor for the PCG0 clock G signal.

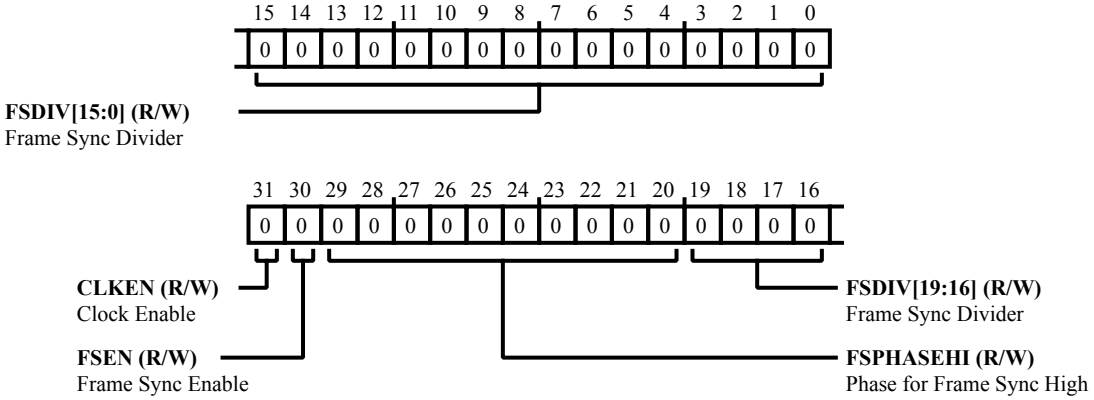


Figure 33-19: PCG_CTLG0 Register Diagram

Table 33-18: PCG_CTLG0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLG0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLG0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLG0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLG0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLG0.FSDIV >1 PCGx is in normal mode, PCG_CTLG0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock G Control 1 Register

The PCG_CTLG1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock G signal.

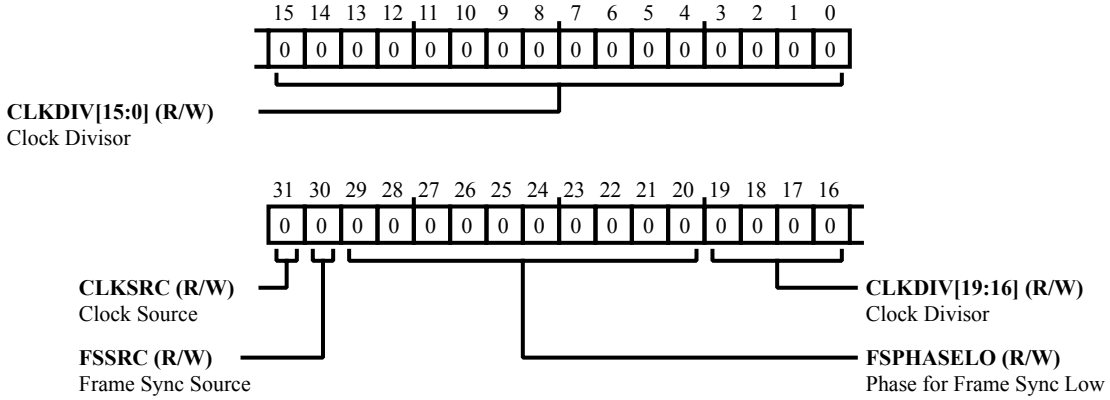


Figure 33-20: PCG_CTLG1 Register Diagram

Table 33-19: PCG_CTLG1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLG1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLG1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLG1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLG1.CLKDIV bit field contains the clock divisor value.	

Precision Clock H Control 0 Register

The PCG_CTLH0 register enables the clock, frame sync, and select divisor for the PCG0 clock H signal.

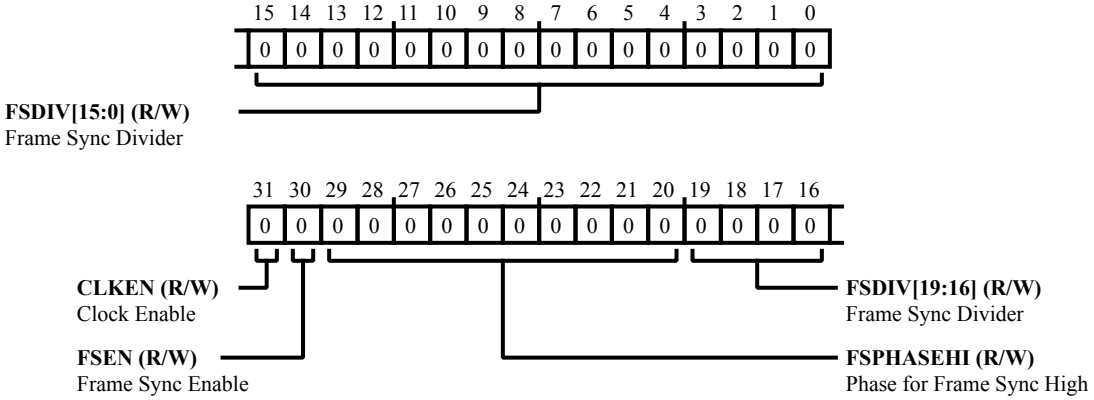


Figure 33-21: PCG_CTLH0 Register Diagram

Table 33-20: PCG_CTLH0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKEN	Clock Enable. The PCG_CTLH0.CLKEN bit enables the clock.	
		0	Clock generation disabled
		1	Clock generation enabled
30 (R/W)	FSEN	Frame Sync Enable. The PCG_CTLH0.FSEN bit enables the frame sync.	
		0	Frame sync generation disabled
		1	Frame sync generation enabled
29:20 (R/W)	FSPHASEHI	Phase for Frame Sync High. The PCG_CTLH0.FSPHASEHI bit field represents the upper half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	FSDIV	Frame Sync Divider. The PCG_CTLH0.FSDIV bit field provides the frame sync divider value. This 20-bit field frame sync divider is multiplexed where: PCG_CTLH0.FSDIV >1 PCGx is in normal mode, PCG_CTLH0.FSDIV = 0, 1 PCGx is in bypass mode.	
		0	PCG is in bypass mode
		1	PCG is in bypass mode
		2-1048575	FSDIV >1 PCG is in normal mode

Precision Clock H Control 1 Register

The PCG_CTLH1 register sets the clock divisor, frame sync source, and clock source for the PCG1 clock H signal.

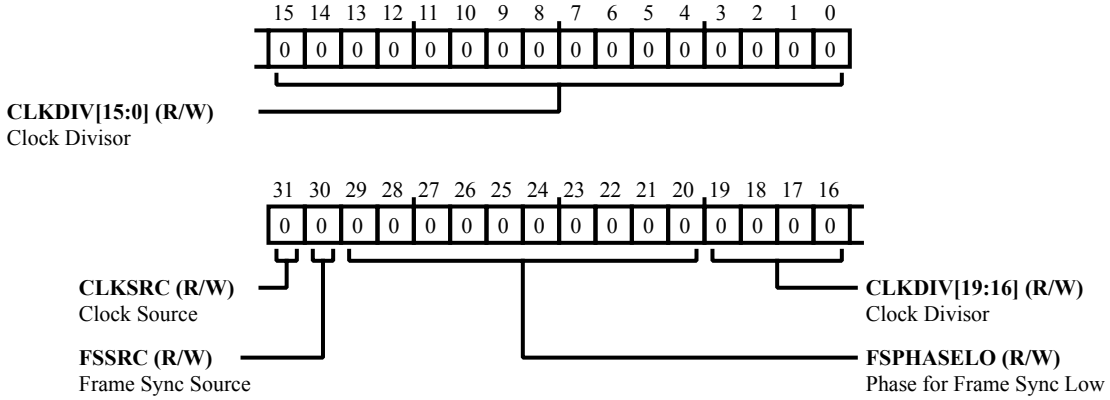


Figure 33-22: PCG_CTLH1 Register Diagram

Table 33-21: PCG_CTLH1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CLKSRC	Clock Source. The PCG_CTLH1.CLKSRC bit specifies the clock source.	
		0	CLKIN0 pin selected for clock
		1	PCG_EXT_DAI selected for clock
30 (R/W)	FSSRC	Frame Sync Source. The PCG_CTLH1.FSSRC bit specifies the frame sync source.	
		0	CLKIN0 pin selected for frame sync
		1	PCG_EXT_DAI selected for frame sync
29:20 (R/W)	FSPHASELO	Phase for Frame Sync Low. The PCG_CTLH1.FSPHASELO bit field represents the lower half of the 20-bit value for the channel A/B/C/D frame sync phase.	
19:0 (R/W)	CLKDIV	Clock Divisor. The PCG_CTLH1.CLKDIV bit field contains the clock divisor value.	

Precision Clock Pulse Width Control 1 Register

The PCG_PW1 register sets the one shot frame sync and the active low frame sync select for PCG A and PCG B.

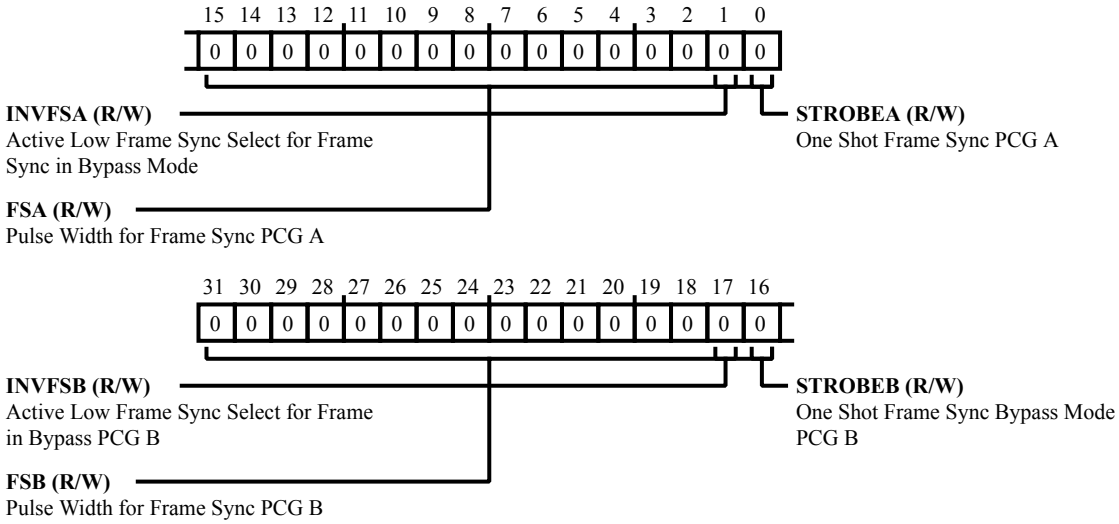


Figure 33-23: PCG_PW1 Register Diagram

Table 33-22: PCG_PW1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	INVFSB	Active Low Frame Sync Select for Frame in Bypass PCG B. The PCG_PW1.INVFSB bit selects active low or active high frame sync in bypass mode for PCG B.
16 (R/W)	STROBEB	One Shot Frame Sync Bypass Mode PCG B. The PCG_PW1.STROBEB bit sets the frame sync pulse in bypass mode for PCG B. This is the duration equal to one period of the DAI_MISCA3_I signal (PCG B) repeating at the beginning of every frame.
31:16 (R/W)	FSB	Pulse Width for Frame Sync PCG B. The PCG_PW1.FSB bit field sets the number of input clock periods for which the frame sync output is high. Pulse width should be less than the divisor of the frame sync.
1 (R/W)	INVFSA	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW1.INVFSA bit selects active low or active high frame sync for PCG A in bypass mode.
15:0 (R/W)	FSA	Pulse Width for Frame Sync PCG A. The PCG_PW1.FSA bit field sets the number of input clock periods for which the frame sync output is high. Pulse width should be less than the divisor of the frame sync.

Table 33-22: PCG_PW1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	STROBEA	One Shot Frame Sync PCG A. The `PCG_PW1.STROBEA` bit sets the frame sync pulse for PCG A in bypass mode. This is the duration equal to one period of the DAI_MISCA2_I signal (PCG A) repeating at the beginning of every frame.

Precision Clock Pulse Width Control 2 Register

The PCG_PW2 register sets the one shot frame sync and the active low frame sync select for PCG C and PCG D.

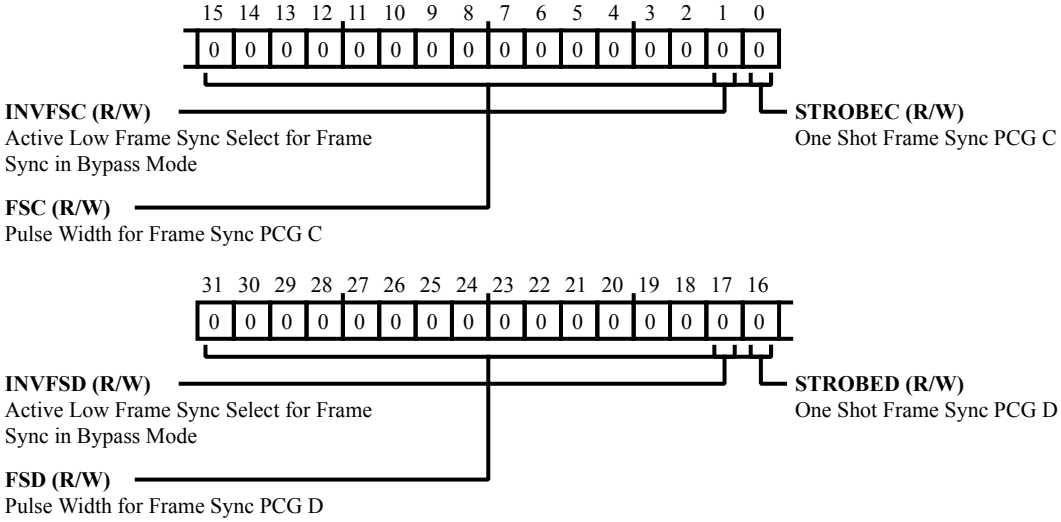


Figure 33-24: PCG_PW2 Register Diagram

Table 33-23: PCG_PW2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	INVFSD	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW2.INVFSD bit selects active low or active high frame sync for PCG D in bypass mode.
16 (R/W)	STROBED	One Shot Frame Sync PCG D. The PCG_PW2.STROBED bit sets the frame sync pulse for PCG D in bypass mode. This is the duration equal to one period of the DAI_MISCA3_I signal (PCG D) repeating at the beginning of every frame.
31:16 (R/W)	FSD	Pulse Width for Frame Sync PCG D. The PCG_PW2.FSD bit field sets the number of input clock periods for which the frame sync output is high for PCG D. Pulse width should be less than the divisor of the frame sync.
1 (R/W)	INVFSC	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW2.INVFSC bit selects active low or active high frame sync for PCG C in bypass mode.
15:0 (R/W)	FSC	Pulse Width for Frame Sync PCG C. The PCG_PW2.FSC bit field sets the number of input clock periods for which the frame sync output is high for PCG C. Pulse width should be less than the divisor of the frame sync.

Table 33-23: PCG_PW2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	STROBEC	One Shot Frame Sync PCG C. The `PCG_PW2.STROBEC` bit sets the frame sync pulse for PCG C in bypass mode. This is the duration equal to one period of the DAI_MISCA2_I signal (PCG C) repeating at the beginning of every frame.

Precision Clock Pulse Width Control 3 Register

The PCG_PW3 register sets the one shot frame sync and the active low frame sync select for PCG E and PCG F.

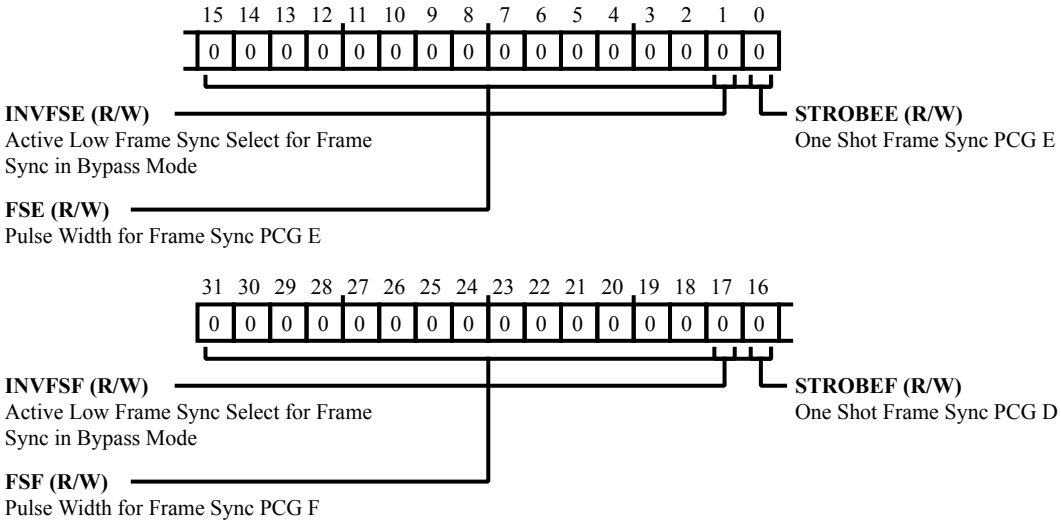


Figure 33-25: PCG_PW3 Register Diagram

Table 33-24: PCG_PW3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	INVFSF	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW3.INVFSF bit selects active low or active high frame sync for PCG F in bypass mode.
16 (R/W)	STROBEF	One Shot Frame Sync PCG D. The PCG_PW3.STROBEF bit sets the frame sync pulse for PCG F in bypass mode. This is the duration equal to one period of the DAI_MISCA1_I signal (PCG F) repeating at the beginning of every frame.
31:16 (R/W)	FSF	Pulse Width for Frame Sync PCG F. The PCG_PW3.FSF bit field sets the number of input clock periods for which the frame sync output is high for PCG F. Pulse width should be less than the divisor of the frame sync.
1 (R/W)	INVFSE	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW3.INVFSE bit selects active low or active high frame sync for PCG E in bypass mode.
0 (R/W)	STROBEE	One Shot Frame Sync PCG E. The PCG_PW3.STROBEE bit sets the frame sync pulse for PCG E in bypass mode. This is the duration equal to one period of the DAI_MISCA0_I signal (PCG E) repeating at the beginning of every frame.

Table 33-24: PCG_PW3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	FSE	Pulse Width for Frame Sync PCG E. The `PCG_PW3.FSE` bit field sets the number of input clock periods for which the frame sync output is high for PCG E. Pulse width should be less than the divisor of the frame sync.

Precision Clock Pulse Width Control 4 Register

The PCG_PW4 register sets the one shot frame sync and the active low frame sync select for PCG G and PCG H.

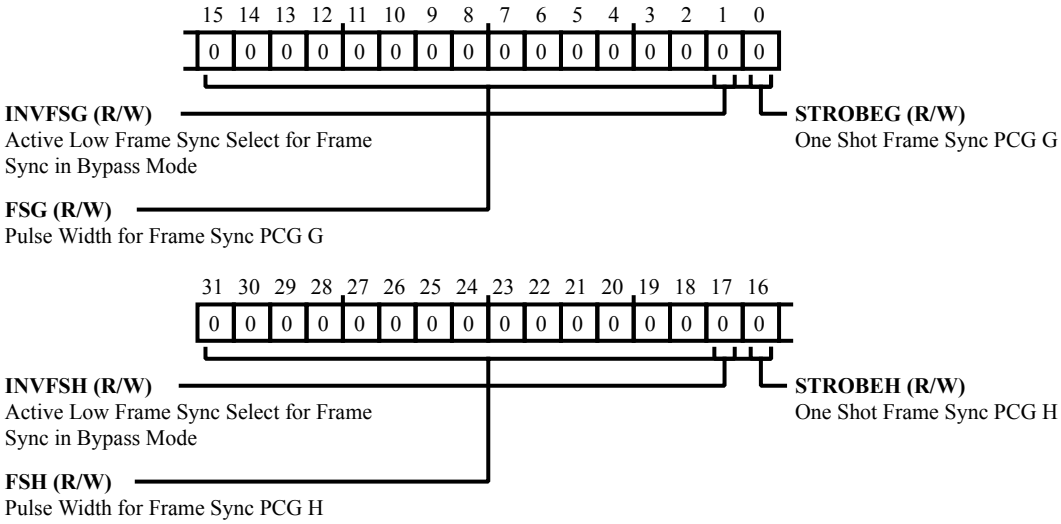


Figure 33-26: PCG_PW4 Register Diagram

Table 33-25: PCG_PW4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
17 (R/W)	INVFSH	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW4.INVFSH bit selects active low or active high frame sync for PCG H in bypass mode.
16 (R/W)	STROBEH	One Shot Frame Sync PCG H. The PCG_PW4.STROBEH bit sets the frame sync pulse for PCG H in bypass mode. This is the duration equal to one period of the DAI_MISCA1_I signal (PCG H) repeating at the beginning of every frame.
31:16 (R/W)	FSH	Pulse Width for Frame Sync PCG H. The PCG_PW4.FSH bit field sets the number of input clock periods for which the frame sync output is high for PCG H. Pulse width should be less than the divisor of the frame sync.
1 (R/W)	INVFSG	Active Low Frame Sync Select for Frame Sync in Bypass Mode. The PCG_PW4.INVFSG bit selects active low or active high frame sync for PCG G in bypass mode.
15:0 (R/W)	FSG	Pulse Width for Frame Sync PCG G. The PCG_PW4.FSG bit field sets the number of input clock periods for which the frame sync output is high for PCG G. Pulse width should be less than the divisor of the frame sync.

Table 33-25: PCG_PW4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	STROBEG	One Shot Frame Sync PCG G. The `PCG_PW4.STROBEG` bit sets the frame sync pulse for PCG G in bypass mode. This is the duration equal to one period of the DAI_MISCA0_I signal (PCG G) repeating at the beginning of every frame.

Precision Clock Frame Sync Synchronization 1 Register

The PCG_SYNC1 register allows programs to synchronize the clock frame sync units with external frame syncs. Note that the PCG_CTLA1.CLKSRC bit is overridden if PCG_SYNC1.CLKASRC bit in the PCG_SYNC1 register is set.

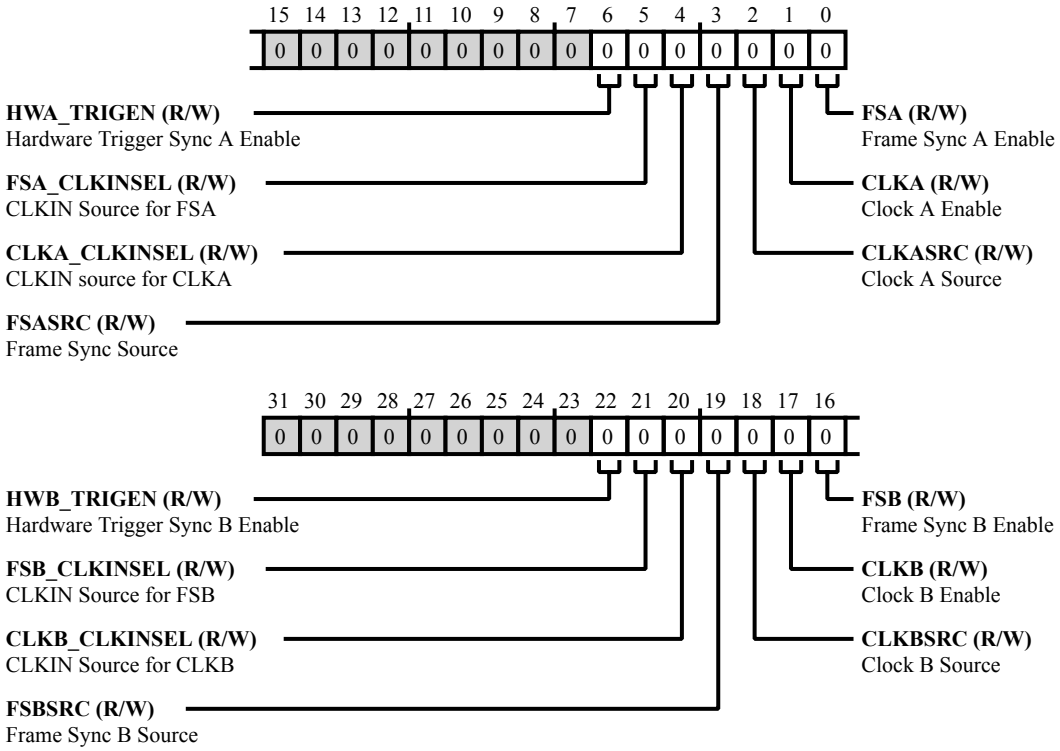


Figure 33-27: PCG_SYNC1 Register Diagram

Table 33-26: PCG_SYNC1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	HWB_TRIGEN	Hardware Trigger Sync B Enable. The PCG_SYNC1.HWB_TRIGEN bit enables synchronization of clock B and frame sync B with the external frame sync only after an internal receiver trigger for PCG B is received.	
21 (R/W)	FSB_CLKINSEL	CLKIN Source for FSB. The PCG_SYNC1.FSB_CLKINSEL bit enables the CLKIN input source for FSB.	
		0	CLKIN0
		1	CLKIN1

Table 33-26: PCG_SYNC1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	CLKB_CLKINSEL	CLKIN Source for CLKB. The `PCG_SYNC1.CLKB_CLKINSEL` bit enables the CLKIN input source for CLKB.	
		0	CLKIN0
		1	CLKIN1
19 (R/W)	FSBSRC	Frame Sync B Source. The `PCG_SYNC1.FSBSRC` bit enables the frame sync B input source.	
		0	Output selected by FSBSOURCE bit
		1	Clock derived from core PLL selected for frame sync B
18 (R/W)	CLKBSRC	Clock B Source. The `PCG_SYNC1.CLKBSRC` bit enables the clock B input source.	
		0	Output selected by CLKBSOURCE bit
		1	Clock derived from core PLL selected for clock B
17 (R/W)	CLKB	Clock B Enable. The `PCG_SYNC1.CLKB` bit enables synchronization of clock B with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
16 (R/W)	FSB	Frame Sync B Enable. The `PCG_SYNC1.FSB` bit enables synchronization of frame sync B with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled
6 (R/W)	HWA_TRIGEN	Hardware Trigger Sync A Enable. The `PCG_SYNC1.HWA_TRIGEN` bit enables synchronization of clock A and frame sync A with the external frame sync only after an internal receiver trigger for PCG A is received.	
5 (R/W)	FSA_CLKINSEL	CLKIN Source for FSA. The `PCG_SYNC1.FSA_CLKINSEL` bit enables the CLKIN input source for FSA.	
		0	CLKIN0
		1	CLKIN1

Table 33-26: PCG_SYNC1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	CLKA_CLKINSEL	CLKIN source for CLKA. The `PCG_SYNC1.CLKA_CLKINSEL` bit enables the CLKIN input source for CLKA	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
3 (R/W)	FSASRC	Frame Sync Source. The `PCG_SYNC1.FSASRC` bit enables the frame sync A input source.	
		0	Output selected by FSASOURCE bit
		1	Clock derived from core PLL selected for frame sync A
2 (R/W)	CLKASRC	Clock A Source. The `PCG_SYNC1.CLKASRC` bit enables the clock A input source.	
		0	Output selected by CLKASOURCE bit
		1	Clock derived from core PLL selected for clock A
1 (R/W)	CLKA	Clock A Enable. The `PCG_SYNC1.CLKA` bit enables synchronization of clock A with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
0 (R/W)	FSA	Frame Sync A Enable. The `PCG_SYNC1.FSA` bit enables synchronization of frame sync A with the external frame sync.	
		0	Frame Sync Disabled
		1	Frame Sync Enabled

Precision Clock Frame Sync Synchronization 2 Register

The PCG_SYNC2 register allows programs to synchronize the clock frame sync units with external frame syncs. Note that the PCG_CTLD1.CLKSRC bit is overridden if PCG_SYNC2.CLKDSRC bit in the PCG_SYNC2 register is set.

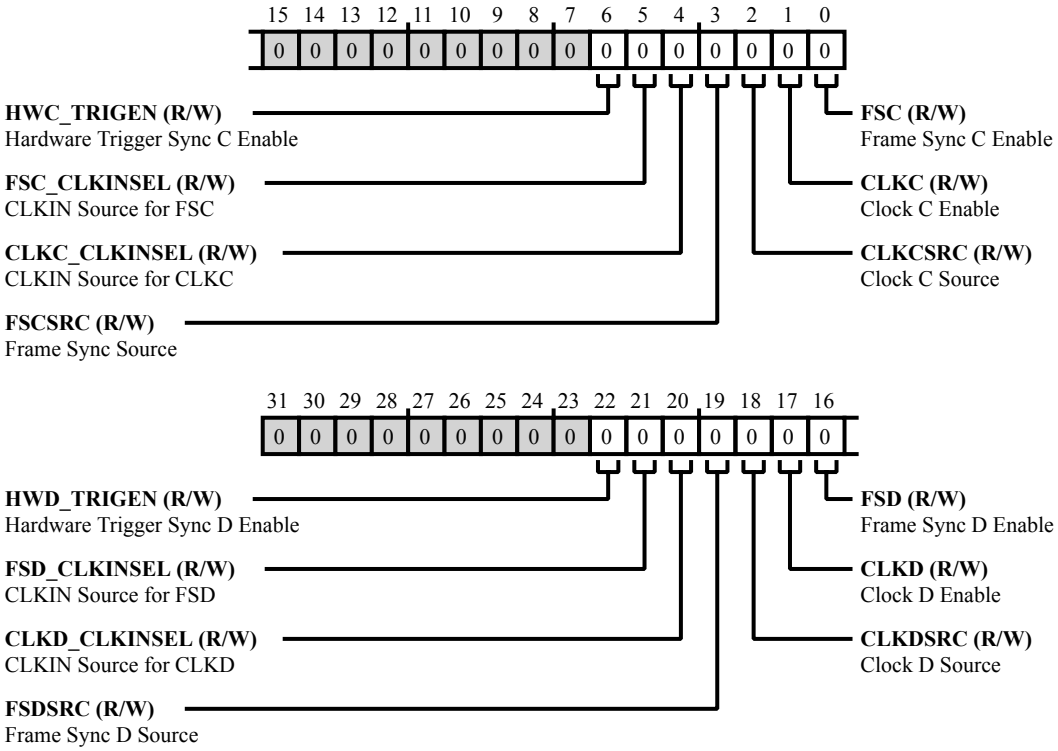


Figure 33-28: PCG_SYNC2 Register Diagram

Table 33-27: PCG_SYNC2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	HWD_TRIGEN	Hardware Trigger Sync D Enable. The PCG_SYNC2.HWD_TRIGEN bit enables synchronization of clock D and frame sync D with the external frame sync only after an internal receiver trigger for PCG D is received.	
21 (R/W)	FSD_CLKINSEL	CLKIN Source for FSD. The PCG_SYNC2.FSD_CLKINSEL bit enables the CLKIN input source for FSD.	
		0	Selects CLKIN0 source for FSB CLKIN
		1	Selects CLKIN1 source for FSB CLKIN

Table 33-27: PCG_SYNC2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	CLKD_CLKINSEL	CLKIN Source for CLKD. The `PCG_SYNC2.CLKD_CLKINSEL` bit enables the CLKIN input source for CLKD.	
		0	Select CLKIN0 source for CLKB CLKIN
		1	Select CLKIN1 source for CLKB CLKIN
19 (R/W)	FSDSRC	Frame Sync D Source. The `PCG_SYNC2.FSDSRC` bit enables the frame sync D input source.	
		0	Output selected by FSDSOURCE bit
		1	Clock derived from core PLL selected for frame sync D
18 (R/W)	CLKDSRC	Clock D Source. The `PCG_SYNC2.CLKDSRC` bit enables the clock D input source.	
		0	Output selected by CLKDSOURCE bit
		1	Clock derived from core PLL selected for clock D
17 (R/W)	CLKD	Clock D Enable. The `PCG_SYNC2.CLKD` bit enables synchronization of clock D with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
16 (R/W)	FSD	Frame Sync D Enable. The `PCG_SYNC2.FSD` bit enables synchronization of frame sync D with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled
6 (R/W)	HWC_TRIGEN	Hardware Trigger Sync C Enable. The `PCG_SYNC2.HWC_TRIGEN` bit enables synchronization of clock C and frame sync C with the external frame sync only after an internal receiver trigger for PCG C is received.	
5 (R/W)	FSC_CLKINSEL	CLKIN Source for FSC. The `PCG_SYNC2.FSC_CLKINSEL` bit enables the CLKIN input source for FSC.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN

Table 33-27: PCG_SYNC2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	CLKC_CLKINSEL	CLKIN Source for CLKC. The `PCG_SYNC2.CLKC_CLKINSEL` bit enables the CLKIN input source for CLKC.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
3 (R/W)	FSCSRC	Frame Sync Source. The `PCG_SYNC2.FSCSRC` bit enables the frame sync C input source.	
		0	Output selected by FSCSOURCE bit
		1	Clock derived from core PLL selected for frame sync C
2 (R/W)	CLKCSRC	Clock C Source. The `PCG_SYNC2.CLKCSRC` bit enables the clock C input source.	
		0	Output selected by CLKCSOURCE bit
		1	Clock derived from core PLL selected for clock C
1 (R/W)	CLKC	Clock C Enable. The `PCG_SYNC2.CLKC` bit enables synchronization of clock C with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
0 (R/W)	FSC	Frame Sync C Enable. The `PCG_SYNC2.FSC` bit enables synchronization of frame sync C with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled

Precision Clock Frame Sync Synchronization 3 Register

The PCG_SYNC3 register allows programs to synchronize the clock frame sync units with external frame syncs. Note that the PCG_CTLD1.CLKSRC bit is overridden if CLKDSRC bit in the PCG_SYNC3 register is set.

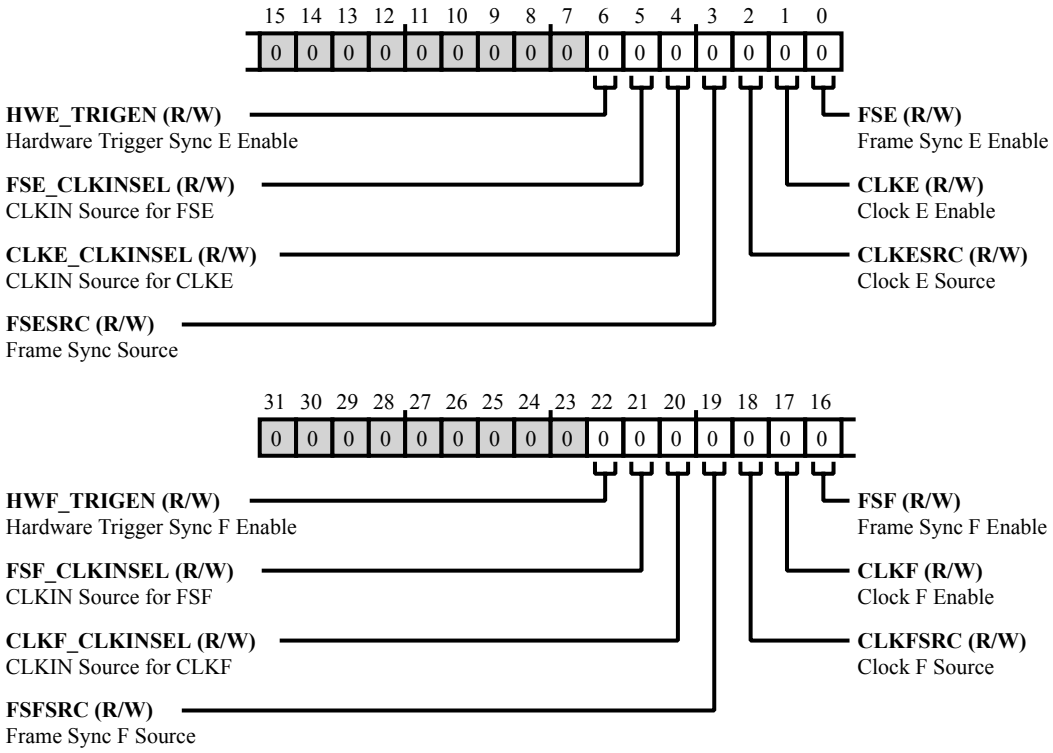


Figure 33-29: PCG_SYNC3 Register Diagram

Table 33-28: PCG_SYNC3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	HWF_TRIGEN	Hardware Trigger Sync F Enable. The PCG_SYNC3.HWF_TRIGEN bit enables synchronization of clock F and frame sync F with the external frame sync only after an internal receiver trigger for PCG-F is received.	
21 (R/W)	FSF_CLKINSEL	CLKIN Source for FSF. The PCG_SYNC3.FSF_CLKINSEL bit enables the CLKIN input source for FSF.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN

Table 33-28: PCG_SYNC3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	CLKF_CLKINSEL	CLKIN Source for CLKF. The `PCG_SYNC3.CLKF_CLKINSEL` bit enables the CLKIN input source for CLKF.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
19 (R/W)	FSFSRC	Frame Sync F Source. The `PCG_SYNC3.FSFSRC` bit enables the frame sync F input source.	
		0	Output selected by FSDSOURCE bit
		1	Clock derived from core PLL selected for frame sync D
18 (R/W)	CLKFSRC	Clock F Source. The `PCG_SYNC3.CLKFSRC` bit enables the clock F input source.	
		0	Output selected by CLKDSOURCE bit
		1	Clock derived from core PLL selected for clock D
17 (R/W)	CLKF	Clock F Enable. The `PCG_SYNC3.CLKF` bit enables synchronization of clock F with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
16 (R/W)	FSF	Frame Sync F Enable. The `PCG_SYNC3.FSF` bit enables synchronization of frame sync F with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled
6 (R/W)	HWE_TRIGEN	Hardware Trigger Sync E Enable. The `PCG_SYNC3.HWE_TRIGEN` bit enables synchronization of clock E and frame sync E with the external frame sync only after an internal receiver trigger for PCG-E is received.	
5 (R/W)	FSE_CLKINSEL	CLKIN Source for FSE. The `PCG_SYNC3.FSE_CLKINSEL` bit enables the CLKIN input source for FSE.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN

Table 33-28: PCG_SYNC3 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	CLKE_CLKINSEL	CLKIN Source for CLKE. The `PCG_SYNC3.CLKE_CLKINSEL` bit enables the CLKIN input source for CLKE.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
3 (R/W)	FSESRC	Frame Sync Source. The `PCG_SYNC3.FSESRC` bit enables the frame sync E input source.	
		0	Output selected by FSCSOURCE bit
		1	Clock derived from core PLL selected for frame sync C
2 (R/W)	CLKESRC	Clock E Source. The `PCG_SYNC3.CLKESRC` bit enables the clock E input source.	
		0	Output selected by CLKCSOURCE bit
		1	Clock derived from core PLL selected for clock C
1 (R/W)	CLKE	Clock E Enable. The `PCG_SYNC3.CLKE` bit enables synchronization of clock E with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
0 (R/W)	FSE	Frame Sync E Enable. The `PCG_SYNC3.FSE` bit enables synchronization of frame sync E with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled

Precision Clock Frame Sync Synchronization 4 Register

The PCG_SYNC4 register allows programs to synchronize the clock frame sync units with external frame syncs. Note that the PCG_CTLD1.CLKSRC bit is overridden if CLKDSRC bit in the PCG_SYNC4 register is set.

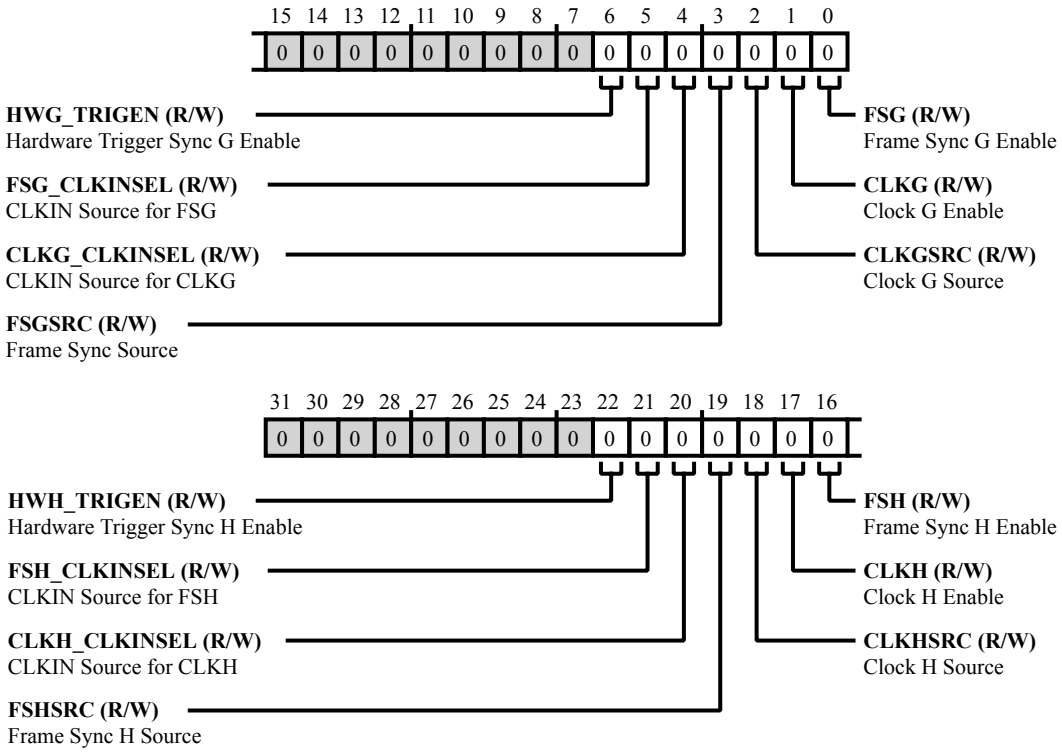


Figure 33-30: PCG_SYNC4 Register Diagram

Table 33-29: PCG_SYNC4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	HWH_TRIGEN	Hardware Trigger Sync H Enable. The PCG_SYNC4.HWH_TRIGEN bit enables synchronization of clock H and frame sync H with the external frame sync only after an internal receiver trigger for PCG-H is received.	
21 (R/W)	FSH_CLKINSEL	CLKIN Source for FSH. The PCG_SYNC4.FSH_CLKINSEL bit enables the CLKIN input source for FSH.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN

Table 33-29: PCG_SYNC4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	CLKH_CLKINSEL	CLKIN Source for CLKH. The `PCG_SYNC4.CLKH_CLKINSEL` bit enables the CLKIN input source for CLKH.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
19 (R/W)	FSHSRC	Frame Sync H Source. The `PCG_SYNC4.FSHSRC` bit enables the frame sync H input source.	
		0	Output selected by FSDSOURCE bit
		1	Clock derived from core PLL selected for frame sync D
18 (R/W)	CLKHSRC	Clock H Source. The `PCG_SYNC4.CLKHSRC` bit enables the clock H input source.	
		0	Output selected by CLKDSOURCE bit
		1	Clock derived from core PLL selected for clock D
17 (R/W)	CLKH	Clock H Enable. The `PCG_SYNC4.CLKH` bit enables synchronization of clock H with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
16 (R/W)	FSH	Frame Sync H Enable. The `PCG_SYNC4.FSH` bit enables synchronization of frame sync H with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled
6 (R/W)	HWG_TRIGEN	Hardware Trigger Sync G Enable. The `PCG_SYNC4.HWG_TRIGEN` bit enables synchronization of clock G and frame sync G with the external frame sync only after an internal receiver trigger for PCG G is received.	
5 (R/W)	FSG_CLKINSEL	CLKIN Source for FSG. The `PCG_SYNC4.FSG_CLKINSEL` bit enables the CLKIN input source for FSG.	
		0	Selects CLKIN0 as source for FSA CLKIN
		1	Selects CLKIN1 as source for FSA CLKIN

Table 33-29: PCG_SYNC4 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	CLKG_CLKINSEL	CLKIN Source for CLKG. The `PCG_SYNC4.CLKG_CLKINSEL` bit enables the CLKIN input source for CLKG.	
		0	Select CLKIN0 source for CLKA CLKIN
		1	Select CLKIN1 source for CLKA CLKIN
3 (R/W)	FSGSRC	Frame Sync Source. The `PCG_SYNC4.FSGSRC` bit enables the frame sync G input source.	
		0	Output selected by FSCSOURCE bit
		1	Clock derived from core PLL selected for frame sync C
2 (R/W)	CLKGSRC	Clock G Source. The `PCG_SYNC4.CLKGSRC` bit enables the clock G input source.	
		0	Output selected by CLKCSOURCE bit
		1	Clock derived from core PLL selected for clock C
1 (R/W)	CLKG	Clock G Enable. The `PCG_SYNC4.CLKG` bit enables synchronization of clock G with the external frame sync.	
		0	Clock disabled
		1	Clock enabled
0 (R/W)	FSG	Frame Sync G Enable. The `PCG_SYNC4.FSG` bit enables synchronization of frame sync G with the external frame sync.	
		0	Frame sync disabled
		1	Frame sync enabled

34 Asynchronous Sample Rate Converter (ASRC)

Sample rate converters (SRC) are frequently used in digital signal processing audio applications. The most frequently used sample rate conversions are off-loaded into hardware modules that are dedicated for filter processing and reduce the instruction processing load on the core, freeing it up for other tasks.

Features

The ASRC has the following features and capabilities:

- 4 asynchronous stereo SRCs operating in completer mode are available in each DAI.
- Simple programming model
- Controllable muting options (hardware, software and automatic)
- Automatically senses input and output sample frequencies
- Supports left-justified, I^2S, right-justified (16-,18-, 20-, 24-bits), and TDM serial port modes
- Daisy-chain configuration in TDM modes (including between DAI0 and DAI1) for input and output ports to create a serial frame
- Different protocols on input/output port allow format conversions
- De-emphasis filter for 32, 44.1 and 48 kHz sampling frequencies
- Up to 192 kHz sample rate input/output continuous sample ratios from 7.5:1 to 1:8
- Group delay (latency of interpolation filter) is 16 samples
- SNR from 128 to 140 dB (depending on processor model)
- Matched phase mode available to compensate for group delays
- Can be used to de-jitter clocks in systems

Functional Description

Conceptually, the sample rate converter interpolates the serial input data at a rate of 220 and samples the interpolated data stream by the output sample rate. In practice, a 64-tap FIR filter with 220 polyphases, a FIFO, a digital servo loop that measures the time difference between the input and output samples within 5 ps, and a digital circuit to track the sample rate ratio are used to perform the interpolation and output sampling.

ADSP-SC59x ASRC Register List

Sample Rate Converter Module

Table 34-1: ADSP-SC59x ASRC Register List

Name	Description
ASRC_CTL01	Control Register for ASRC 0 and 1
ASRC_CTL23	Control Register for ASRC 2 and 3
ASRC_MUTE	Mute Register
ASRC_RAT01	Ratio Register for ASRC 0 and 1
ASRC_RAT23	Ratio Register for ASRC 2 and 3

ASRC Interrupt List

The ASRC interrupts are controlled through the DAI.

Table 34-2: ASRC Interrupt List

Interrupt ID	Interrupt Name	Interrupt Condition
20	DAI0_IRQH	ASRC initialization ASRC sample rate change
21	DAI1_IRQH	
116	DAI0_IRQL	
117	DAI1_IRQL	

ASRC Block Diagram

The ***ASRC Block Diagram*** figure shows a top level block diagram of the ASRC module and the ***Core Architecture*** figure shows architecture details.

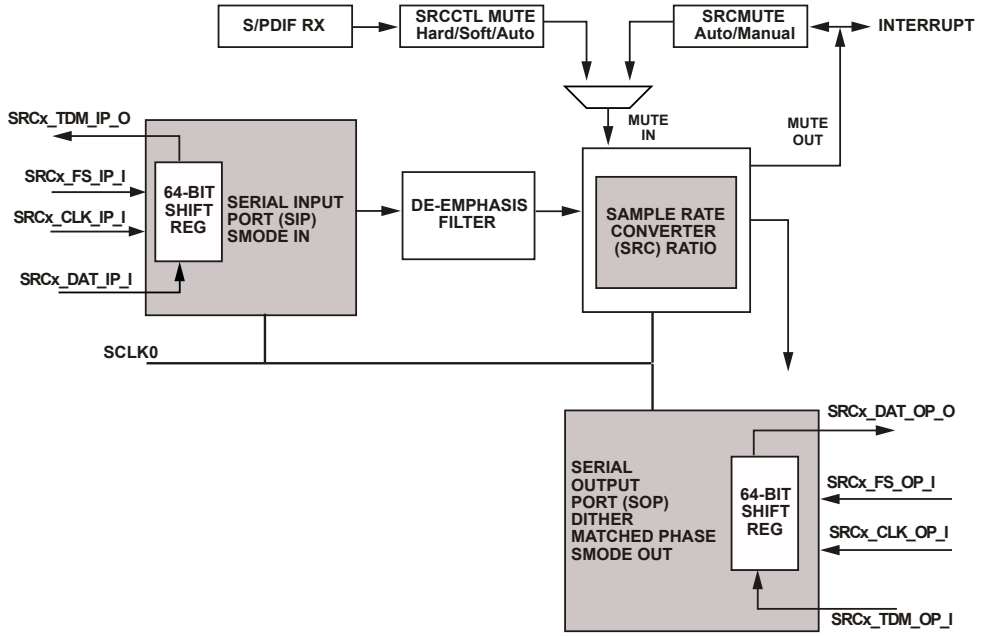


Figure 34-1: ASRC Block Diagram

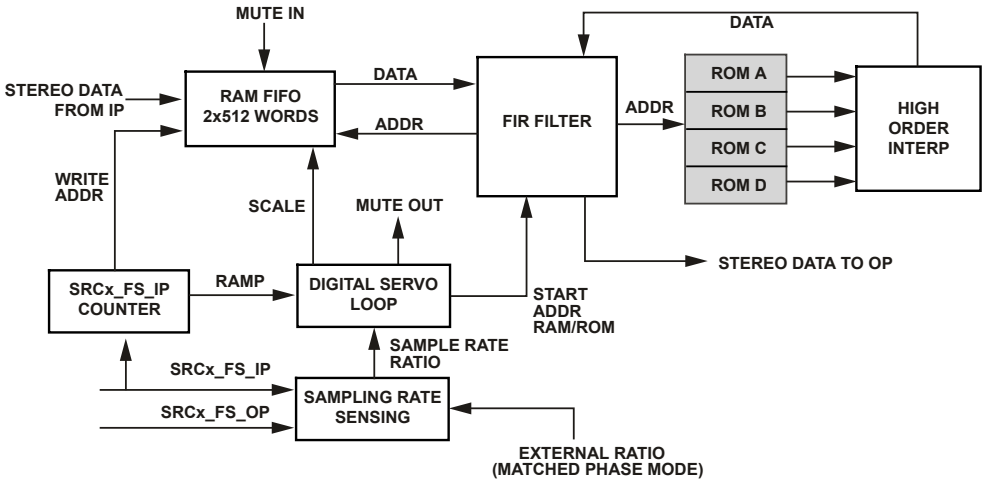


Figure 34-2: ASRC Core Architecture

SRU Programming

The SRU (signal routing unit) needs to be programmed to connect the ASRCs to the output pins or any other peripherals. For more information, see the Digital Audio Interface (DAI) chapter.

Clocking

The ASRC module is in the SCLK0 clock domain. An internal divided version of the SCLK0 clock is generated and used as the fundamental clock for the ASRC module.

I/O Ports

The I/O ports provide the interface through which data is transferred asynchronously into and out of the SRC modules. The SRC has a 3-wire interface for the serial input and output ports that supports left-justified, I^2S, and right-justified (16-, 18-, 20-, 24-bit) modes. Additionally, the serial interfaces support TDM mode for daisy-chaining multiple SRCs to form a frame. The serial output data is dithered down to 20, 18, or 16 bits when 20-, 18-, or 16-bit output data is selected.

NOTE: The SRC converts the data from the serial input port to the sample rate of the serial output port. The sample rate at the serial input port can be asynchronous with respect to the output sample rate of the output serial port.

De-Emphasis Filter

The de-emphasis filter is used to de-emphasize audio data that has been emphasized.

Mute Control

When either the SRC starts up (or there is a change in sample ratio), the mute out signal (`SRCx_MUTEOUT`) is asserted (=1). The mute out signal stays high until the SRC settles on the new sample rates. While mute out is asserted high, the mute in signal should be asserted high as well. The mute in signal performs a soft mute of the audio input data when asserted and un-mutes the input audio data softly when deasserted.

Note that it takes 4096 input port FS samples until the audio input data is completely muted and 4096 FS samples until the audio input data is completely unmuted.

SRC Core

As shown in the ***ASRC Core Architecture*** figure, the sample rate converter's RAM FIFO block adjusts the left and right input samples and stores them for the FIR filter's convolution cycle. The `ASRCx_FS_IP` counter provides the write address (for scaling) to the FIFO block and the ramp input to the digital-servo loop. The ROM stores the coefficients for the FIR filter convolution and performs a high-order interpolation between the stored coefficients. The sample rate ratio block measures the sample rate by dynamically altering the ROM coefficients and scaling the FIR filter length and input data. The digital-servo loop automatically tracks the `SRCx_FS_IP` and `SRCx_FS_OP` sample rates and provides the RAM and ROM start addresses for the start of the FIR filter convolution.

NOTE: Unlike other peripherals, the sample rate converters own local memories (RAM and ROM) which are dedicated for the purpose of sample rate conversion only.

The sample rate converter only operates asynchronously and is always a completer to the input and output ports.

RAM FIFO

The RAM FIFO receives the left and right input data and adjusts the amplitude of the data for both the soft muting of the SRC and the scaling of the input data by the sample rate ratio before storing the samples in RAM. The input data is scaled by the sample rate ratio because as the FIR filter length of the convolution increases, so does the

amplitude of the convolution output. To keep the output of the FIR filter from saturating, the input data is scaled down by multiplying it by (SRCx_FS_OP)/(SRCx_FS_IP) when SRCx_FS_OP<SRCx_FS_IP. The FIFO also scales the input data to mute and stop muting the SRC.

Digital Servo Loop

The digital-servo loop is a ramp filter that provides the initial pointer to the address in RAM and ROM for the start of the FIR convolution. The RAM pointer is the integer output of the ramp filter while the ROM pointer is the fractional part. The digital-servo loop must be able to provide excellent rejection of jitter on the SRCx_FS_IP and SRCx_FS_OP clocks as well as measure the arrival of the SRCx_FS_OP clock within 5 ps. The digital-servo loop also divides the fractional part of the ramp output by the ratio of (SRCx_FS_IP)/(SRCx_FS_OP) for the case when SRCx_FS_IP> SRCx_FS_OP, to dynamically alter the ROM coefficients.

The digital-servo loop is implemented with a multi-rate filter. To settle the digital-servo loop filter quickly at startup or at a change in the sample rate, a fast mode has been added to the filter. When the digital-servo loop starts up or the sample rate is changed, the digital-servo loop kicks into fast mode to adjust and settle on the new sample rate. Upon sensing the digital-servo loop settling down to some reasonable value, the digital-servo loop kicks into normal or slow mode. During fast mode, the SRCx_MUTE_OUT bit of the ASRC is asserted to mute the ASRC input which avoids clicks and pops.

FIR Filter

The FIR filter is a 64-tap filter in the case of SRCx_FS_OP<SRCx_FS_IP and is (SRCx_FS_IP)/(SRCx_FS_OP) × 64 taps for the case when SRCx_FS_IP>SRCx_FS_OP. The FIR filter performs its convolution by loading in the starting address of the RAM address pointer and the ROM address pointer from the digital-servo loop at the start of the SRCx_FS_OP period. The FIR filter then steps through the RAM by decrementing its address by 1 for each tap, and the ROM pointer increments its address by the (SRCx_FS_OP/SRCx_FS_IP) × 2^{20} ratio for SRCx_FS_IP>SRCx_FS_OP or 2^{20} for SRCx_FS_OP <SRCx_FS_IP. Once the ROM address rolls over, the convolution is complete. The convolution is performed for both the left and right channels, and the multiply/accumulate circuit used for the convolution is shared between the channels.

Sample Rate Sensing

The (SRCx_FS_IP)/(SRCx_FS_OP) sample rate ratio circuit is used to dynamically alter the coefficients in the ROM for the case when SRCx_FS_IP>SRCx_FS_OP. The ratio is calculated by comparing the output of an SRCx_FS_OP counter to the output of an SRCx_FS_IP counter. If ASRCx_FS_OP>SRCx_FS_IP, the ratio is held at one. If SRCx_FS_IP>SRCx_FS_OP, the sample rate ratio is updated if it is different by more than two SRCx_FS_OP periods from the previous SRCx_FS_OP to SRCx_FS_IP comparison. This is done to provide some hysteresis to prevent the filter length from oscillating and causing distortion.

Digital Filter Group Delay

The RAM in the FIFO is 512 words deep for both left and right channels. An offset of 16 samples to the write address, provided by the SRCx_FS_IP counter, is added to prevent the RAM read pointer from overlapping the write address. The maximum decimation rate can be calculated from the RAM word: depth = (512 – 16) ÷ 64 taps = 7.75:1.

The 64 samples effect latency in the interpolation filter. This latency (group delay) depends on interpolation or decimation ratio and is determined as follows:

Interpolation or Decimation Ratio (1): GDL = $16/f_{S_IN} + 32/f_{S_IN}$ seconds for `SRC_FS_OP` > `SRC_FS_IP`

Interpolation or Decimation Ratio (2): GDL = $16/f_{S_IN} + 32/f_{S_IN} \times f_{S_IN}/f_{S_OUT}$ seconds for `SRC_FS_OP` < `SRC_FS_IP`

Data Format

The ***ASRC Data Frame Format by Protocol*** figure shows the data input format for a frame (stereo data). The frame format is valid for all protocols. For models which do not support matched phase mode the 8-bit data field is ignored.

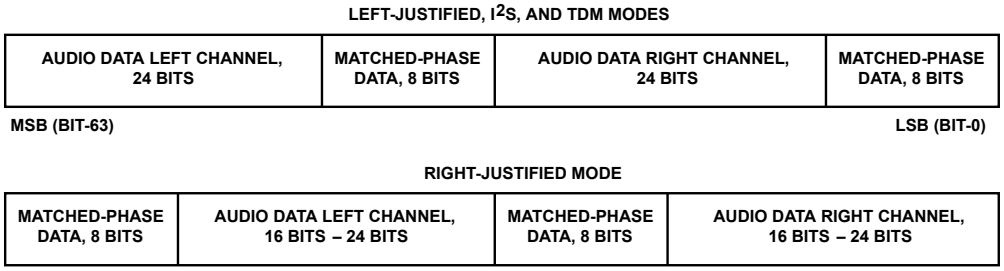


Figure 34-3: ASRC Data Frame Format by Protocol

Operating Modes

The ASRC can operate in TDM, I²S, left-justified, right-justified, and bypass modes. The serial ports of the processor can be used for moving the ASRC data to/from the internal memory.

In I²S, left-justified and right-justified modes, the ASRCs operate individually. The serial data provided in the input port is converted to the sample rate of the output port.

TDM Input Mode

In TDM input port, several ASRCs can be daisy-chained together and connected to the serial input port of a SHARC processor or other processor (see the ***TDM Input/Output Modes*** figure). The ASRC IP contains a 64-bit parallel load shift register. When the `SRCx_FS_IP_I` pulse arrives, each ASRC parallel loads its left and right data into the 64-bit shift register. The input to the shift register is connected to `SRCx_DATA_IP_I`, while the output is connected to `SRCx_TDM_IP_O`. By connecting the `SRCx_TDM_IP_O` to the `SRCx_DATA_IP_I` of the next ASRC, a large shift register is created, which is clocked by `SRCx_CLK_IP_I`.

NOTE: In TDM mode, the ASRC drives at the rising edge and samples at the falling edge of the serial clock. In all other modes, the serial clock rising edge is the sampling edge, and the falling edge is the driving edge.

NOTE: The number of ASRCs that can be daisy-chained together is limited by the maximum frequency of `SRCx_CLK_xx_I`, refer to the data sheet for exact values. For example, if the maximum frequency of

SRCx_CLK_xx_I is *x* MHz, and the output sample rate is *fS*, then number of ASRCs (n) that can be connected in daisy chained fashion is: n 64 FS ≤ *x* MHz.

TDM Output Mode

As shown in the ***TDM Input/Output Modes*** figure, using the TDM output port several ASRCs can be daisy-chained together and connected to the SPORT of this or another processor. The ASRC OP contains a 64-bit parallel load shift register. When the SRCx_FS_OP_I pulse arrives, each ASRC loads its left and right data into the 64-bit shift register. The input to the shift register is connected to SRCx_TDM_OP_I, and the output is connected to SRCx_DAT_OP_O. By connecting the SRCx_DAT_OP_O to the SRCx_TDM_OP_I of the next ASRC, a large shift register is created, which is clocked by SRCx_CLK_OP_I.

As shown in ***TDM Input/Output Modes***, with three ASRCs in a daisy-chain connection, the serial clock for input/or output port is defined as: SYSCLK = 3 × 64 FS = 192 FS.

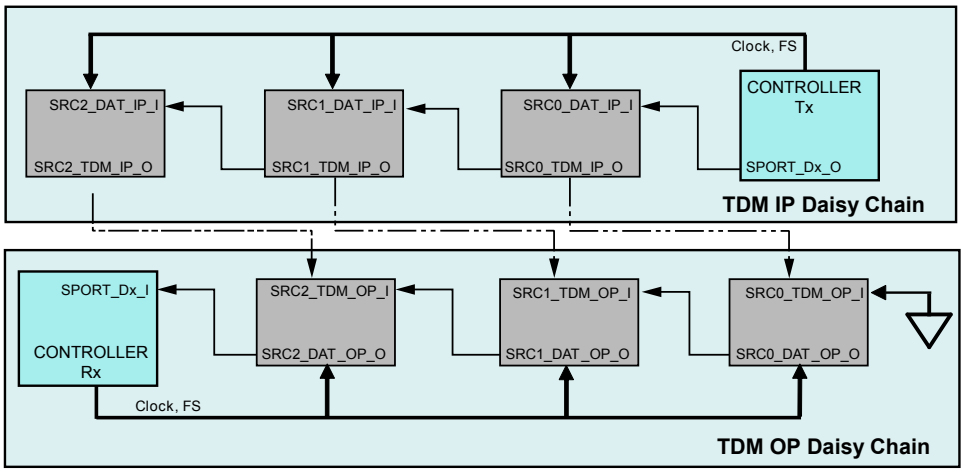


Figure 34-4: TDM Input/Output Modes

Matched-Phase Mode

The matched-phase mode of the sample rate converter, shown in ***Typical Configuration for Matched-Phase Mode Operation***, is enabled by the ASRC_CTL01.MPHASE0, ASRC_CTL01.MPHASE1, ASRC_CTL23.MPHASE2 and ASRC_CTL23.MPHASE3 bits. This mode is used to match the phase (group delay) between two or more adjacent sample rate converters that are operating with the same input and output clocks.

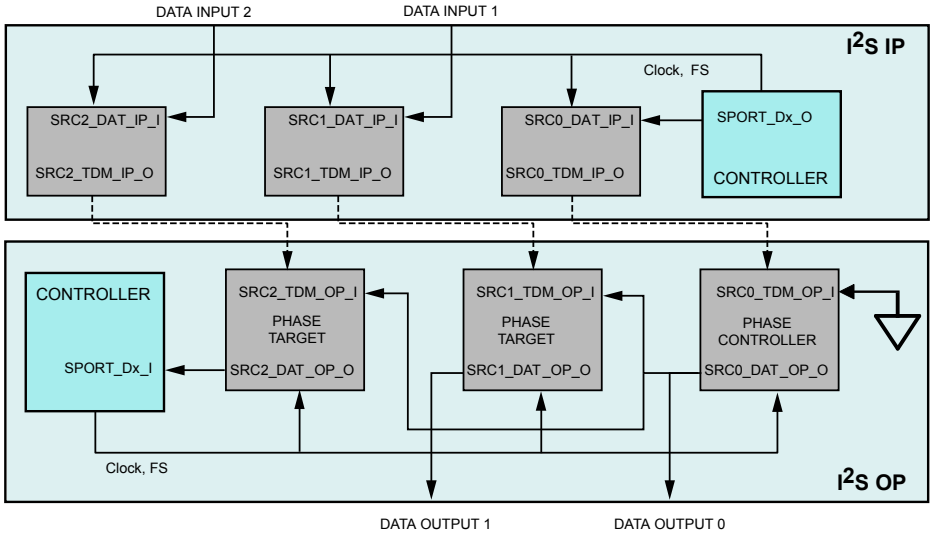


Figure 34-5: Typical Configuration for Matched-Phase Mode Operation

Hysteresis of the (SRCx_FS_OP)/(SRCx_FS_IP) ratio circuit can cause phase mismatching between two ASRCs operating with the same input and output clocks. Since the hysteresis requires a difference of more than two SRCx_FS_OP periods to update the SRCx_FS_OP and SRCx_FS_IP ratios, two ASRCs may have differences in their ratios from 0 to 4 SRCx_FS_OP period counts. The (SRCx_FS_OP)/(SRCx_FS_IP) ratio adjusts the filter length of the ASRC, which corresponds directly with the group delay. Thus, the magnitude in the phase difference depends upon the resolution of the SRCx_FS_OP and SRCx_FS_IP counters. The greater the resolution of the counters, the smaller the phase difference error.

When the completer SRC matched-phase mode bit is set (=1), it accepts the sample rate ratio transmitted by another SRC, (the matched-phase requester) which has its matched-phase mode bit cleared (=0), through its serial output.

The phase requester ASRC device transmits its SRCx_FS_OP/SRCx_FS_IP ratio through the data output pin (SRCx_DAT_OP_O) to the completer's ASRC's data input pins (SRCx_TDM_OP_I). The transmitted data (32-bit subframe) contains 24-bit data and 8-bits matched phase (see the ***ASRC Data Frame Format by Protocol*** figure).

The completer SRCs receive the 8-bit matched phase bits (instead of their own internally-derived ratio) if their SRCx_MPHASE bits are set to 1, respectively. The SRCx_FS_IP and SRCx_FS_OP signals may be asynchronous with respect to each other in this mode. Note that there must be 64 SRCx_CLK_OP cycles per frame in matched-phase mode (two 24-bits data and two 8-bits phase match).

NOTE: By default, matched phased data is sent on the SRCx_DAT_OP_O pin, but only if the SRCx_TDM_OP_I pin is tied low. The completers' simply ignore the matched phased data if their ASRC_CTL01.MPHASE1 through ASRC_CTL23.MPHASE3 bits are cleared (= 0).

Bypass Mode

When the ASRC_CTL01.BYP0, ASRC_CTL01.BYP1, ASRC_CTL23.BYP2 and ASRC_CTL23.BYP3 bits are set (=1), the input data bypasses the sample rate converter and is sent directly to the serial output port. Dithering is disabled. This mode is ideal when the input and output sample rates are the same and the SRCx_FS_IP_I and

SRCx_FS_OP_I signals are synchronous with respect to each other. In matched phase bypass mode, the SRCx_FS_OP_I signal should come at least one SRCx_CLK_xx_I period before SRCx_FS_IP_I. Cases where this is not met could result in data loss. For example, if internal SPORTS are used then the SRCx_FS_OP_I and SRCx_FS_IP_I signals could be driven by different SPORTs so that the timing of these signals can be controlled by enabling them at different times. This mode can also be used for passing through non-audio data since no processing is performed on the input data.

De-Emphasis Mode

The ASRC_CTL01.DEEMPHASIS0, ASRC_CTL01.DEEMPHASIS1, ASRC_CTL23.DEEMPHASIS2 and ASRC_CTL23.DEEMPHASIS3 bits choose the type of de-emphasis filter based on the input sample rate for 32, 44.1 or 48 kHz sampling rates.

Dithering Mode

The ASRC_CTL01.DITHER0, ASRC_CTL01.DITHER1, ASRC_CTL23.DITHER2, and ASRC_CTL23.DITHER3 control this mode of operation. Serial output data is dithered down to 20, 18, or 16 bits when 20-, 18-, or 16-bit output data is selected. In the case of 20-, 18-, and 16-bit word lengths, the least significant bits of the 24-bit word coming from the SRC into the serial output port are truncated. The DITHER_EN signal (not user configurable) automatically adds dithering to the 24-bit word before truncating to the appropriate output word length. The 21BIT_DITHER signal is used for the consumer version of the SRC to reduce the dynamic range performance to approximately 128 dB.

NOTE: The ASRC can be programmed to add the triangular Probability Distribution Function (PDF) dither to the digital audio samples. It is advisable to add dither when the input word width exceeds the output word width, for example the input word is 20 bits, and the output word is 16 bits. A triangular PDF is considered to create the most favorable noise shaping of the residual quantization noise.

Muting Modes

The mute feature of the ASRC can be controlled automatically in hardware using the MUTE_IN signal by connecting it to the MUTE_OUT signal. Automatic muting can be disabled by setting (=1) the ASRC_MUTE.MUTE0 through ASRC_MUTE.MUTE3 bits.

NOTE: Note that by default, the ASRC_MUTE register connects the MUTE_IN signal to the MUTE_OUT signal, but not conversely.

Soft Mute

When the ASRC_CTL01.SOFTMUTE0, ASRC_CTL01.SOFTMUTE1, ASRC_CTL23.SOFTMUTE2 and ASRC_CTL23.SOFTMUTE3 bits are set, the MUTE_IN signal is asserted, and the ASRC performs a soft mute by linearly decreasing the input data to the ASRC FIFO to zero, (-144 dB) attenuation as described for automatic hardware muting.

A 12-bit counter, clocked by SRCx_FS_IP_I, is used to control the mute attenuation. Therefore, the time it takes from the assertion of the MUTE_IN signal to -144 dB, full mute attenuation is 4096 FS cycles. Likewise, the time it takes to reach 0 dB mute attenuation from the deassertion of the MUTE_IN signal is 4096 FS cycles.

Hard Mute

When the ASRC_CTL01.HARDMUTE0, ASRC_CTL01.HARDMUTE1, ASRC_CTL23.HARDMUTE2 and ASRC_CTL23.HARDMUTE3 bits are set, the ASRC immediately mutes the input data to the ASRC FIFO to zero, (-144 dB) attenuation.

Auto Mute

When the ASRC_CTL01.AUTOMUTE0, ASRC_CTL01.AUTOMUTE1, ASRC_CTL23.AUTOMUTE2 and ASRC_CTL23.AUTOMUTE3 bits are set, the ASRC communicates with the S/PDIF receiver peripheral to determine when the input should mute.

This mode is useful for automatic detection of non-PCM audio data received from the S/PDIF receiver.

Interrupts

The following sections provide information about interrupt sources, masking, and servicing.

Sources

Each ASRC module drives one interrupt signal (mute out asserted). All these signals are connected into the DAI_IRPTL_H or DAI_IRPTL_L latch registers. The ASRC ports generate interrupts as described below.

SRC Mute Out

The SRC mute out signal can be used to generate interrupts on their rising edge, falling edge, or both, depending on how the DAI interrupt mask registers (DAI_IMSK_RE/DAI_IMSK_FE) are programmed. This programming allows the generation of DAI_IRPTL_H/DAI_IRPTL_L interrupts either entering mute, exiting muting or both. The SRCx_MUTE_OUT interrupt is generated only once when the SRC is locked (after 4096 FS input samples) and after changes to the sample ratio. Hard mute, soft mute, and auto mute only control the muting of the input data to the SRC.

Masking

The DAI_IMSK_FE, DAI_IMSK_RE, and DAI_IMSK_PRI registers must be unmasked accordingly. The DAI_IRQH and DAI_IRQL signals are routed to the system event controller (SEC) and general interrupt controller (GIC).

Service

The ISR reads the DAI_IRPTL_H and DAI_IRPTL_L registers to clear the interrupt request.

Programming Model

The following is basic information on programming the ASRC module.

1. Program the ASRC_CTL01 and ASRC_CTL23 registers and keep the ASRC_CTL01.EN0 through ASRC_CTL23.EN3 bits cleared.
2. Set the ASRC_CTL01.EN0 through ASRC_CTL23.EN3 bits. After 4096 input port FS cycles the ASRC has un-muted.

Debug Features

The asynchronous sample rate converter allow the bypass mode. When the ASRC_CTL01.BYP0 through ASRC_CTL23.BYP3 bits are set (=1), the input data bypasses the sample rate converter and is sent directly to the serial output port. This mode can be used for testing both ports when the input and output sample rates are at the same frequency, therefore both input and output ports can be routed to the same serial clock and frame sync.

ADSP-SC59x ASRC Register Descriptions

Sample Rate Converter Module (ASRC) contains the following registers.

Table 34-3: ADSP-SC59x ASRC Register List

Name	Description
ASRC_CTL01	Control Register for ASRC 0 and 1
ASRC_CTL23	Control Register for ASRC 2 and 3
ASRC_MUTE	Mute Register
ASRC_RAT01	Ratio Register for ASRC 0 and 1
ASRC_RAT23	Ratio Register for ASRC 2 and 3

Control Register for ASRC 0 and 1

The ASRC_CTL01 register (read/write) controls the operating modes, filters, and data formats used in the ASRC modules 0 and 1.

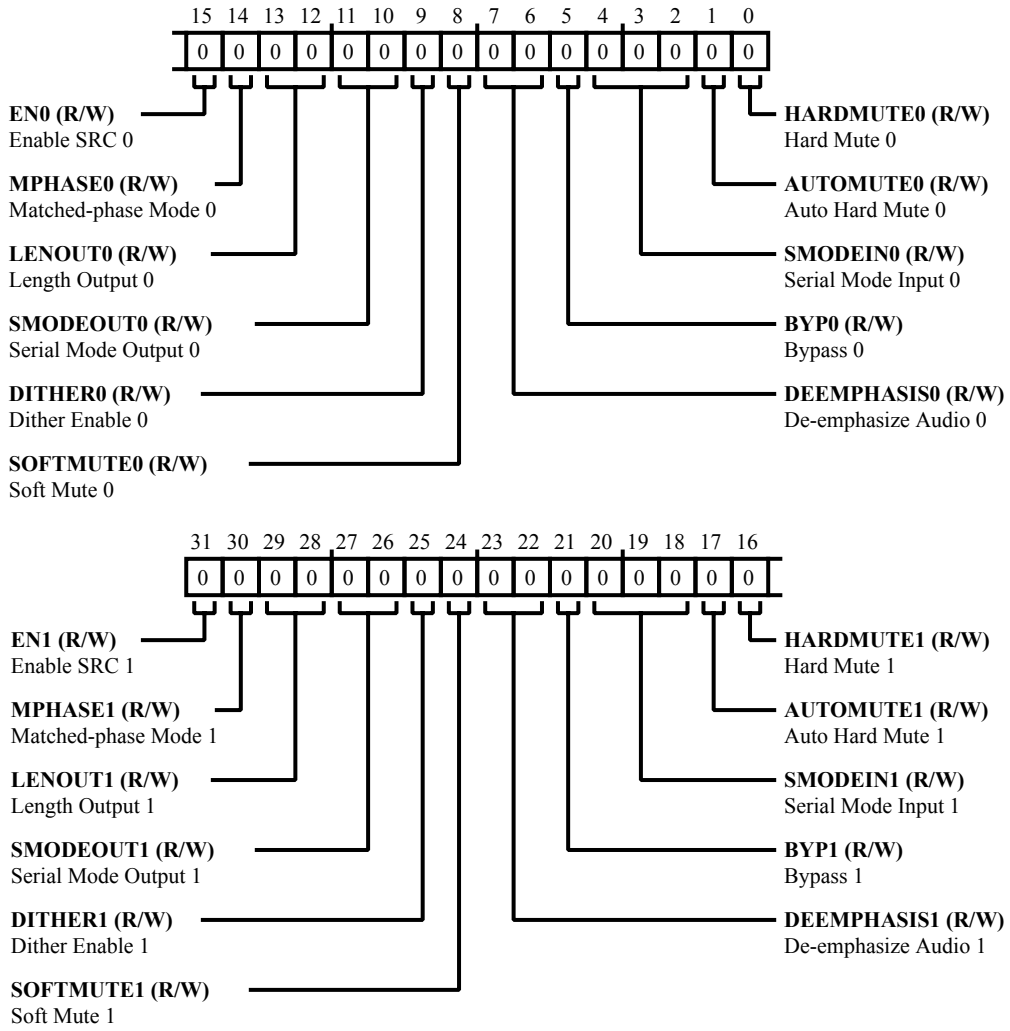


Figure 34-6: ASRC_CTL01 Register Diagram

Table 34-4: ASRC_CTL01 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EN1	Enable SRC 1. The `ASRC_CTL01.EN1` bit enables SRC 1. When (set = 1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (144 dB). Note that SRC power-up completion is finished by clearing the `ASRC_RAT01.MUTEOUT1` bit. Writes to the `ASRC_CTL01` register should be at least one cycle before setting the `ASRC_CTL01.EN1` bit. When setting and clearing this bit, it should be held low for a minimum of 5 SYSCLK cycles.	
30 (R/W)	MPHASE1	Matched-phase Mode 1. The `ASRC_CTL01.MPHASE1` bit configures SRC1 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
29:28 (R/W)	LENOUT1	Length Output 1. The `ASRC_CTL01.LENOUT1` bit field selects the serial output word length on SRC1.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
27:26 (R/W)	SMODEOUT1	Serial Mode Output 1. The `ASRC_CTL01.SMODEOUT1` bit field selects the serial output format on SRC1.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
25 (R/W)	DITHER1	Dither Enable 1. The `ASRC_CTL01.DITHER1` bit enables dithering before truncation on SRC1 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 34-4: ASRC_CTL01 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	SOFTMUTE1	Soft Mute 1. The `ASRC_CTL01.SOFTMUTE1` bit enables soft mute on SRC1.	
		0	Unmute
		1	Mute
23:22 (R/W)	DEEMPHASIS1	De-emphasize Audio 1. The `ASRC_CTL01.DEEMPHASIS1` bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
21 (R/W)	BYP1	Bypass 1. The `ASRC_CTL01.BYP1` bit makes the output of SRC1 the same as the input.	
20:18 (R/W)	SMODEIN1	Serial Mode Input 1. The `ASRC_CTL01.SMODEIN1` bit field selects the serial input format for SRC1.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
17 (R/W)	AUTOMUTE1	Auto Hard Mute 1. The `ASRC_CTL01.AUTOMUTE1` bit auto hard mutes SRC1 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
16 (R/W)	HARDMUTE1	Hard Mute 1. The `ASRC_CTL01.HARDMUTE1` bit hard mutes SRC1.	
		0	Unmute
		1	Mute

Table 34-4: ASRC_CTL01 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	EN0	Enable SRC 0. The ASRC_CTL01.EN0 bit enables SRC 0. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT01.MUTEOUT0 bit. Writes to the ASRC_CTL01 register should be at least one cycle before setting the ASRC_CTL01.EN0 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
14 (R/W)	MPHASE0	Matched-phase Mode 0. The ASRC_CTL01.MPHASE0 bit configures SRC0 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
13:12 (R/W)	LENOUT0	Length Output 0. The ASRC_CTL01.LENOUT0 bit field selects the serial output word length on SRC0.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
11:10 (R/W)	SMODEOUT0	Serial Mode Output 0. The ASRC_CTL01.SMODEOUT0 bit field selects the serial output format on SRC0.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
9 (R/W)	DITHER0	Dither Enable 0. The ASRC_CTL01.DITHER0 bit enables dithering before truncation on SRC0 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 34-4: ASRC_CTL01 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	SOFTMUTE0	Soft Mute 0. The `ASRC_CTL01.SOFTMUTE0` bit enables soft mute on SRC0.	
		0	Unmute
		1	Mute
7:6 (R/W)	DEEMPHASIS0	De-emphasize Audio 0. The `ASRC_CTL01.DEEMPHASIS0` bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
5 (R/W)	BYP0	Bypass 0. The `ASRC_CTL01.BYP0` bit makes the output of SRC0 the same as the input.	
4:2 (R/W)	SMODEIN0	Serial Mode Input 0. The `ASRC_CTL01.SMODEIN0` bit field selects the serial input format for SRC0.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
1 (R/W)	AUTOMUTE0	Auto Hard Mute 0. The `ASRC_CTL01.AUTOMUTE0` bit auto hard mutes SRC0 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
0 (R/W)	HARDMUTE0	Hard Mute 0. The `ASRC_CTL01.HARDMUTE0` bit hard mutes SRC0.	
		0	Unmute
		1	Mute (default)

Control Register for ASRC 2 and 3

The ASRC_CTL23 register (read/write) controls the operating modes, filters, and data formats used in the sample rate converter modules 2 and 3.

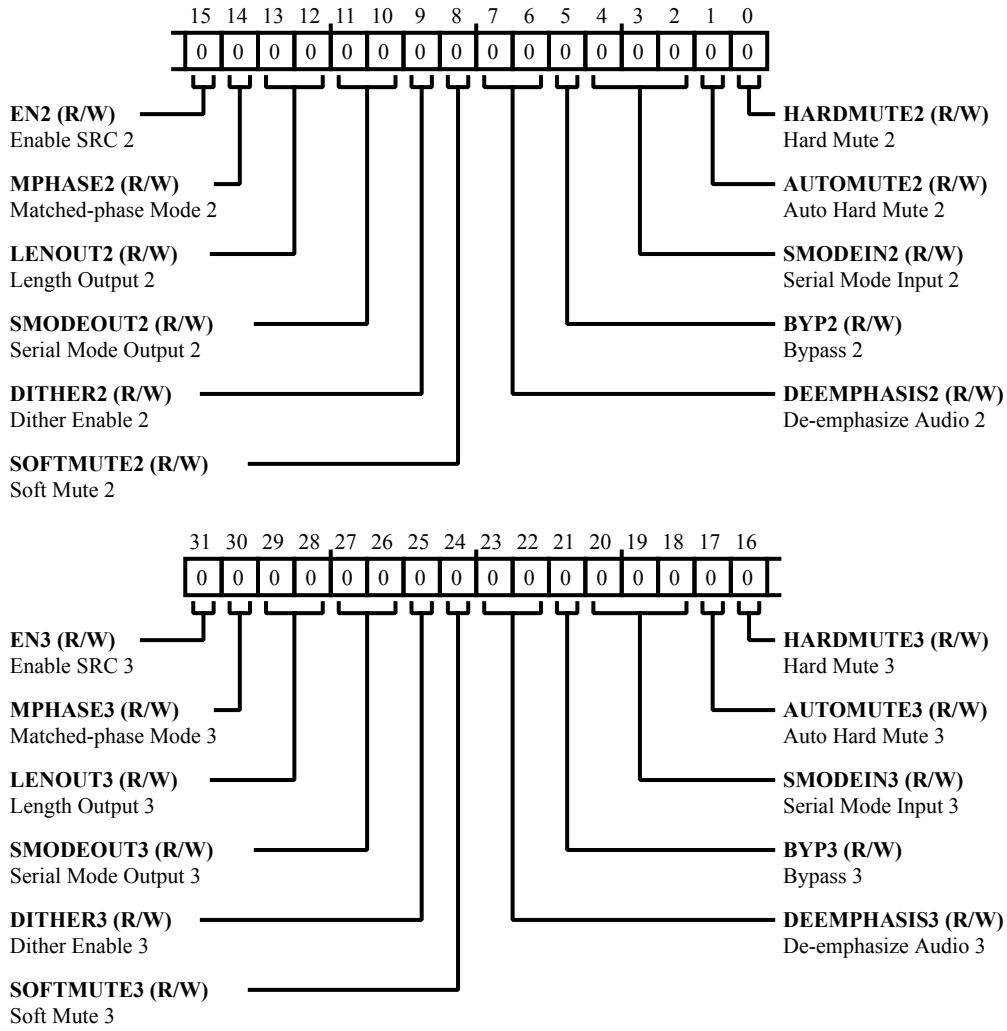


Figure 34-7: ASRC_CTL23 Register Diagram

Table 34-5: ASRC_CTL23 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	EN3	Enable SRC 3. The ASRC_CTL23.EN3 bit enables SRC 3. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT23.MUTEOUT3 bit. Writes to the ASRC_CTL23 register should be at least one cycle before setting the ASRC_CTL23.EN3 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
30 (R/W)	MPHASE3	Matched-phase Mode 3. The ASRC_CTL23.MPHASE3 bit configures SRC3 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
29:28 (R/W)	LENOUT3	Length Output 3. The ASRC_CTL23.LENOUT3 bit field selects the serial output word length on SRC3.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
27:26 (R/W)	SMODEOUT3	Serial Mode Output 3. The ASRC_CTL23.SMODEOUT3 bit field selects the serial output format on SRC3.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
25 (R/W)	DITHER3	Dither Enable 3. The ASRC_CTL23.DITHER3 bit enables dithering before truncation on SRC3 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 34-5: ASRC_CTL23 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	SOFTMUTE3	Soft Mute 3. The `ASRC_CTL23.SOFTMUTE3` bit enables soft mute on SRC3.	
		0	Unmute
		1	Mute
23:22 (R/W)	DEEMPHASIS3	De-emphasize Audio 3. The `ASRC_CTL23.DEEMPHASIS3` bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
21 (R/W)	BYP3	Bypass 3. The `ASRC_CTL23.BYP3` bit makes the output of SRC3 the same as the input.	
20:18 (R/W)	SMODEIN3	Serial Mode Input 3. The `ASRC_CTL23.SMODEIN3` bit field selects the serial input format for SRC3.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
17 (R/W)	AUTOMUTE3	Auto Hard Mute 3. The `ASRC_CTL23.AUTOMUTE3` bit auto hard mutes SRC3 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
16 (R/W)	HARDMUTE3	Hard Mute 3. The `ASRC_CTL23.HARDMUTE3` bit hard mutes SRC3.	
		0	Unmute
		1	Mute

Table 34-5: ASRC_CTL23 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	EN2	Enable SRC 2. The ASRC_CTL23.EN2 bit enables SRC 2. When (set =1), or when the sample rate (frame sync) between the input and output changes, the SRC begins its initialization routine where; 1) MUTE_OUT is asserted, 2) soft mute control counter for input samples is set to maximum attenuation (144 dB). Note that SRC power-up completion is finished by clearing the ASRC_RAT23.MUTEOUT2 bit. Writes to the ASRC_CTL23 register should be at least one cycle before setting the ASRC_CTL23.EN2 bit. When setting and clearing this bit, it should be held low for a minimum of 5 CLK cycles.	
14 (R/W)	MPHASE2	Matched-phase Mode 2. The ASRC_CTL23.MPHASE2 bit configures SRC2 to not use its own internally-generated sample rate ratio but use an externally-generated ratio. Used with TDM data.	
		0	Matched phase completer disabled
		1	Matched phase completer enabled
13:12 (R/W)	LENOUT2	Length Output 2. The ASRC_CTL23.LENOUT2 bit field selects the serial output word length on SRC2.	
		0	24 bits
		1	20 bits
		2	18 bits
		3	16 bits
11:10 (R/W)	SMODEOUT2	Serial Mode Output 2. The ASRC_CTL23.SMODEOUT2 bit field selects the serial output format on SRC2.	
		0	Left-justified
		1	I2S
		2	TDM
		3	Right-justified
9 (R/W)	DITHER2	Dither Enable 2. The ASRC_CTL23.DITHER2 bit enables dithering before truncation on SRC2 when a word length less than 24 bits is selected.	
		0	Truncation only
		1	Dithering before truncation

Table 34-5: ASRC_CTL23 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W)	SOFTMUTE2	Soft Mute 2. The `ASRC_CTL23.SOFTMUTE2` bit enables soft mute on SRC2.	
		0	Unmute
		1	Mute
7:6 (R/W)	DEEMPHASIS2	De-emphasize Audio 2. The `ASRC_CTL23.DEEMPHASIS2` bits are used to de-emphasize audio data that has been emphasized. The type of de-emphasis filter is based on the input sample rate (SRCx_FS_IP_I signal).	
		0	No de-emphasis
		1	32 kHz
		2	44.1 kHz
		3	48 kHz
5 (R/W)	BYP2	Bypass 2. The `ASRC_CTL23.BYP2` bit makes the output of SRC2 the same as the input.	
4:2 (R/W)	SMODEIN2	Serial Mode Input 2. The `ASRC_CTL23.SMODEIN2` bit field selects the serial input format for SRC2.	
		0	left-justified
		1	I2S
		2	TDM
		4	24-bit right-justified
		5	20-bit right-justified
		6	18-bit right-justified
		7	16-bit right-justified
1 (R/W)	AUTOMUTE2	Auto Hard Mute 2. The `ASRC_CTL23.AUTOMUTE2` bit auto hard mutes SRC2 when non audio is asserted by the SPDIF receiver.	
		0	Unmute
		1	Mute
0 (R/W)	HARDMUTE2	Hard Mute 2. The `ASRC_CTL23.HARDMUTE2` bit hard mutes SRC2.	
		0	Unmute
		1	Mute

Mute Register

This register connects an ASRCx mute input and output when the mute bit is cleared (=0). This allows ASRCx to automatically mute input while the ASRC is initializing (0 = automatic muting and 1 = manual muting). Bit 0 controls ASRC0, bit 1 controls ASRC1, bit 2 controls ASRC2, and bit 3 controls ASRC3.

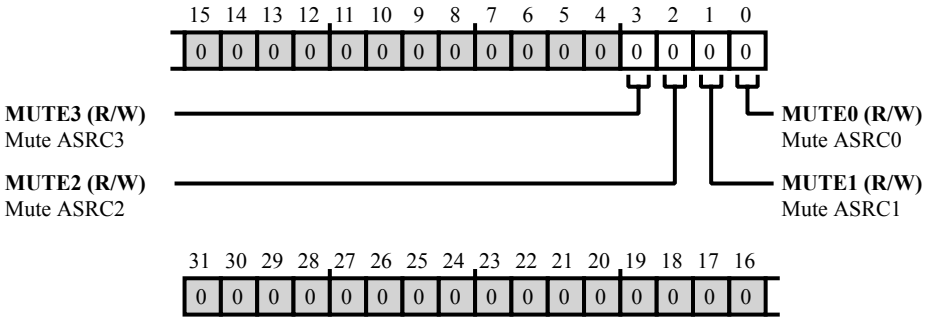


Figure 34-8: ASRC_MUTE Register Diagram

Table 34-6: ASRC_MUTE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W)	MUTE3	Mute ASRC3. The `ASRC_MUTE.MUTE3` bit automatically mutes ASCR3 output when cleared (=0).
2 (R/W)	MUTE2	Mute ASRC2. The `ASRC_MUTE.MUTE2` bit automatically mutes ASCR2 output when cleared (=0).
1 (R/W)	MUTE1	Mute ASRC1. The `ASRC_MUTE.MUTE1` bit automatically mutes ASCR1 output when cleared (=0).
0 (R/W)	MUTE0	Mute ASRC0. The `ASRC_MUTE.MUTE0` bit automatically mutes ASCR0 output when cleared (=0).

Ratio Register for ASRC 0 and 1

The ASRC_RAT01 register report the mute and I/O sample ratio for ASRC0 and ASRC1.

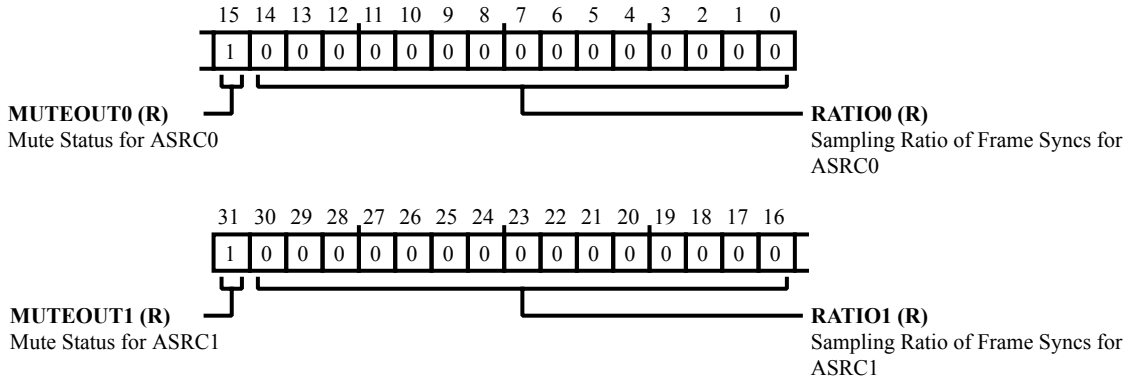


Figure 34-9: ASRC_RAT01 Register Diagram

Table 34-7: ASRC_RAT01 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MUTEOUT1	Mute Status for ASRC1. The ASRC_RAT01.MUTEOUT1 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC1 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loops internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.
30:16 (R/NW)	RATIO1	Sampling Ratio of Frame Syncs for ASRC1. The ASRC_RAT01.RATIO1 bit field is read to find the ratio of output to input sampling frequency for ASRC1 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.
15 (R/NW)	MUTEOUT0	Mute Status for ASRC0. The ASRC_RAT01.MUTEOUT0 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC0 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loops internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.

Table 34-7: ASRC_RAT01 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14:0 (R/NW)	RATIO0	Sampling Ratio of Frame Syncs for ASRC0. The `ASRC_RAT01.RATIO0` bit field is read to find the ratio of output to input sampling frequency for ASRC0 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.

Ratio Register for ASRC 2 and 3

The ASRC_RAT23 register report the mute and I/O sample ratio for ASRC0 and ASRC1.

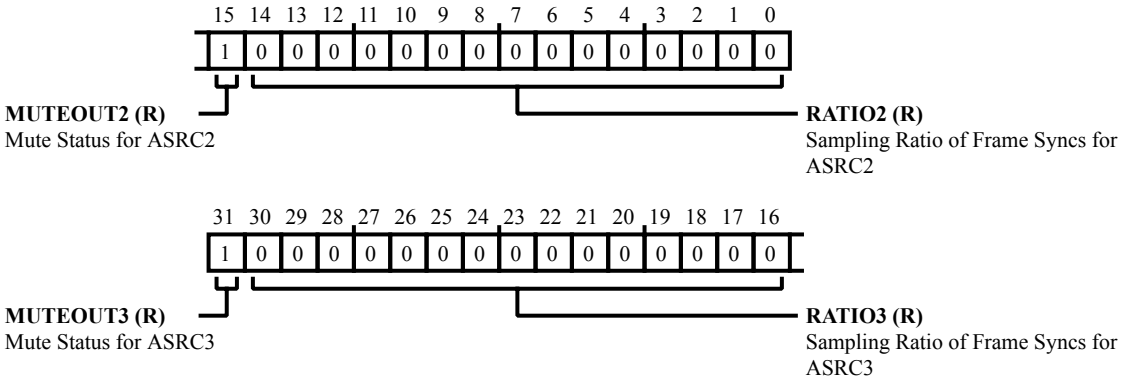


Figure 34-10: ASRC_RAT23 Register Diagram

Table 34-8: ASRC_RAT23 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/NW)	MUTEOUT3	Mute Status for ASRC3. The ASRC_RAT23.MUTEOUT3 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC3 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loops internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.
30:16 (R/NW)	RATIO3	Sampling Ratio of Frame Syncs for ASRC3. The ASRC_RAT23.RATIO3 bit field is read to find the ratio of output to input sampling frequency for ASRC3 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.
15 (R/NW)	MUTEOUT2	Mute Status for ASRC2. The ASRC_RAT23.MUTEOUT2 bit field reports the status of the MUTE_OUT signal. Once the SRCx_MUTEOUT signal is cleared, the ratio can be read. When ASRC2 is enabled or there is a change in the sample ratio, the MUTE_OUT signal is asserted. The MUTE_OUT signal remains asserted until the digital servo loops internal fast settling mode is complete. When the digital servo loop has switched to slow settling mode, the MUTE_OUT signal is deasserted.

Table 34-8: ASRC_RAT23 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
14:0 (R/NW)	RATIO2	Sampling Ratio of Frame Syncs for ASRC2. The `ASRC_RAT23.RATIO2` bit field is read to find the ratio of output to input sampling frequency for ASRC2 (SRCx_FS_OP_I/SRCx_FS_IP_I). This ratio is reported in 4.11 (integer.fraction) format where the 15-bit value of the normal binary number is comprised of 4 bits for the integer and 11 bits for the fraction.

35 Sony/Philips Digital Interface (S/PDIF)

The Sony/Philips Digital Interface (S/PDIF) is a standard audio data transfer format that allows the transfer of digital audio signals from one device to another without having to convert them to an analog signal. The digital audio interface carries three types of information; audio data, non-audio data (compressed data), and timing information.

Features

The S/PDIF interface has the following features.

- Supports one stereo channel or compressed audio streams.
- AES3-compatible S/PDIF transmitter and receiver.
- Transmitting a biphase mark encoded signal that may contain any number of audio channels (compressed or linear pulse code modulation) or non-audio data.
- S/PDIF receiver managing clock recovery with separate S/PDIF on-chip PLL.
- S/PDIF receiver supports the detection of DTS frames of 256, 512, 1024, 2048, and 4096.
- Manage user status information and provide error-handling capabilities in both the transmitter and receiver.
- DAI allows interactions over DAI by serial ports and the external DAI pins to interface to other S/PDIF devices. This includes using the receiver to decode incoming biphase encoded audio streams and passing them via the SPORTs to internal memory for processing-or using the transmitter to encode audio or digital data and transfer it to another S/PDIF receiver in the audio system.

It is important to be familiar with serial digital audio interface standards IEC-60958, EIAJ CP-340, AES3 and AES11.

ADSP-SC59x SPDIF Register List

The S/PDIF module is a standard audio data transfer format that allows the transfer of digital audio signals from one device to another without having to convert them to an analog signal. A set of registers govern S/PDIF operations. For more information on S/PDIF functionality, see the S/PDIF register descriptions.

Table 35-1: ADSP-SC59x SPDIF Register List

Name	Description
SPDIF_RX_CTL	Receive Control
SPDIF_RX_STAT	Receive Status Register
SPDIF_RX_STAT0_A	Receive Status A0 Register
SPDIF_RX_STAT0_B	Receive Status B0 Register
SPDIF_RX_STAT1_A	Receive Status A1 Register
SPDIF_RX_STAT1_B	Receive Status B1 Register
SPDIF_TX_CTL	Transmit Control Register
SPDIF_TX_STAT_A0	Transmit Status A0 Register
SPDIF_TX_STAT_A1	Transmit Status A1 Register
SPDIF_TX_STAT_A2	Transmit Status A2 Register
SPDIF_TX_STAT_A3	Transmit Status A3 Register
SPDIF_TX_STAT_A4	Transmit Status A4 Register
SPDIF_TX_STAT_A5	Transmit Status A5 Register
SPDIF_TX_STAT_B0	Transmit Status B0 Register
SPDIF_TX_STAT_B1	Transmit Status B1 Register
SPDIF_TX_STAT_B2	Transmit Status B2 Register
SPDIF_TX_STAT_B3	Transmit Status B3 Register
SPDIF_TX_STAT_B4	Transmit Status B4 Register
SPDIF_TX_STAT_B5	Transmit Status B5 Register
SPDIF_TX_UBUFF_A0	Transmit User Buffer A0 Register
SPDIF_TX_UBUFF_A1	Transmit User Buffer A1 Register
SPDIF_TX_UBUFF_A2	Transmit User Buffer A2 Register
SPDIF_TX_UBUFF_A3	Transmit User Buffer A3 Register
SPDIF_TX_UBUFF_A4	Transmit User Buffer A4 Register
SPDIF_TX_UBUFF_A5	Transmit User Buffer A5 Register
SPDIF_TX_UBUFF_B0	Transmit User Buffer B0 Register
SPDIF_TX_UBUFF_B1	Transmit User Buffer B1 Register
SPDIF_TX_UBUFF_B2	Transmit User Buffer B2 Register
SPDIF_TX_UBUFF_B3	Transmit User Buffer B3 Register
SPDIF_TX_UBUFF_B4	Transmit User Buffer B4 Register
SPDIF_TX_UBUFF_B5	Transmit User Buffer B5 Register

Table 35-1: ADSP-SC59x SPDIF Register List (Continued)

Name	Description
SPDIF_TX_USRUPDT	User Bit Update Register

SRU Programming

The SRU (signal routing unit) is used to connect the S/PDIF transmitter biphase data out to the output pins or to the S/PDIF receiver. The serial clock, frame sync, data, and external sync (if external synchronization is required) inputs also need to be routed through the SRU. For details of the routing, see DAI Routing Capabilities in the *Digital Audio Interface (DAI)* chapter.

The SRU needs to be programmed in order to connect the S/PDIF receiver to the output pins or any other peripherals and also for the connection to the input biphase stream.

Program the corresponding SRU registers to connect the outputs to the required destinations (see DAI Routing Capabilities inputs to the receiver are routed through the SRU. The extracted clock, frame sync, and data are also routed through the SRU.

S/PDIF Interrupt List

Table 35-2: S/PDIF Interrupt List

Interrupt Name	Interrupt Condition	Return DAI Register	Return SEC Register	SEC ID
DAI0_IRQH	Block Start	DAIx_IRPTL	SEC_ID	24
DAI0_IRQL	Non Audio			145

Clocking

The fundamental timing clock of the S/PDIF receiver is CLKO5 from the CDU. When CLKO5 is configured, it supports sampling frequencies of 24 kHz to 192 kHz. The fundamental timing clock of the S/PDIF transmitter is *SCLK0*/4.

For information on clock programming, see CDU Programming Model.

S/PDIF Transmitter

The following sections provide information on the S/PDIF transmitter.

Functional Description

The S/PDIF transmitter, shown in the ***S/PDIF Transmitter Block Diagram***, resides within the DAI, and its inputs and outputs can be routed via the SRU. It receives audio data in serial format, encloses the specified user status information, and converts it into the biphase encoded signal. The serial data input to the transmitter can be

formatted as left-justified, I^{2}S, or right-justified with word widths of 16, 18, 20 or 24 bits. ***AES3 Output Block*** shows the detail of the AES block.

The serial data, clock, and frame sync inputs to the S/PDIF transmitter are routed through the signal routing unit (SRU).

The S/PDIF transmitter output may be routed to an output pin via the SRU and then routed to another S/PDIF receiver or to components for off-board connections to other S/PDIF receivers. The output is also available to the S/PDIF receiver for loop-back testing through SRU.

In addition to encoding the audio data in the bi-phase format, the transmitter also provides a way to easily add the channel status information to the outgoing bi-phase stream. There are status/user registers for a frame (192-bits/24 bytes) in the transmitter that correspond to each channel or subframe.

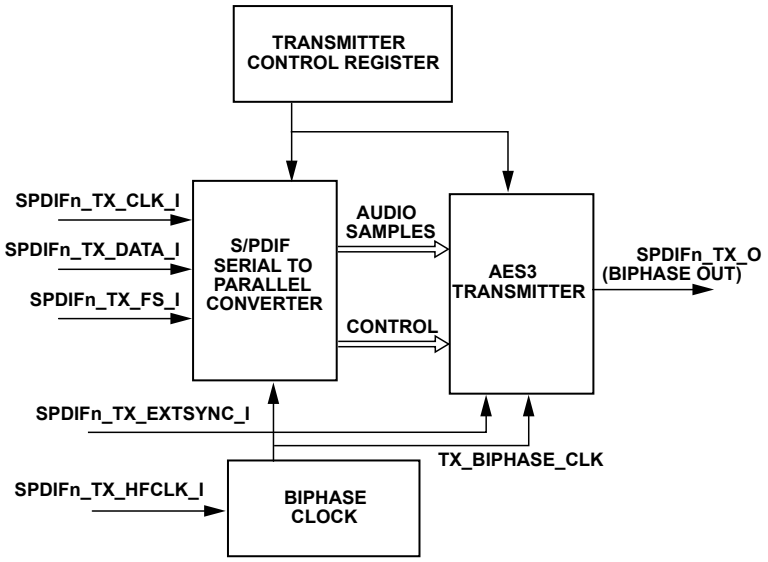


Figure 35-1: S/PDIF Transmitter Block Diagram

Validity bits for both channels may also be controlled by the transmitter control register. Optionally, the user bit, validity bit, and channel status bit are sent to the transmitter with each left/right sample. For each subframe the parity bit is automatically generated and inserted into the biphase encoded data.

A mute control and support for double-frequency single-channel mode are also provided. The serial data input format may be selected as left-justified, I^{2}S, or right-justified with 16-, 18-, 20-, or 24-bit word widths. The over sampling clock is also selected by the transmitter control register.

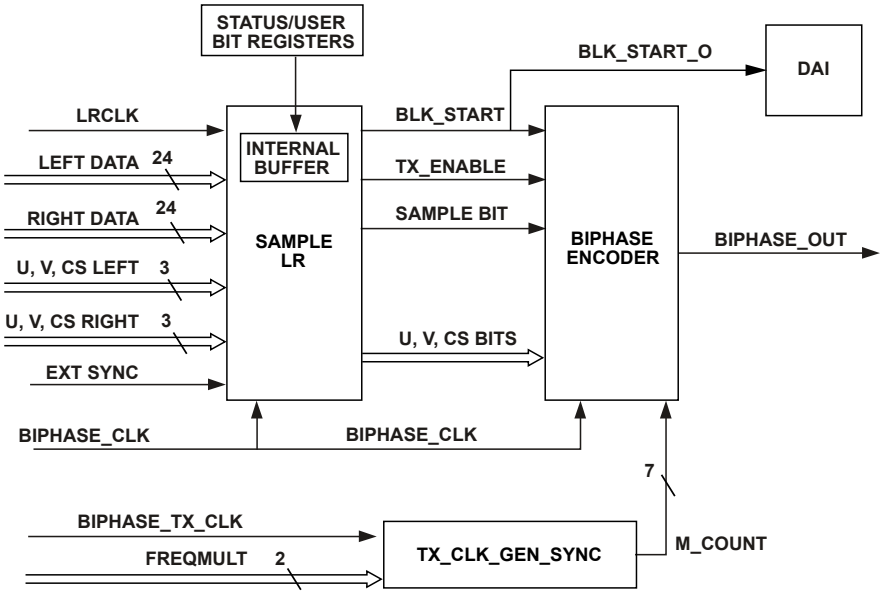


Figure 35-2: AES3 Output Block

Input Data Formats

The *I2S and Left-Justified Formats* and *Right-Justified Formats* figures show the format of data that is sent to the S/PDIF transmitter using a variety of protocol standards.

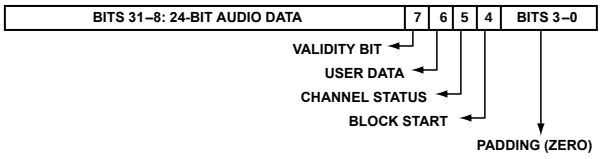


Figure 35-3: I^2S and Left-Justified Formats

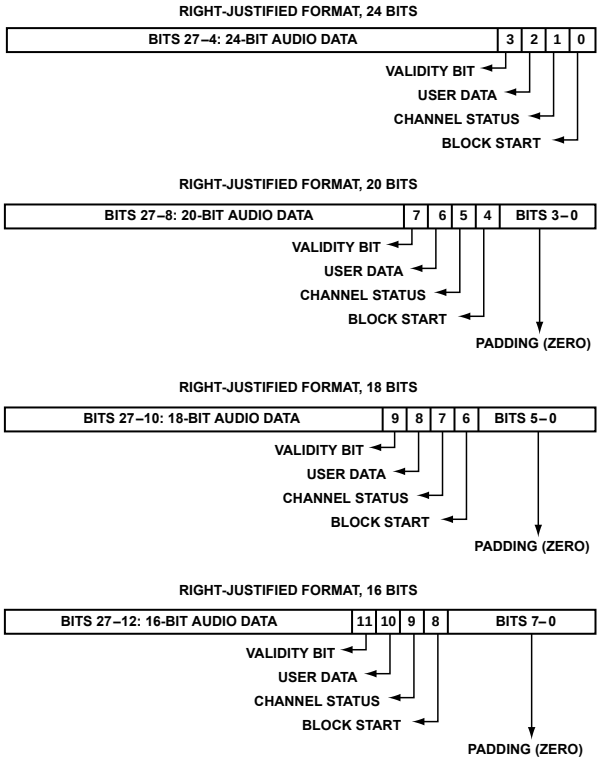


Figure 35-4: Right-Justified Formats

Operating Modes

The S/PDIF transmitter can operate in standalone and full serial modes. The following sections describe these modes in detail.

Full Serial Mode

This mode is selected by clearing the `SPDIF_TX_CTL.AUTO` bit. In this mode all the status bits, audio data and the block start bit (indicating start of a frame), come through the serial data stream (`SPDIF_TX_DATA_I`) pin. The transmitter should be enabled after or at the same time as all of the other control bits.

Standalone Mode

This mode is selected by setting the `SPDIF_TX_CTL.AUTO` bit. In this mode, the block start bit (indicating the start of a frame) is generated internally. The channel status bits come from the channel status buffer registers. The user status bits come from the user bits buffers as shown in the ***AES3 Output Block*** figure.

The validity bits are the `SPDIF_TX_CTL.VALIDR` and `SPDIF_TX_CTL.VALIDL`. In this mode only audio data comes from the `SPDIF_TX_DATA_I` pin. All other data, including the status bit and block start bit is either generated internally or taken from the internal register.

Once the user bits buffer registers (`SPDIF_TX_UBUFF_A0` - `SPDIF_TX_UBUFF_B5`) are programmed, they are used only for the next block of data. This allows programs to change the user bit information in every block of data.

To allow user bit updates, write a 0x1 to the SPDIF_TX_USRUPDT register that is used for further processing. If the SPDIF_TX_CTL.AUTO bit is set:

- and if SPDIF_TX_USRUPDT =1, at every 192nd frame end the user status bits are taken from user bits buffers and transmitted. Simultaneously, the SPDIF_TX_USRUPDT register is cleared automatically by hardware.
- and if SPDIF_TX_USRUPDT =0, at every 192nd frame end the user status bits are updated as zeros and transmitted. The SPDIF_TX_USRUPDT register remains low.

In general, for the next block, programs can update user bits buffers at any time during the transfer of the current block (1 block = 192 frames). There are internal buffers to store the user status bits of the current block of transfer. In other words, at the beginning of every new block, the user status bit (SPDIF_TX_CTL.USRPEND bit) from user bits buffers are copied to internal buffers and transmitted in each frame during the transfer.

Note that since a frame contains 192 bits/8 = 24 bytes, six status/user registers are required to store each four bytes.

Data Output Mode

Two output data formats are supported by the transmitter; *two channel mode* and *single-channel double-frequency* (SCDF) mode. The output format is determined by the transmitter control register (SPDIF_TX_CTL).

In two channel mode, the left channel (channel A) is transmitted when the SPDIF_TX_FS_I is high, and the right channel (channel B) is transmitted when the SPDIF_TX_FS_I is low.

In SCDF mode, the transmitter sends successive audio samples of the same signal across both sub frames, instead of channel A and B. The transmitter transmits at half the sample rate of the input bit stream. The SPDIF_TX_CTL.SCDF bit selects SCDF mode. When in SCDF mode, the SPDIF_TX_CTL.SCDFLR bit determines whether left or right channel data is transmitted.

S/PDIF Receiver

The S/PDIF receiver (***S/PDIF Receiver Block Diagram***) is compliant with all common serial digital audio interface standards including IEC-60958, IEC-61937, AES3, and AES11. For the IEC-60958 standard, all the user-data and channel-status bits (as outlined in this document) are not decoded by the S/PDIF receiver. The interface does make all 192 user-data and channel-status pairs available as an output of the block, for post-decoding.

For the IEC-61937 standard, the S/PDIF only detects compressed AC-3 and DTS formats. No decompression is performed.

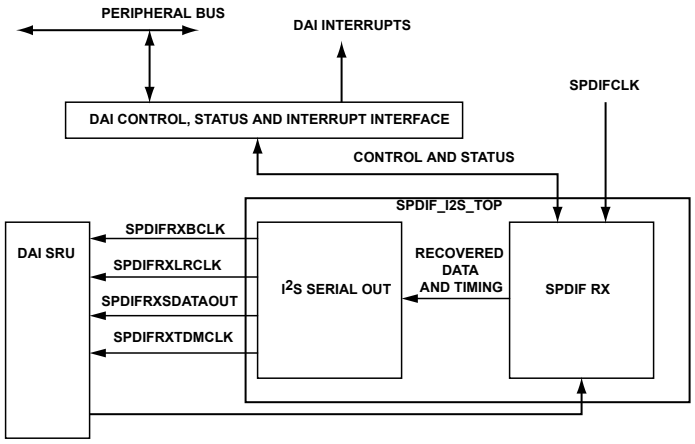


Figure 35-5: S/PDIF Receiver Block Diagram

Functional Description

If the receiver is used, programs need to enable it using the `SPDIF_RX_CTL` register. After the SRU programming is complete, write to the register with control values. At this point, the receiver attempts to lock.

NOTE: The S/PDIF receiver is disabled at default. If the receiver is used in an application, programs should enable the receiver.

The input to the receiver (SPDIFn_RX_I) is a biphase encoded signal that may contain two audio channels (compressed or linear PCM) or non-audio data. The receiver decodes the single biphase encoded stream, producing an I^2S compatible serial data output that consists of a serial clock, a left-right frame sync, and data (channel A/B). It provides the programmer with several methods of managing the incoming status bit information.

The S/PDIF receiver receives any S/PDIF stream with a sampling frequency range of 24 kHz to 192 kHz. Refer to Clocking for more details.

The channel status bits are collected into memory-mapped registers, while other channel status and user bytes must be handled manually. The block start bit, which replaces the parity bit in the serial I^2S stream, indicates the reception of the Z preamble and the start of a new block of channel status and data bits.

Clock Recovery

The S/PDIF receiver recovers the clock that generated the AES3/SPDIF biphase encoded stream from the incoming S/PDIF stream.

This clock is used by the receiver to clock in the biphase encoded data stream and also to provide clocks for either the SPORTs, sample rate converter, or the AES3 and S/PDIF transmitter. The recovered clock may also be used externally to the chip for clocking D/A and A/D converters.

In order to maintain performance, jitter on the clock is sourced to several peripherals.

To comply with the AES11 standard, the recovered left or right clock must be aligned with the preambles within a + or - 5% of the frame period. Since the PLL clock generates a clock 512 times the frame rate clock (512 f_{SCLK}), this clock can be used and divided down to create the phase aligned jitter-free left or right clock.

Output Data Format

The extracted 24-bit audio data, V, U, C, and block start bits are sent on the SPDIF_RX_DAT pin in 32-bit I^2S format as shown in ***I2S and Left-Justified Formats***. The frame sync is transmitted on the SPDIF_RX_FS pin and serial clock is transmitted on the SPDIF_RX_CLK pin. All three pins are routed through the SRU.

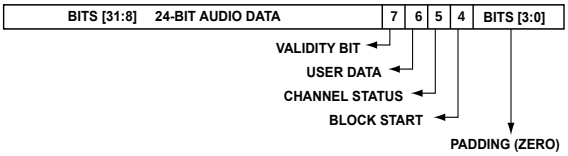


Figure 35-6: I^2S Format

Channel Status

The channel status for the first bytes 4-0 (consumer mode) are collected into memory-mapped registers (SPDIF_RX_STAT0_A, SPDIF_RX_STAT0_B, SPDIF_RX_STAT1_A and SPDIF_RX_STAT1_B). All other channel status bytes 23-5 (professional mode) must be manually extracted from the receiver data stream.

NOTE: Only the first 5 channel status bytes (40-bit) for consumer mode of a frame are stored into the S/PDIF receiver status registers.

Operating Modes

This section describes the receiver channel status for the different modes.

Compressed or Non-linear Audio Data

The S/PDIF receiver processes compressed as well as non-linear audio data according to the supported standards. The following sections describe how this peripheral handles different data.

MPEG-2, AC-3, DTS, and AAC compressed data may be transmitted without setting either the SPDIF_RX_STAT.VALID bit or bit 1 of byte 0. To detect this data, the IEC61937 and SPMTE 337M standards dictate that there be a 96-bit sync code in the 16-, 20- or 24-bit audio data stream. This sync code consists of four words of zeros followed by a word consisting of 0xF872 and another word consisting of 0x4E1F. When this sync code is detected, the SPDIF_RX_STAT.COMPMODE bits hold the information regarding type of compression.

The last two words of the sync code, 0xF872 and 0x4E1F, are called the preamble-A and preamble-B of the burst preamble. Preamble-C of the burst preamble contains burst information and is captured and stored by the receiver. Preamble-D of the burst preamble contains the length code and is captured by the receiver. Even if the validity bit or bit 1 of byte 0 has been set, the receiver still looks for the sync code in order to record the preamble-C and D values. Once the sync code has not been detected in 4096 frames, the preamble-C and D registers are set to zero.

Emphasized Audio Data

Determination as to whether the received audio data is emphasized or not is done in software using the channel status bits as detailed below.

- In professional mode, (bit 0 of byte 0 = 1), channel status bits 2-4 of byte 0 indicate the audio data is emphasized if they are equal to 110 or 111.
- In consumer mode, (bit 0 of byte 0 = 0), channel status bits 3-5 indicate the audio data is emphasized if they are equal to 100, 010 or 110.

Single-Channel Double-Frequency Mode

Unlike previous processors, support for single-channel, double-frequency mode (SCDF) is not supported through specific bits within the `SPDIF_RX_CTL` register, but must be implemented in software using the information provided by the CS (channel status) bits.

- 0111–single channel double frequency mode
- 1000–single channel double frequency mode-stereo left
- 1001–single channel double frequency mode-stereo right

Clock Recovery Modes

The S/PDIF receiver extracts audio data, channel status, and user bits from the biphase encoded AES3 and S/PDIF stream. In addition, a 50% duty cycle reference clock running at the sampling rate of the audio input data is generated for the receiver to recover the oversampling clock.

Number Controlled Oscillator

The receiver can recover the clock from the biphase encoded stream using an on-chip NCO shown in the following figure. The dedicated NCO is separate from the PLL that supplies the clock to the processor core, which is the default operation of the receiver.

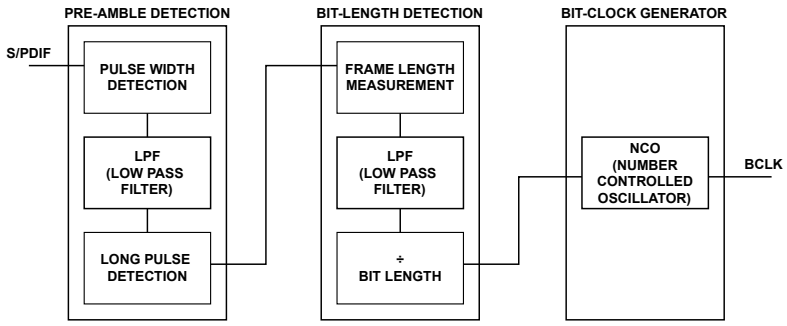


Figure 35-7: S/PDIF Clock Recovery Mechanism

The left/right frame reference clock for the NCO is generated using the preambles. The recovered low jitter left/right frame clock from the NCO attempts to align with the reference clock. However, this recovered left/right clock, like the reference clock, is not phase aligned with the preambles.

Interrupts

The following sections provide information about interrupt sources, masking, and servicing.

Sources

The S/PDIF module of each DAI drives five interrupt signals. Four are status signals driven from SPDIFn_RX and one signal is driven from SPDIFn_TX (block start). These signals are connected into the DAI_IRPTL_L/ DAI_IRPTL_H latch register.

Transmit Block Start

The SPDIFn_TX_BLKSTART output signal, if routed to any miscellaneous interrupt bits (DAIn_INT_31-22 in the DAI_MISC0/DAI_MISC1 registers), triggers a block start interrupt during the last frame of current block.

Receiver Status

The following receiver status bits generate an interrupt.

- Validity (SPDIF_RX_STAT.VALID)
- Receiver locked (SPDIF_RX_STAT.LOCK)
- No audio (SPDIF_RX_STAT.AUDIOTYPE)

Receiver Error

The loss of lock (SPDIF_RX_STAT.LOCKLOSS) bit generates an interrupt.

Masking

For the S/PDIF receiver the DAI_IMSK_RE register must be unmasked accordingly. For the S/PDIF transmit the DAIn_IMASK_x register must be unmasked accordingly.

The INTR_DAI_IRQH and INTR_DAI_IRQL signals are routed to the SEC.

Service

The ISR reads the DAI_IRPTL_H and DAI_IRPTL_L registers to clear the interrupt request.

Programming Model

The following sections provide information on programming the transmitter and receiver.

Programming the Transmitter

Since the S/PDIF transmitter data input is not available to the core, programming the transmitter is as simple as: 1) connecting the SRU to the on-chip (serial ports or input data port) or off-chip (DAI pins) serial devices that provide the clock and data to be encoded, and 2) selecting the desired mode in the transmitter control register. This setup can be accomplished in three steps.

1. Connect the transmitter's four required input signals and one biphase encoded output in the SRU. The four input signals are the serial clock (SPDIF_TX_CLK_I), the serial frame sync (SPDIF_TX_FS_I), the serial data (SPDIF_TX_DAT_I), and the high frequency clock (SPDIF_TX_HFCLK_I) used for the encoding. The only output of the transmitter is SPDIF_TX_O.
2. If user bits are required, write 0x1 to the SPDIF_TX_USRUPDT register for the first block of transfer. Also route the SPDIF_TX_BLK_START_O signal to the DAI_INT_31-22 (DAI_IRPTLx register). This generates interrupts during the last frame of the block (192), allowing changes of user bits for the next block.
3. Initialize the SPDIF_TX_CTL register to enable the data encoding.
4. Manually set the block start bit in the data stream once per block (every 384 words). This is necessary if automatic generation of block start information is not enabled using the SPDIF_TX_CTL.AUTO bit = 0.

NOTE: For more information, see the "DAI Routing Capabilities" section of the *Digital Audio Interface (DAI)* chapter.

Programming the Receiver

Since the S/PDIF receiver data output is not available to the core, programming the peripheral is as simple as connecting the SRU to the on-chip (serial ports) or off-chip (DAI pins) serial devices that provide the clock and data to be decoded, and selecting the desired mode in the receiver control register. This setup can be accomplished in two steps.

1. Connect the input signal and three output signals in the SRU. The only input of the receiver is the biphase encoded stream, SPDIFn_RX_I. The three required output signals are the serial clock (SPDIFn_RX_CLK), the serial frame sync (SPDIFn_RX_FS), and the serial data (SPDIFn_RX_DAT). The high frequency clock (SPDIFn_RX_TDMCLK) derived from the encoded stream is also available if the system requires it.
2. Initialize the SPDIF_RX_CTL register to enable the data decoding. Note that this peripheral is disabled by default.

NOTE: For more information, see the "DAI Routing Capabilities" section of the *Digital Audio Interface (DAI)* chapter.

Interrupted Data Streams on the Receiver

When using the S/PDIF receiver with data streams that are likely to be interrupted, (in other words unplugged and reconnected), it is necessary to take some extra steps to ensure that the S/PDIF receiver's digital PLL will relock to the stream. The steps to accomplish this are described below.

1. Set up interrupts within the DAI so that the S/PDIF receiver can generate an interrupt when the stream is reconnected.
2. Within the interrupt service routine (ISR), stop and restart the NCO. This is accomplished by setting and then clearing the SPDIF_RX_CTL.RST bit.
3. Return from the ISR and continue normal operation.

This method of resetting the NCO has been shown to provide extremely reliable performance when the S/PDIF inputs are interrupted or unplugged momentarily.

Debug Features

The following feature supports S/PDIF debugging.

Loopback Routing

The S/PDIF supports an internal loopback mode by using the SRU. For more information about loopback, see "Loopback Routing" in the *Digital Audio Interface (DAI)* chapter.

ADSP-SC59x SPDIF Register Descriptions

The S/PDIF module (SPDIF) contains the following registers.

Table 35-3: ADSP-SC59x SPDIF Register List

Name	Description
SPDIF_RX_CTL	Receive Control
SPDIF_RX_STAT	Receive Status Register
SPDIF_RX_STAT0_A	Receive Status A0 Register
SPDIF_RX_STAT0_B	Receive Status B0 Register
SPDIF_RX_STAT1_A	Receive Status A1 Register
SPDIF_RX_STAT1_B	Receive Status B1 Register
SPDIF_TX_CTL	Transmit Control Register
SPDIF_TX_STAT_A0	Transmit Status A0 Register
SPDIF_TX_STAT_A1	Transmit Status A1 Register
SPDIF_TX_STAT_A2	Transmit Status A2 Register
SPDIF_TX_STAT_A3	Transmit Status A3 Register
SPDIF_TX_STAT_A4	Transmit Status A4 Register
SPDIF_TX_STAT_A5	Transmit Status A5 Register
SPDIF_TX_STAT_B0	Transmit Status B0 Register
SPDIF_TX_STAT_B1	Transmit Status B1 Register
SPDIF_TX_STAT_B2	Transmit Status B2 Register
SPDIF_TX_STAT_B3	Transmit Status B3 Register
SPDIF_TX_STAT_B4	Transmit Status B4 Register
SPDIF_TX_STAT_B5	Transmit Status B5 Register

Table 35-3: ADSP-SC59x SPDIF Register List (Continued)

Name	Description
SPDIF_TX_UBUFF_A0	Transmit User Buffer A0 Register
SPDIF_TX_UBUFF_A1	Transmit User Buffer A1 Register
SPDIF_TX_UBUFF_A2	Transmit User Buffer A2 Register
SPDIF_TX_UBUFF_A3	Transmit User Buffer A3 Register
SPDIF_TX_UBUFF_A4	Transmit User Buffer A4 Register
SPDIF_TX_UBUFF_A5	Transmit User Buffer A5 Register
SPDIF_TX_UBUFF_B0	Transmit User Buffer B0 Register
SPDIF_TX_UBUFF_B1	Transmit User Buffer B1 Register
SPDIF_TX_UBUFF_B2	Transmit User Buffer B2 Register
SPDIF_TX_UBUFF_B3	Transmit User Buffer B3 Register
SPDIF_TX_UBUFF_B4	Transmit User Buffer B4 Register
SPDIF_TX_UBUFF_B5	Transmit User Buffer B5 Register
SPDIF_TX_USRUPDT	User Bit Update Register

Receive Control

The SPDIF_RX_CTL register is used to enable and control the S/PDIF receiver.

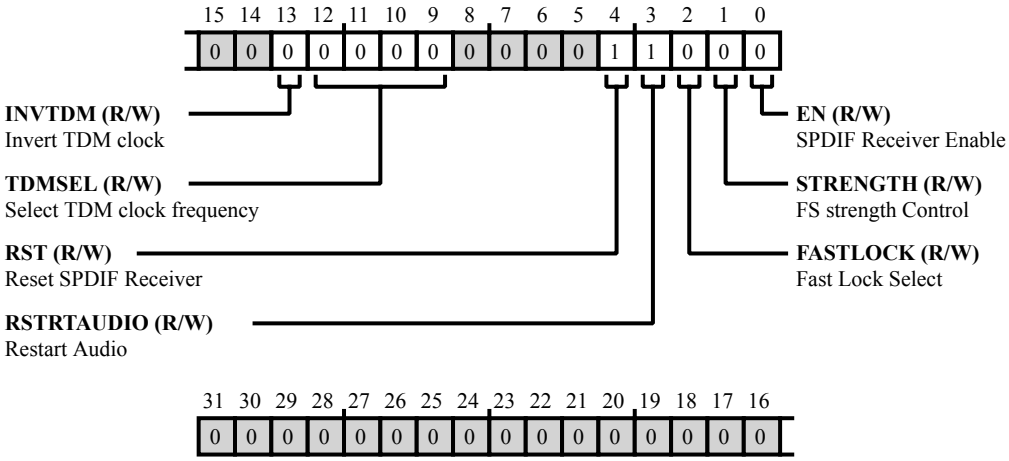


Figure 35-8: SPDIF_RX_CTL Register Diagram

Table 35-4: SPDIF_RX_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	INVTDM	Invert TDM clock. When set to 0, inverted TDM clock or non-inverted Bit clock is selected. When set to 1, non-inverted TDM clock or inverted Blit clock is selected.	
12:9 (R/W)	TDMSEL	Select TDM clock frequency. Select TDM clock frequency	
4 (R/W)	RST	Reset SPDIF Receiver. The SPDIF_RX_CTL.RST bit resets the receiver.	
		0	Takes the receiver out of reset
		1	Resets the SPDIF receiver
3 (R/W)	RSTRTAUDIO	Restart Audio. The SPDIF_RX_CTL.RSTRTAUDIO bit restarts the audio once a re-lock has occurred. When the S/PDIF receiver loses lock the audio output is set to 0. This bit determines the behavior of the audio once lock is re-established. Audio can be manually restarted by toggling this bit high and then low.	
		0	Manually restart audio
		1	Automatically restart audio

Table 35-4: SPDIF_RX_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	FASTLOCK	Fast Lock Select. The `SPDIF_RX_CTL.FASTLOCK` bit allows the lock mechanism to lock at normal speed or at faster speed. This has the advantage of recovering very quickly whenever the S/PDIF receiver looses lock due to glitches in the signal. In normal mode the S/PDIF receiver locks after 64 consecutive valid samples, in fast mode the S/PDIF receiver locks after 8 consecutive valid samples	
		0	Enable normal mode
		1	Enable fast mode
1 (R/W)	STRENGTH	FS strength Control. The `SPDIF_RX_CTL.STRENGTH` bit controls the strength of the bit clock and Frame sync outputs from the SPDIF receiver. In strong mode these output signals are continued (as best possible) when the receiver notices a loss-of-lock condition. Note that 'as best possible' refers to the fact that this recovered signal may not be accurate, given the loss-of-lock condition. In weak mode these output signals are interrupted as soon as the receiver notices a loss-of-lock condition.	
		0	Enable strong mode
		1	Enable weak mode
0 (R/W)	EN	SPDIF Receiver Enable. When the `SPDIF_RX_CTL.EN` bit =0 the clock to SPDIF is switched off for power savings.	
		0	Disable receiver
		1	Enable receiver

Receive Status Register

The SPDIF_RX_STAT register consists of bits that indicate the status of various functions supported by S/PDIF receiver.

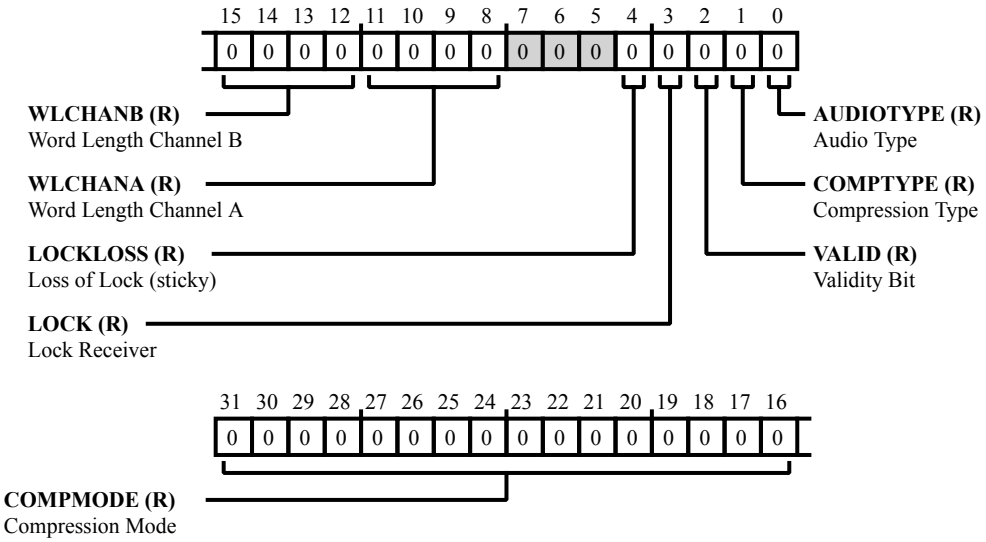


Figure 35-9: SPDIF_RX_STAT Register Diagram

Table 35-5: SPDIF_RX_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16 (R/NW)	COMPMODE	Compression Mode. The SPDIF_RX_STAT.COMPMODE bit field indicates the type of compression mode used in the Digital audio stream. The value in this field indicates the 16-bit burst information as specified by the IEC 62937-2 standard. Use this document to decode the value in this bit field.	
15:12 (R/NW)	WLCHANB	Word Length Channel B. The SPDIF_RX_STAT.WLCHANB bit field indicates the S/PDIF word length for channel B. May be decoded as follows (from the S/PDIF standard).	
		0	Reserved
		1	Reserved
		2	SPDIF_LENGTH_16
		3	Reserved
		4	SPDIF_LENGTH_18
		5	SPDIF_LENGTH_22
		6	Reserved

Table 35-5: SPDIF_RX_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		7	Reserved
		8	SPDIF_LENGTH_19
		9	SPDIF_LENGTH_23
		10	SPDIF_LENGTH_20
		11	SPDIF_LENGTH_24
		12	SPDIF_LENGTH_17
		13	SPDIF_LENGTH_21
		14	Reserved
		15	Reserved
11:8 (R/NW)	WLCHANA	Word Length Channel A. The `SPDIF_RX_STAT.WLCHANA` bit indicates the S/PDIF word length for channel A. May be decoded as follows (from the S/PDIF standard).	
		0	Reserved
		1	Reserved
		2	SPDIF_LENGTH_16
		3	Reserved
		4	SPDIF_LENGTH_18
		5	SPDIF_LENGTH_22
		6	Reserved
		7	Reserved
		8	SPDIF_LENGTH_19
		9	SPDIF_LENGTH_23
		10	SPDIF_LENGTH_20
		11	SPDIF_LENGTH_24
		12	SPDIF_LENGTH_17
		13	SPDIF_LENGTH_21
		14	Reserved
		15	Reserved

Table 35-5: SPDIF_RX_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/NW)	LOCKLOSS	Loss of Lock (sticky). The SPDIF_RX_STAT.LOCKLOSS bit indicates that the system has lost lock. This is different to the lock register, as it is sticky it goes high when system looses lock, but returns to low once SPDIF_RX_CTL.RSTRTAUDIO bit is toggled. This is to allow the programs to poll the lock status.	
		0	SPDIF receiver locked
		1	SPDIF Receiver lost lock
3 (R/NW)	LOCK	Lock Receiver. The SPDIF_RX_STAT.LOCK bit indicates the S/PDIF receiver has successfully locked to the S/PDIF stream and is outputting valid data.	
		0	Receiver not locked
		1	Receiver locked
2 (R/NW)	VALID	Validity Bit. The SPDIF_RX_STAT.VALID bit indicates the ORed bits of channel A and B.	
		0	Linear PCM data
		1	Non-linear audio data
1 (R/NW)	COMPTYPE	Compression Type. The SPDIF_RX_STAT.COMPTYPE bit indicates AC3 or DTS compression. Valid only if SPDIF_RX_STAT.AUDIOTYPE indicates compressed data.	
		0	AC3 compressed data
		1	DTS compressed data
0 (R/NW)	AUDIOTYPE	Audio Type. The SPDIF_RX_STAT.AUDIOTYPE bit indicates PCM or compressed data.	
		0	PCM data
		1	Compressed Data

Receive Status A0 Register

The SPDIF_RX_STAT0_A register holds the receive channel 0 status for bytes 0-3 for sub frame A.

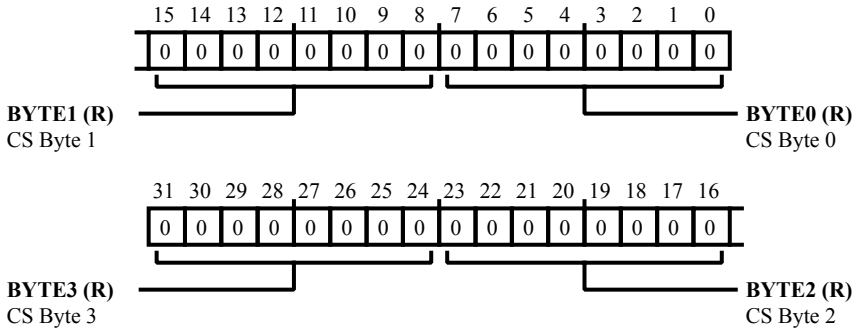


Figure 35-10: SPDIF_RX_STAT0_A Register Diagram

Table 35-6: SPDIF_RX_STAT0_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	BYTE3	CS Byte 3. The SPDIF_RX_STAT0_A.BYTE3 bit field contains byte 3 of received channel A status.
23:16 (R/NW)	BYTE2	CS Byte 2. The SPDIF_RX_STAT0_A.BYTE2 bit field contains byte 2 of received channel A status.
15:8 (R/NW)	BYTE1	CS Byte 1. The SPDIF_RX_STAT0_A.BYTE1 bit field contains byte 1 of received channel A status.
7:0 (R/NW)	BYTE0	CS Byte 0. The SPDIF_RX_STAT0_A.BYTE0 bit field contains the status of byte 0-3 of received channel A.

Receive Status B0 Register

The SPDIF_RX_STAT0_B register holds the receive channel 0 status for bytes 0-3 for sub frame B.

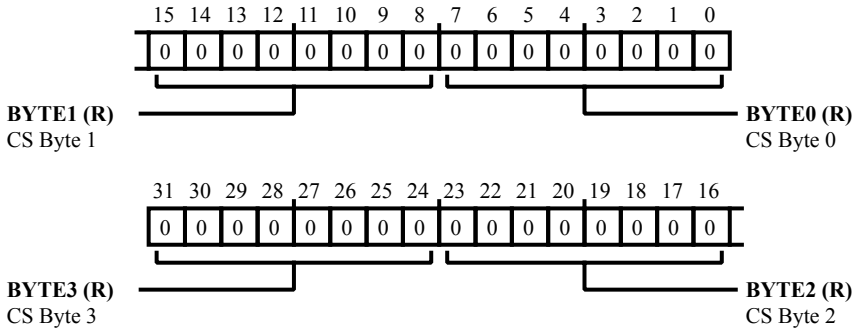


Figure 35-11: SPDIF_RX_STAT0_B Register Diagram

Table 35-7: SPDIF_RX_STAT0_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	BYTE3	CS Byte 3. The SPDIF_RX_STAT0_B.BYTE3 bit field contains byte 3 of received channel B status.
23:16 (R/NW)	BYTE2	CS Byte 2. The SPDIF_RX_STAT0_B.BYTE2 bit field contains byte 2 of received channel B status.
15:8 (R/NW)	BYTE1	CS Byte 1. The SPDIF_RX_STAT0_B.BYTE1 bit field contains byte 1 of received channel B status.
7:0 (R/NW)	BYTE0	CS Byte 0. The SPDIF_RX_STAT0_B.BYTE0 bit field contains byte 0 of received channel B status.

Receive Status A1 Register

The SPDIF_RX_STAT1_A register holds the receive channel 1 status for byte 4 for sub frame A.

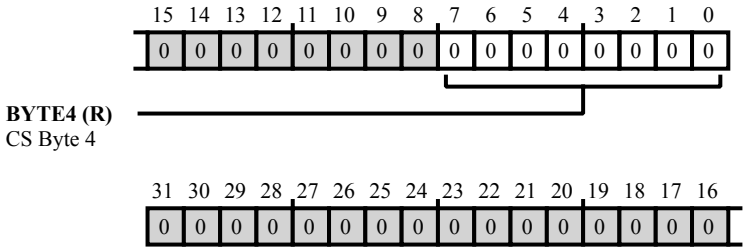


Figure 35-12: SPDIF_RX_STAT1_A Register Diagram

Table 35-8: SPDIF_RX_STAT1_A Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	BYTE4	CS Byte 4. The SPDIF_RX_STAT1_A.BYTE4 bit field contains byte 4 of received channel A status.

Receive Status B1 Register

The SPDIF_RX_STAT1_B register holds the receive channel 1 status for byte 4 for sub frame B.

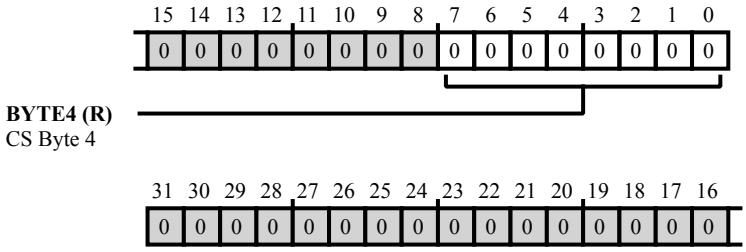


Figure 35-13: SPDIF_RX_STAT1_B Register Diagram

Table 35-9: SPDIF_RX_STAT1_B Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	BYTE4	CS Byte 4. The SPDIF_RX_STAT1_B.BYTE4 bit field contains byte 4 of received channel B status.

Transmit Control Register

The SPDIF_TX_CTL register provides bits the enable/disable the transmitter and configure several options related to data transmission.

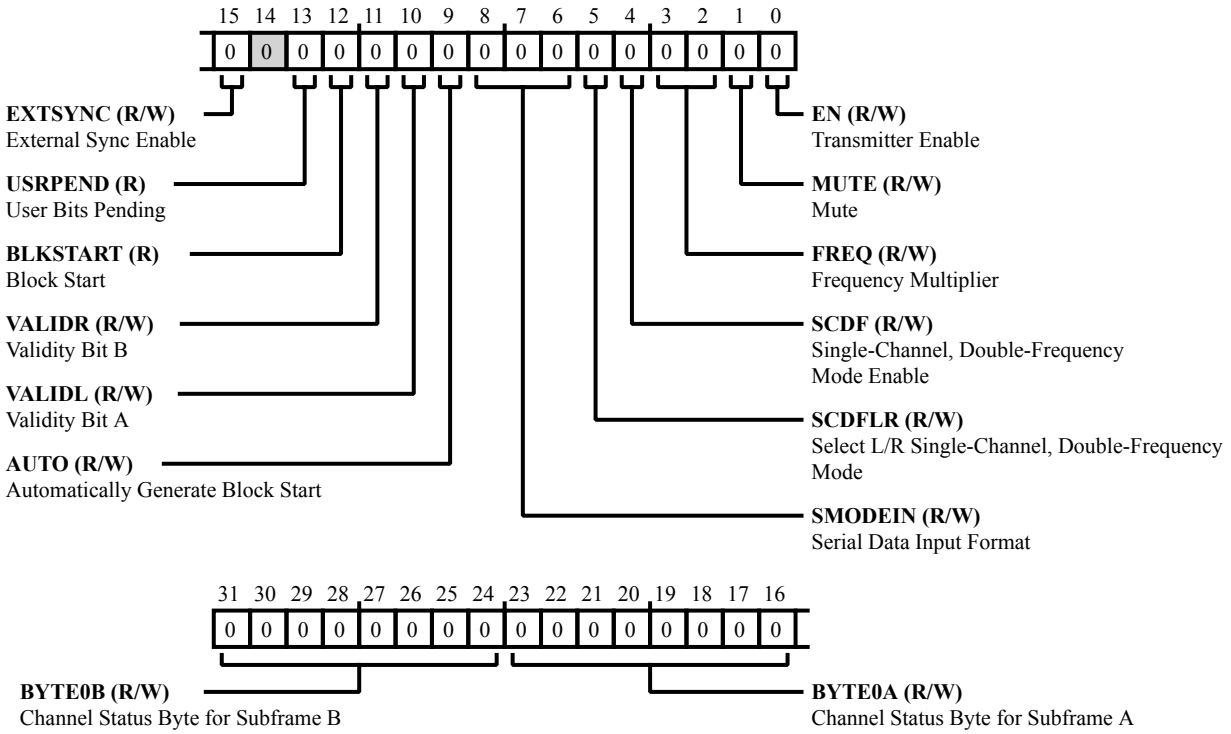


Figure 35-14: SPDIF_TX_CTL Register Diagram

Table 35-10: SPDIF_TX_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE0B	Channel Status Byte for Subframe B. The SPDIF_TX_CTL.BYTE0B bit field contains the channel status for the second bytes 95.
23:16 (R/W)	BYTE0A	Channel Status Byte for Subframe A. The SPDIF_TX_CTL.BYTE0A bit field contains the channel status for the first bytes 40.
15 (R/W)	EXTSYNC	External Sync Enable. When the SPDIF_TX_CTL.EXTSYNC bit is set (regardless of the SPDIF_TX_CTL.AUTO bit setting) the internal frame counter is set to zero at an internal LRCLK rising edge followed by an EXTSYNC_I rising edge.

Table 35-10: SPDIF_TX_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/NW)	USRPEND	User Bits Pending. The SPDIF_TX_CTL.USRPEND bit is set if the update of the internal buffer from the Transmit User Bits Buffer registers has not yet completed.	
12 (R/NW)	BLKSTART	Block Start. The SPDIF_TX_CTL.BLKSTART bit is a status bit that indicates block start (when the SPDIF_TX_CTL.AUTO bit = 1).	
		0	Current word is not block start
		1	Current word is block start
11 (R/W)	VALIDR	Validity Bit B. Use the SPDIF_TX_CTL.VALIDR bit with the channel status buffer.	
10 (R/W)	VALIDL	Validity Bit A. The SPDIF_TX_CTL.VALIDL bit either manually start block transfer according to input stream status bits or automatically start block transfer. Use the SPDIF_TX_CTL.VALIDL bit with the channel status buffer.	
		0	Manually start block transfer
		1	Automatically start block transfer
9 (R/W)	AUTO	Automatically Generate Block Start. When enabled, the transmitter is in standalone mode where it inserts block start, channel status, and validity bits on its own. If the channel status or validity buffer needs to be enabled (after the DAI programming is complete), first write to the buffers with the required data and then enable the buffers by setting the SPDIF_TX_CTL.AUTO bit.	
8:6 (R/W)	SMODEIN	Serial Data Input Format. The SPDIF_TX_CTL.SMODEIN bit selects the data input format.	
		0	Left-justified
		1	I2S
		2	Reserved
		3	Reserved
		4	Right-justified, 24 bits
		5	Right-justified, 20 bits
		6	Right-justified, 18 bits
		7	Right-justified, 16 bits

Table 35-10: SPDIF_TX_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	SCDFLR	Select L/R Single-Channel, Double-Frequency Mode. The `SPDIF_TX_CTL.SCDFLR` bit selects the left or right channel in SCDF mode.	
		0	Left channel
		1	Right channel
4 (R/W)	SCDF	Single-Channel, Double-Frequency Mode Enable. The `SPDIF_TX_CTL.SCDF` bit enables single-channel, double-frequency mode.	
		0	Two-channel mode
		1	SCDF mode
3:2 (R/W)	FREQ	Frequency Multiplier. The `SPDIF_TX_CTL.FREQ` bit field sets the oversampling ratio.	
		0	256 x frame sync oversampling
		1	384 x frame sync oversampling
		2-3	Reserved
1 (R/W)	MUTE	Mute. The `SPDIF_TX_CTL.MUTE` bit mutes the serial data output.	
		0	Disable Mute
		1	Enable Mute
0 (R/W)	EN	Transmitter Enable. The `SPDIF_TX_CTL.EN` bit enables the transmitter and resets the control registers to their defaults.	
		0	Transmitter disabled
		1	Transmitter enabled

Transmit Status A0 Register

The SPDIF_TX_STAT_A0 register holds the transmit channel 0 status for bytes 1-4 for sub frame A.

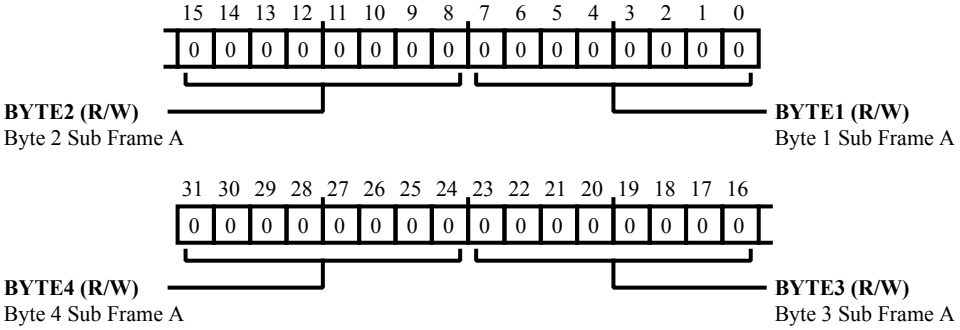


Figure 35-15: SPDIF_TX_STAT_A0 Register Diagram

Table 35-11: SPDIF_TX_STAT_A0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE4	Byte 4 Sub Frame A. The SPDIF_TX_STAT_A0.BYTE4 bit field holds the transmit channel 0 status for byte 4 for sub frame A.
23:16 (R/W)	BYTE3	Byte 3 Sub Frame A. The SPDIF_TX_STAT_A0.BYTE3 bit field holds the transmit channel 0 status for byte 3 for sub frame A.
15:8 (R/W)	BYTE2	Byte 2 Sub Frame A. The SPDIF_TX_STAT_A0.BYTE2 bit field holds the transmit channel 0 status for byte 2 for sub frame A.
7:0 (R/W)	BYTE1	Byte 1 Sub Frame A. The SPDIF_TX_STAT_A0.BYTE1 bit field holds the transmit channel 0 status for byte 1 for sub frame A.

Transmit Status A1 Register

The SPDIF_TX_STAT_A1 register holds the transmit status for bytes 5-8 for sub frame A.

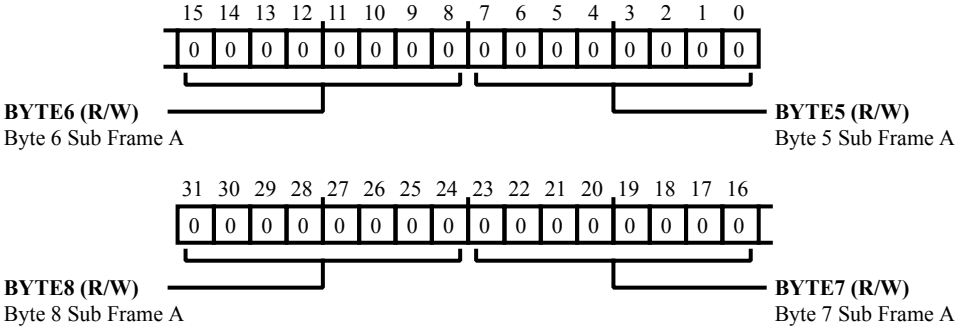


Figure 35-16: SPDIF_TX_STAT_A1 Register Diagram

Table 35-12: SPDIF_TX_STAT_A1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE8	Byte 8 Sub Frame A. The SPDIF_TX_STAT_A1.BYTE8 bit field contains transmit status for byte 8 of sub frame A.
23:16 (R/W)	BYTE7	Byte 7 Sub Frame A. The SPDIF_TX_STAT_A1.BYTE7 bit field contains transmit status for byte 7 of sub frame A.
15:8 (R/W)	BYTE6	Byte 6 Sub Frame A. The SPDIF_TX_STAT_A1.BYTE6 bit field contains transmit status for byte 6 of sub frame A.
7:0 (R/W)	BYTE5	Byte 5 Sub Frame A. The SPDIF_TX_STAT_A1.BYTE5 bit field contains transmit status for byte 5 of sub frame A.

Transmit Status A2 Register

The SPDIF_TX_STAT_A2 register holds the transmit status for bytes 9-12 for sub frame A.

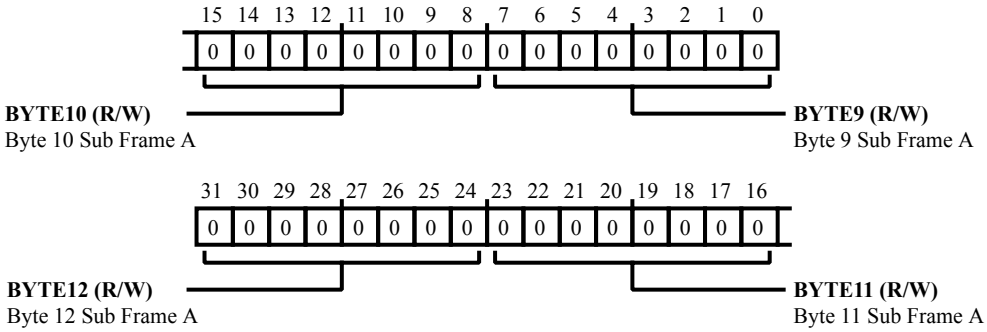


Figure 35-17: SPDIF_TX_STAT_A2 Register Diagram

Table 35-13: SPDIF_TX_STAT_A2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE12	Byte 12 Sub Frame A. The SPDIF_TX_STAT_A2.BYTE12 bit field contains transmit status for byte 12 of sub frame A.
23:16 (R/W)	BYTE11	Byte 11 Sub Frame A. The SPDIF_TX_STAT_A2.BYTE11 bit field contains transmit status for byte 11 of sub frame A.
15:8 (R/W)	BYTE10	Byte 10 Sub Frame A. The SPDIF_TX_STAT_A2.BYTE10 bit field contains transmit status for byte 10 of sub frame A.
7:0 (R/W)	BYTE9	Byte 9 Sub Frame A. The SPDIF_TX_STAT_A2.BYTE9 bit field contains transmit status for byte 9 of sub frame A.

Transmit Status A3 Register

The SPDIF_TX_STAT_A3 register holds the transmit status for bytes 13-16 for sub frame A.

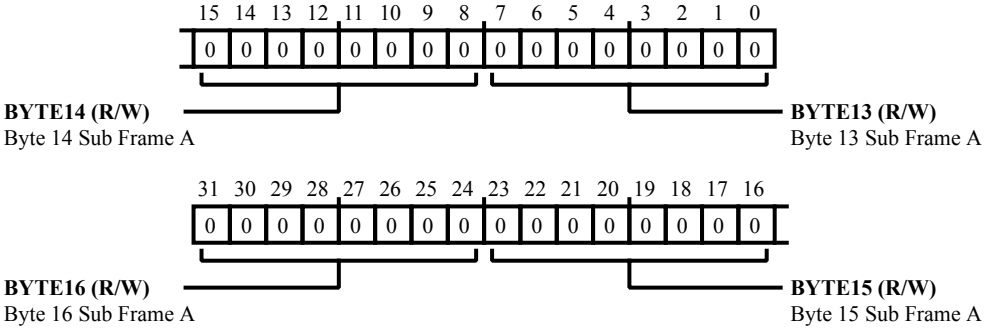


Figure 35-18: SPDIF_TX_STAT_A3 Register Diagram

Table 35-14: SPDIF_TX_STAT_A3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE16	Byte 16 Sub Frame A. The SPDIF_TX_STAT_A3.BYTE16 bit field contains transmit status for byte 16 of sub frame A.
23:16 (R/W)	BYTE15	Byte 15 Sub Frame A. The SPDIF_TX_STAT_A3.BYTE15 bit field contains transmit status for byte 15 of sub frame A.
15:8 (R/W)	BYTE14	Byte 14 Sub Frame A. The SPDIF_TX_STAT_A3.BYTE14 bit field contains transmit status for byte 14 of sub frame A.
7:0 (R/W)	BYTE13	Byte 13 Sub Frame A. The SPDIF_TX_STAT_A3.BYTE13 bit field contains transmit status for byte 13 of sub frame A.

Transmit Status A4 Register

The SPDIF_TX_STAT_A4 register holds the transmit status for bytes 17-20 for sub frame A.

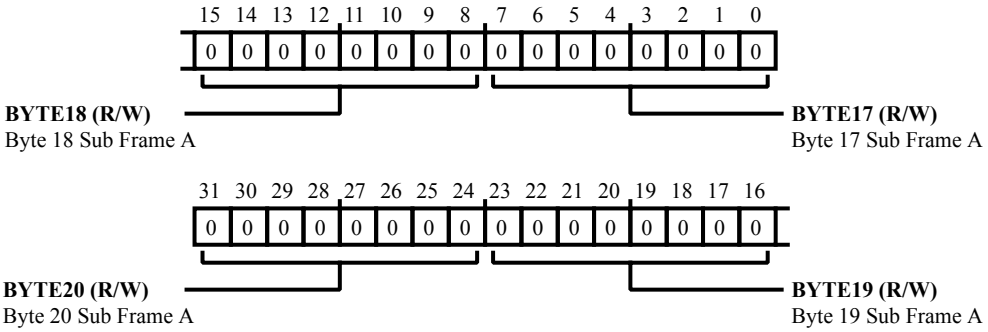


Figure 35-19: SPDIF_TX_STAT_A4 Register Diagram

Table 35-15: SPDIF_TX_STAT_A4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE20	Byte 20 Sub Frame A. The SPDIF_TX_STAT_A4.BYTE20 bit field contains transmit status for byte 20 of sub frame A.
23:16 (R/W)	BYTE19	Byte 19 Sub Frame A. The SPDIF_TX_STAT_A4.BYTE19 bit field contains transmit status for byte 19 of sub frame A.
15:8 (R/W)	BYTE18	Byte 18 Sub Frame A. The SPDIF_TX_STAT_A4.BYTE18 bit field contains transmit status for byte 18 of sub frame A.
7:0 (R/W)	BYTE17	Byte 17 Sub Frame A. The SPDIF_TX_STAT_A4.BYTE17 bit field contains transmit status for byte 17 of sub frame A.

Transmit Status A5 Register

The SPDIF_TX_STAT_A5 register holds the transmit status for bytes 21-23 for sub frame A.

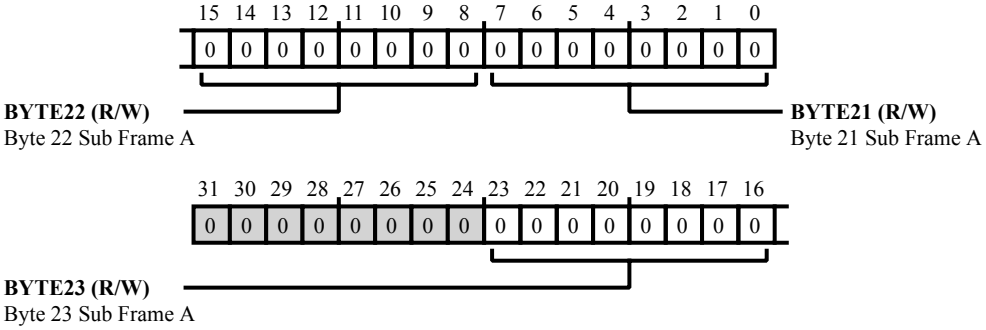


Figure 35-20: SPDIF_TX_STAT_A5 Register Diagram

Table 35-16: SPDIF_TX_STAT_A5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	BYTE23	Byte 23 Sub Frame A. The SPDIF_TX_STAT_A5.BYTE23 bit field contains transmit status for byte 23 of sub frame A.
15:8 (R/W)	BYTE22	Byte 22 Sub Frame A. The SPDIF_TX_STAT_A5.BYTE22 bit field contains transmit status for byte 22 of sub frame A.
7:0 (R/W)	BYTE21	Byte 21 Sub Frame A. The SPDIF_TX_STAT_A5.BYTE21 bit field contains transmit status for byte 21 of sub frame A.

Transmit Status B0 Register

The SPDIF_TX_STAT_B0 register holds the transmit channel 0 status for bytes 1-4 for sub frame B.

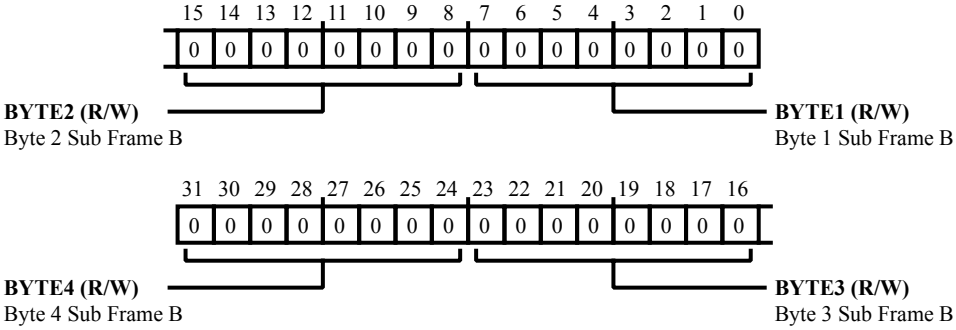


Figure 35-21: SPDIF_TX_STAT_B0 Register Diagram

Table 35-17: SPDIF_TX_STAT_B0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE4	Byte 4 Sub Frame B. The SPDIF_TX_STAT_B0.BYTE4 bit field holds the transmit channel 0 status for byte 4 for sub frame B.
23:16 (R/W)	BYTE3	Byte 3 Sub Frame B. The SPDIF_TX_STAT_B0.BYTE3 bit field holds the transmit channel 0 status for byte 3 for sub frame B.
15:8 (R/W)	BYTE2	Byte 2 Sub Frame B. The SPDIF_TX_STAT_B0.BYTE2 bit field holds the transmit channel 0 status for byte 2 for sub frame B.
7:0 (R/W)	BYTE1	Byte 1 Sub Frame B. The SPDIF_TX_STAT_B0.BYTE1 bit field holds the transmit channel 0 status for byte 1 for sub frame B.

Transmit Status B1 Register

The SPDIF_TX_STAT_B1 register holds the transmit status for bytes 5-8 for sub frame B.

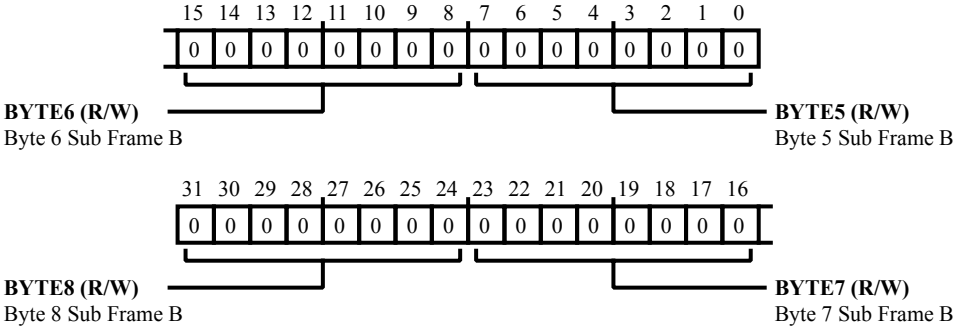


Figure 35-22: SPDIF_TX_STAT_B1 Register Diagram

Table 35-18: SPDIF_TX_STAT_B1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE8	Byte 8 Sub Frame B. The SPDIF_TX_STAT_B1.BYTE8 bit field contains transmit status for byte 8 of sub frame B.
23:16 (R/W)	BYTE7	Byte 7 Sub Frame B. The SPDIF_TX_STAT_B1.BYTE7 bit field contains transmit status for byte 7 of sub frame B.
15:8 (R/W)	BYTE6	Byte 6 Sub Frame B. The SPDIF_TX_STAT_B1.BYTE6 bit field contains transmit status for byte 6 of sub frame B.
7:0 (R/W)	BYTE5	Byte 5 Sub Frame B. The SPDIF_TX_STAT_B1.BYTE5 bit field contains transmit status for byte 5 of sub frame B.

Transmit Status B2 Register

The SPDIF_TX_STAT_B2 register holds the transmit status for bytes 9-12 for sub frame B.

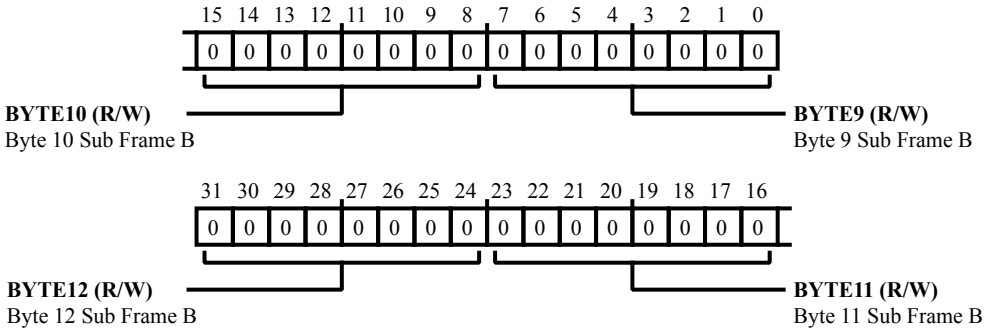


Figure 35-23: SPDIF_TX_STAT_B2 Register Diagram

Table 35-19: SPDIF_TX_STAT_B2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE12	Byte 12 Sub Frame B. The SPDIF_TX_STAT_B2.BYTE12 bit field contains transmit status for byte 12 of sub frame B.
23:16 (R/W)	BYTE11	Byte 11 Sub Frame B. The SPDIF_TX_STAT_B2.BYTE11 bit field contains transmit status for byte 11 of sub frame B.
15:8 (R/W)	BYTE10	Byte 10 Sub Frame B. The SPDIF_TX_STAT_B2.BYTE10 bit field contains transmit status for byte 10 of sub frame B.
7:0 (R/W)	BYTE9	Byte 9 Sub Frame B. The SPDIF_TX_STAT_B2.BYTE9 bit field contains transmit status for byte 9 of sub frame B.

Transmit Status B3 Register

The SPDIF_TX_STAT_B3 register holds the transmit status for bytes 13-16 for sub frame B.

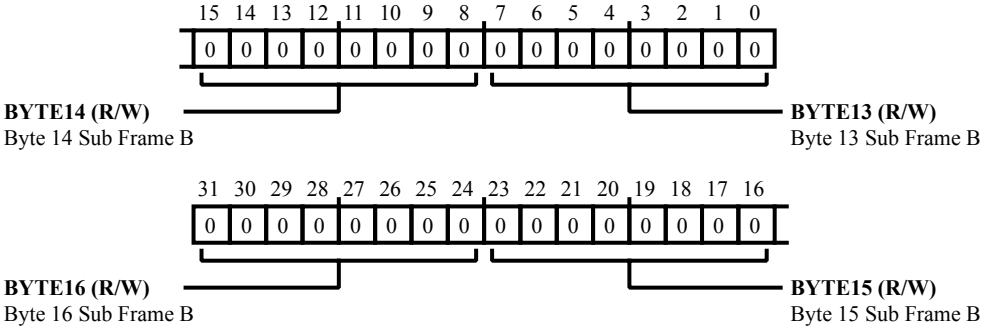


Figure 35-24: SPDIF_TX_STAT_B3 Register Diagram

Table 35-20: SPDIF_TX_STAT_B3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE16	Byte 16 Sub Frame B. The SPDIF_TX_STAT_B3.BYTE16 bit field contains transmit status for byte 16 of sub frame B.
23:16 (R/W)	BYTE15	Byte 15 Sub Frame B. The SPDIF_TX_STAT_B3.BYTE15 bit field contains transmit status for byte 15 of sub frame B.
15:8 (R/W)	BYTE14	Byte 14 Sub Frame B. The SPDIF_TX_STAT_B3.BYTE14 bit field contains transmit status for byte 14 of sub frame B.
7:0 (R/W)	BYTE13	Byte 13 Sub Frame B. The SPDIF_TX_STAT_B3.BYTE13 bit field contains transmit status for byte 13 of sub frame B.

Transmit Status B4 Register

The SPDIF_TX_STAT_B4 register holds the transmit status for bytes 17-20 for sub frame B.

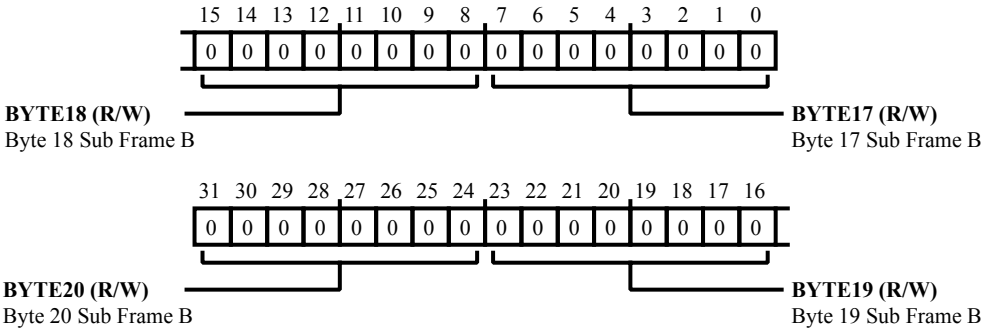


Figure 35-25: SPDIF_TX_STAT_B4 Register Diagram

Table 35-21: SPDIF_TX_STAT_B4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE20	Byte 20 Sub Frame B. The SPDIF_TX_STAT_B4.BYTE20 bit field contains transmit status for byte 20 of sub frame B.
23:16 (R/W)	BYTE19	Byte 19 Sub Frame B. The SPDIF_TX_STAT_B4.BYTE19 bit field contains transmit status for byte 19 of sub frame B.
15:8 (R/W)	BYTE18	Byte 18 Sub Frame B. The SPDIF_TX_STAT_B4.BYTE18 bit field contains transmit status for byte 18 of sub frame B.
7:0 (R/W)	BYTE17	Byte 17 Sub Frame B. The SPDIF_TX_STAT_B4.BYTE17 bit field contains transmit status for byte 17 of sub frame B.

Transmit Status B5 Register

The SPDIF_TX_STAT_B5 register holds the transmit status for bytes 21-23 for sub frame B.

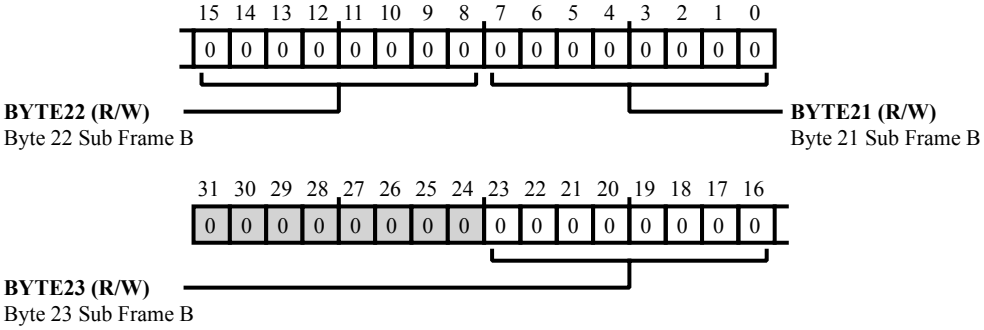


Figure 35-26: SPDIF_TX_STAT_B5 Register Diagram

Table 35-22: SPDIF_TX_STAT_B5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	BYTE23	Byte 23 Sub Frame B. The SPDIF_TX_STAT_B5.BYTE23 bit field contains transmit status for byte 23 of sub frame B.
15:8 (R/W)	BYTE22	Byte 22 Sub Frame B. The SPDIF_TX_STAT_B5.BYTE22 bit field contains transmit status for byte 22 of sub frame B.
7:0 (R/W)	BYTE21	Byte 21 Sub Frame B. The SPDIF_TX_STAT_B5.BYTE21 bit field contains transmit status for byte 21 of sub frame B.

Transmit User Buffer A0 Register

The SPDIF_TX_UBUFF_A0 register holds the transmit user buffer data for bytes 0-3 for sub frame A.

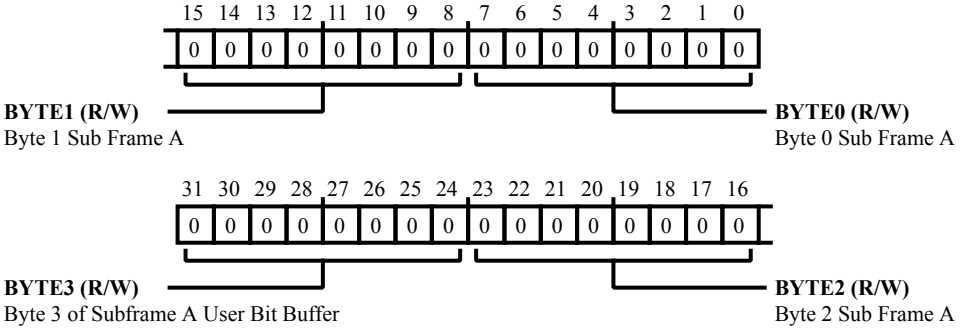


Figure 35-27: SPDIF_TX_UBUFF_A0 Register Diagram

Table 35-23: SPDIF_TX_UBUFF_A0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE3	Byte 3 of Subframe A User Bit Buffer. The SPDIF_TX_UBUFF_A0.BYTE3 bit field contains user bit data for byte 0 of sub frame A.
23:16 (R/W)	BYTE2	Byte 2 Sub Frame A. The SPDIF_TX_UBUFF_A0.BYTE2 bit field contains user bit data for byte 2 of sub frame A.
15:8 (R/W)	BYTE1	Byte 1 Sub Frame A. The SPDIF_TX_UBUFF_A0.BYTE1 bit field contains user bit data for byte 1 of sub frame A.
7:0 (R/W)	BYTE0	Byte 0 Sub Frame A. The SPDIF_TX_UBUFF_A0.BYTE0 bit field contains user bit data for byte 0 of sub frame A.

Transmit User Buffer A1 Register

The SPDIF_TX_UBUFF_A1 register holds the transmit user buffer data for bytes 4-7 for sub frame A.

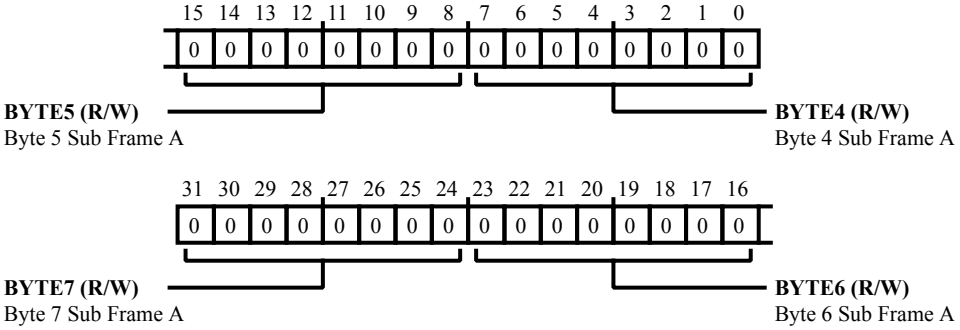


Figure 35-28: SPDIF_TX_UBUFF_A1 Register Diagram

Table 35-24: SPDIF_TX_UBUFF_A1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE7	Byte 7 Sub Frame A. The SPDIF_TX_UBUFF_A1.BYTE7 bit field contains user bit data for byte 7 of sub frame A.
23:16 (R/W)	BYTE6	Byte 6 Sub Frame A. The SPDIF_TX_UBUFF_A1.BYTE6 bit field contains user bit data for byte 6 of sub frame A.
15:8 (R/W)	BYTE5	Byte 5 Sub Frame A. The SPDIF_TX_UBUFF_A1.BYTE5 bit field contains user bit data for byte 5 of sub frame A.
7:0 (R/W)	BYTE4	Byte 4 Sub Frame A. The SPDIF_TX_UBUFF_A1.BYTE4 bit field contains user bit data for byte 0 of sub frame A.

Transmit User Buffer A2 Register

The SPDIF_TX_UBUFF_A2 register holds the transmit user buffer data for bytes 8-11 for sub frame A.

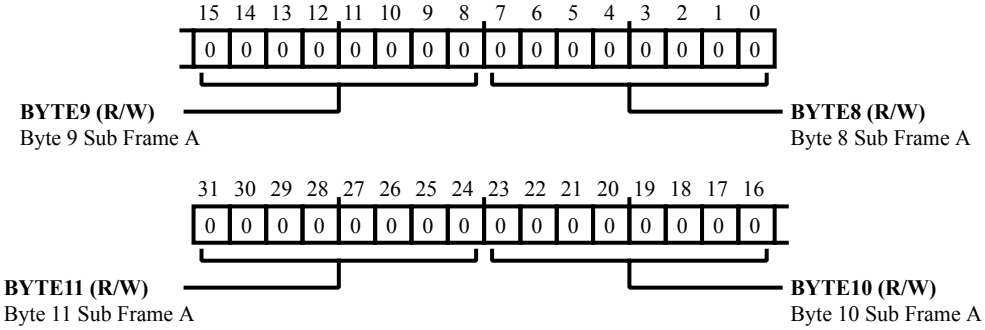


Figure 35-29: SPDIF_TX_UBUFF_A2 Register Diagram

Table 35-25: SPDIF_TX_UBUFF_A2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE11	Byte 11 Sub Frame A. The SPDIF_TX_UBUFF_A2.BYTE11 bit field contains user bit data for byte 11 of sub frame A.
23:16 (R/W)	BYTE10	Byte 10 Sub Frame A. The SPDIF_TX_UBUFF_A2.BYTE10 bit field contains user bit data for byte 10 of sub frame A.
15:8 (R/W)	BYTE9	Byte 9 Sub Frame A. The SPDIF_TX_UBUFF_A2.BYTE9 bit field contains user bit data for byte 9 of sub frame A.
7:0 (R/W)	BYTE8	Byte 8 Sub Frame A. The SPDIF_TX_UBUFF_A2.BYTE8 bit field contains user bit data for byte 8 of sub frame A.

Transmit User Buffer A3 Register

The SPDIF_TX_UBUFF_A3 register holds the transmit user buffer data for bytes 12-15 for sub frame A.

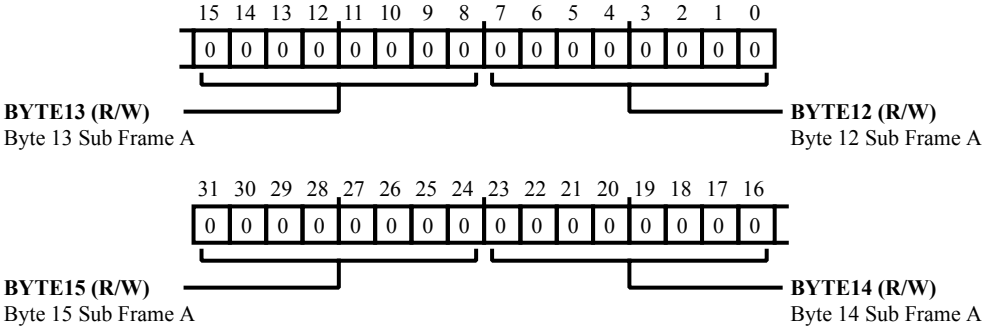


Figure 35-30: SPDIF_TX_UBUFF_A3 Register Diagram

Table 35-26: SPDIF_TX_UBUFF_A3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE15	Byte 15 Sub Frame A. The SPDIF_TX_UBUFF_A3.BYTE15 bit field contains user bit data for byte 15 of sub frame A.
23:16 (R/W)	BYTE14	Byte 14 Sub Frame A. The SPDIF_TX_UBUFF_A3.BYTE14 bit field contains user bit data for byte 14 of sub frame A.
15:8 (R/W)	BYTE13	Byte 13 Sub Frame A. The SPDIF_TX_UBUFF_A3.BYTE13 bit field contains user bit data for byte 13 of sub frame A.
7:0 (R/W)	BYTE12	Byte 12 Sub Frame A. The SPDIF_TX_UBUFF_A3.BYTE12 bit field contains user bit data for byte 12 of sub frame A.

Transmit User Buffer A4 Register

The SPDIF_TX_UBUFF_A4 register holds the transmit user buffer data for bytes 16-19 for sub frame A.

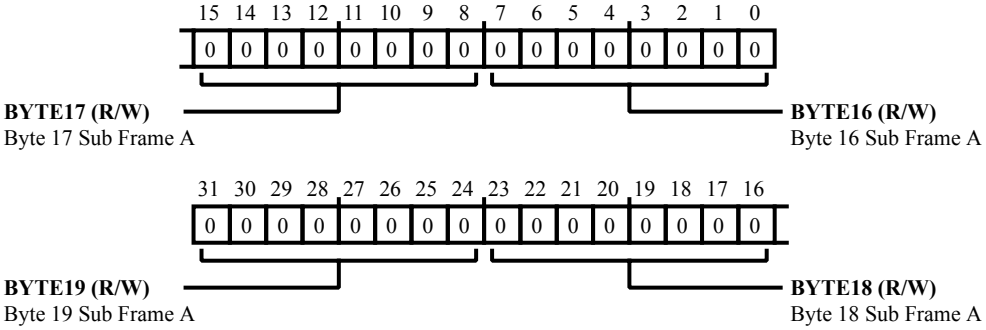


Figure 35-31: SPDIF_TX_UBUFF_A4 Register Diagram

Table 35-27: SPDIF_TX_UBUFF_A4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE19	Byte 19 Sub Frame A. The SPDIF_TX_UBUFF_A4.BYTE19 bit field contains user bit data for byte 19 of sub frame A.
23:16 (R/W)	BYTE18	Byte 18 Sub Frame A. The SPDIF_TX_UBUFF_A4.BYTE18 bit field contains user bit data for byte 18 of sub frame A.
15:8 (R/W)	BYTE17	Byte 17 Sub Frame A. The SPDIF_TX_UBUFF_A4.BYTE17 bit field contains user bit data for byte 17 of sub frame A.
7:0 (R/W)	BYTE16	Byte 16 Sub Frame A. The SPDIF_TX_UBUFF_A4.BYTE16 bit field contains user bit data for byte 16 of sub frame A.

Transmit User Buffer A5 Register

The SPDIF_TX_UBUFF_A5 register holds the transmit user buffer data for bytes 20-23 for sub frame A.

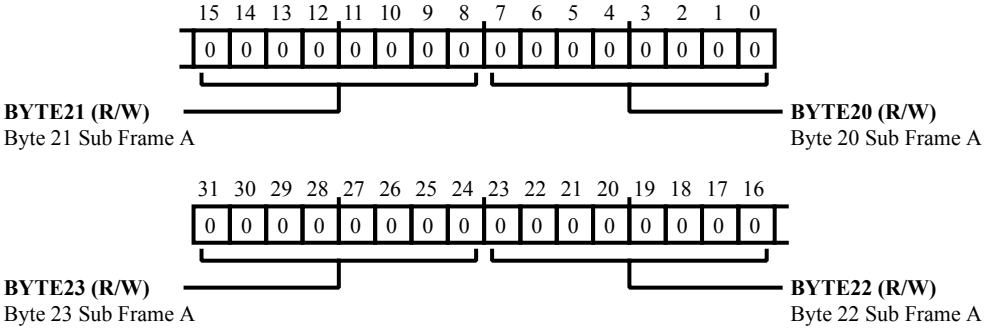


Figure 35-32: SPDIF_TX_UBUFF_A5 Register Diagram

Table 35-28: SPDIF_TX_UBUFF_A5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE23	Byte 23 Sub Frame A. The SPDIF_TX_UBUFF_A5.BYTE23 bit field contains user bit data for byte 23 of sub frame A.
23:16 (R/W)	BYTE22	Byte 22 Sub Frame A. The SPDIF_TX_UBUFF_A5.BYTE22 bit field contains user bit data for byte 22 of sub frame A.
15:8 (R/W)	BYTE21	Byte 21 Sub Frame A. The SPDIF_TX_UBUFF_A5.BYTE21 bit field contains user bit data for byte 21 of sub frame A.
7:0 (R/W)	BYTE20	Byte 20 Sub Frame A. The SPDIF_TX_UBUFF_A5.BYTE20 bit field contains user bit data for byte 20 of sub frame A.

Transmit User Buffer B0 Register

The SPDIF_TX_UBUFF_B0 register holds the transmit user buffer data for bytes 0-3 for sub frame B.

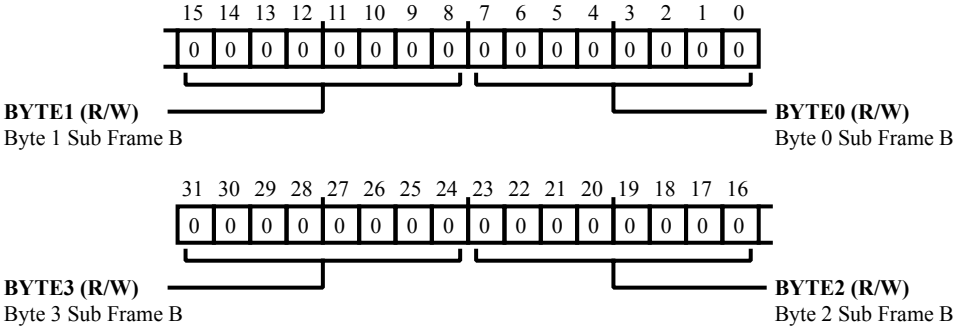


Figure 35-33: SPDIF_TX_UBUFF_B0 Register Diagram

Table 35-29: SPDIF_TX_UBUFF_B0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE3	Byte 3 Sub Frame B. The SPDIF_TX_UBUFF_B0.BYTE3 bit field contains user bit data for byte 3 of sub frame B.
23:16 (R/W)	BYTE2	Byte 2 Sub Frame B. The SPDIF_TX_UBUFF_B0.BYTE2 bit field contains user bit data for byte 2 of sub frame B.
15:8 (R/W)	BYTE1	Byte 1 Sub Frame B. The SPDIF_TX_UBUFF_B0.BYTE1 bit field contains user bit data for byte 1 of sub frame B.
7:0 (R/W)	BYTE0	Byte 0 Sub Frame B. The SPDIF_TX_UBUFF_B0.BYTE0 bit field contains user bit data for byte 0 of sub frame B.

Transmit User Buffer B1 Register

The SPDIF_TX_UBUFF_B1 register holds the transmit user buffer data for bytes 4-7 for sub frame B.

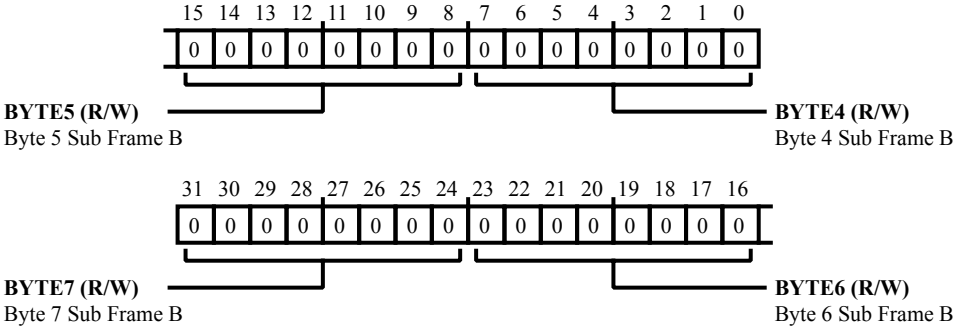


Figure 35-34: SPDIF_TX_UBUFF_B1 Register Diagram

Table 35-30: SPDIF_TX_UBUFF_B1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE7	Byte 7 Sub Frame B. The SPDIF_TX_UBUFF_B1.BYTE7 bit field contains user bit data for byte 7 of sub frame B.
23:16 (R/W)	BYTE6	Byte 6 Sub Frame B. The SPDIF_TX_UBUFF_B1.BYTE6 bit field contains user bit data for byte 6 of sub frame B.
15:8 (R/W)	BYTE5	Byte 5 Sub Frame B. The SPDIF_TX_UBUFF_B1.BYTE5 bit field contains user bit data for byte 5 of sub frame B.
7:0 (R/W)	BYTE4	Byte 4 Sub Frame B. The SPDIF_TX_UBUFF_B1.BYTE4 bit field contains user bit data for byte 4 of sub frame B.

Transmit User Buffer B2 Register

The SPDIF_TX_UBUFF_B2 register holds the transmit user buffer data for bytes 8-11 for sub frame B.

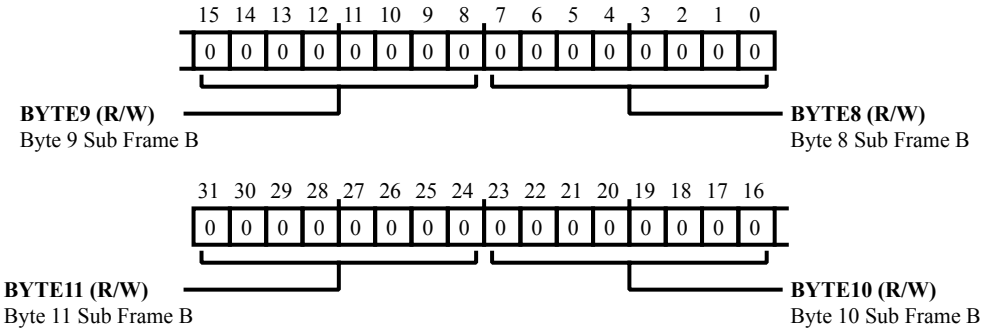


Figure 35-35: SPDIF_TX_UBUFF_B2 Register Diagram

Table 35-31: SPDIF_TX_UBUFF_B2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE11	Byte 11 Sub Frame B. The SPDIF_TX_UBUFF_B2.BYTE11 bit field contains user bit data for byte 11 of sub frame B.
23:16 (R/W)	BYTE10	Byte 10 Sub Frame B. The SPDIF_TX_UBUFF_B2.BYTE10 bit field contains user bit data for byte 10 of sub frame B.
15:8 (R/W)	BYTE9	Byte 9 Sub Frame B. The SPDIF_TX_UBUFF_B2.BYTE9 bit field contains user bit data for byte 9 of sub frame B.
7:0 (R/W)	BYTE8	Byte 8 Sub Frame B. The SPDIF_TX_UBUFF_B2.BYTE8 bit field contains user bit data for byte 8 of sub frame B.

Transmit User Buffer B3 Register

The SPDIF_TX_UBUFF_B3 register holds the transmit user buffer data for bytes 12-15 for sub frame B.

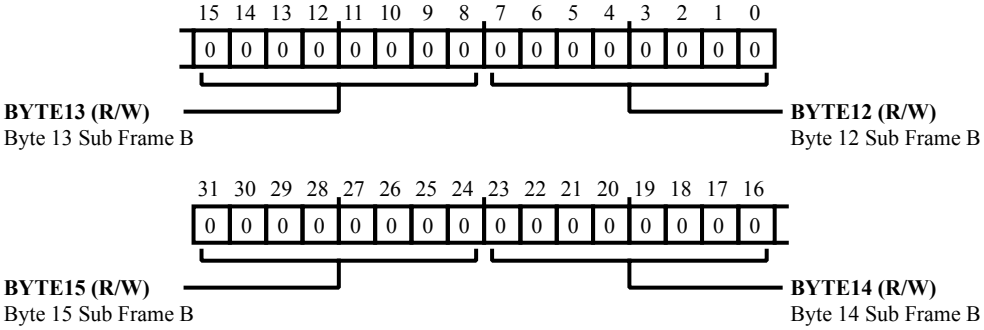


Figure 35-36: SPDIF_TX_UBUFF_B3 Register Diagram

Table 35-32: SPDIF_TX_UBUFF_B3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE15	Byte 15 Sub Frame B. The SPDIF_TX_UBUFF_B3.BYTE15 bit field contains user bit data for byte 15 of sub frame B.
23:16 (R/W)	BYTE14	Byte 14 Sub Frame B. The SPDIF_TX_UBUFF_B3.BYTE14 bit field contains user bit data for byte 14 of sub frame B.
15:8 (R/W)	BYTE13	Byte 13 Sub Frame B. The SPDIF_TX_UBUFF_B3.BYTE13 bit field contains user bit data for byte 13 of sub frame B.
7:0 (R/W)	BYTE12	Byte 12 Sub Frame B. The SPDIF_TX_UBUFF_B3.BYTE12 bit field contains user bit data for byte 12 of sub frame B.

Transmit User Buffer B4 Register

The SPDIF_TX_UBUFF_B4 register holds the transmit user buffer data for bytes 16-19 for sub frame B.

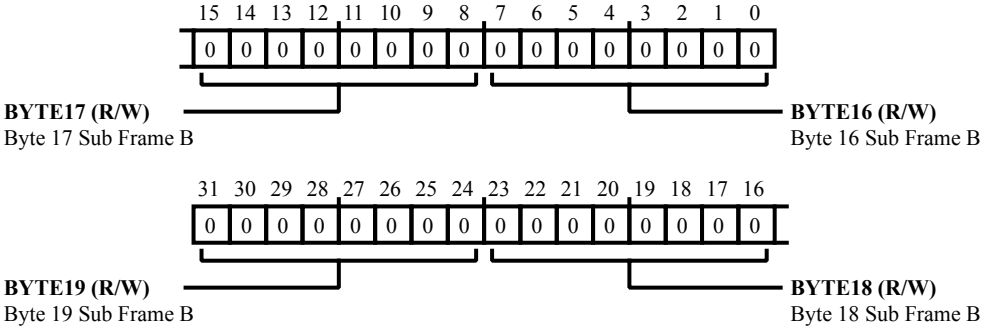


Figure 35-37: SPDIF_TX_UBUFF_B4 Register Diagram

Table 35-33: SPDIF_TX_UBUFF_B4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE19	Byte 19 Sub Frame B. The SPDIF_TX_UBUFF_B4.BYTE19 bit field contains user bit data for byte 19 of sub frame B.
23:16 (R/W)	BYTE18	Byte 18 Sub Frame B. The SPDIF_TX_UBUFF_B4.BYTE18 bit field contains user bit data for byte 18 of sub frame B.
15:8 (R/W)	BYTE17	Byte 17 Sub Frame B. The SPDIF_TX_UBUFF_B4.BYTE17 bit field contains user bit data for byte 17 of sub frame B.
7:0 (R/W)	BYTE16	Byte 16 Sub Frame B. The SPDIF_TX_UBUFF_B4.BYTE16 bit field contains user bit data for byte 16 of sub frame B.

Transmit User Buffer B5 Register

The SPDIF_TX_UBUFF_B5 register holds the transmit user buffer data for bytes 20-23 for sub frame B.

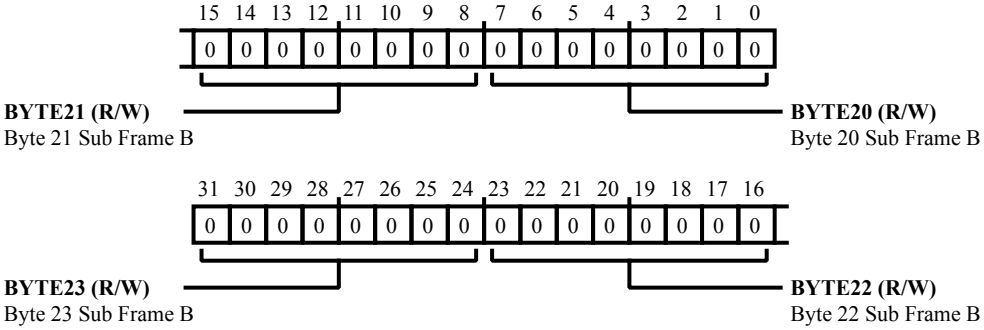


Figure 35-38: SPDIF_TX_UBUFF_B5 Register Diagram

Table 35-34: SPDIF_TX_UBUFF_B5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYTE23	Byte 23 Sub Frame B. The SPDIF_TX_UBUFF_B5.BYTE23 bit field contains user bit data for byte 23 of sub frame B.
23:16 (R/W)	BYTE22	Byte 22 Sub Frame B. The SPDIF_TX_UBUFF_B5.BYTE22 bit field contains user bit data for byte 22 of sub frame B.
15:8 (R/W)	BYTE21	Byte 21 Sub Frame B. The SPDIF_TX_UBUFF_B5.BYTE21 bit field contains user bit data for byte 21 of sub frame B.
7:0 (R/W)	BYTE20	Byte 20 Sub Frame B. The SPDIF_TX_UBUFF_B5.BYTE20 bit field contains user bit data for byte 20 of sub frame B.

User Bit Update Register

After writing to the transmit user buffer registers, a value of 0x1 must be written into the SPDIF_TX_USRUPDT register to enable the use of these user buffer bits in the next transfer block.

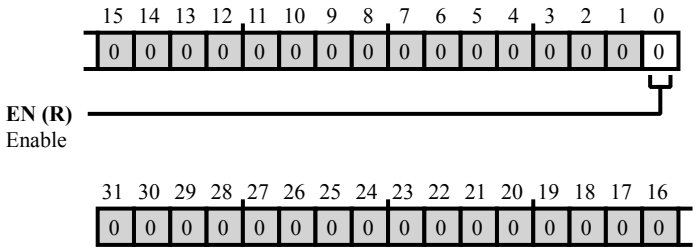


Figure 35-39: SPDIF_TX_USRUPDT Register Diagram

Table 35-35: SPDIF_TX_USRUPDT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/NW)	EN	Enable. After writing to the transmit user buffer registers, a value of 0x1 must be written into the SPDIF_TX_USRUPDT register to enable the use of these SPDIF_TX_USRUPDT.EN bits in the next transfer block.

36 Direct Memory Access (DMA)

The processor architecture distributes the DMA channels throughout the infrastructure. Often, the channels cluster together through system crossbars (SCB), sharing a single interface with the main system crossbar.

The DMA channels can perform transfers between memory and a peripheral or between one memory and another memory. Memory-to-memory DMA transfers (MDMA) require two DMA channels. One channel is the source channel, and the second, the destination channel.

All DMA channels can transport data to and from virtually all on-chip and off-chip memories.

DMA transfers on the processor use either a descriptor-based method or register-based method. Register-based DMA allows the processor directly to program DMA controller registers to initiate a DMA transfer. On completion, the controller registers can automatically update with their original setup values for continuous transfer, if needed. Descriptor-based DMA transfers require a set of parameters stored within memory to initiate a DMA sequence. Descriptor-based transfers allow the chaining together of multiple DMA sequences. In descriptor-based DMA operations, DMA channel programming can automatically set up and start another DMA transfer after the current sequence completes.

The DMA channel does not connect external memories and devices directly. Rather, data passes through an external-memory interface port. DMA operations can access any device the external memory interface supports. These interfaces typically include:

- Flash memory
- SRAM
- FIFOs
- Memory-mapped peripheral devices
- Dynamic Memory (if present)

DMA Channel Features

The processor uses Direct Memory Access (DMA) to transfer data within memory spaces or between a memory space and a peripheral. The processor can specify data transfer operations and return to normal processing while the fully integrated DMA channel carries out the data transfers independent of processor activity. The DMA channels are dispersed throughout the infrastructure and interface with the system crossbar unit (SCB).

The following is a list of DMA interface features.

- Supports integer byte strides including byte strides of 0 and negative byte strides
- Register-based configuration
  - Core writes DMA configuration
  - Supports automatic reloading for continuous operation
- Flexible descriptor-based configuration
  - DMA descriptors are fetched from memory
  - Support for variable descriptor sizes
- Flexible flow control – Transitions between the various descriptor-based modes and for DMA termination
- Orthogonal transfers
  - Support for three transfer dimensions
  - One and two dimensional (1D and 2D) transfers supported per descriptor set
  - Three dimensional (3D) support provided by chained descriptor sets
- Configurable memory and peripheral-transfer word sizes
  - Memory interface supports 8-bit, 16-bit, 32-bit, 64-bit, 128-bit, and 256-bit transfers
  - Peripheral interface supports for 8-bit, 16-bit, and 32-bit transfers
- Interrupt notification
  - Row or work unit completion
  - Error conditions
- Incoming and outgoing trigger support
  - Trigger generation for row or work unit completion
  - Work unit can wait for incoming trigger
- MMR access bus – Provides access to memory-mapped registers for configuration, monitoring, and debug
- SCB crossbar interface connects the DMA channel to the system crossbar
- Peripheral DMA bus – Interfaces the DMA channel to a peripheral or another DMA channel
- Peripheral data-request interrupt support
- Bandwidth monitoring and limiting for MDMA channels

DMA Channel Functional Description

This section provides a functional description of the DMA channel interface.

ADSP-SC59x DMA Register List

The DMA channel controller (DMA) supports data transfers within memory spaces or between a memory space and a peripheral. The processor can specify data transfer operations and return to normal processing while the fully integrated DMA channel carries out the data transfers independent of processor activity. The DMA channels are dispersed throughout the infrastructure, as DMAn's. A set of registers governs DMA operations. For more information on DMA functionality, see the DMA register descriptions.

Table 36-1: ADSP-SC59x DMA Register List

Name	Description
DMA_ADDRSTART	Start Address of Current Buffer Register
DMA_ADDR_CUR	Current Address Register
DMA_BWLCNT	Bandwidth Limit Count Register
DMA_BWLCNT_CUR	Bandwidth Limit Count Current Register
DMA_BWMCNT	Bandwidth Monitor Count Register
DMA_BWMCNT_CUR	Bandwidth Monitor Count Current Register
DMA_CFG	Configuration Register
DMA_DSCPTR_CUR	Current Descriptor Pointer Register
DMA_DSCPTR_NXT	Pointer to Next Initial Descriptor Register
DMA_DSCPTR_PRV	Previous Initial Descriptor Pointer Register
DMA_STAT	Status Register
DMA_XCNT	Inner Loop Count Start Value Register
DMA_XCNT_CUR	Current Count (1D) or Intra-row XCNT (2D) Register
DMA_XMOD	Inner Loop Address Increment Register
DMA_YCNT	Outer Loop Count Start Value (2D only) Register
DMA_YCNT_CUR	Current Row Count (2D only) Register
DMA_YMOD	Outer Loop Address Increment (2D only) Register

ADSP-SC59x DMA Channel List

Table 36-2: ADSP-SC59x DMA Channel List

DMA ID	DMA Channel Name	Description
DMA0	SPORT0_A_DMA	SPORT0 Channel A DMA
DMA1	SPORT0_B_DMA	SPORT0 Channel B DMA

Table 36-2: ADSP-SC59x DMA Channel List (Continued)

DMA ID	DMA Channel Name	Description
DMA2	SPORT1_A_DMA	SPORT1 Channel A DMA
DMA3	SPORT1_B_DMA	SPORT1 Channel B DMA
DMA4	SPORT2_A_DMA	SPORT2 Channel A DMA
DMA5	SPORT2_B_DMA	SPORT2 Channel B DMA
DMA6	SPORT3_A_DMA	SPORT3 Channel A DMA
DMA7	SPORT3_B_DMA	SPORT3 Channel B DMA
DMA8	MDMA0_SRC	Memory DMA Stream 0 Source Channel
DMA9	MDMA0_DST	Memory DMA Stream 0 Destination Channel
DMA10	SPORT4_A_DMA	SPORT4 Channel A DMA
DMA11	SPORT4_B_DMA	SPORT4 Channel B DMA
DMA12	SPORT5_A_DMA	SPORT5 Channel A DMA
DMA13	SPORT5_B_DMA	SPORT5 Channel B DMA
DMA14	SPORT6_A_DMA	SPORT6 Channel A DMA
DMA15	SPORT6_B_DMA	SPORT6 Channel B DMA
DMA16	SPORT7_A_DMA	SPORT7 Channel A DMA
DMA17	SPORT7_B_DMA	SPORT7 Channel B DMA
DMA18	MDMA1_SRC	Memory DMA Stream 1 Source Channel
DMA19	MDMA1_DST	Memory DMA Stream 1 Destination Channel
DMA20	UART0_TXDMA	UART0 Transmit DMA
DMA21	UART0_RXDMA	UART0 Receive DMA
DMA22	SPI0_TXDMA	SPI0 TX DMA Channel
DMA23	SPI0_RXDMA	SPI0 RX DMA Channel
DMA24	SPI1_TXDMA	SPI1 TX DMA Channel
DMA25	SPI1_RXDMA	SPI1 RX DMA Channel
DMA26	SPI2_TXDMA	SPI2 TX DMA Channel
DMA27	SPI2_RXDMA	SPI2 RX DMA Channel
DMA28	EPPI0_CH0_DMA	EPPI0 Channel 0 DMA
DMA29	EPPI0_CH1_DMA	EPPI0 Channel 1 DMA
DMA30	LP0_DMA	LP0 DMA Channel

Table 36-2: ADSP-SC59x DMA Channel List (Continued)

DMA ID	DMA Channel Name	Description
DMA34	UART1_TXDMA	UART1 Transmit DMA
DMA35	UART1_RXDMA	UART1 Receive DMA
DMA36	LP1_DMA	LP1 DMA Channel
DMA37	UART2_TXDMA	UART2 Transmit DMA
DMA38	UART2_RXDMA	UART2 Receive DMA
DMA39	MDMA2_SRC	Memory DMA Stream 2 Source Channel
DMA40	MDMA2_DST	Memory DMA Stream 2 Destination Channel
DMA43	MDMA3_SRC	Memory DMA Stream 3 Source Channel
DMA44	MDMA3_DST	Memory DMA Stream 3 Destination Channel
DMA45	MDMA4_SRC	Memory DMA Stream 4 Source Channel
DMA46	MDMA4_DST	Memory DMA Stream 4 Destination Channel
DMA47	MDMA5_SRC	Memory DMA Stream 5 Source Channel
DMA48	MDMA5_DST	Memory DMA Stream 5 Destination Channel
DMA49	MDMA6_SRC	Memory DMA Stream 6 Source Channel
DMA50	MDMA6_DST	Memory DMA Stream 6 Destination Channel
DMA51	MDMA7_SRC	Memory DMA Stream 7 Source Channel
DMA52	MDMA7_DST	Memory DMA Stream 7 Destination Channel
DMA53	UART3_TXDMA	UART3 Transmit DMA
DMA54	UART3_RXDMA	UART3 Receive DMA
DMA55	SPI3_TXDMA	SPI3 TX DMA Channel
DMA56	SPI3_RXDMA	SPI3 RX DMA Channel

DMA Definitions

To make the best use of the DMA controller, it is useful to understand the following terms.

Descriptor

A single element of a descriptor set that maps to a specific register of a DMA channel.

Descriptor Fetch

The action of retrieving descriptors from memory through memory read operations and loading then into the DMA channel registers upon their read return.

Descriptor Set

An array of descriptors associated with a single work unit. The user can configure the size of the descriptor set. See Descriptor-Based Flow Modes .

Disabled State

The channel is disabled because the enable bit = 0 or because of an error.

DMAC

An acronym used for a DMA cluster.

DMA Channel

A single DMA engine that has all the capabilities and registers as defined for a given processor. A DMA channel or engine is connected to a single peripheral.

DMA Cluster

A grouping of multiple DMA channels with a shared SCB crossbar interface, controller, and arbiter. Also known as a DMAC.

Initial Descriptor

The first descriptor in the descriptor set.

MDMA

Memory-to-Memory DMA data transfer. Two DMA channels are paired to perform a memory read from one address location and a memory write of that data to another address location.

Stop State

A time where the channel is enabled but not currently programmed to perform a data transfer. Programming the flow to STOP causes the channel to enter the stop state at the end of the work unit.

User

Any person, debug, emulator, software routine, or action taken by the core that accesses the MMR registers of the DMA channel or peripherals, or sets up data and descriptors in memory.

Wait State

If instructed to wait for a trigger, the channel enters this state once it has completed a work unit. The channel remains in this state until a trigger occurs. If a trigger came in before reaching the wait state, the channel skips over the wait state upon completion of the work unit.

Work Unit

A single data transaction or series of data transactions performed based on the configuration of the DMA channel. For autobuffer mode, a new work unit is defined at the time all current count registers are initialized to start values. Once all the current count registers count down to zero, the work unit has completed.

Work Unit Chain

A single work unit or a series of work units separated by a `STOP` or disabled state. The work units in the chain (except the last one) are programmed to the required descriptor flow. The last work unit in the chain is programmed to a flow of `STOP` or `AUTO`. `STOP` terminates the state at the end of that work unit. `AUTO` must be terminated by disabling the DMA channel. A work unit chain is also known as a descriptor chain.

Block Diagram

The ***DMA Channel Block Diagram*** shows the functional blocks within the DMA interface.

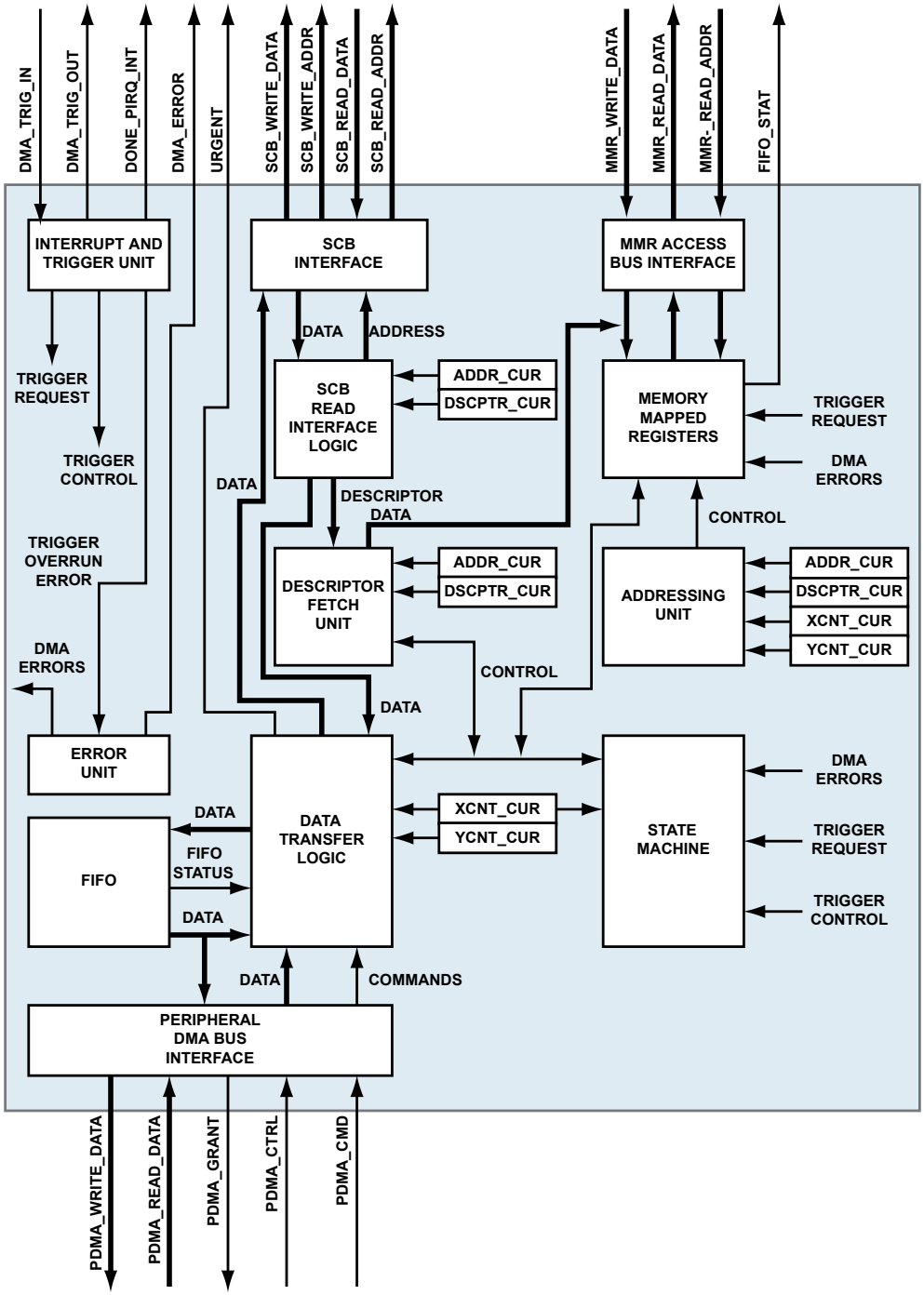


Figure 36-1: DMA Channel Block Diagram

For more information on the interfaces, see:

- DMA Channel Peripheral DMA Bus
- Medium Band Width DMA Channel MMR Access Bus
- DMA Channel Event Control

- DMA Channel SCB Interface

Architectural Concepts

The DMA channel provides a method to transfer data between memory spaces or between memory and a peripheral using a number of system interfaces. The DMA channel provides an efficient method of distributing data throughout the system, freeing up the processor core for other operations. Each peripheral that supports DMA transfers has its own dedicated DMA channel or channels with its own register set. The register set configures and controls the operating modes of the DMA transfers.

DMA Channel SCB Interface

The SCB interface connects the DMA channel to the SCB crossbar allowing for transfers to and from the processors internal memory and other suitable system resources.

The DMA channel connects to the system interconnect through the SCB interface. This connection lets the DMA channel perform work-unit data transfers with memories such as L1, L2 (internal), and L3 (external). In addition to work unit data transfers, the SCB interface also is used for fetching descriptor sets for all the descriptor-based transfer modes.

The DMA channel can support data bus widths of 16, 32, 64, or 128 bits. The data bus widths for a given DMA channel on a specific processor can vary and are not configurable. Read the `DMA_STAT.MBWID` field to determine the assigned bus widths.

SCB Interface Signals

The DMA channel operates at one of the *SCLKn* frequencies, as does the SCB interface. SYSCLK clocks all eight DMA channels. The SCB crossbar handles the internal arbitration of the transfer requests of all the controllers interfaced to the SCB crossbar instance as shown in the ***SCB Interface Signals*** table.

Table 36-3: SCB Interface Signals

Signal	Width (bits)	Description
SCB_WRITE_DATA	16, 32, 64, or 128	Data bus used for write operations. The width of the bus can be determined from `DMA_STAT.MBWID`.
SCB_WRITE_ADDRESS	32	Write address bus. Provides the address of the first transfer in a burst transaction.
SCB_READ_DATA	16, 32, 64, or 128	Data bus used for read operations. The width of the bus can be determined from `DMA_STAT.MBWID`.
SCB_READ_ADDRESS	32	Read address bus. Provides the address of the first transfer in a burst transaction.

SCB Burst Transfers

The SCB interface supports burst transfers for memory read and write operations. The burst length is a function of the configurable memory size of the DMA channel for the work unit and the fixed bus width of the SCB data bus of the DMA channel.

- If the DMA channel configuration selects a memory transfer size less than or equal to the DMA channels bus width, the burst length is always 1.
- If the configured memory size is greater than the SCB interface bus width, the burst length is sufficient to transfer a transaction as specified by the configured memory size.

Table 36-4: DMA Channel SCB Burst Lengths

Configured Memory Size	Burst Length			
	16-bit Bus	*32-bit Bus*	*64-Bit Bus*	*128-bit Bus*
1 Byte	1	1	1	1
2 Bytes	1	1	1	1
4 Bytes	2	1	1	1
8 Bytes	4	2	1	1
16 Bytes	8	4	2	1
32 Bytes	16	8	4	2

Data Address Alignment

To prevent addressing errors and to maximize bandwidth of the SCB interface to the DMA channel, data addresses align with a multiple of the programmable memory size of the DMA channels configuration. These configuration options appear in the Descriptor Set Address Alignment table.

There are situations in which entire work units may not transfer at the maximum configurable memory size. In this case, the entire work unit can transfer by reducing the configured memory size at the expense of bus bandwidth using descriptor sets as follows:

- The first descriptor set can be configured to transfer data until the larger memory size alignments are met.
- A second descriptor set with a larger memory size configuration then can be used to transfer the bulk of the data in the work unit.
- Finally, a third descriptor set can be used with a smaller memory size to complete any final data transfers that cannot meet the alignment requirements of the previous descriptor set configuration.

Table 36-5: DMA Channel Address Alignment Requirements

Configured Memory Size	Address Restriction
1 Byte	No restriction
2 Bytes	ADDR[0] == 0
4 Bytes	ADDR[1:0] == 0
8 Bytes	ADDR[2:0] == 0
16 Bytes	ADDR[3:0] == 0
32 Bytes	ADDR[4:0] == 0

Descriptor Set Address Alignment

All descriptor set addresses and descriptors within a descriptor set must align to a 32-bit address. For descriptor set fetches, the DMA engine ignores the memory-size configuration of the DMA channel. This feature avoids the need to align descriptor sets based on the memory width configuration of the previous descriptor set.

For descriptor sets containing only a single descriptor, the transfer takes place as a single 32-bit transfer. For descriptor sets containing multiple descriptors, the DMA engine fetches each 32-bit descriptor individually and treats it as multiple 32-bit transfers.

Peripheral Control Commands

The peripheral DMA bus of the DMA channel provides a means for peripherals on the processor to issue commands to the DMA channel. These commands provide greater control over the DMA channel operation. This control improves real-time performance and relieves control and interrupt demands on the core. Peripherals can send commands to the DMA controller over the 3-bit PERI_CMD bus. The DMA control commands extend the set of operations available to the peripheral beyond the simple “request data” command used by peripherals in general. Refer to the appropriate peripheral chapter for a description on how that peripheral uses DMA control commands.

These DMA control commands (see the ***PDMA_CMD Peripheral DMA Control Commands*** table) are not visible to or controlled by the program. But, their use by a peripheral has implications for the structure of the DMA transfers that the peripheral can support. It is important to write application software such that it complies with certain restrictions, regarding work units and descriptor chains. Complying with this guideline makes the peripheral operate properly whenever it issues DMA control commands.

The ***PDMA_CMD Peripheral DMA Control Commands*** table describes the commands the DMA controller issues. The following sections describe these commands in more detail.

Table 36-6: PDMA_CMD Peripheral DMA Control Commands

Command	Name	Description
b#000	NOP	No operation
b#001	Restart	Restarts the current work unit from the beginning
b#010	Finish	Finishes the current work unit and starts the next
b#011	Interrupt	Immediately sets the DMA completion interrupt in the DMA channel
b#100	Request Data	Typical DMA data request
b#101	Request Data Urgent	Urgent DMA data request
b#110	Reserved	Reserved
b#111	Reserved	Reserved

Idle Command

The DMA channel drives this command when the enabled peripheral has no data requests required.

Request-Data Command

The request data command is a request for data transfers between the DMA channel and the peripheral. The request is held by the peripheral until granted or acknowledged by the DMA channel.

Request-Data Urgent Command

The request-data urgent command behaves identically to the request data command, except that---during the commands assertion---the DMA channel performs its memory accesses with urgent priority. This priority includes both data and descriptor fetch memory accesses. For example, a DMA management capable peripheral can use this control command if an internal FIFO approaches a critical condition.

The request is held by the peripheral until granted or acknowledged by the DMA channel.

Peripheral-Control Command Restrictions

The proper operation of the DMA channel FIFO leads to certain restrictions in the sequence of DMA peripheral control commands issued by a peripheral. The following sections describe these restrictions.

DMA Channel Peripheral DMA Bus

The DMA channel connects to peripherals or other DMA channels through the peripheral DMA bus. This bus is a dedicated point-to-point interface supporting data bus widths of 8, 16, 32, or 64 bits. The data bus widths for a given DMA channel on a particular processor can vary and are not configurable. Reading the `DMA_STAT.PBWID` field permits determining the assigned bus width.

The DMA channel operates at one of the SCLK frequencies, as does the peripheral DMA bus. The ***Peripheral DMA Bus Signals*** table provides descriptions of the peripheral DMA bus signals.

Table 36-7: Peripheral DMA Bus Signals

Signal	Width (bits)	Description
PDMA_WRITE_DATA	8, 16, 32, or 64	Data bus used for write operations. The width of the bus can be determined from `DMA_STAT.PBWID`.
PDMA_READ_DATA	8, 16, 32, or 64	Data bus used for read operations. The width of the bus can be determined from `DMA_STAT.PBWID`.
PDMA_DMA_GRANT		Control signals to indicate that data is valid for DMA channel read operations (peripheral transmit). These signals indicate that the DMA channel is ready to receive data for write operations (peripheral receive).
PDMA_CMD	3	The peripheral uses the signal for issuing DMA channel control commands.
PDMA_CTRL		The peripheral uses the control signals to send various commands to the DMA channel and control the direction of flow.

Memory DMA and Triggering

A memory DMA (MDMA) channel provides a means of doing memory-to-memory DMA transfers among the various memory spaces that have DMA support.

The DMA controller implements memory DMA (MDMA) channels by interfacing two DMA channels through the peripheral DMA bus interface. One DMA channel serves for memory read operations, and the second channel servers for memory writes. Depending on the processor, a memory DMA channel can have an additional peripheral, such as a CRC peripheral. The additional peripheral is inserted into the peripheral DMA bus that optionally can be enabled.

MDMA channel configurations that do not involve an additional peripheral impose no restrictions on which of the DMA channels is used for the read operation or the write operation. But, the configuration of both channels cannot have the same transfer direction. For MDMA channel configurations that enable a peripheral between the read and write channels, be aware of possible restrictions imposed on which channel can be used for a given transfer direction.

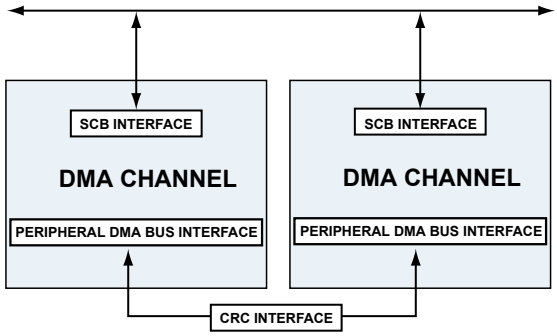


Figure 36-2: MDMA Channel Dedicated Pair

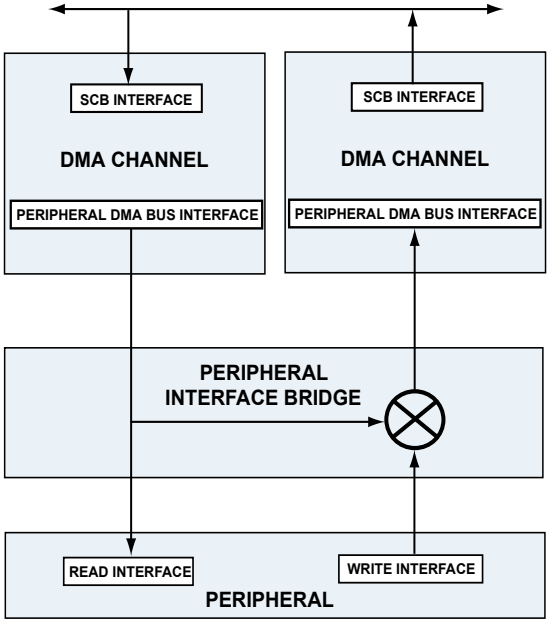


Figure 36-3: MDMA Channel Pair with Peripheral

A memory-to-memory transfer always requires enabled source and destination channels. Because the channels interface through the peripheral DMA bus and can have an additional peripheral inserted into the peripheral DMA bus, programs must make sure to set the same values in the `DMA_CFG.PSIZE` of both the source and destination channels.

The memory DMA channels support the full range of the DMA_CFG.MSIZE options for the DMA transfers to and from the memories.

Because the MDMA channel consists of two DMA channels, the entire MDMA channel has two sets of FIFOs, one in the read channel and one in the write channel. This FIFO usage allows for more efficient bursting of both read and write transactions using the available bandwidth. While the DMA_CFG.PSIZE configuration must be identical for both source and destination DMA channels, this restriction does not apply for the DMA_CFG.MSIZE configuration.

Configure the DMA_CFG.PSIZE bits to a value no larger than the supported bus width of the peripheral DMA bus. From a performance perspective, use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the PSIZE value used.

NOTE: This is applicable for all of the DMA channels, except the enhanced and high-speed MDMA channels. For the enhanced and high-speed MDMAs, the minimum *PSIZE* and *MSIZE* is 4 bytes (32 bits).

The independent source and destination DMA channels also have their own dedicated interrupt and trigger events. While it is normal practice to have only event generation performed at destination DMA completion, programs also can use other means of interrupt generation.

Configuration of an MDMA transfer is done in a similar manner to peripheral DMA transfers, except for writing two DMA channel registers instead of one.

To control the pace of data transfers, use triggers on either the memory read or the memory write channel pair used in an MDMA operation. Setting the DMA_CFG.TWAIT bit in the memory read channel prevents both channels from transferring data before the system is ready. However, only configuring the memory write channel to wait for a trigger allows for data fetch from the memory in anticipation of the memory write operation.

The *MDMA Streams* table shows the details for each supported MDMA stream.

Table 36-8: MDMA Streams

MDMA Stream	Source Channel	Destination Channel	CRC Support	Other Names	Operating Clock	Bus Width (bits)	Maximum Theoretical BW (Mbytes/ second)
0	8	9	Yes	CRC0	SYSCLK (500 MHz max)	32	2000
1	18	19	Yes	CRC1			
2	39	40	No	MSMDMA			
3	43	44	No	HSMDMA		64	4000
4	45	46	Yes	CRC2		32	2000
5	47	48	Yes	CRC3			
6	49	50	No	MSMDMA1			

Table 36-8: MDMA Streams (Continued)

MDMA Stream	Source Channel	Destination Channel	CRC Support	Other Names	Operating Clock	Bus Width (bits)	Maximum Theoretical BW (Mbytes/ second)
7	51	52	No	HSMDMA1		64	4000

Medium Band Width DMA Channel MMR Access Bus

The MMR access bus provides access to all the DMA channels memory-mapped registers for DMA channel configuration, monitoring, and debug. The interface has a fixed 32-bit data bus for read and write accesses.

The ***MMR Access Bus Signals*** table provides descriptions of the MMR access bus signals.

Table 36-9: MMR Access Bus Signals

Signal	Width (bits)	Description
MMR_WRITE_DATA	32	Data bus used for write operations to the MMRs from the core
MMR_READ_DATA	32	Data bus used to return read data from the MMRs
MMR_READ_ADDR	7	Address used to select the MMR to access

DMA Channel Operation Flow

A detailed description of the flow of operation of the DMA channel appears in the following topics:

- Startup Flow
- Refresh Flow
- DMA Operating Modes
- Stop Mode
- DMA Channel Errors

Startup Flow

Enabling a DMA operation on a given channel first requires directly writing some or all of the DMA parameter registers. The minimum set of register required to be initialized depends on the desired mode of operation as described in the following sections.

Startup Minimum-Enable Requirements

To start a DMA operation on a given channel, some or all of the DMA parameter registers must first be initialized and configured to the desired DMA channels operating mode.

- For descriptor-array-based flow modes, at minimum, write the `DMA_DSCPTR_CUR` register prior to writing to the `DMA_CFG` register, which is the special action required to start the DMA channel.

- For descriptor-list-based flow modes, at minimum, write the DMA_DSCPTR_NXT register prior to writing to the DMA_CFG register, which is the special action required to start the DMA channel.
- For non-descriptor-based flow modes, write the DMA_ADDRSTART, DMA_XCNT, and DMA_XMOD registers prior to writing the DMA_CFG register.

Programs can write other registers that can remain static throughout the course of the DMA activity. The write to the DMA_CFG register begins the DMA operation.

ATTENTION: When software directly writes the DMA_CFG register, the DMA controller recognizes this action as the special startup condition. This condition occurs when starting the DMA controller for the first time on this channel or occurs after the DMA channel stops. It is possible for the channel to flag a DMA error condition regardless of the DMA_CFG.EN bit setting.

Startup Operation

The startup operation is initiated by software directly writing the DMA_CFG register when starting DMA for the first time on a channel or after the channel has entered to the stop state.

When the descriptor fetch is complete and the DMA channel is enabled, the DMA_CFG descriptor element replaces the DMA_CFG register content and assumes control. Before this point, the direct write to the DMA_CFG register had control.

At startup, the selected flow mode and the descriptor size determine the course of the DMA initialization process. The DMA_CFG.FLOW field determines whether to load more current registers from descriptor sets in memory. The DMA_CFG.NDSIZE field details how many descriptor elements to fetch before starting the DMA operation. This process does not affect DMA registers that are not in the descriptor; no modifications are made to their prior values.

For descriptor-list flow modes, the channel copies the DMA_DSCPTR_NXT register value into the DMA_DSCPTR_CUR register. Then, the channel fetches new descriptor elements from memory. The DMA_DSCPTR_CUR register indexes each fetch, and the channel increments the index after each fetch. After completion of the descriptor fetch, the DMA_DSCPTR_CUR register points to the next 32-bit word in memory past the end of the descriptor.

If the descriptor fetch is for a descriptor-array mode transfer, the channel does *not* copy the DMA_DSCPTR_NXT register into the DMA_DSCPTR_CUR register. *Instead*, the descriptor fetch indexing begins with the value in the DMA_DSCPTR_CUR register.

If DMA_CFG is not part of the fetched descriptor set, the previous value (originally as written on startup) controls the work unit operation. If the DMA_CFG register is part of the fetched descriptor set, the value programmed by the MMR access controls only the loading of the first descriptor set fetched from memory. The configuration of the DMA_CFG register controls the subsequent DMA work units of the fetched descriptor set.

After the descriptor fetch is complete or if the flow configuration was originally for one of the register-based flow modes, the DMA operation begins. The DMA channel immediately fills its FIFO. For a memory-write operation, the DMA channel begins accepting data from the peripheral. For a memory-read operation, the DMA channel begins memory reads when the SCB bus grants access to the DMA channel.

When the DMA channel performs its first data-memory access, its address and count computations take their input operands from the start registers. These registers can include DMA_ADDRSTART, DMA_XCNT, and DMA_YCNT, if necessary. The channel writes results back to the current registers. These registers include DMA_ADDR_CUR, DMA_XCNT_CUR, and DMA_YCNT_CUR. Note that the current registers are not valid until the channel performs the first memory access, which can be some time after the write to the DMA_CFG register starts the channel. Once started, the channel automatically loads the current registers from the appropriate descriptor elements, overwriting their previous contents. These automatic-load operations include:

- The channel copies the DMA_ADDRSTART value to DMA_ADDR_CUR.
- The channel copies the DMA_XCNT value to DMA_XCNT_CUR.
- The channel copies the DMA_YCNT to DMA_YCNT_CUR.

Refresh Flow

When the channel completes processing of a work unit, the DMA channel performs the following operations:

- Completes the transfer of all data between memory and the DMA channel.
- Performs a synchronized transition (if the DMA channel configuration is a memory read operation with the DMA_CFG.SYNC bit enabled) *and* transfers all data to the peripheral before continuing.
- Forwards the signals from the DMA channel (if interrupts or triggers are enabled) *and* updates the DMA_STAT register to indicate the interrupt request or trigger events.
- Clears the DMA_STAT.RUN bit field to stop DMA operation (if the flow was set to stop mode) *and* transfers any remaining data in the FIFO of the DMA channel to the peripheral.
- Loads a new descriptor from memory into the DMA registers by way of the contents of the DMA_DSCPTR_CUR register (for descriptor-array mode) *and* increments the DMA_DSCPTR_CUR register

  The channel takes the descriptor size from the DMA_CFG.NDSIZE value before the fetch.
- Copies the DMA_DSCPTR_NXT register into the DMA_DSCPTR_CUR register (for descriptor-list mode), fetches the descriptor from the new contents of the DMA_DSCPTR_CUR register, *and* places these contents into the DMA registers while incrementing the DMA_DSCPTR_CUR register.
- Checks for detection of an incoming trigger event (for descriptor-on-demand array mode):
  - If the channel detects a trigger event, the DMA channel loads a new descriptor from memory into the DMA registers from the contents of the DMA_DSCPTR_CUR register, while incrementing the DMA_DSCPTR_CUR register. The channel takes the descriptor size from the DMA_CFG.NDSIZE value before the fetch.
  - If the channel detects no trigger event, the DMA channel begins the next work unit by reloading the current registers.
- Checks for detection of an incoming trigger event (for descriptor-on-demand list mode):

  - If the channel detects a trigger event, the DMA channel copies the DMA_DSCPTR_NXT register value to the DMA_DSCPTR_CUR register, fetches the descriptor memory from the DMA_DSCPTR_CUR register, *and* places the contents into the DMA registers while incrementing the DMA_DSCPTR_CUR register.
  - If the channel detects no trigger event, the DMA channel begins the next work unit by reloading the current registers as described in the next step.
- Begins the next work unit (if flow configuration is anything other than stop mode) by reloading the current registers (DMA_ADDR_CUR, DMA_XCNT_CUR, and DMA_YCNT_CUR) from their descriptor registers (DMA_ADDRSTART, DMA_XCNT, and DMA_YCNT)

Work Unit Transition Flow

The DMA_CFG.SYNC bit controls transitions from one work unit to the next work unit. In general, continuous transitions have lower latency at the cost of restrictions on changes of data format or addressed memory space in the two work units. These latency gains and data restrictions arise from the way the channel handles the DMA FIFO while fetching the next descriptor.

In continuous transitions, with disabled synchronization, the DMA FIFO pipeline continues to transfer data to and from the peripheral or destination memory. These transfers continue during the descriptor fetch and during the DMA channel pause between descriptor chains. By comparison, synchronized transitions provide better real-time synchronization of interrupts and triggers with a given peripheral state. Synchronized transitions also provide greater flexibility in the data formats and memory spaces of the two work units. This flexibility comes at the cost of higher latency in the transition. In synchronized transitions, the DMA FIFO pipeline drains to the destination or flushes (received data discarded) between work units.

NOTE: The DMA_CFG.SYNC bit of the MDMA source channel controls work unit transitions for MDMA streams. Clear this reserved bit of the MDMA destination channel, placing it in the disabled state. In transmit (memory read) channels, the DMA_CFG.SYNC bit of the last descriptor before the transition controls the transition behavior. In contrast, in receive channels, the DMA_CFG.SYNC bit of the first descriptor of the next descriptor chain controls the transition.

Work Unit Transmit and MDMA Source Transitions

In DMA transmit (memory read) and MDMA source channels, the DMA_CFG.SYNC bit controls the interrupt timing at the end of the work unit. This bit also controls the handling of the DMA FIFO between the current and the next work unit.

If the DMA_CFG.SYNC bit configuration disables synchronization, the DMA channel operates in continuous transition. In a continuous transition, just after reading the last data item from memory, the DMA channel starts all of the following operations parallel:

- Signals the interrupt request or trigger
- Updates the DMA_STAT register to indicate DMA completion status
- Begins fetching the next descriptor
- Delivers the final data items from the DMA FIFO to the destination memory or peripheral

This process lets the DMA channel provide data from the FIFO to the peripheral continuously during the descriptor fetch latency period.

If the configuration disables synchronization, the final interrupt request or trigger (if enabled) occurs when the channel reads the last data from memory. This event occurs at the earliest time that the channel safely can modify the output memory buffer without affecting the previous data transmission. There can be a number of data items remaining in the FIFO and not yet at the peripheral. This number depends on the FIFO depth of the DMA channel. In this configuration, do not use the DMA interrupt request as the sole means of synchronizing the shutdown or reconfiguration of the peripheral following a transmission.

NOTE: If the configuration selects continuous transition on a transmit (memory read) descriptor, the next descriptor must have the same:

- Peripheral transfer size (`DMA_CFG.PSIZE`)
- Read or write direction
- Source memory (internal versus external) as the current descriptor

It is possible to disable synchronization by selecting continuous transition on a work unit with configuration for stop-flow mode and with enabled interrupts or triggers. This approach can result in the execution of the event service routine while draining of the final data is ongoing from the FIFO to the peripheral. If data transfers are in-progress, the FIFO is not yet empty. The `DMA_STAT.RUN` bits of the DMA channels indicate this status. Do not start a new work unit with a different peripheral transfer size or direction while data transfers are in-progress.

CAUTION: Disabling the channel with the `DMA_CFG.EN` bit while data transfers are in-progress causes the loss of the data in the FIFO.

A synchronized transition configuration directs the channel to drain the DMA FIFO to the destination memory or peripheral. This FIFO operation occurs before the channel signals any interrupt and before the channel fetches any subsequent descriptor or data. This operation incurs greater latency, but provides direct synchronization between the DMA interrupt and the state of the data at the peripheral.

If the configuration enables synchronization and enables interrupts, on the last descriptor in a work unit, the interrupt occurs when the channel transfers the final data to the peripheral. This event allows the service routine to switch properly to non-DMA transmit operation. When the event vectors to the interrupt service routine, the DMA channel FIFO is empty, and the DMA channel is no longer running (indicated by the `DMA_STAT.RUN` bits).

A synchronized transition also allows greater flexibility in the format of the DMA descriptor chain. When enabled, the next descriptor can have any `DMA_CFG.PSIZE` configuration or read/write direction supported by the peripheral and can come from either memory space (internal or external). This feature can be useful in managing MDMA work unit queues, since it is no longer necessary to interrupt the queue between dissimilar work units.

Work Unit Receive and MDMA Destination Transitions

In DMA receive channels (memory write operations), the `DMA_CFG.SYNC` bit controls the handling of the DMA FIFO between descriptor chains (not individual descriptor sets), during the DMA channel pause. The DMA channel pauses after the descriptor sets configured with stop flow mode are complete. Restart the channel (for example,

after an interrupt) by writing the DMA_CFG register of the channel with a value that enables the DMA channel. If the configuration disables synchronization in the DMA_CFG value of the new work unit, the configuration selects a continuous transition. In this mode, the DMA FIFO retains any data items received during the channel pause, and they are the first items written to memory in the new work unit. This mode of operation provides lower latency at work unit transitions and ensures no dropping of data items during a DMA pause. The channel provides this operation at the cost of certain restrictions on the DMA descriptors.

NOTE: If the DMA_CFG.SYNC bit disables synchronization on the first descriptor of a chain after a DMA pause, do not change the configuration of the DMA_CFG.PSIZE field of the new chain from the previous descriptor chain (active before the pause). This restriction applies unless the DMA channel is reset between chains by disabling and then re-enabling the DMA channel.

If the DMA_CFG.SYNC bit configuration enables synchronization, the channel uses a synchronized transition. In this mode, only the data that the DMA channel receives from the peripheral after the write to the DMA_CFG register gets to memory. The channel discards any prior data items transferred from the peripheral to the DMA FIFO before this register write occurs. This operation provides direct synchronization between the data stream received from the peripheral and the timing of the channel restart, which occurs on the write to the DMA_CFG register.

For receive DMA operations, the synchronization has no effect in transitions between work units in the same descriptor chain. When the flow mode of previous descriptor was not stopped, the DMA channel did not pause.

If a descriptor chain begins with synchronization enabled, there is no restriction on the DMA_CFG.PSIZE of the new chain in comparison with the previous chain.

NOTE: The peripheral transfer size (DMA_CFG.PSIZE) must not change between one descriptor and the next in any DMA receive (memory write) channel within a single descriptor chain, regardless of the DMA_CFG.SYNC bit setting. In other words, all memory write descriptor sets in a descriptor chain must have the same DMA_CFG.PSIZE value. For any DMA receive channel (memory write operation), there is no restriction on changes of peripheral transfer size (internal versus external) between descriptors or descriptor chains.

Transfer Termination and Shutdown Flow

This section describes channel transfer termination and shutdown in stop flow mode and in autobuffer flow mode.

Stop Flow Mode

In stop flow mode, the DMA channel stops automatically after the work unit is complete. If using a list or array of descriptors to control DMA transfers and if every descriptor contains a DMA_CFG descriptor element, configure the flow of the final DMA_CFG descriptor element to stop mode, stopping the channel gracefully. After completion, the DMA channel remains in the stop state. Do not confuse this state with the disabled state, which either occurs due to a DMA error or occurs through disabling the DMA channel by configuring the DMA_CFG.EN bit.

The intention of disabling the DMA channel through a write to the DMA_CFG.EN bit is to shut down the DMA channel and to enter the disabled state. All memory and peripheral data transfers cease, and only peripheral interrupts pass through the DMA channels interrupt signals. However, the DMA channel maintains the

DMA_STAT.RUN bits. For a write to memory, the outstanding memory-transaction counter tracks returning memory write acknowledgments and updates as required.

For memory reads, the outstanding memory-transaction count also tracks returning memory reads. The channel does not write the memory reads into the FIFO. The channel updates the counter to reflect the completion of the transaction, but the channel ignores the data. The DMA_STAT.RUN bits remain in the *waiting for write ACK or FIFO drain to peripheral* state and do not change to *stop or idle state* until the return of all outstanding transactions.

When the DMA_CFG.EN bit again enables the DMA channel, the channel performs a full reset and clears all counters. If an outstanding memory transaction returns an acknowledgment or read data after this event, a memory transaction error occurred, which generates an error event. Programs must ensure that all outstanding memory transactions complete before reconfiguring the DMA channel. For example, programs can poll the DMA_STAT.RUN bits to return to the stop or idle state before proceeding.

Autobuffer Flow Mode

In this mode, the flow does not use any descriptors in stored memory. Instead, the channel performs DMA in a continuous circular buffer fashion, based on user-programmed DMA register settings. On completion of the work unit, the channel reloads the parameter registers into the current registers, and the DMA controller resumes immediately with zero overhead. Consider this mode as a succession of automatically restarted work units.

For autobuffer-flow modes, the only way to cease operations is to disable the DMA channel through the DMA_CFG.EN bit. One method of changing to a new work unit is:

- Disable the DMA channel
- Set up all the registers (and descriptors in memory, if used) except for DMA_CFG
- Poll DMA_STAT.RUN to wait for the status to reflect stop or idle state, and
- Write DMA_CFG to the new configuration to begin the next work unit

In autobuffer-flow mode or for a list or array of descriptor sets without DMA_CFG descriptors, use an MMR write to the DMA_CFG register to terminate the DMA transfer process. Configure the value of the DMA_CFG.EN bit in this register to disable the DMA channel.

CAUTION: When the configuration disables a DMA channel, the DMA controller disables interrupt logic that is based on work unit transitions. Be aware of the system environment and current actions, so that additional interrupts are not required from the DMA channel.

CAUTION: If disabled through DMA_CFG.EN in the middle of a transaction, the DMA channel completes any transactions that have begun and avoids generating bus errors. However, the channel considers the action of re-enabling the DMA as a hard reset for all internal DMA channel components. Therefore, pay attention to that particular action to avoid unexpected results.

DMA Channel Errors

When an error occurs, the DMA channel maintains all the state and register values that allow programs to diagnose error causes more thoroughly. The greatest benefit to the programmer is to know exactly what operational state the DMA channel was in at the exact moment the error occurred.

Take care to address the root cause of the error, whether or not the problem originated in the DMA channel. If not properly resolved, the error can result in an additional error shortly after operations resume. The problem can cause other errors elsewhere in the DMA channel or associated modules and circuitry. So, take care also to address those potential problems. Ensure that all outstanding memory reads and writes are complete or cleared before resuming DMA channel operation.

After addressing all issues and neutralizing all side effects of any errors, clear the DMA_STAT.ERRC status field and restart the DMA channel by disabling then re-enabling the DMA channel through the DMA_CFG.EN bit.

The following sections describe the error types.

Status and Debug Errors

DMA channel error conditions can cause the DMA process to end abnormally. The DMA channel provides error detection as a tool for system development and debug, helping to identify DMA-related programming errors. When the DMA channel detects an error, the channel immediately stops and discards any returned memory-read transactions. The DMA_STAT.RUN field of the DMA channel indicates the idle state after acknowledging all outstanding memory transactions. In addition, the channel asserts an error interrupt request and updates the DMA_STAT.IRQERR field. Also, the channel updates the DMA_STAT.ERRC field, indicating the error cause of the first detected error. Unless the error occurs at the exact moment that modification of register values occurs, the registers contain the error values.

All the DMA error interrupt requests are combined into a single shared interrupt request output. Combined error signals require reading the DMA_STAT register of each DMA channel associated with a combined error interrupt request to determine the DMA channel responsible for the generation of the interrupt.

The DMA channel error interrupt handler performs the following actions:

- Read the DMA_STAT register of each DMA channel, seeking a channel with the DMA_STAT.IRQERR set to indicate an error.
- Read the DMA_STAT.ERRC field of each DMA channel, determining the cause of the error.
- Clear the problem with the DMA channel. For example, fix the register values.
- Clear the error in the DMA channel through a write-1-to-clear operation to the DMA_STAT.IRQERR bit.

If the channel flags any uncleared error other than a bandwidth monitor error, the channel reports no other error. If the channel reports an uncleared bandwidth monitor error, the channel reports any newly detected error through updating the DMA_STAT.ERRC field.

DMA Configuration Register Errors

The channel only flags these configuration errors when the DMA_CFG.EN bit enables the DMA channel. Error flagging occurs when the configuration:

- Uses a reserved setting
- Enables DMA_CFG.TWAIT in descriptor on-demand flow mode
- Uses an illegal DMA_CFG.NDSIZE
- Uses an illegal DMA_CFG.MSIZE
- Configures DMA_XCNT = 0 or, when DMA_YCNT = 0 in 2D DMA mode
- Uses non-zero value in DMA_CFG.NDSIZE when DMA is configured in stop mode or auto mode
- Enables interrupt or outgoing triggers on DMA_YCNT when DMA is configured in 1D mode
- Use a DMA_CFG.MSIZE that exceeds the FIFO size of the DMA channel
- Uses an illegal DMA_CFG.PSIZE
- Uses a DMA_CFG.PSIZE that exceeds the FIFO size
- Uses a DMA_CFG.PSIZE that exceeds the bus width
- Attempts to change from a transmit operation (memory read) to a receive operation without properly synching in the previous work unit or when it is the first work unit in a new chain
- Attempts to change DMA_CFG.PSIZE of a transmit operation (memory read) without properly synching in previous work unit or when it is the first work unit in a new chain
- Attempts to change from receive operation (memory write) to a transmit operation during a descriptor chain.

  The channel only can change from receive to transmit if the new transmit is synchronized and is the first work unit.
- Attempts to change DMA_CFG.PSIZE of a receive operation (memory write) when the operation was not the first work unit (with DMA_CFG.SYNC enabled)

Illegal Register Write During Run

The channel generates an error when a write occurs to writable registers of an enabled, running DMA channel. The channel blocks the write.

Address Alignment Error

The channel generates an address alignment error when any of the following apply:

- Alignment of a descriptor address is not on a 32-bit boundary.
- The current DMA_CFG.MSIZE configuration contains an unaligned transfer address. The DMA_ADDRSTART register is not aligned according to the DMA_CFG.MSIZE field.

Memory Access Error

The channel generates a memory access error when the DMA process:

- attempts to access an unpopulated address,

- attempts to access a location that provokes a security violation

The error returned from the memory triggers the memory access error.

Trigger Overrun Error

A trigger overrun error is generated when a new trigger input occurred while an outstanding trigger is waiting. This error is only generated if `DMA_CFG.TOVEN` is enabled.

Bandwidth-Monitor Error

The channel generates this error when the bandwidth-monitor count expires. This error is not fatal, and the DMA channel continues operation.

Control Interface Error

The channel reports control-interface errors as bus errors to the bus controller. This error can result from:

- An address error
- A register write error (write to a read-only register)

DMA Operating Modes

The DMA channel supports a number of different flow modes that control how the DMA channel progresses from one work unit to the next.

The flow mode of a DMA channel is not a global setting. A DMA descriptor set can include the descriptor responsible for configuring the flow of the work unit. There is no restriction, limiting the flow configuration to be the same for the entire descriptor chain. If the descriptor chain is not endless, the last descriptor set configures the flow to stop mode, which results in termination of the descriptor chain after the work unit completes. Another example for mixing flow modes is to create an endless descriptor-array. The configuration of the last descriptor set in the array selects the descriptor-list mode. The next descriptor pointer in this set of descriptors points to the first descriptor in the array.

Register-Based Flow Modes

Register-based DMA operations require configuration by directly writing to the memory-mapped registers of the DMA channel.

Register-based DMA is the traditional method of DMA operation. Software writes all of the configuration of the DMA channel into the memory-mapped registers. This configuration includes information such as the source or destination address and length of the data in the transfer. The DMA controller then starts channel operation. The DMA channel supports the following register-based flow modes.

- Stop Mode
- Autobuffer Mode

The DMA channel supports variable descriptor set sizes within the configuration. The size of a descriptor set can contain as little as a single descriptor. The supported descriptor set sizes can differ between the various descriptor-based flow modes. In addition to the descriptor set size being configurable, descriptor-based DMA also allows altering the flow mode of the next descriptor set. This feature allows for the transition from descriptor-array mode to descriptor-list mode and permits configuring the flow to stop or autobuffer mode.

Stop Mode

In stop mode, the DMA operation executes only once. If started, the DMA channel transfers the desired number of data words and stops itself again when finished. If the DMA channel is no longer used, software configures the enable bit to disable a paused channel. The channel also can generate interrupts and triggers for each row or work unit completion, depending on the desired operation.

Autobuffer Mode

In autobuffer mode, the DMA operates repeatedly in a circular manner. If the transfer of all data words completes, the channel reloads the address pointer (`DMA_ADDR_CUR`) automatically with the `DMA_ADDRSTART` value. The channel also can generate an interrupt.

The `DMA_CFG.FLOW` field enables autobuffer mode. The configuration must load the `DMA_CFG.NDSIZE` field value, such that the next descriptor size is zero.

Descriptor-Based Flow Modes

Descriptor-based DMA operations fetch descriptor sets from memory allowing for autonomous loading of work units on other work units. Software does not need to set up the DMA sequences directly by writing into the DMA controller registers. Rather, software keeps DMA descriptor sets in memory.

Descriptor-based DMA operations have the following additional attributes.

- The DMA controller autonomously loads the descriptor set from memory to the affected DMA controller registers on demand.
- The channel can fetch descriptor sets from any memory space that supports DMA read operations.
- The descriptor set describes the next operation that the DMA controller performs.
- The descriptor set can include information such as the DMA configuration word as well as data source or destination address, transfer count, and address modify values.

A descriptor set describes a single work unit. The next work unit can reuse some values from the previous one descriptor set. But, this reusage is possible only if they are not overwritten in the subsequent descriptor set fetches and only if the work unit requires the use of this descriptor.

The DMA channel supports the following flow modes with descriptor-based operations.

- Descriptor-Array Mode
- Descriptor-List Mode
- Descriptor-On-Demand Modes

The DMA channel supports variable descriptor set sizes within the configuration. The size of a descriptor set can contain as little as a single descriptor and the supported descriptor set sizes can differ between the various descriptor-based flow modes. In addition to configurable descriptor set size, descriptor-based DMA also allows for altering of the flow mode of the next descriptor set. Programs can transition from one descriptor-based mode to another descriptor-based mode and can also transition to any of the register-based flow modes.

Descriptor-Array Mode

When configured in this mode, the descriptor sets do not contain further descriptor pointers. Software writes the initial descriptor-pointer value, which points to an array of descriptors. This operation assumes that the individual descriptors reside next to each other and assumes that their addresses are known.

The ***Offsets for Descriptor-Array Mode Parameters and Descriptors*** table illustrates how to structure a descriptor set in memory. The descriptor sets must reside in a contiguous block or memory in the format shown in the table. Locate the first descriptor of the next descriptor set in the memory location immediately following the last descriptor of the current descriptor set. The values have the same order as the corresponding offset addresses of the memory-mapped register.

Table 36-10: Offsets for Descriptor -Array Mode Parameters and Descriptors

Descriptor Offset	Parameter Register
0x00	DMA_ADDRSTART
0x04	DMA_CFG
0x08	DMA_XCNT
0x0C	DMA_XMOD
0x10	DMA_YCNT
0x14	DMA_YMOD

All other DMA channel registers that were not loaded as a result of the descriptor set fetch, retain their previous values. The channel reloads all of the current registers between the descriptor set fetch and the start of the DMA operation for the work unit.

NOTE: At a minimum, write the DMA_DSCPTR_CUR register prior to writing to the DMA_CFG register, which is the special action required to start the DMA channel.

Descriptor-List Mode

In this flow mode, multiple descriptors form a chained list in which each descriptor set contains a pointer to the next descriptor set, allowing greater flexibility in memory layout options. When the channel fetches the descriptor set, the operation loads this pointer value into the next descriptor pointer register of the DMA channel.

Descriptor Sets

The ***Offsets for Descriptor-List Mode Parameters and Descriptors*** table shows how to structure a descriptor set in memory. Disperse the placement of the descriptor sets throughout memory, having sets reside in different memory

blocks. But, each descriptor of the descriptor set must reside in a contiguous section of memory in the format shown in the table. The values have the same order as the corresponding offset addresses of the memory-mapped registers.

Table 36-11: Offsets for Descriptor-List Mode Parameters and Descriptors

Descriptor Offset	Parameter Register
0x00	DMA_DSCPTR_NXT
0x04	DMA_ADDRSTART
0x08	DMA_CFG
0x0C	DMA_XCNT
0x10	DMA_XMOD
0x14	DMA_YCNT
0x18	DMA_YMOD

All other DMA channel registers that were not loaded as a result of the descriptor set fetch, retain their previous values. The channel reloads all of the current values of the registers between the descriptor set fetch and the start of the DMA operation for the work unit.

Minimum Startup Requirements

At a minimum, write the `DMA_DSCPTR_NXT` register prior to write to the `DMA_CFG` register, which is the special action required to start the DMA channel.

Descriptor-On-Demand Modes

The Descriptor-Array Mode and Descriptor-List Mode each have an on-demand mode of operation.

In on-demand mode, at the end of the work unit, if the DMA channel has not detected an incoming trigger event, the channel repeats the current work unit. If the DMA channel receives an incoming trigger before completion of the work unit, the channel fetches a new descriptor set.

The ***Offsets for Descriptor-Array Mode Parameters and Descriptors*** and ***Offsets for Descriptor-List Mode Parameters and Descriptors*** tables illustrate how to structure each descriptor set in memory.

Table 36-12: Offsets for Descriptor-Array Mode Parameters and Descriptors

Descriptor Offset	Parameter Register
0x00	DMA_ADDRSTART
0x04	DMA_CFG
0x08	DMA_XCNT
0x0C	DMA_XMOD

Table 36-12: Offsets for Descriptor-Array Mode Parameters and Descriptors (Continued)

Descriptor Offset	Parameter Register
0x10	DMA_YCNT
0x14	DMA_YMOD

NOTE: For descriptor-array mode, at a minimum, write the DMA_DSCPTR_CUR register prior to writing to the DMA_CFG register, which is the special action required to start the DMA channel.

Table 36-13: Offsets for Descriptor-List Mode Parameters and Descriptors

Descriptor Offset	Parameter Register
0x00	DMA_DSCPTR_NXT
0x04	DMA_ADDRSTART
0x08	DMA_CFG
0x0C	DMA_XCNT
0x10	DMA_XMOD
0x14	DMA_YCNT
0x18	DMA_YMOD

NOTE: For descriptor-list mode, at a minimum, write the DMA_DSCPTR_NXT register prior to write to the DMA_CFG register, which is the special action required to start the DMA channel.

Data Transfer Modes

In addition to supporting basic one-dimensional DMA transfers, the DMA channel also supports two-dimensional functionality.

Two-Dimensional DMA

Register-based flow modes and descriptor-based flow modes support two-dimensional data transfers.

In two-dimensional (2D) mode, the X-direction count (DMA_XCNT), the X-direction modifier (DMA_XMOD), the Y-direction count (DMA_YCNT), and the Y-direction modifier (DMA_YMOD) support arbitrary row and column sizes. Also, the modify values can be negative, allowing implementation of interleaved data streams. The DMA_XCNT value specifies the row size, and the DMA_YCNT value specifies the column size; where the DMA_XCNT value must be 2 or greater.

The DMA start address (DMA_ADDRSTART), the X-direction modifier (DMA_XMOD), and the Y-direction modifier (DMA_YMOD) specifications all are in bytes. The alignment must be a multiple of the DMA transfer word size; configured using the DMA_CFG.MSIZE bit. Misalignment results in a DMA channel error.

The DMA_XMOD register value is the byte-address increment that the channel applies after each transfer, decrementing the DMA_XCNT register. The channel does not apply the DMA_XCNT when the inner loop count ends with the DMA_XCNT_CUR register decrementing to 0 from 1. Except, the channel does apply the DMA_XCNT on the final transfer, when the DMA_YCNT register is 1 and the DMA_XCNT register decrements from 1 to 0.

The DMA_YMOD register value is the byte-address increment that the channel applies after each decrement of the value in DMA_YCNT_CUR. However, the channel does not apply the DMA_YMOD value to the last item in the array on which the outer loop count (DMA_YCNT_CUR) also expires by decrementing from 1 to 0.

After the last transfer completes, DMA_YCNT_CUR is 1 and the DMA_XCNT_CUR register is 0. The DMA channels current address points to the last items address plus the DMA_XMOD register value. If the DMA channel programming selects automatic refresh (such as in autobuffer mode), the channel reloads the DMA_XCNT_CUR, DMA_YCNT_CUR, and DMA_ADDR_CUR for the first data transfer of the next work unit.

Interrupt notification is configurable for end-of-row or end-of-work unit completion.

For example, two-dimensional DMA can be used to extract interleaved data (such as RGB values for a video frame) by modifying both DMA_XMOD and DMA_YMOD values. The ***Capturing a Video Data Stream 2D DMA Example*** depicts the process of receiving a stream of the R, G, B values from an N*M frame. The inner loop of the 2D DMA configuration has three values (DMA_XCNT = 3) and a stride (DMA_XMOD) of N*M, chosen such that successive elements in each row are 1-2-3, 4-5-6 and so forth. The outer loop of the 2D DMA configuration has N*M values (DMA_YCNT = N*M) and a negative stride (DMA_YMOD) of 1-2*N*M chosen to instruct the DMA controller to jump from element 3 to 4, 6 to 7 and so forth at the end of each inner loop.

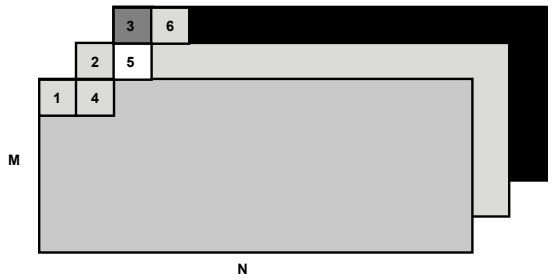


Figure 36-4: Capturing a Video Data Stream 2D DMA Example

DMA Channel Event Control

The DMA channel supports a number of events that provide notification of work unit state, peripheral data request, peripheral interrupt request and completion events, and DMA channel error conditions. In addition to flexible interrupt configuration, the DMA channel also supports incoming and outgoing triggers which are useful in synchronizing the DMA channel with other system resources.

The DMA channel has two interrupt signals for support of a number of events such as work-unit state events, peripheral interrupt request (PIRQ) events, peripheral data request (PDR) events, and DMA channel errors. The channel reports DMA channel errors on a dedicated interrupt signal. All other interrupt sources share an interrupt signal.

In addition to flexible interrupt configuration, the DMA channel also supports incoming and outgoing triggers which are useful in synchronizing the DMA channel with other system resources.

The channel can signal the processor on DMA channel events using status information and optional interrupt requests. Programs can use these events to update the progress of data transfers and to request intervention from the processor core. Configure most DMA channel interrupts using bits in the DMA_CFG register. Dedicated bits in the DMA_STAT register report the occurrence of various events. Use write-one-to-clear (W1C) operations to clear interrupt requests from the status register.

NOTE: Hardware does not clear the interrupt status bits automatically, even when programs disable then reenable the DMA channel. In this situation, the channel deasserts the interrupt signal, after the program disables the DMA channel. But, the status bit remains set until software either re-enables the DMA channel or clears the status bit.

The DMA channel supports the following categories of events on the interrupt signals:

- Work-unit state events generate interrupts on row or on work unit DMA completion.
- A peripheral uses peripheral interrupt request (PIRQ) events to signal when it has completed the transfer of all data.
- A peripheral uses peripheral data request (PDR) events to request data from a disabled or idle DMA channel.
- Error events signal a failure in the work unit.

ATTENTION: While in an error state, the DMA channel does not generate an interrupt to the processor for a work-unit state event or a PIRQ event, nor does the channel forward a PDR event.

Event Signals

The ***Event Signals*** table provides descriptions of DMA channel events.

Table 36-14: Event Signals

Signal	Width (bits)	Description
DMA_ERROR	1	Used to signal an error condition in the DMA channel. The source of the error can be determined by reading the DMA_STAT.ERRC bit.
DONE_PIRQ_INT	1	Signal used to indicate DMA completions events, PIRQ events and also for forwarding PDR events based on configuration. Read the corresponding fields in DMA_STAT to determine the source of the event.
DMA_TRIG_OUT	1	Trigger output that gets routed to the TRU and can be configured to provide notification on row or work unit completion.
DMA_TRIG_IN	1	Trigger input from the TRU that can be used to control the start of a work unit.

Work Unit State Events

Completing a row or a work unit generates a work-unit state event. For either of these events to generate an interrupt request, the configuration of the interrupt of the DMA channel must select one of the available work-unit completion modes.

- Current X count reaching 0 for row completion or 1D DMA work unit completion
- Current Y count reaching 0 for work unit completion of 2D DMA

NOTE: For 1D DMA, a DMA channel configuration error results if the configuration generates the interrupt request when the current Y counter reaches 0.

The DMA channel issues the last memory read or write transaction for the row or work unit, then pauses until the return of the read or write acknowledge. After successful acknowledge of the transfer, the DMA channel issues the interrupt request and continues to process the next row or work unit.

Waiting for acknowledgement of the memory access results in a delay. However, programs can read or modify data in the memory without adversely affecting or being affected by the DMA transfer.

NOTE: While the DMA channel pauses waiting for acknowledgement of the memory transfer, the DMA channel is still capable of fetching the next descriptor set. This fetch gets the channel ready to process the next work unit as soon as the memory access completes.

The channel configuration of the synchronization feature also affects interrupt timing. For memory-read operations with synchronization enabled, the channel delays the interrupt request until the completion of the last transfer from the DMA channel FIFO to the peripheral. The synchronization feature does not affect interrupt timing for memory write operations.

Peripheral Interrupt Request Events

For peripheral-transmit operations, a peripheral connected to the DMA channel can use peripheral interrupt request (PIRQ) events to indicate that data has left the channel FIFO and to indicate transfer completion.

In order to support PIRQ interrupts, correctly configure the interrupt of the DMA channel. This configuration disables the generation of interrupt requests based on the work unit state and, instead, results in generating an interrupt request when the DMA channel receives the command from the peripheral.

The channel only generates the interrupt request under the following conditions:

- The configuration enables the DMA channel
- The DMA channel is in the stop state
- The configuration of the DMA channel interrupt selects PIRQ operation

Peripheral Data Request Events

Peripheral data request (PDR) events occur when an interfaced peripheral requests data from the DMA channel and the DMA channel (either disabled or enabled) is in the stop state.

When a peripheral sends a data request command to a disabled DMA channel, the DMA channel generates an interrupt to the System Event Controller (SEC). There is no status information reported about this event in the status register of the DMA channel. Instead, the channel identifies the PDR event from the fact that the DMA channel generated an interrupt while disabled. It is possible to further confirm event status by verifying the status of the peripheral interfaced to the DMA channel.

This operation forwards data requests as interrupts when the DMA channel is in the disabled state. The DMA channel can forward PDR events as an interrupt request when the DMA channel is in the stop state after the completion of a work unit. The forwarding of this interrupt when the DMA channel is in the stop state is optional and configured by the program during DMA channel configuration.

DMA Channel Triggers

DMA channel triggers are useful for synchronizing the DMA channel with other events in the system. One usage is to combine channel triggers with each other to create ping-pong buffers. Another usage is to combine the triggers with interrupt requests to notify the processor on reaching a particular milestone that requires service. The channel also can use triggers to enforce a handshake DMA operation in which the trigger acts as a signal for a DMA request.

NOTE: Using the trigger to control the pace of data transfers, such as for handshake DMA, requires that all the data for the entire work unit is ready for transfer.

The DMA channel has a single incoming trigger that can control the pace of the data transfers performed by the DMA channel. The configuration can direct the DMA channel to wait for the incoming trigger before starting the work unit transfer or fetching a descriptor set from memory.

The DMA channel also has a single outgoing trigger signal. This configuration can direct this trigger to signal the end of row or an entire work unit. The DMA channel issues the last memory read or memory write transaction for the row or work unit, then pauses until return of the transfer acknowledge. After acknowledgement of the transfer, the DMA channel issues the trigger before processing the next row or work unit.

Issuing Triggers

The DMA channel configuration can direct the channel to generate an outgoing trigger signal at the end of row or the end of a work unit. The DMA channel issues the last memory read or memory write transaction for the row or work unit, then pauses until the return of the transfer acknowledge. After acknowledgement of the transfer, the DMA channel issues the trigger before processing the next row or work unit.

NOTE: While the DMA channel pauses waiting for acknowledgement of the memory transfer, the DMA channel is still capable of fetching the next descriptor set. This fetch gets the channel ready to process the next work unit as soon as the memory access completes.

Waiting For Triggers

Programs can use triggering to control the pace of data transfers performed by the DMA channel. The DMA channel enters a wait state before beginning the next work unit if the configuration enables `DMA_CFG.TWAIT` and either of the following apply:

- The channel receives a trigger since the last time the DMA channel left the wait state.
- The channel receives a trigger since its transition from disable to enable.

In the wait state, the DMA channel also does not perform a descriptor fetch. After receiving a trigger, the DMA channel leaves the wait state and begins the next work unit or fetches the next descriptor if configured for a descriptor-based mode of operation.

If a memory-mapped register write operation programs the channel with stop flow mode enabled (`DMA_CFG.TWAIT` bit) and the channel has not already received a trigger, the DMA channel enters a wait state before performing the data transfer. On receiving the trigger, the DMA channel begins the data transfer portion of the work unit. Once the data transfer is complete, the DMA channel enters the stop state.

If a memory-mapped register write operation programs the DMA channel with the flow mode configured to one of the descriptor-based modes, the DMA channel enters the wait state before performing the descriptor fetch. After completing the descriptor fetch, the DMA channel immediately proceeds to the data transfer, regardless of the value of the `DMA_CFG.TWAIT` bit. If another (next) descriptor fetch follows the descriptor fetch, the DMA channel enters a wait state before fetching the next descriptor.

If the descriptor fetch returns a descriptor with stop flow mode, the `DMA_CFG.TWAIT` value for that descriptor does not affect the DMA as the channel enters the stop state after completing the data transfer. The DMA channel only enters the wait state based on `DMA_CFG.TWAIT` before the next work unit or descriptor fetch.

If the descriptor fetch returns a descriptor configured for autobuffer flow mode, the `DMA_CFG.TWAIT` for that descriptor does not affect the DMA for the first work unit of the autobuffer transfer. After completing the first work unit and not receiving another trigger, the DMA channel enters the wait state before reinitializing its counters and address registers (if not configured for current addressing). The channel performs the next work unit after receiving the trigger.

The incoming trigger can occur when the DMA channel has not entered the wait state. The trigger can occur while the DMA channel is executing a work unit, is performing descriptor fetch, or is in the stop state. The trigger is held internally. After the work unit is complete, the DMA channel skips the wait state and proceeds directly to executing the following work unit. If the `DMA_CFG.TWAIT` bit is not enabled, the DMA channel also skips the wait state. However, the trigger is held internally and is used the next time the configuration enables `DMA_CFG.TWAIT`. This trigger retention allows programs to enable the `DMA_CFG.TWAIT` functionality several work units apart without concern for losing a trigger. The DMA channels trigger-overrun enable functionality can be enabled in all work units to ensure that multiple triggers do not occur between the work units with the `DMA_CFG.TWAIT` bit enabled.

DMA Channel Programming Model

Several synchronization and control methods are available for use in development of software tasks which manage peripheral DMA and memory DMA. Software must accept requests for new DMA transfers from other software tasks, integrate these transfers into existing transfer queues, and reliably notify other tasks when the transfers are complete.

In the processor, it is possible to manage each peripheral DMA and memory DMA stream with a separate task or to manage them together with any other stream. Each DMA channel has independent, orthogonal control registers, resources, and interrupts. So, the selection of the control scheme for one channel does not affect the choice of control scheme on other channels. For example, one peripheral can use a linked-descriptor-list, interrupt-driven scheme while another peripheral can simultaneously use a demand-driven, buffer-at-a-time scheme synchronized by polling DMA events.

The topics that follow describe the steps required to configure the DMA channel for the various modes in addition to the programming concepts required for software synchronization.

Mode Configuration

Use the step-by-step directions that follow to set up the DMA channel for operating modes.

Register-Based Linear-Buffer Stop Flow Mode

This procedure configures the DMA channel of a peripheral to read data from internal memory and to send it to the peripheral for transmission. Upon DMA completion, the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

Assume that the peripheral is in a state where it is ready to transmit data received from the DMA channel.

The task involves writing to a number of DMA channel MMR registers to configure a DMA channel to:

- Read data from internal memory, and
- Send it to a peripheral connected to the peripheral DMA bus.

1. Write the DMA_ADDRSTART register.

   ADDITIONAL INFORMATION: Software can use the address to calculate the most optimum possible DMA_CFG.MSIZE.

2. Calculate the optimum DMA_CFG.MSIZE based on the DMA_ADDRSTART register and number of bytes in work unit.

   ADDITIONAL INFORMATION: The number of bytes in the work unit must be a multiple of the selected DMA_CFG.MSIZE, and the calculation also must consider the start address alignment.

3. Write the DMA_XCNT register based on the calculated DMA_CFG.MSIZE.

   ADDITIONAL INFORMATION: The DMA_XCNT value is the number of DMA_CFG.MSIZE transfers to make up the entire work unit.

4. Write the DMA_XMOD register.

   ADDITIONAL INFORMATION: For a linear buffer transfer, determine the value in DMA_XMOD from the selected DMA_CFG.MSIZE. Always specify this register as a number of bytes.

5. Write the DMA_CFG register with DMA_CFG.EN configured to enable the DMA channel.

ADDITIONAL INFORMATION: Set the DMA_CFG.FLOW bit for STOP mode. Configure the DMA_CFG.WNR bit for memory read operation. Configure the DMA_CFG.PSIZE bits to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is recommended to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used.

- The DMA_CFG.SYNC bit can be configured to control DMA completion notification timing.
- Interrupts and triggers also can be configured at this step depending on requirements.

Now, the DMA channel is enabled, and the buffer is transferred. The DMA channel enters the IDLE state upon completion of the work unit.

Register-Based Autobuffer Flow Mode

This procedure configures the DMA channel of a peripheral to read data from internal memory and send it to the peripheral for transmission. The transmission of the buffer repeats endlessly. Upon DMA completion, the DMA channel restarts the DMA operation, creating an endless circular buffer transfer.

Assume the peripheral is in a state where it is ready to transmit data received from the DMA channel.

The task involves writing to a number of DMA channel MMR registers to configure a DMA channel to:

- Read data from internal memory, and
- Send it to a peripheral connected to the peripheral DMA bus.

1. Write the DMA_ADDRSTART register.

   ADDITIONAL INFORMATION: Use the address to calculate the optimum possible DMA_CFG.MSIZE.

2. Calculate the optimum DMA_CFG.MSIZE based on the DMA_ADDRSTART register and number of bytes in work unit.

   ADDITIONAL INFORMATION: The number of bytes in the work unit must be a multiple of the selected DMA_CFG.MSIZE, and the calculation must consider the start address alignment.

3. Write the DMA_XCNT register based on calculated DMA_CFG.MSIZE.

   ADDITIONAL INFORMATION: The DMA_XCNT register value is the number of DMA_CFG.MSIZE transfers to make up the entire work unit.

4. Write the DMA_XMOD register.

   ADDITIONAL INFORMATION: For a linear buffer transfer, determine the value in DMA_XMOD from the selected DMA_CFG.MSIZE. Always specify this register as a number of bytes.

5. Write the DMA_CFG register with the DMA_CFG.EN bit configured to enable the DMA channel.

ADDITIONAL INFORMATION: Set the DMA_CFG.FLOW bit for autobuffer mode. Configure the DMA_CFG.WNR bit for memory read operation. Configure the DMA_CFG.PSIZE bit to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is recommended to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used.

- The DMA_CFG.SYNC bit can be configured to control DMA completion notification timing.
- Interrupts and triggers also can be configured at this step depending on requirements.

Now, the DMA channel is enabled, and the buffer transfers until the DMA channel is disabled.

Descriptor-Array Flow Mode

This procedure configures the DMA channel of a peripheral to:

- Read data from memory as described by the descriptor sets in the array, and
- Send the data to the peripheral for transmission.

Descriptor sets are read from an array in memory to configure the individual work units.

Assume the peripheral is in a state where it is ready to transmit data received from the DMA channel. Assume that the array of descriptors is to be initialized with the last descriptor set configured for STOP flow mode.

The task involves writing to a number of DMA channel MMR registers to:

- Configure a DMA channel to read the array in memory, containing the first descriptor set that configured the DMA channel to retrieve, and
- Send the data to a peripheral connected to the peripheral DMA bus.

On DMA completion, the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

1. Write the DMA_DSCPTR_CUR register with the address of the array in which the descriptor sets are stored.

   ADDITIONAL INFORMATION: The array address must meet any processor alignments restrictions imposed by descriptor fetches.

2. Write the DMA_CFG register with the DMA_CFG.EN bit configured to enable the DMA channel.

   ADDITIONAL INFORMATION: Configure the DMA_CFG.PSIZE bits to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is recommended to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used. Set the DMA_CFG.FLOW bit for descriptor-array mode. Configure the

DMA_CFG.NDSIZE bits to describe the number of descriptor elements contained within the first descriptor set. Configure the DMA_CFG.WNR bit for memory read operation.

- The descriptor set that is fetched controls the DMA_CFG.SYNC configuration and the interrupt or trigger configurations.

The first descriptor set is fetched from memory location provided by the DMA_DSCPTR_CUR register and loaded to the MMR registers of the DMA channel.

Now, the DMA channel is processing all the work units provided in the descriptor array. The DMA channel enters the IDLE state on completion of the final work unit that was configured for STOP flow mode.

Descriptor-List Flow Mode

This procedure configures the DMA channel of a peripheral to:

- Read data from memory as described by the descriptor sets in the list, and
- Send it to the peripheral for transmission.

The DMA controller reads the descriptor sets from a list of descriptors. With the list, each descriptor set has a descriptor that points to the next descriptor set location in memory.

Assume the peripheral must be in a state where it is ready to transmit data received from the DMA channel. Assume that the list of descriptors must be initialized with the last descriptor set in the list configured for Stop flow mode.

The task involves writing to a number of DMA channel MMR registers to:

- Configure a DMA channel to read the list in memory, containing the first descriptor set that configured the DMA channel to retrieve, and
- Send the data to a peripheral connected to the peripheral DMA bus.

On DMA completion, the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

1. Write the DMA_DSCPTR_NXT register with the address of the first descriptor in the list to be processed.

   ADDITIONAL INFORMATION: The array address must meet any processor alignments restrictions imposed by descriptor fetches.

2. Write the DMA_CFG register with the DMA_CFG.EN configured to enable the DMA channel.

   ADDITIONAL INFORMATION: Set the DMA_CFG.FLOW for descriptor-list mode. Configure the DMA_CFG.NDSIZE bit to describe the number of descriptor elements contained within the first descriptor set. Configure the DMA_CFG.WNR bit for memory read operation. Configure the DMA_CFG.PSIZE bit to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is

recommended to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used.

- The descriptor set that is fetched controls the DMA_CFG.SYNC configuration and controls the interrupt or trigger configurations.

The first descriptor set is fetched from the memory location provided by DMA_DSCPTR_NXT and is loaded to the MMR registers of the DMA channel.

Now, the DMA channel is processing all the work units provided in the descriptor list. The DMA channel enters the idle state when the final work unit that was configured for stop-flow mode is complete.

Register-Based Memory-to-Memory Transfer in Stop Flow Mode

This procedure configures a memory DMA channel pair in stop flow mode. One DMA channel is configured for memory read operations, while the other DMA channel is configured for memory write.

The task involves writing to a number of DMA channels on two DMA channels that create a memory DMA pair. On DMA completion, the DMA channel enters the idle state, until either the DMA channel is disabled or is reconfigured for a new transfer.

1. Write the DMA_ADDRSTART register of the source DMA channel.

   ADDITIONAL INFORMATION: The address can be used to calculate the optimum DMA_CFG.MSIZE possible.

2. Calculate the optimum DMA_CFG.MSIZE based on the DMA_ADDRSTART register and number of bytes in work unit.

   ADDITIONAL INFORMATION: The number of bytes in the work unit must be a multiple of the selected DMA_CFG.MSIZE and the start address alignment must also be considered.

3. Write the DMA_XCNT register of the source DMA channel based on calculated DMA_CFG.MSIZE.

   ADDITIONAL INFORMATION: DMA_XCNT is the number of DMA_CFG.MSIZE transfers to make up the entire work unit.

4. Write the DMA_XMOD register of the source DMA channel.

   ADDITIONAL INFORMATION: For a linear buffer transfer, determine the value in DMA_XMOD from the selected DMA_CFG.MSIZE. This register is always specified in the number of bytes.

5. Write the DMA_ADDRSTART register of the destination DMA channel.

   ADDITIONAL INFORMATION: The address can be used to calculate the most optimum DMA_CFG.MSIZE possible.

6. Calculate the optimum DMA_CFG.MSIZE based on the DMA_ADDRSTART register and number of bytes in work unit.

ADDITIONAL INFORMATION: The number of bytes in the work unit must be a multiple of the selected DMA_CFG.MSIZE and the start address alignment must also be considered.

7. Write the DMA_XCNT register of the destination DMA channel based on the calculated DMA_CFG.MSIZE.

   ADDITIONAL INFORMATION: DMA_XCNT is the number of DMA_CFG.MSIZE transfers to make up the entire work unit.

8. Write the DMA_XMOD register of the destination DMA channel.

   ADDITIONAL INFORMATION: For a linear buffer transfer, determine the value in DMA_XMOD from the selected DMA_CFG.MSIZE. This register is always specified in the number of bytes.

9. Write the DMA_CFG register of the source DMA channel with DMA_CFG.EN configured to enable the DMA channel.

   ADDITIONAL INFORMATION: The DMA_CFG.FLOW bit field must be configured for stop mode. The DMA_CFG.WNR bit must be cleared for memory read operation. The DMA_CFG.PSIZE bits must be configured to a value no larger than the supported bus width of the peripheral DMA bus. From performance perspective, it is recommended to use the largest possible DMA_CFG.PSIZE value (for example, equal to the supported peripheral bus width (DMA_STAT.PBWID)). However, ensure that the number of bytes in the work unit is a multiple of the DMA_CFG.PSIZE value used.

   - The DMA_CFG.SYNC bit can be configured to control DMA completion notification timing.
   - Interrupts and triggers also can be configured at this step, depending on requirements. The interrupts and triggers are enabled within the destination DMA channel configuration.

   The memory read DMA transfer begins.

10. Write the DMA_CFG register of the destination DMA channel with DMA_CFG.EN configured to enable the DMA channel.

    ADDITIONAL INFORMATION: The DMA_CFG.FLOW bit field must be configured for stop mode. The DMA_CFG.WNR bit must be set for memory write operation. The DMA_CFG.PSIZE bits must be configured to a value no larger than the supported bus width of the peripheral DMA bus. This value must also match the value written for the source DMA channel configuration.

    - Interrupts and triggers also can be configured at this step depending on requirements.

    The memory write DMA transfer begins.

Both memory DMA channels are now running and the data is transferred from the source address to the destination address. The DMA channel enters the IDLE state upon completion of the work unit.

Programming Concepts

Using the features, operating modes, and event control for the DMA channel to their greatest potential requires an understanding of some DMA channel-related concepts.

Synchronization of Software and DMA

A critical element of software DMA management is the synchronization of DMA work unit completion with software. This synchronization can be achieved using DMA channel interrupt request and trigger events and using a poll of the status bits of these events within the DMA channel registers, or combining these techniques. Processor polling of DMA address/count/status for completion is not a recommended programming practice. The requirements and limitations of processor polling place significant responsibility onto the code developer to be deeply aware of the underlying hardware. The interrupt requests and triggers are designed for efficient code development and reuse.

Interrupt and Trigger Event-Based Synchronization

Interrupt and trigger based synchronization methods must avoid overrun. An overrun occurs when some events fail to invoke the event handler of a DMA channel for every event due to excessive latency in processing of events. The system design must ensure to either:

- Schedule only one event per channel (for example, at the end of a descriptor list), or
- Space the generated events sufficiently far apart in time that system processing budgets can guarantee service of every event.

The DMA channel issues status information through an interrupt request or trigger event or changes event status bits in the DMA_STAT register. This status guarantees that the last memory operation of the work unit is complete. For memory read DMA transactions, this status means that the FIFO of the DMA channel safely receives the final memory read data. For DMA transactions writing to memory, this status indicates that the DMA channel received an acknowledge of completion of the last write transfer of the work unit.

Register Polling Based Synchronization

Do not poll the DMA channel registers (DMA_ADDR_CUR, DMA_DSCPTR_CUR, DMA_XCNT_CUR, or DMA_YCNT_CUR) as a method of precisely synchronizing DMA with data processing. This approach is inaccurate due to the operation of the DMA channel FIFOs and DMA or memory pipelining. The current address, pointer, and count registers change several cycles in advance of the completion of the corresponding memory operation. This timing is measurable from the time at which the results of the operation are first visible to the core by memory read or write instructions.

For example, in a DMA channel memory write operation to external memory, assume DMA channel *A* initiates a DMA channel write operation. For memories with access latency, this operation requires many system-clock cycles. Meanwhile, DMA channel *B* (which does not in itself incur latency) initiates a transfer, which stalls behind the slow operation of channel *A*. Software monitoring channel *B* could not safely conclude whether the memory location pointed to by the DMA_ADDR_CUR of channel *B*. Also, the software cannot conclude whether the register has been written based solely on the contents of this register.

Polling of the current address, pointer, and count registers can permit loose synchronization of DMA with software. But, the software must allow for the lengths of the DMA or memory pipeline. Also, software must consider the length of the DMA FIFO for a particular peripheral. If the FIFOs are filled with incomplete work, the DMA channel does not advance current address, pointer, or count registers. The incomplete work includes reads that have been started but have not yet finished.

Additionally, software must consider the length of the pipelines to the destination memory. If the DMA FIFO length and channel memory-pipeline length are added, software can estimate the maximum number of incomplete memory operations in progress.

NOTE: The estimate would be a maximum, as the DMA or memory pipeline can include traffic from other DMA channels.

Descriptor Queues

A system designer may want to write a DMA manager facility which accepts DMA requests from other software. The DMA manager software does not know in advance when new work requests are received or what these requests contain. The software could manage these transfers using a circular linked list of DMA descriptors. In such a list, each descriptor sets the DMA_DSCPTR_NXT descriptor, which points to the next descriptor set. And the last descriptor set in the list points to the first descriptor set.

The code that writes into this descriptor list could use the circular addressing modes of the processor. This approach does not need to use comparison and conditional instructions to manage the circular structure. In this case, the DMA_DSCPTR_NXT descriptor of each descriptor set can be written once at startup, and skipped over as new contents are written for each descriptor.

The recommended method for synchronization of a descriptor queue is to use an interrupt or trigger. The descriptor queue is structured, such that (at least) the final valid descriptor set is always programmed to generate an interrupt or trigger event upon completion. More detail is provided in the following sections.

- Queues Using Event Generation for Every Descriptor Set
- Queues Using Minimal Events

Queues Using Event Generation for Every Descriptor Set

In this system, the DMA manager software synchronizes with the DMA channel by enabling an interrupt request or trigger on every descriptor set. Only use this method if the system design can guarantee that each work unit completion event is serviced separately (no interrupt or trigger overrun).

To maintain synchronization of the descriptor set queue, the non-interrupt software maintains a count of descriptor sets added to the queue. The event handler (either interrupt or trigger) maintains a count of completed descriptor sets removed from the queue. The counts are equal only when the DMA channel is paused after having processed all the descriptor sets.

When each new work unit event is received, the DMA manager software initializes a new descriptor set, taking care to set the flow to stop mode. Next, the software compares the descriptor set counts to determine whether the DMA channel is running. If the DMA channel is paused (counts equal), the software increments its count. Then, the software starts the DMA channel by writing the DMA_CFG of the new descriptor set.

If the counts are unequal, the software instead modifies the DMA_CFG of the next-to-last descriptor set, such that it now describes the newly queued descriptor set. This operation does not disrupt the DMA channel provided the rest of the descriptors of the set are initialized in advance. It is necessary to synchronize the software to the DMA to determine whether the DMA channel read the new or the old DMA_CFG value.

The event handler performs the synchronization operation. When an event is detected, the handler reads the DMA_STAT register of the DMA channel. If the DMA_STAT.RUN bit indicates that the DMA channel is running, the channel has moved on to processing another descriptor. The event handler can increment its count and exit. If the DMA_STAT.RUN bit indicates that the channel is not running, the channel is paused because either:

- There are no more descriptor sets to process, or
- The last descriptor set was queued too late

Where *too late* means that the modification of the DMA_CFG of the next-to-last descriptor set occurred *after* that descriptor was read into the DMA channel. In this case, the event handler does the following:

- Writes the DMA_CFG value appropriate for the last descriptor set to DMA_CFG register of the DMA channel,
- Increments the completed descriptor count, and
- Exits

If the event latencies of the system are large enough to cause any of the events to be dropped, this system can fail. An event handler capable of safely synchronizing multiple descriptor set interrupt requests is complex, performing several MMR accesses to ensure robust operation. In such a system environment, a minimal event synchronization method is preferred.

Queues Using Minimal Events

In this system, only one DMA interrupt request or trigger event is generated in the queue at any time. The DMA event handler for this system can also be extremely short. Here, the descriptor queue is organized into an *active* and a *waiting* portion, where events are enabled only on the last descriptor set in each portion.

When each new DMA request is processed, the software fills in the content of a new descriptor set and adds it to the waiting portion of the queue. The DMA_CFG descriptor of the descriptor set must have the flow set to stop mode. If more than one request is received before the DMA queue completion event occurs, the non-interrupt code queues later descriptor sets. It forms a waiting portion of the queue separate from the active portion of the queue that the DMA channel is processing. In other words, all but the last active descriptor sets contain flow values for a descriptor-based mode and have no event enable set.

The last active descriptor set has the stop flow mode and an event generation enabled. Also, all but the last waiting descriptor sets are configured for descriptor-based flow modes with no event generation. Only the last waiting descriptor set is configured for stop flow mode and event generation enabled. This configuration ensures that the DMA channel can automatically process the whole active queue before then issuing one event. Also, this arrangement makes it easy to start the waiting queue within the event handler by a single DMA_CFG register write.

After queuing a new waiting descriptor, the non-interrupt software leaves a message for its interrupt handler in a memory mailbox location. The location contains the desired DMA_CFG value for starting the first waiting descriptor set in the waiting queue (or 0, indicating no waiting descriptors).

The software must not modify the contents of the active descriptor set queue directly once processing by the DMA channel has started, unless careful synchronization measures are taken. In the most straightforward implementation of a descriptor set queue, the DMA manager software never modifies descriptors on the active queue. Instead, the

DMA manager waits until the DMA queue completion event indicates that the processing of the entire active queue is complete.

When a DMA queue completion event is received, the event handler reads the mailbox from the non-interrupt software and writes the value to the DMA_CFG register of the DMA channel. This write to a register restarts the queue, effectively transforming the waiting queue to an active queue. The event handler then passes a message back to the non-interrupt software indicating the location of the last descriptor set accepted into the active queue.

However, the event handler can read its mailbox and find a DMA_CFG value of zero, indicating there is no more work to perform. It then passes an appropriate message back to the non-interrupt software indicating that the queue has stopped.

The non-interrupt software which accepts new DMA work unit requests must synchronize the activation of a new work unit with the interrupt handler. If the queue has stopped (the mailbox from the event handler is zero), the non-interrupt software must start the queue. (The queue starts by writing the first descriptor sets DMA_CFG value to the DMA_CFG register of the channel). If the queue is not stopped, the non-interrupt software must not write the DMA_CFG register. (This write causes a DMA error). Instead, it must queue the descriptor onto the waiting queue and update its mailbox directed to the event handler.

ADSP-SC59x DMA Register Descriptions

The Direct Memory Access module (DMA) contains the following registers.

Table 36-15: ADSP-SC59x DMA Register List

Name	Description
DMA_ADDRSTART	Start Address of Current Buffer Register
DMA_ADDR_CUR	Current Address Register
DMA_BWLCNT	Bandwidth Limit Count Register
DMA_BWLCNT_CUR	Bandwidth Limit Count Current Register
DMA_BWMCNT	Bandwidth Monitor Count Register
DMA_BWMCNT_CUR	Bandwidth Monitor Count Current Register
DMA_CFG	Configuration Register
DMA_DSCPTR_CUR	Current Descriptor Pointer Register
DMA_DSCPTR_NXT	Pointer to Next Initial Descriptor Register
DMA_DSCPTR_PRV	Previous Initial Descriptor Pointer Register
DMA_STAT	Status Register
DMA_XCNT	Inner Loop Count Start Value Register
DMA_XCNT_CUR	Current Count (1D) or Intra-row XCNT (2D) Register
DMA_XMOD	Inner Loop Address Increment Register

Table 36-15: ADSP-SC59x DMA Register List (Continued)

Name	Description
DMA_YCNT	Outer Loop Count Start Value (2D only) Register
DMA_YCNT_CUR	Current Row Count (2D only) Register
DMA_YMOD	Outer Loop Address Increment (2D only) Register

Start Address of Current Buffer Register

The DMA_ADDRSTART register contains the start address of the work unit currently targeted for DMA. This register is read/write prior to enabling the channel, but is read-only after enabling channel.

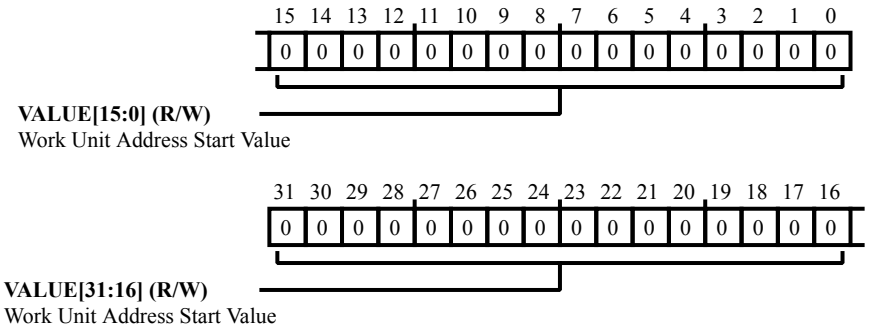


Figure 36-5: DMA_ADDRSTART Register Diagram

Table 36-16: DMA_ADDRSTART Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Work Unit Address Start Value. The DMA_ADDRSTART.VALUE bit field contains the start address of the work unit currently targeted for DMA.

Current Address Register

The DMA_ADDR_CUR register contains the present memory transfer address for a given work unit. At the start of a work unit, the DMA_ADDR_CUR register is loaded from the DMA_ADDRSTART register, and the DMA_ADDR_CUR register is incremented as each transfer occurs. The DMA_ADDR_CUR register is read/write prior to enabling the channel, but is read-only after enabling the channel.

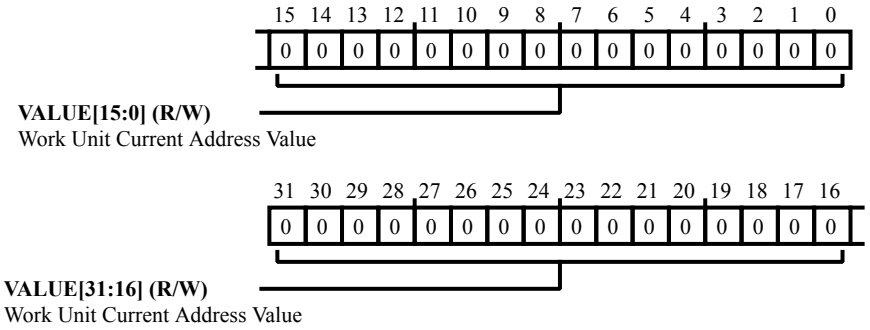


Figure 36-6: DMA_ADDR_CUR Register Diagram

Table 36-17: DMA_ADDR_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Work Unit Current Address Value. The DMA_ADDR_CUR.VALUE bit field contains the present memory transfer address for a given work unit.

Bandwidth Limit Count Register

The DMA_BWLCNT register contains a count that determines how often the DMA issues memory transactions. The DMA loads the value from the DMA_BWLCNT register into the DMA_BWLCNT_CUR register and decrements the current value each SYSCLK cycle. When DMA_BWLCNT_CUR reaches 0x0000, the next request is issued, and the DMA reloads DMA_BWLCNT_CUR. This bandwidth limit functionality is not applied to descriptor fetch requests. Programming 0x0000 allows the DMA to request as often as possible. 0xFFFF is a special case and causes requests to stop.

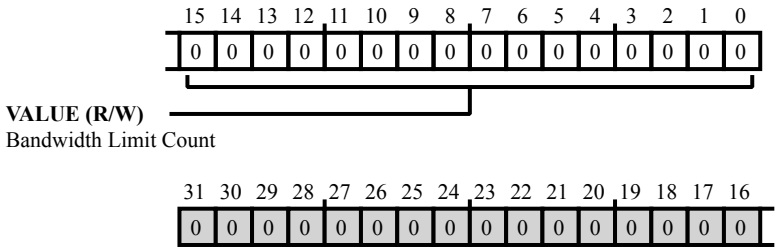


Figure 36-7: DMA_BWLCNT Register Diagram

Table 36-18: DMA_BWLCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Bandwidth Limit Count. The DMA_BWLCNT.VALUE bit field contains a count that determines how often the DMA issues memory transactions.

Bandwidth Limit Count Current Register

The DMA_BWLCNT_CUR register contains the number of SYSCLK count cycles remaining before the next request is issued.

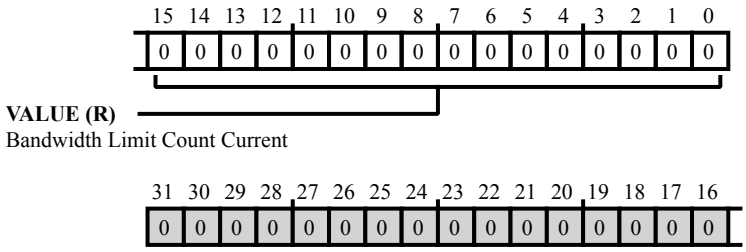


Figure 36-8: DMA_BWLCNT_CUR Register Diagram

Table 36-19: DMA_BWLCNT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/NW)	VALUE	Bandwidth Limit Count Current. The DMA_BWLCNT_CUR.VALUE bit field contains the number of SYSCLK count cycles remaining before the next request is issued.

Bandwidth Monitor Count Register

The DMA_BWMCNT register contains the maximum number of SYSCLK cycles allowed for a work unit to complete. Each time the DMA_CFG register is written (MMR access only), a work unit ends or an autobuffer wraps. The DMA loads the value in this register into the DMA_BWMCNT_CUR register.

The DMA decrements DMA_BWMCNT_CUR every SYSCLK a work unit is active. If the DMA_BWMCNT_CUR register reaches 0x0000_0000, the DMA_STAT.IRQERR bit is set, and the DMA_STAT.ERRC bit field is set to 0x6. The DMA_BWMCNT_CUR remains at 0x0000_0000 until it is reloaded when the work unit completes.

Unlike other errors, a bandwidth monitor error does not stop work unit processing. Programming 0x0000_0000 disables bandwidth monitor functionality.

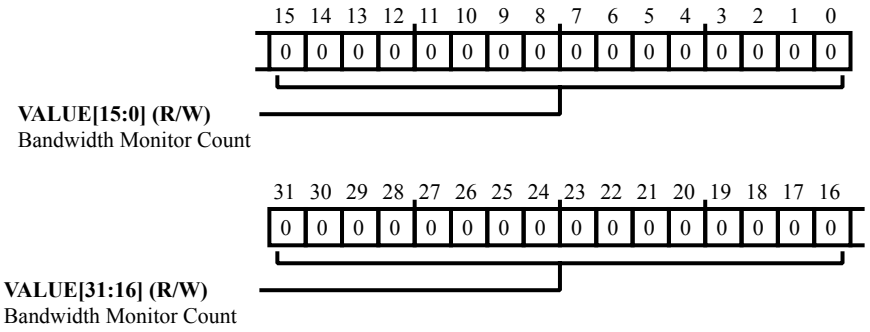


Figure 36-9: DMA_BWMCNT Register Diagram

Table 36-20: DMA_BWMCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Bandwidth Monitor Count. The DMA_BWMCNT.VALUE bit field contains the maximum number of SYSCLK cycles allowed for a work unit to complete.

Bandwidth Monitor Count Current Register

The DMA_BWMCNT_CUR register contains the number of cycles remaining for the current descriptor to complete.

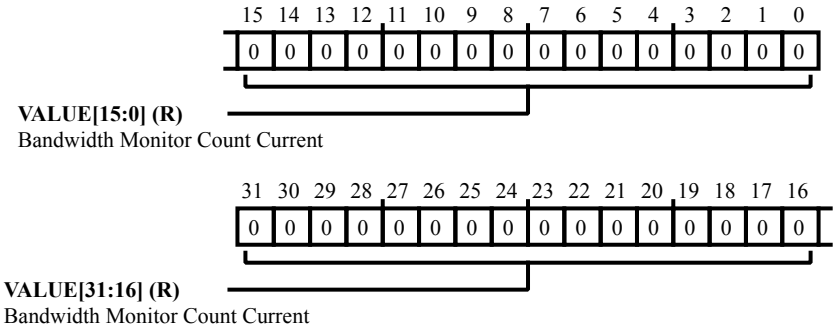


Figure 36-10: DMA_BWMCNT_CUR Register Diagram

Table 36-21: DMA_BWMCNT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Bandwidth Monitor Count Current. The DMA_BWMCNT_CUR.VALUE bit field contains the number of cycles remaining for the current descriptor to complete.

Configuration Register

The DMA_CFG register sets up DMA parameters and operation modes. Writing to the DMA_CFG register while a DMA process is already running causes a DMA error (except when clearing the DMA_CFG.EN bit).

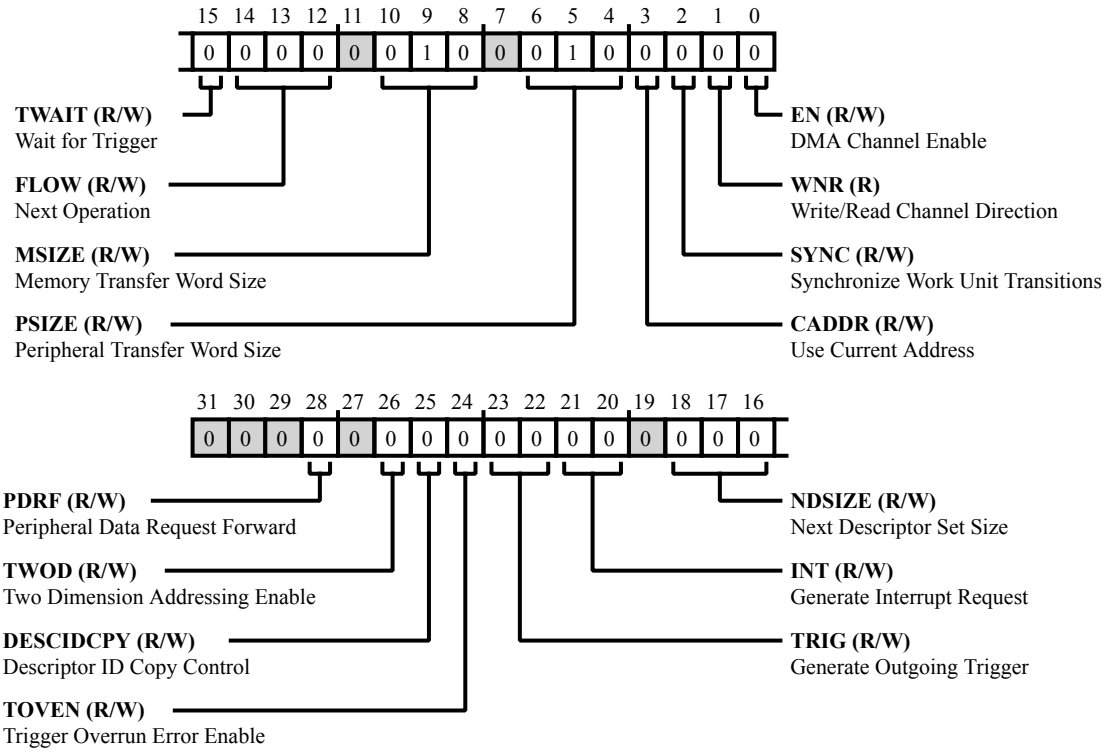


Figure 36-11: DMA_CFG Register Diagram

Table 36-22: DMA_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	PDRF	Peripheral Data Request Forward. The DMA_CFG.PDRF bit defines how the DMA handles data requests from the peripheral while in idle state after a stop mode or memory read work unit. If set, the DMA forwards the peripheral data request as an interrupt. This bit applies only to the DMA_CFG.FLOW bits configured for stop and DMA_CFG.WNR bits configured for memory read.	
		0	Peripheral Data Request Not Forwarded
		1	Peripheral Data Request Forwarded

Table 36-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/W)	TWOD	Two Dimension Addressing Enable. The DMA_CFG.TWOD bit selects whether the DMA addressing involves only DMA_XCNT and DMA_XMOD (one-dimensional DMA) or also involves DMA_YCNT and DMA_YMOD (two-dimensional DMA).	
		0	One-Dimensional Addressing
		1	Two-Dimensional Addressing
25 (R/W)	DESCIDCPY	Descriptor ID Copy Control. The DMA_CFG.DESCIDCPY bit specifies when to copy the initial descriptor pointer to the DMA_DSCPTR_PRV register. A bus write to the DMA_CFG register to clear the DMA_CFG.EN bit causes the DMA to immediately use the new value of the DMA_CFG.DESCIDCPY bit. To preserve consistency (if required by application), match the new value of this bit to the previous value.	
		0	Never Copy
		1	Copy on Work Unit Complete
24 (R/W)	TOVEN	Trigger Overrun Error Enable. A trigger overrun occurs if more than one trigger was received before the DMA reached the trigger wait state. If DMA_CFG.TOVEN is set, a trigger overrun causes the DMA to flag an error. In cases where a trigger overrun is not a problem, clearing DMA_CFG.TOVEN prevents the overrun from causing an error and halting the DMA. The DMA_CFG.TOVEN operates independently of the DMA_CFG.TWAIT bit selection.	
		0	Ignore Trigger Overrun
		1	Error on Trigger Overrun

Table 36-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:22 (R/W)	TRIG	Generate Outgoing Trigger. The DMA_CFG.TRIG selects whether the DMA issues an outgoing trigger, based on the work unit counter values. In one-dimensional mode, the only options are to trigger at the end of the whole work unit (trigger when DMA_XCNT_CUR reaches 0) or not to trigger at all. If in one-dimensional addressing mode, programming the DMA_CFG.TRIG bit field to trigger when DMA_YCNT_CUR reaches 0 (or to reserved) causes the DMA to flag a configuration error. In two-dimensional addressing mode, the trigger options are: at the end of each row of the inner loop (when DMA_XCNT_CUR reaches 0), only after completing the whole work unit (when DMA_YCNT_CUR reaches 0), or no trigger. If in two-dimensional mode and set to trigger when DMA_XCNT_CUR reaches 0, the DMA also issues a trigger at the end of the work unit. If in two-dimensional addressing mode, programming DMA_CFG.TRIG to reserved causes the DMA to flag a configuration error. If DMA_CFG.TRIG is non-zero and the peripheral issues a finish command, the DMA issues a trigger after the finish procedure is complete. For more information about trigger generation timing, see the trigger section of the DMA functional description.	
		0	Never Assert Trigger
		1	Trigger When XCNTCUR Reaches 0
		2	Trigger When YCNTCUR Reaches 0
		3	Reserved

Table 36-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:20 (R/W)	INT	Generate Interrupt Request. The DMA_CFG.INT bit field selects whether an interrupt request goes to the core based on work unit status or a peripheral interrupt request. For one-dimensional mode, setting the DMA_CFG.INT bits to generate an interrupt request when the DMA_YCNT_CUR register reaches 0 causes the DMA to flag a configuration error. The peripheral interrupt setting lets the DMA generate the last grant indication and to accept and or forward the peripheral interrupt command. The peripheral interrupt selection applies only to the DMA_CFG.FLOW bits set for stop and the DMA_CFG.WNR bits set for memory read. If the DMA_CFG.INT is set for interrupt request on count completion (DMA_XCNT_CUR or DMA_YCNT_CUR reach 0) and the peripheral issues a finish command, the DMA issues an interrupt request after the finish procedure is complete. For more information, see the sections on interrupt generation and peripheral control in the DMA functional description.	
		0	Never Assert Interrupt
		1	Interrupt When X Count Expires
		2	Interrupt When Y Count Expires
		3	Peripheral Interrupt request
18:16 (R/W)	NDSIZE	Next Descriptor Set Size. The DMA_CFG.NDSIZE bit field specifies the number of descriptor elements in memory to load during the next descriptor fetch. The DMA loads the descriptors in a specific order. The descriptor set contains the next descriptor pointer when it is a descriptor list. The descriptor set does not contain the next descriptor pointer when it is a descriptor array.	
		0	Fetch One Descriptor Element
		1	Fetch Two Descriptor Elements
		2	Fetch Three Descriptor Elements
		3	Fetch Four Descriptor Elements
		4	Fetch Five Descriptor Elements
		5	Fetch Six Descriptor Elements
		6	Fetch Seven Descriptor Elements
		7	Reserved

Table 36-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W)	TWAIT	Wait for Trigger. The DMA_CFG.TWAIT bit controls whether the DMA waits for an incoming trigger from another channel or user. If the DMA_CFG.TWAIT bit is set, the DMA enters the wait state before starting the next work unit, including descriptor fetch when in descriptor mode. Do not use the wait for trigger control for descriptor-on-demand mode which causes an error. For more information, see the trigger section of the DMA functional description.	
		0	Begin Work Unit Automatically (No Wait)
		1	Wait for Trigger (Halt before Work Unit)
14:12 (R/W)	FLOW	Next Operation. The DMA_CFG.FLOW bit field selects the descriptor handling options.	
		0	STOP. See the Stop Flow Mode section.
		1	AUTO. See the Autobuffer Flow Mode section.
		2	Reserved
		3	Reserved
		4	DSCL. See the Descriptor List Mode section.
		5	DSCA. See the Descriptor Array Mode section.
		6	Descriptor On-Demand List. See the Descriptor List Mode section.
		7	Descriptor On Demand Array. See the Descriptor On Demand section.
10:8 (R/W)	MSIZE	Memory Transfer Word Size. The DMA_CFG.MSIZE bits select memory transfer sizes of 8-, 16-, 32-, 64-, 128-, or 256-bit words. The transfer start address (DMA_ADDRSTART) and transfer increment values (DMA_XMOD, and, if needed, DMA_YMOD) must be a multiple of the memory transfer unit size.	
		2	4 Bytes
		3	8 Bytes
		4	16 Bytes
		5	32 Bytes

Table 36-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:4 (R/W)	PSIZE	Peripheral Transfer Word Size. The DMA_CFG.PSIZE bits select peripheral transfer sizes as 8, 16, 32, or 64 bits wide. Each request and grant results in a single peripheral access. There are no bursts on the peripheral bus, so the DMA_CFG.PSIZE selection must be less than or equal to the width of the bus. If the selection is greater than the bus width, a configuration error occurs. The peripheral bus of the processor is dedicated to DMA and peripheral accesses.	
		2	4 Bytes
		3	8 Bytes
3 (R/W)	CADDR	Use Current Address. When the DMA_CFG.CADDR bit is cleared, the DMA loads the DMA_ADDRSTART register on the first access of the work unit. When the DMA_CFG.CADDR bit is set, the DMA uses the DMA_ADDR_CUR register value for the starting address for the work unit and writes the same value to the DMA_ADDRSTART register. This operation permits continuation of a previous work unit. When DMA uses this mode, the fetched start address value (as part of the descriptor set at the end of a descriptor list or array) is ignored.	
		0	Load Starting Address
		1	Use Current Address

Table 36-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	SYNC	Synchronize Work Unit Transitions. Setting the DMA_CFG.SYNC bit clears the DMA FIFO and pointers before starting the first work unit of a work unit chain. When the transfer direction is memory read/transmit (DMA_CFG.WNR =0), the DMA waits until all data transmits to a peripheral before moving on to the next work unit, clearing the FIFO and pointers. When the transfer direction is memory write/receive (DMA_CFG.WNR =1), the DMA ignores the DMA_CFG.SYNC bit value after processing the first work unit of a work unit chain. As a channel can receive data when turned on but idle, data from the peripheral can still be in the FIFO even though the channel was not programmed. When the DMA_CFG.SYNC bit field is set at the beginning of a work unit chain (during the first work unit), the DMA clears the FIFO, erasing the data put into the FIFO while the channel was idle. Syncing lets programs change the DMA_CFG.PSIZE between individual work units and (in some cases) work unit chains. The sync resets the pointers in the FIFO, preventing misaligned FIFO access. Programs can change the DMA_CFG.MSIZE field between consecutive work units, independent of the DMA_CFG.SYNC bit setting. Syncing also permits changes to transfer direction. Because the data in the FIFO is eliminated, the data that went into the FIFO from one direction (transmit or receive) is not sent back in the other direction after the direction change.	
		0	No Synchronization
		1	Synchronize Channel
1 (R/NW)	WNR	Write/Read Channel Direction. The DMA_CFG.WNR selects receive (write to memory) or transmit (read from memory) channel direction.	
		0	Transmit (Read from memory)
		1	Receive (Write to memory)

Table 36-22: DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	EN	DMA Channel Enable. The DMA_CFG.EN bit enables the selected DMA channel. When a peripheral DMA channel is enabled, data requests from the peripheral denote DMA requests. When a channel is disabled, the DMA unit ignores the peripheral data request and passes it directly to the system event controller. To avoid unexpected results, enable the DMA channel before enabling the peripheral, and disable the peripheral before disabling the DMA channel. A transition of the DMA_CFG.EN bit from 0 to 1 creates a hard reset of all internal counters and states, including the DMA_STAT register. (All other register values remain unaffected.) A transition from 1 to 0 maintains all counters and registers for reading and analysis. Note that if a descriptor loads when this bit is cleared (see the DMA_CFG.FLOW field), the DMA transitions to the off or idle state after the descriptor load is complete.	
		0	Disable
		1	Enable

Current Descriptor Pointer Register

The DMA_DSCPTR_CUR register contains the memory address for the next descriptor to be loaded. The DMA_DSCPTR_CUR register is read/write prior to enabling the channel, but is read-only after enabling the channel. For DMA_CFG.FLOW mode settings that involve descriptor fetches, this register is used to read descriptors into appropriate MMRs before a work unit begins. For descriptor list mode, the DMA_DSCPTR_CUR register is initialized from the DMA_DSCPTR_NXT register before fetching each descriptor set. Then, the address in the DMA_DSCPTR_CUR register increments as each descriptor is read.

When the entire descriptor set has been read, the DMA_DSCPTR_CUR register contains this value:

DMA_DSCPTR_CUR = Descriptor Start Address + Descriptor Size (# of elements)

For descriptor array mode, the DMA_DSCPTR_CUR register, and not the DMA_DSCPTR_NXT register, must be programmed by a MMR access before starting DMA operation.

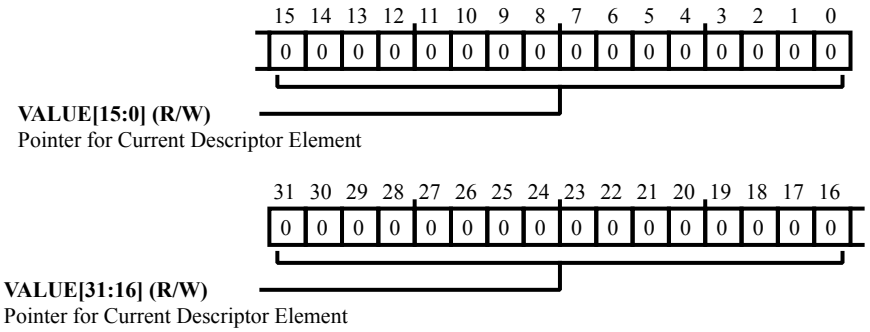


Figure 36-12: DMA_DSCPTR_CUR Register Diagram

Table 36-23: DMA_DSCPTR_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Pointer for Current Descriptor Element. The DMA_DSCPTR_CUR.VALUE bit field contains the memory address for the next descriptor to be loaded.

Pointer to Next Initial Descriptor Register

The DMA_DSCPTR_NXT register specifies the start location of the next descriptor set, which begins when the DMA activity specified by the current descriptor set completes. This register is read/write prior to enabling the channel, but is read-only after enabling channel.

The DMA_DSCPTR_NXT register is only used in descriptor list mode. At the start of a descriptor fetch in this mode, the DMA_DSCPTR_NXT register is copied into the DMA_DSCPTR_CUR register. During descriptor fetch, the DMA increments the DMA_DSCPTR_CUR register value after reading each element of the descriptor set.

In descriptor list mode, the DMA_DSCPTR_NXT register (not the DMA_DSCPTR_CUR register) must be programmed directly through MMR access, before the DMA operation is started. In descriptor array mode, the DMA disregards the DMA_DSCPTR_NXT register and uses the DMA_DSCPTR_CUR register to control descriptor fetch.

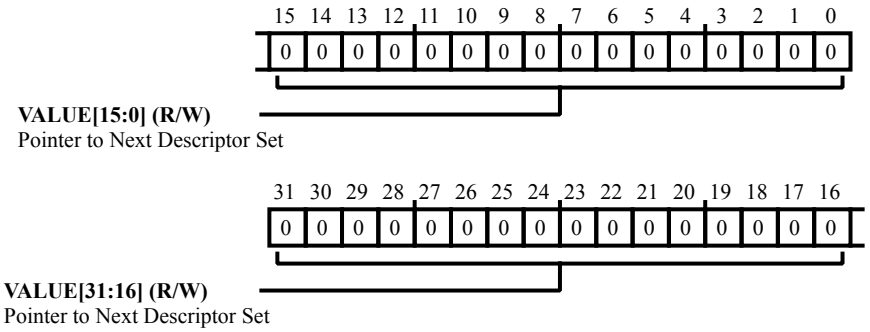


Figure 36-13: DMA_DSCPTR_NXT Register Diagram

Table 36-24: DMA_DSCPTR_NXT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Pointer to Next Descriptor Set. The DMA_DSCPTR_NXT.VALUE bit field specifies the start location of the next descriptor set.

Previous Initial Descriptor Pointer Register

The DMA_DSCPTR_PRV register contains the initial descriptor pointer for the previous work unit. If DMA_CFG.DESCIDCPYis set, the DMA copies the initial descriptor pointer to DMA_DSCPTR_PRV after the work unit completes. Otherwise, the value is not updated.

To indicate an overrun, bit 0 of the DMA_DSCPTR_PRV register is used as a previous descriptor pointer overrun (PDPO) status bit. Due to aligned addressing combined with all descriptors being 32 bits in width, bits 0 and 1 of all descriptor pointers must be zero. Otherwise, an alignment error occurs when used for descriptor fetches. As a result, bit 1 and 0 of the DMA_DSCPTR_PRV register can be used for status. For more information, see the section on descriptor pointer capture in the DMA functional description.

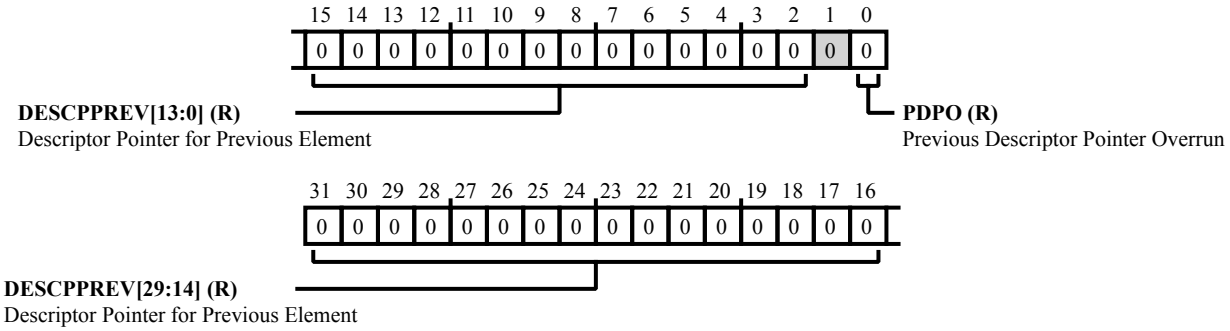


Figure 36-14: DMA_DSCPTR_PRV Register Diagram

Table 36-25: DMA_DSCPTR_PRV Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:2 (R/NW)	DESCPPREV	Descriptor Pointer for Previous Element. The DMA_DSCPTR_PRV.DESCPPREV bit field contains the initial descriptor pointer for the previous work unit.	
0 (R/NW)	PDPO	Previous Descriptor Pointer Overrun. The DMA_DSCPTR_PRV.PDPO bit signifies an alignment error. Due to aligned addressing combined with all descriptors being 32 bits in width, bits 0 and 1 of all descriptor pointers must be zero. Otherwise, an alignment error would occur when used for descriptor fetches.	
		0	No Error Occurred
		1	Error Occurred

Status Register

The DMA_STAT register indicates the status of DMA work units, the FIFO, errors, interrupts, and triggers.

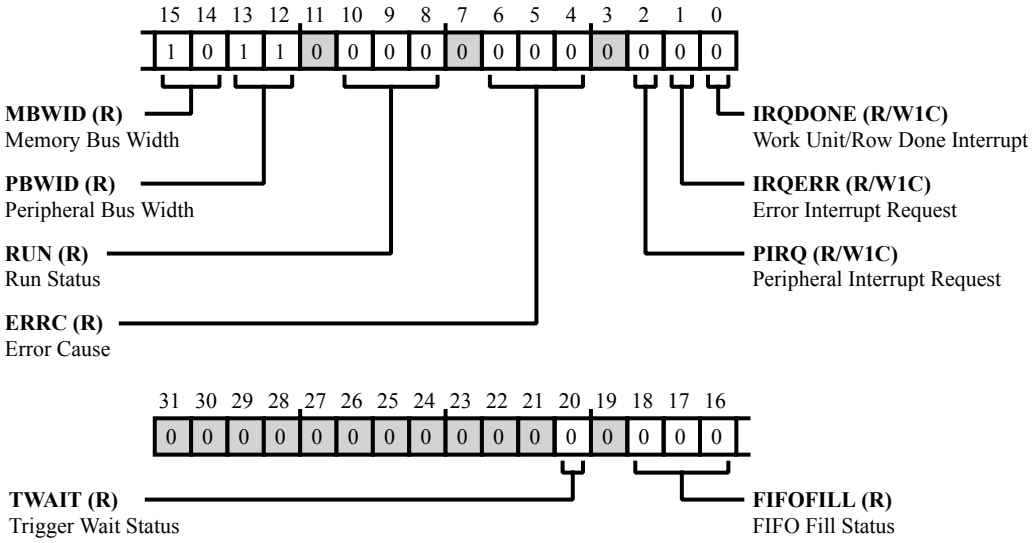


Figure 36-15: DMA_STAT Register Diagram

Table 36-26: DMA_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/NW)	TWAIT	Trigger Wait Status. The DMA_STAT.TWAIT bit indicates whether the DMA has or has not received a trigger. This bit is set until it reaches the next wait state. At that point, the bit is cleared, the DMA stops processing that work unit, and the following work unit processes. The DMA does not distinguish between one or more triggers received.	
		0	No Trigger Received
		1	Trigger Received
18:16 (R/NW)	FIFOFILL	FIFO Fill Status. The DMA_STAT.FIFOFILL bit field reports the quantity of data in the FIFO relative to available space.	
		0	Empty
		1	Empty < FIFO = 1/4 Full
		2	1/4 Full < FIFO = 1/2 Full
		3	1/2 Full < FIFO = 3/4 Full
		4	3/4 Full < FIFO = Full
		5	Reserved

Table 36-26: DMA_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		6	Reserved
		7	Full
15:14 (R/NW)	MBWID	Memory Bus Width. The DMA_STAT.MBWID bit field indicates the width of the memory bus connected to this DMA.	
		0	2 Bytes
		1	4 Bytes
		2	8 Bytes
		3	16 Bytes
13:12 (R/NW)	PBWID	Peripheral Bus Width. The DMA_STAT.PBWID bit field indicates the width of the peripheral bus connected to this DMA.	
		0	1 Byte
		1	2 Bytes
		2	4 Bytes
		3	8 Bytes
10:8 (R/NW)	RUN	Run Status. The DMA_STAT.RUN bit field reports the DMA's current operational state. If the DMA is in idle or stop state, the DMA_CFG.EN bit is either 0 or 1. This bit field does not clear when a transition of the DMA_CFG.EN bit from 0 to 1 occurs. The DMA_STAT.RUN clears automatically when the DMA completes.	
		0	Idle/Stop State
		1	Descriptor Fetch
		2	Data Transfer
		3	Waiting for Trigger
		4	Waiting for Write ACK/FIFO Drain to Peripheral
		5	Reserved
		6	Reserved
		7	Reserved
6:4 (R/NW)	ERRC	Error Cause. When an interrupt request error occurs (DMA_STAT.IRQERR), the DMA updates the DMA_STAT.ERRC bit field to identify the type of error. For more information, see the errors section of the DMA functional description.	

Table 36-26: DMA_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	Configuration Error
		1	Illegal Write Occurred While Channel Running
		2	Address Alignment Error
		3	Memory Access or Fabric Error
		4	Reserved
		5	Trigger Overrun
		6	Bandwidth Monitor Error
		7	Reserved
2 (R/W1C)	PIRQ	Peripheral Interrupt Request. The DMA_STAT.PIRQ bit indicates a peripheral interrupt request. Programs can use the DMA_STAT.PIRQ bit status to help determine which DMA asserted the interrupt request. This bit also helps to distinguish between an interrupt request caused by the state of the work unit and an interrupt request caused by the peripheral.	
		0	No Interrupt request
		1	Interrupt Request signaled by peripheral
1 (R/W1C)	IRQERR	Error Interrupt Request. The DMA_STAT.IRQERR bit indicates that the DMA has detected a documented rule violation during DMA programming or operation. The DMA cannot, however, flag all possible programming or operation issues to indicate errors. Programmers can use the DMA_STAT.IRQERR bit to help determine which DMA issued the error interrupt request. The DMA_STAT.IRQERR does not clear a transition of the DMA_CFG.EN bit from 0 to 1. Clear the DMA_STAT.IRQERR bit with a write-1-to-clear operation prior to the DMA_CFG.EN transition for the fields to be reset.	
		0	No Error
		1	Error Occurred
0 (R/W1C)	IRQDONE	Work Unit/Row Done Interrupt. The DMA_STAT.IRQDONE bit indicates that the DMA has detected the completion of a work unit or row (inner loop count) and has issued an interrupt request. Programs can use the DMA_STAT.IRQDONE status to help determine which DMA asserted the interrupt request. Programs can also use these bits to help distinguish between an interrupt request based on the state of the work unit and an interrupt request made by the peripheral. For more information, see the interrupts section of the DMA functional description.	
		0	Inactive
		1	Active

Inner Loop Count Start Value Register

For 2D DMA, the DMA_XCNT register contains the inner loop count. This value selects the number of DMA_CFG.MSIZE size data transfers that make up the length of a row. For 1D DMA, the DMA_XCNT register specifies the number of DMA_CFG.MSIZE size data transfers for the entire work unit. The DMA_XCNT register is read/write prior to enabling the channel, but is read-only after enabling channel. Note that the DMA generates a configuration error if this register is 0x0 when a work unit begins.

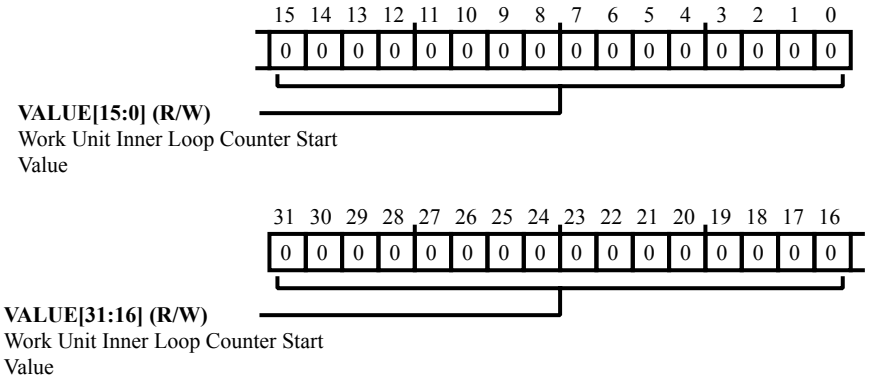


Figure 36-16: DMA_XCNT Register Diagram

Table 36-27: DMA_XCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Work Unit Inner Loop Counter Start Value. For 2D DMA, the DMA_XCNT.VALUE bit field contains the inner loop count. For 1D DMA, DMA_XCNT.VALUE specifies the number of DMA_CFG.MSIZE size data transfers for the entire work unit.

Current Count (1D) or Intra-row XCNT (2D) Register

For 1D DMA, the DMA loads the DMA_XCNT_CUR register from the DMA_XCNT register at the beginning of each work unit. For 2D DMA, the DMA loads the DMA_XCNT_CUR register from the DMA_XCNT register after the end of each row. The DMA decrements the value in DMA_XCNT_CUR register each time a DMA_CFG.MSIZE size data transfer occurs. When the count in DMA_XCNT_CUR register expires, the work unit is complete. In 2D DMA, the DMA_XCNT_CUR value is 0 only when the entire transfer is complete.

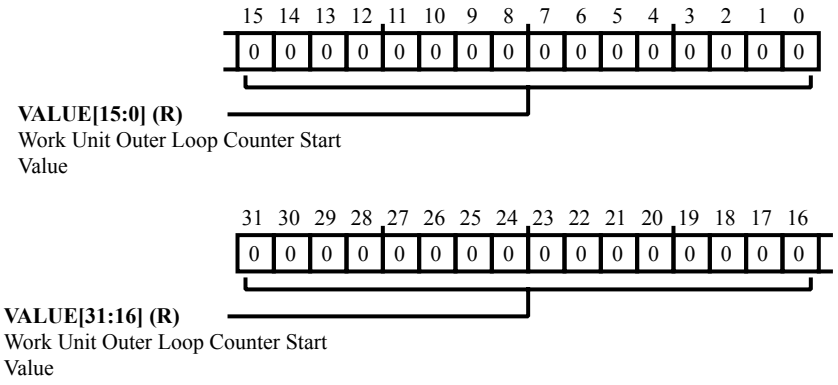


Figure 36-17: DMA_XCNT_CUR Register Diagram

Table 36-28: DMA_XCNT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Work Unit Outer Loop Counter Start Value. For 1D DMA, the DMA_XCNT_CUR.VALUE bit field holds the DMA_XCNT value at the beginning of each work unit. For 2D DMA, the DMA_XCNT_CUR.VALUE bit field holds the DMA_XCNT value after the end of each row.

Inner Loop Address Increment Register

The DMA_XMOD register contains a signed, two's-complement byte address increment. In 1D DMA, this increment is the stride that is applied after each DMA_CFG.MSIZE size data transfer. The DMA_XMOD register is read/write prior to enabling the channel, but is read-only after enabling the channel.

The DMA_XMOD register value is specified in bytes, regardless of the work unit size. In 2D DMA, this increment is applied after each DMA_CFG.MSIZE size data transfer in the inner loop, up to but not including the last DMA_CFG.MSIZE size data transfer in each inner loop. After the last DMA_CFG.MSIZE size data transfer in each inner loop, the DMA_YMOD register is applied instead, including the last DMA_CFG.MSIZE size data transfer of a work unit.

The DMA_XMOD field can be set to 0. In this case, DMA is performed repeatedly to or from the same address. This approach can be useful for transferring data between a data register and an external memory-mapped peripheral.

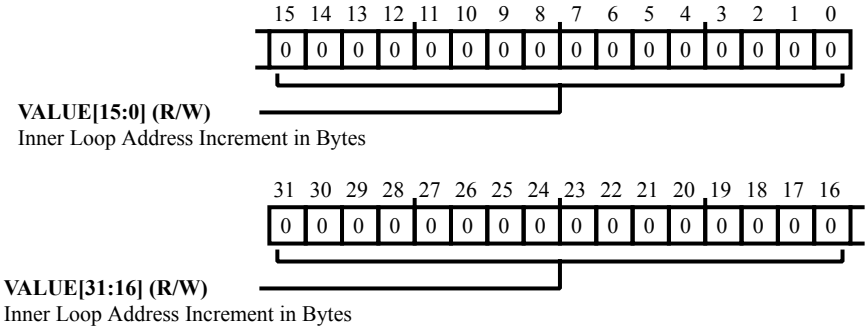


Figure 36-18: DMA_XMOD Register Diagram

Table 36-29: DMA_XMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Inner Loop Address Increment in Bytes. The DMA_XMOD.VALUE bit field contains a signed, two's-complement byte address increment.

Outer Loop Count Start Value (2D only) Register

For 2D DMA, the DMA_YCNT register contains the outer loop count. This register is not used in 1D DMA mode. The DMA_YCNT register specifies the number of rows in the outer loop of a 2D DMA sequence. The DMA_YCNT register is read/write prior to enabling the channel, but is read-only after enabling channel. Note that the DMA generates a configuration error if this register is 0x0 when a work unit begins.

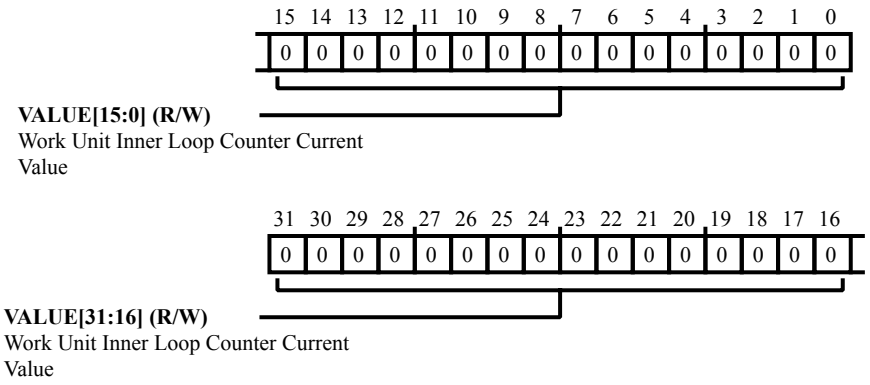


Figure 36-19: DMA_YCNT Register Diagram

Table 36-30: DMA_YCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Work Unit Inner Loop Counter Current Value. For 2D DMA, the DMA_YCNT.VALUE bit field contains the outer loop count.

Current Row Count (2D only) Register

For 2D DMA, the DMA loads the DMA_YCNT_CUR register from the DMA_YCNT register at the beginning of each 2D DMA session. The DMA_YCNT_CUR register is not used for 1D DMA. The DMA decrements the DMA_YCNT_CUR register each time the DMA_XCNT_CUR register expires during 2D DMA operation, signifying the completion of an entire row transfer.

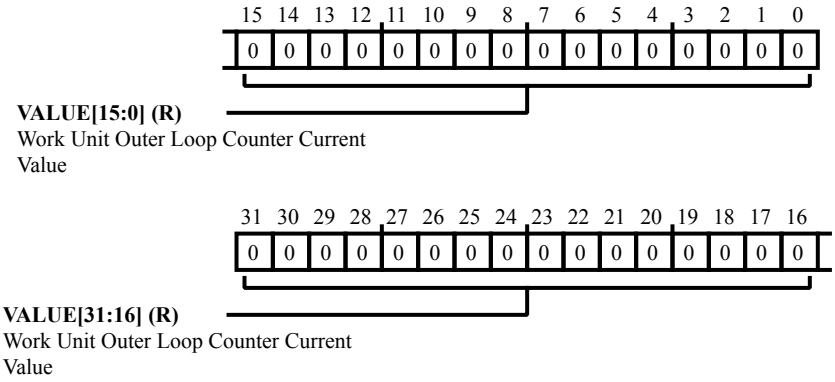


Figure 36-20: DMA_YCNT_CUR Register Diagram

Table 36-31: DMA_YCNT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Work Unit Outer Loop Counter Current Value. For 2D DMA, the DMA_YCNT_CUR.VALUE bit field holds the value from the DMA_YCNT register at the beginning of each 2D DMA session.

Outer Loop Address Increment (2D only) Register

The DMA_YMOD register contains a signed, two's-complement value. This byte address increment is applied after each decrement of the DMA_YCNT_CUR register. The value is the offset between the last word of one row and the first word of the next row. Note that DMA_YMOD is specified in bytes, regardless of the work unit size. The DMA_YMOD register is read/write prior to enabling the channel, but is read-only after enabling the channel.

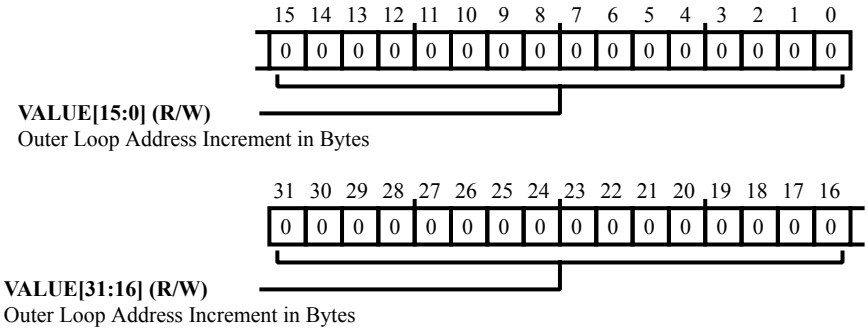


Figure 36-21: DMA_YMOD Register Diagram

Table 36-32: DMA_YMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Outer Loop Address Increment in Bytes. The DMA_YMOD.VALUE bit field contains a signed, two's-complement value.

37 Extended Memory DMA (EMDMA)

The Extended Memory DMA engine can be used in applications that copy data in a non-sequential manner. This includes delay lines, scatter and gather, and circular access types.

NOTE: Previous SHARC processors featured external port DMA. Current SHARC products use EMDMA that can access all memory locations for source and destination DMA operations.

EMDMA Features

EMDMA frees the processor core, allowing it to perform other operations while the data transfers between memories occurs as a background task.

EMDMA has the following features and capabilities.

- Standard mode DMA Transfer
- Chained mode with direction on-the-fly
- Tap list mode (scatter/gather)
- Delay line mode (write to read)
- All the DMA modes can operate in circular fashion
- In circular operation, some modes allow a write back of the index pointer for correct addressing of the next Transfer Control Block (TCB)

EMDMA Functional Description

The following sections provide information on the functional operations and operating modes of EMDMA.

Internal-to-internal DMA

This is accomplished by indexing all external parameter registers with internal addresses.

DMA bursting

DMA supports burst transfers only when the modifier is 1, for any other modifier values, single accesses are performed. The burst size is not user configurable and is chosen by the processor for optimal performance. The maximum burst size is 8 and the minimum is 1.

Transfer control blocks

The structure of a TCB is conceptually the same as that of a traditional linked list. Each TCB has several data values and a pointer to the next TCB. Further, the chain pointer of a TCB may point to itself to continuously re-run the same DMA.

Chain pointer DMA

In chained DMA operations, the processor automatically initializes and then begins another DMA transfer when the current DMA transfer is complete.

ADSP-SC59x EMDMA Register List

The EMDMA controllers support a variety of direct memory access operations which can access any system memory and transfer the entire block of data. A set of registers govern EMDMA operations. For more information on EMDMA functionality, see the EMDMA register descriptions.

Table 37-1: ADSP-SC59x EMDMA Register List

Name	Description
EMDMA_BASE	External Base Address Register
EMDMA_BUFLEN	Circular Buffer Length Register
EMDMA_CHNPTR	Chain Pointer Register
EMDMA_CNT0	Internal Count Register
EMDMA_CNT1	External Count Register
EMDMA_CTL	External Memory DMA Control Register
EMDMA_INDX0	Internal Index Register
EMDMA_INDX1	External Index Register
EMDMA_MOD0	Internal Modifier Register
EMDMA_MOD1	External Modifier Register
EMDMA_TCNT	Delay Line Tap Count Register
EMDMA_TPTR	Tap List Pointer Register

ADSP-SC59x EMDMA Interrupt List

Table 37-2: ADSP-SC59x EMDMA Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
210	EMDMA0_DONE	EMDMA0 Transfer Done	Edge	
211	EMDMA1_DONE	EMDMA1 Transfer Done	Edge	

ADSP-SC59x EMDMA Trigger List

Table 37-3: ADSP-SC59x EMDMA Trigger List Generators

Trigger ID	Name	Description	Sensitivity
27	EMDMA0_DONE	EMDMA0 DMA Done	Edge
28	EMDMA1_DONE	EMDMA1 DMA Done	Edge

Table 37-4: ADSP-SC59x EMDMA Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
None			

DMA Addressing

Besides the traditional internal-to-external addressing type, the DMA module also supports internal-to-internal transfers. This is accomplished by indexing all external parameter registers with internal addresses. The DMA controller recognizes the transfer by addresses and not by an additional control bit setting.

All the DMA addresses given by EMDMA are word-aligned byte addresses. The programming for the index registers is provided in the *Register Descriptions* section.

DMA Burst Transfers

DMA supports burst transfers only when the modifier is 1. For any other modifier values, single accesses are performed.

The burst size is not user-configurable and is chosen by the processor for optimal performance. The maximum burst size is 8 and the minimum is 1.

The EMDMA uses appropriate burst transfer sizes for optimal throughput. For example, if the word count is 15, then 5 bursts are performed with burst sizes of 8+4+1+1+1 transfers.

Transfer Control Block (TCB) Memory Storage

The location of the DMA parameters for the next sequence comes from the chain pointer register that points to the next set of DMA parameters stored in the processor's internal memory. In chained DMA operations, the processor automatically initializes and then begins another DMA transfer when the current DMA transfer is complete. Each new set of parameters is stored in a user-initialized memory buffer or TCB for a chosen peripheral.

Chain Assignment

The structure of a TCB is conceptually the same as that of a traditional linked list. Each TCB has several data values and a pointer to the next TCB. Further, the chain pointer of a TCB may point to itself to continuously rerun the same DMA. The EMDMA reads each word of the TCB and loads it into the corresponding register. The end of the chain (no further TCBs are loaded) is indicated by a TCB with a chain pointer register value of zero.

The address field of the chain pointer registers is only 30 bits wide. If a program writes a symbolic address to bit 30 of the chain pointer there may be a conflict with the `EMDMA_CHNPTR.PCI` bit. Programs should clear the upper bits of the address then AND the `EMDMA_CHNPTR.PCI` bit separately, if needed.

Starting Chain Loading

A DMA sequence is defined as the sum of the DMA transfer from when the parameter registers initialize to when the count register decrements to zero. The EMDMA module has a chaining enable bit (`EMDMA_CTL.CHEN`).

To start the chain, write the internal (channel 0) index address of the first TCB to the chain pointer register (`EMDMA_CHNPTR`). When chaining is enabled, DMA transfers are initiated by writing a memory address to the chain pointer register. This is also an easy way to start a single DMA sequence, with no subsequent chained DMAs.

During TCB chain loading, the EMDMA loads the DMA channel parameter registers with values retrieved from system memory.

The address in the chain pointer register points to the highest address of the TCB. This contains the internal (channel 0) index parameter. This means that if a program declares an array to hold the TCB, the chain pointer register should point to the last location of the array and not to the first TCB location.

Buffered Chain Loading Register

The chain pointer register (`EMDMA_CHNPTR`) is buffered. Before the chain loading starts, the buffer is copied into the chain pointer register and is decremented after each register is loaded.

The chain pointer register can be loaded with a new address at any time during the DMA sequence (`EMDMA_CTL.CHEN` =1). This allows a DMA channel to have chaining status deactivated (chain pointer register = 0x0) until some event occurs that loads the chain pointer register with a non-zero value. Writing all zeros to the address field of the chain pointer register also deactivates chaining for the next TCB.

TCB Storage

The EMDMA supports several types of DMA, resulting in different lengths of TCBs. The TCB size varies from six locations (chained DMA) to 13 locations (delay line DMA). The ***EMDMA TCBs*** table shows the required TCBs for chained DMA.

In the following tables, TCB refers to the start address of the TCB array.

Table 37-5: EMDMA TCBs for Standard DMA

Address	Register
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_MOD1
TCB + 0x2	EMDMA_INDX1
TCB + 0x3	EMDMA_CNT0
TCB + 0x4	EMDMA_MOD0
TCB + 0x5	EMDMA_INDX0

The order the descriptors are fetched with circular buffering enabled is shown in the ***EMDMA TCBs for Standard Circular DMA*** table.

Table 37-6: EMDMA TCBs for Standard Circular DMA

Address	Register
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_BUFLEN
TCB + 0x2	EMDMA_BASE
TCB + 0x3	EMDMA_MOD1
TCB + 0x4	EMDMA_INDX1
TCB + 0x5	EMDMA_CNT0
TCB + 0x6	EMDMA_MOD0
TCB + 0x7	EMDMA_INDX0

For delay line DMA, TCB loading is split into two sequences to improve overall priority. The first TCB loads the write parameters (EMDMA_INDX0 – EMDMA_BUFLEN) and the second loads the read parameters (EMDMA_INDX0 – EMDMA_CHNPTR). This two stage loading is transparent to the application. The order the descriptors are fetched for delay line DMA, as shown in the ***EMDMA TCBs for Delay Line DMA*** table.

Table 37-7: EMDMA TCBs for Delay Line DMA

Address	Register
Delay Line Read	
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_TPTR
TCB + 0x2	EMDMA_TCNT
TCB + 0x3	EMDMA_MOD1
TCB + 0x4	EMDMA_CNT0

Table 37-7: EMDMA TCBs for Delay Line DMA (Continued)

Address	Register
TCB + 0x5	EMDMA_INDX0
Delay Line Write	
TCB + 0x6	EMDMA_BUFLEN
TCB + 0x7	EMDMA_BASE
TCB + 0x8	EMDMA_MOD1
TCB + 0x9	EMDMA_INDX1
TCB + 0xA	EMDMA_CNT0
TCB + 0xB	EMDMA_MOD0
TCB + 0xC	EMDMA_INDX0

The order the descriptors are fetched for scatter/gather DMA with circular buffering disabled is shown in the ***EMDMA TCBs for Scatter/Gather DMA*** table.

Table 37-8: EMDMA TCBs for Scatter/Gather DMA

Address	Register
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_TPTR
TCB + 0x2	EMDMA_TCNT
TCB + 0x3	EMDMA_MOD1
TCB + 0x4	EMDMA_INDX1
TCB + 0x5	EMDMA_CNT0
TCB + 0x6	EMDMA_MOD0
TCB + 0x7	EMDMA_INDX0

The order the descriptors are fetched for scatter/gather DMA with circular buffering enabled is shown in the ***EMDMA TCBs for Circular Scatter/Gather DMA*** table.

Table 37-9: EMDMA TCBs for Circular Scatter/Gather DMA

Address	Register
TCB	EMDMA_CHNPTR
TCB + 0x1	EMDMA_BUFLEN
TCB + 0x2	EMDMA_BASE
TCB + 0x3	EMDMA_TPTR
TCB + 0x4	EMDMA_TCNT

Table 37-9: EMDMA TCBs for Circular Scatter/Gather DMA (Continued)

Address	Register
TCB + 0x5	EMDMA_MOD1
TCB + 0x6	EMDMA_INDX1
TCB + 0x7	EMDMA_CNT0
TCB + 0x8	EMDMA_MOD0
TCB + 0x9	EMDMA_INDX0

EMDMA Operating Modes

This section and the ***EMDMA_CTL Register Bit to Operating Modes*** table show the different DMA modes which can be used. The complete register bit descriptions are in the ***External Memory DMA Control Registers (EMDMA_CTL)***.

Table 37-10: EMDMA_CTL Register Bit to Operating Modes

Bit (Name)	Standard	Chained	Scatter/Gather	Delay Line
Control Bits				
0 (EN)	Valid			
1 (TRAN)	Valid			N/A
2 (CHEN)	Valid			
3 (DLEN)	N/A			Valid
4 (CBEN)	Valid			
5 (DFLSH)	Valid			
6	N/A			
7 (WRBEN)	N/A	Valid	(=0)	(=1)
8 (OFCEN)	Valid			N/A
9 (TLEN)	N/A		Valid	N/A
11–10	N/A			
12 (INTDONE0)	Valid			
15–13	N/A			
Status Bits				
17–16 (DFS)	Valid			
19–18	N/A			
20 (DMAS0)	Valid			
21 (CHS)	N/A	Valid	N/A	Valid

Table 37-10: EMDMA_CTL Register Bit to Operating Modes (Continued)

Bit (Name)	Standard	Chained	Scatter/Gather	Delay Line
22 (TLS)	N/A		Valid	N/A
23 (WBS)	N/A			Valid
24 (DMAS1)	Valid			
25 (DIRS)	Valid			
31–26	N/A			

NOTE: Reading additional bit-field information from N/A (Not applicable) bits does not generate a meaningful result.

A program sets up a DMA channel by writing the transfer's parameters to the DMA parameter registers.

NOTE: The `EMDMA_CNT1` parameter register (read only) is a copy of the `EMDMA_CNT0` register. If `EMDMA_CNT0` is written, the `EMDMA_CNT1` register is updated automatically.

Standard DMA

A standard DMA (once it is configured) transfers data from location A to location B. An interrupt can be used to indicate the end of the transfer. To start a new DMA sequence after the current one is finished, a program must first clear the DMA enable bit (`EMDMA_CTL.EN`), write new parameters to the index, modify, and count registers (parameter registers), then set the DMA enable bit to re-enable DMA.

This DMA type resembles the traditional DMA type to initialize the different internal and external parameters (channel0 and channel1) (index, modify and count) registers and configuration of the DMA control registers.

Circular Buffered DMA

Circular buffered DMA resembles the traditional core DAG circular buffered mode by using registers for circular buffering. In this mode, the DMA needs two additional registers (base and length) to support reads and writes to a circular buffer. The ***Circular Buffering Write DMA*** and ***Circular Buffering Read DMA*** figures illustrate circular buffered DMA, in contrast with the ***Standard Write DMA*** figure.

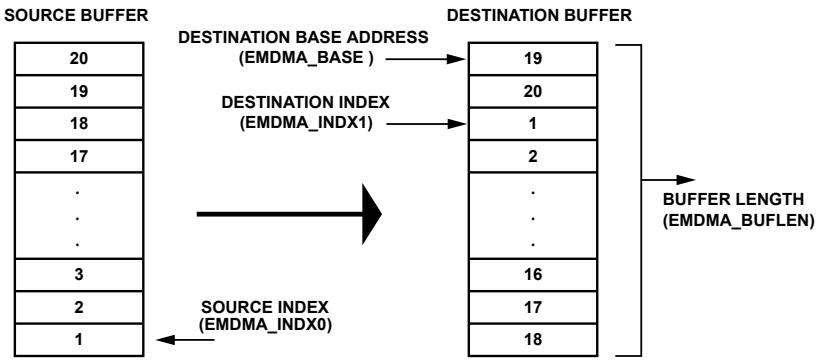


Figure 37-1: Circular Buffering Write DMA

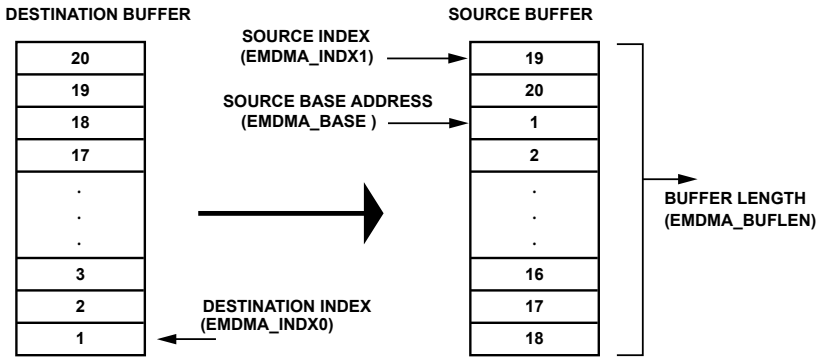


Figure 37-2: Circular Buffering Read DMA

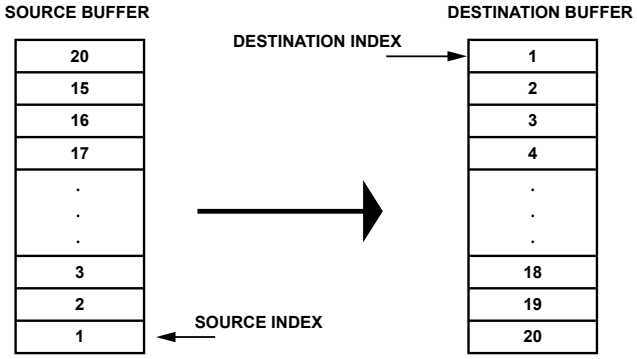


Figure 37-3: Standard Write DMA

NOTE: Circular buffering is available for all operating modes (standard, chained, tap list and delay line DMA).

Chained DMA Mode

Chained DMA sequences are a set of multiple DMA operations, each auto-initializing the next in line. It is used to support automated access by a linked list (repetitive reads and writes to a defined location defined by the individual TCBs). To start a new DMA sequence after the current one is finished, the EMDMA automatically loads new index, modify, and count values pointed to by that channel's `EMDMA_CHNPTR` register. Using chaining, programs can set up consecutive DMA operations and each operation can have different attributes.

DMA data transfers can be set up as continuous or periodic. With chained DMA, the attributes of a specific DMA are stored in internal memory and are referred to as a Transfer Control Block or TCB. Extended Memory DMA loads these attributes in chains for execution. This allows for multiple chains that are finite or infinite.

NOTE: When chaining is enabled, polling should not be used to determine DMA status only because the DMA appears inactive if it is sampled while the next TCB is loading. In such cases where chaining is enabled, along with the polling of DMA status bit, polling of the chaining status bit should also occur to so that the correct status of the DMA is known. For example, the `EMDMA_CTL.CHS` bit should be polled as well as the `EMDMA_CTL.DMAS0` and `EMDMA_CTL.DMAS1` bits when EMDMA is configured in DMA chaining mode.

Data Direction On-the-Fly

A change of external memory data direction for each individual TCB in a chain sequence is allowed.

The EMDMA_CHNPTR.CPDR bit changes the data flow direction. If the EMDMA_CHNPTR.CPDR =0, writes through channel 0 are performed; if EMDMA_CHNPTR.CPDR =1, channel 0 reads are performed. This works similarly to the EMDMA_CHNPTR.PCI bit. The EMDMA_CTL.OFCEN and EMDMA_CTL.CHEN bits must be set (=1) to enable this functionality.

NOTE: If chaining is enabled with the EMDMA_CTL.OFCEN =1, then the EMDMA_CTL.TRAN bit has no effect, and the data direction is determined by the EMDMA_CHNPTR.CPDR bit.

Write Back Circular Index Pointer

Operating the DMA in circular mode requires some special considerations. The index pointer of start address within the buffer may wrap around for the case when IC × IM > EL or it does not finish if IC × IM < EL where:

IC is the value of the EMDMA_INDX0 register

IM is the value of the EMDMA_MOD0 register

EL is the value of the EMDMA_BUFLEN register

In both cases, the TCB start address is no longer valid.

Setting the EMDMA_CTL.WRBEN bit writes (at the end of current TCB block) the current index address + 1 into the TCB memory which is the start address for the next TCB. This bit is only selectable for chained DMA mode. For tap list and delay line modes, this bit is hardwired to 0 or 1.

Scatter/Gather DMA

The purpose of scatter/gather DMA is the transfer of data from/to non-contiguous memory blocks.

The scatter/gather DMA type is a fixed block size scatter/gather DMA that relies on tap list entries in system memory to calculate the (Channel 1 DMA Address) to scatter/gather the DMA. When the DMA direction is Channel 1 write (EMDMA_CTL.TRAN =1) it is a scatter DMA. When (EMDMA_CTL.TRAN =0), it is a gather DMA. This mode also supports chained and circular buffer chained DMAs.

See the ***Scatter DMA (Writes)***, ***Gather DMA (Reads)***, ***Circular Buffering Scatter DMA (Writes)***, and ***Circular Gather DMA (Reads)*** figures.

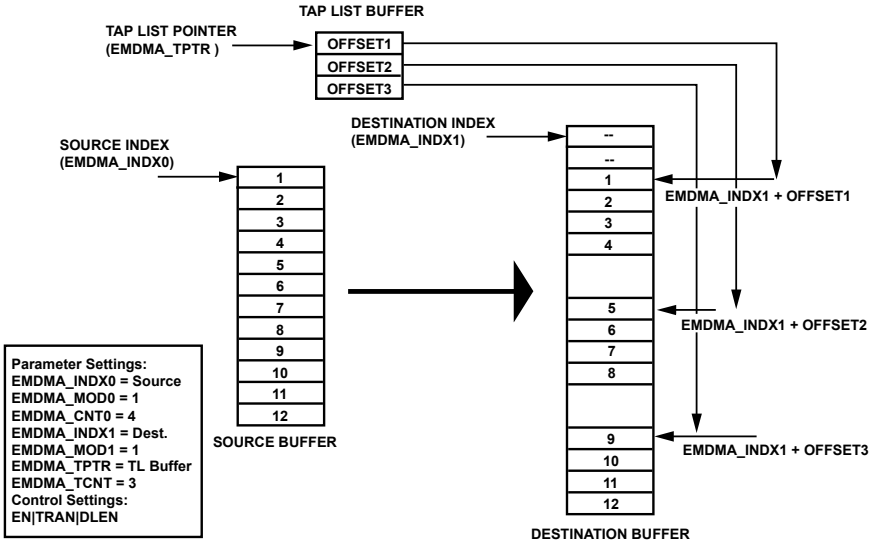


Figure 37-4: Scatter DMA (Writes)

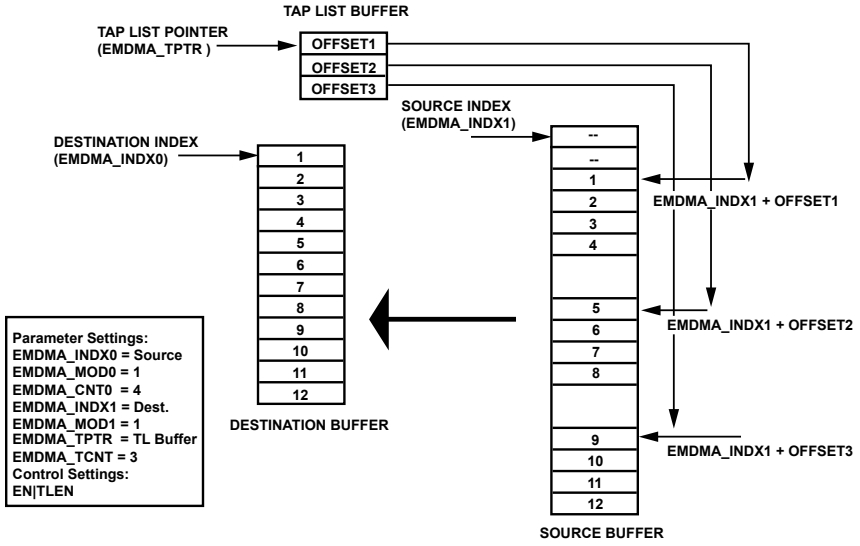


Figure 37-5: Gather DMA (Reads)

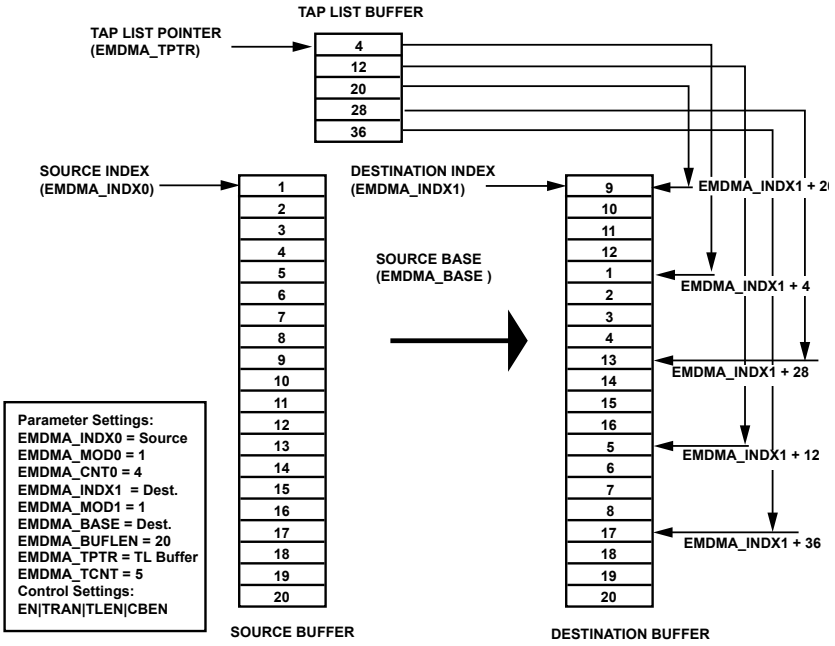


Figure 37-6: Circular Buffering Scatter DMA (Writes)

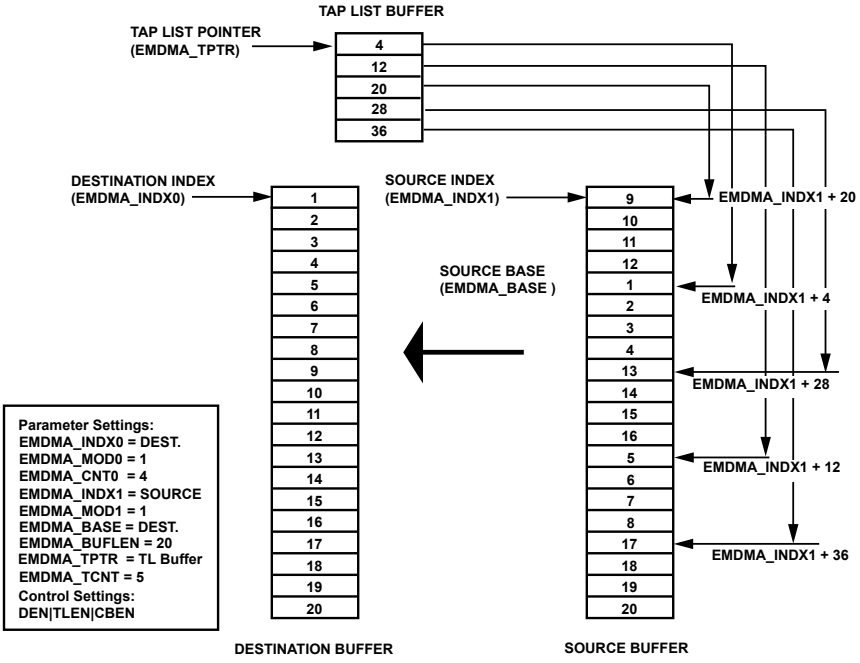


Figure 37-7: Circular Gather DMA (Reads)

For each 32-bit tap read, the Channel 1 read index is shown in the ***Read/Write Index Pre-Modify (Scatter/Gather DMA)*** table. Note that one tap list entry starts multiple reads.

Table 37-11: Read/Write Index Pre-Modify (Scatter/Gather DMA)

Pre-Modify Address Equation **EMDMA_INDX0** + **EMDMA_TPTR** [**EMDMA_TCNT**] + (**EMDMA_MOD1** × **EMDMA_CNT1**)	Result	
	Block Size	Tap
EMDMA_INDX0 + EMDMA_TPTR[0] + EMDMA_MOD1 × 0	N	0
EMDMA_INDX0 + EMDMA_TPTR[0] + EMDMA_MOD1 × 1		
EMDMA_INDX0 + EMDMA_TPTR[0] + EMDMA_MOD1 × 2		
EMDMA_INDX0 + EMDMA_TPTR[0] + EMDMA_MOD1 × N		
EMDMA_INDX0 + EMDMA_TPTR[1] + EMDMA_MOD1 × 0	N	1
EMDMA_INDX0 + EMDMA_TPTR[1] + EMDMA_MOD1 × 1		
EMDMA_INDX0 + EMDMA_TPTR[1] + EMDMA_MOD1 × 2		
EMDMA_INDX0 + EMDMA_TPTR[1] + EMDMA_MOD1 × N		
EMDMA_INDX0 + EMDMA_TPTR[M] + EMDMA_MOD1 × 0	N	M
EMDMA_INDX0 + EMDMA_TPTR[M] + EMDMA_MOD1 × 1		
EMDMA_INDX0 + EMDMA_TPTR[M] + EMDMA_MOD1 × 2		
EMDMA_INDX0 + EMDMA_TPTR[M] + EMDMA_MOD1 × N		

Pre Modified Read/Write Index

For scatter/gather DMA, the tap list modifiers are employed, and the number of taps is determined by the tap list count register (EMDMA_TCNT). The number of sequential reads (block size) from every tap is determined by the internal count register (EMDMA_CNT0), and is the same for every tap. The read/write pointer in external index register (EMDMA_INDX1) serves as the index address for these read/writes.

TL[N] is the first tap list entry in the memory as pointed to by the tap list pointer register (EMDMA_TPTR). The tap list entries are 27-bit signed integers. For each read/write block, the DMA state machine fetches the offset from the tap list. The offset is added to the EMDMA_INDX1 register value to get the start address of the next block. The Channel 1 addresses are circular buffered if circular buffering is enabled (see the ***Circular Buffering Scatter DMA (Writes)*** and ***Circular Buffering Gather DMA (Reads)*** figures in Scatter/Gather DMA).

Once the EMDMA_CNT0 register for the final tap decrements to zero (EMDMA_TCNT and EMDMA_CNT0 are zero), then the tap list DMA access is complete, and the DMA completion interrupt is generated (when chaining is enabled the interrupt depends on the EMDMA_CHNPTR.PCI bit setting).

The write back mode (EMDMA_CTL.WRBEN bit) is hardwired to zero for tap list based DMA (as the addressing is pre-modify, and therefore the EMDMA_INDX1 value coincides with the TCB value even at the end of the DMA).

Delay Line DMA

Delay line DMA is used to support reads and writes to delay line buffers with limited core interaction. In this sense, delay line DMA is a quantity of integrated writes followed by reads from channel 1 (a delay line DMA access). Delay line DMA is described in the following sections.

NOTE: Delay line DMA can only operate by using chained DMA mode (EMDMA_CTL.CHEN bit set). In order to use delay line DMA for a single DMA sequence, initialize the EMDMA_CHNPTR register to zero in the TCB.

NOTE: Delay line DMA can be used in any system memory.

The delay line DMA operation follows these steps:

1. Load the first half of TCB for writing (seven parameters).
2. DMA writes to the delay line buffer until IC = 0.
3. Update the EI index pointer if circular mode is enabled.
4. Load the second half of TCB for reading (six parameters).
5. DMA tap based reads from the delay line buffer until RC = 0.

   Jump to step 1.

Writes to delay line. The number of writes is determined by the EMDMA_CNT0 register. The data is fetched from the EMDMA_INDX0 register and the EMDMA_MOD0 register is used as the internal modifier. The EMDMA_INDX1 register serves as the external index and is incremented by the EMDMA_MOD1 register after each write. These writes are circular buffered if circular buffering is enabled. See the ***Write to Delay Line Buffer*** figure.

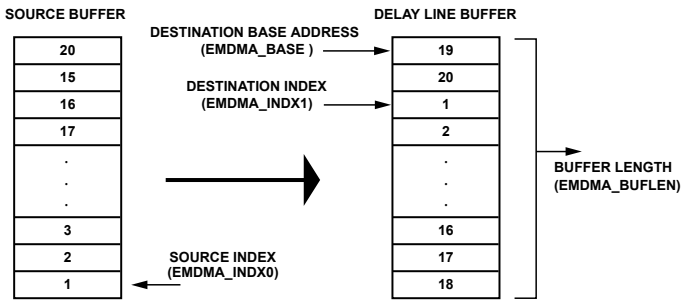


Figure 37-8: Write to Delay Line Buffer

When the writes are complete, (EMDMA_CNT0 =0) the EMDMA_INDX1 register, which serves as the write pointer of the delay line, is written back (EMDMA_CTL.WRBEN is hardwired to 1) to the TCB location from where it was fetched.

Reads from the delay line. For reads, the tap list (TL) modifiers are used, and the number of reads is determined by the EMDMA_CNT0 register. The write pointer in the EMDMA_INDX0 register serves as the index address for these reads (reads start from where writes end). The EMDMA_INDX0 register, along with tap list modifiers, are used in a pre-modify addressing mode to create the external address for the reads. Therefore, for each read, the DMA controller fetches the external modifier (EMDMA_TCNT register) from the tap list and the reads are circular buffered (if enabled). See the ***Read From Delay Line Buffer*** figure.

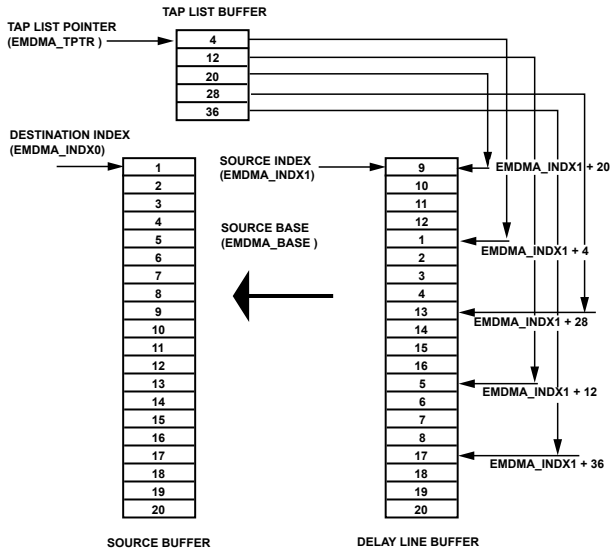


Figure 37-9: Read From Delay Line Buffer

Pre-Modified Read Index

Note that TL[N] is the first tap list entry in memory pointed to by the tap list pointer register (EMDMA_TPTR). Tap list entries are 27-bit signed integers. Therefore, for each read-block, the DMA state machine fetches the offset external modifier from the tap list. The reads are circular buffered if circular buffering is enabled.

NOTE: The channel 1 DMA address generation follows pre-modify addressing for reads in delay line DMA and therefore the EMDMA_INDX1 register values are not updated. Also the EMDMA_MOD1 register does not have any effect during these delay line reads. Once the read count completes, the EMDMA_CNT0 register decrements to zero (both EMDMA_CNT0 and EMDMA_TCNT are zero) for the final tap. Finally, the delay line DMA access completes, and the DMA completion interrupt is generated. IWhen chaining is enabled, the interrupt is dependent on the EMDMA_CHNPTR.PCI bit setting. The delay line DMA can only be initialized using the TCB. To use the delay line DMA for a single DMA sequence, initialize the EMDMA_CHNPTR register to zero in the TCB.

For each 32-bit tap read, the channel 1 read index is shown in the ***Read/Write Index Pre-Modify (Scatter/Gather DMA)*** table. Note that one tap list entry starts multiple reads.

Table 37-12: Read/Write Index Pre-Modify (Scatter/Gather DMA)

Pre-Modify Address Equation	Result	
EMDMA_INDX1 + **EMDMA_TPTR**[**EMDMA_TCNT**] + (**EMDMA_MOD1** × **EMDMA_CNT1**)	Block Size	Tap
EMDMA_INDX1 + EMDMA_TPTR[0] + EMDMA_MOD1 × 0	N	0
EMDMA_INDX1 + EMDMA_TPTR[0] + EMDMA_MOD1 × 1		
EMDMA_INDX1 + EMDMA_TPTR[0] + EMDMA_MOD1 × 2		
EMDMA_INDX1 + EMDMA_TPTR[0] + EMDMA_MOD1 × N		

Table 37-12: Read/Write Index Pre-Modify (Scatter/Gather DMA) (Continued)

Pre-Modify Address Equation **EMDMA_INDX1** + **EMDMA_TPTR**[**EMDMA_TCNT**] + (**EMDMA_MOD1** × **EMDMA_CNT1**)	Result	
	Block Size	Tap
EMDMA_INDX1 + EMDMA_TPTR[1] + EMDMA_MOD1 × 0	N	1
EMDMA_INDX1 + EMDMA_TPTR[1] + EMDMA_MOD1 × 1		
EMDMA_INDX1 + EMDMA_TPTR[1] + EMDMA_MOD1 × 2		
EMDMA_INDX1 + EMDMA_TPTR[1] + EMDMA_MOD1 × N		
EMDMA_INDX1 + EMDMA_TPTR[M] + EMDMA_MOD1 × 0	N	M
EMDMA_INDX1 + EMDMA_TPTR[M] + EMDMA_MOD1 × 1		
EMDMA_INDX1 + EMDMA_TPTR[M] + EMDMA_MOD1 × 2		
EMDMA_INDX1 + EMDMA_TPTR[M] + EMDMA_MOD1 × N		

ADSP-SC59x EMDMA Register Descriptions

Extended Memory DMA (EMDMA) contains the following registers.

Table 37-13: ADSP-SC59x EMDMA Register List

Name	Description
EMDMA_BASE	External Base Address Register
EMDMA_BUFLEN	Circular Buffer Length Register
EMDMA_CHNPTR	Chain Pointer Register
EMDMA_CNT0	Internal Count Register
EMDMA_CNT1	External Count Register
EMDMA_CTL	External Memory DMA Control Register
EMDMA_INDX0	Internal Index Register
EMDMA_INDX1	External Index Register
EMDMA_MOD0	Internal Modifier Register
EMDMA_MOD1	External Modifier Register
EMDMA_TCNT	Delay Line Tap Count Register
EMDMA_TPTR	Tap List Pointer Register

External Base Address Register

The EMDMA_BASE register contains the external base address of the Delay Line buffer. This is used for maintaining circular buffered read/writes to the delay line.

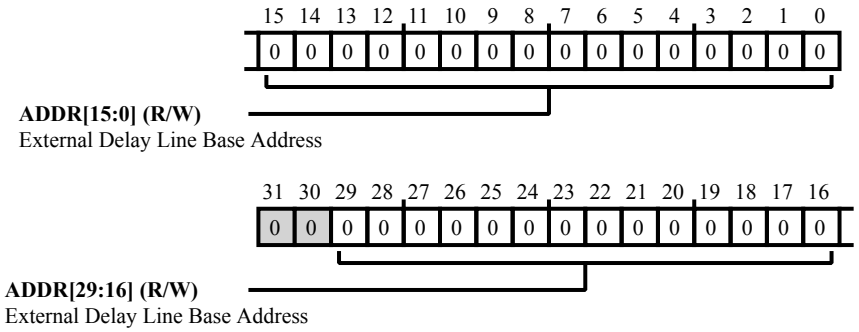


Figure 37-10: EMDMA_BASE Register Diagram

Table 37-14: EMDMA_BASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	ADDR	External Delay Line Base Address. The EMDMA_BASE.ADDR bit field contains the external base address of the Delay Line buffer.

Circular Buffer Length Register

The EMDMA_BUFLEN register holds the circular buffer length for the Delay-line DMA.

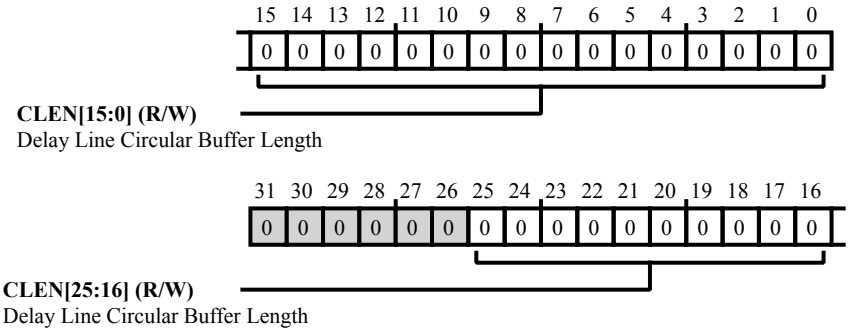


Figure 37-11: EMDMA_BUFLEN Register Diagram

Table 37-15: EMDMA_BUFLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:0 (R/W)	CLEN	Delay Line Circular Buffer Length. The EMDMA_BUFLEN.CLEN bit field holds the circular buffer length for the Delay-line DMA.

Chain Pointer Register

The EMDMA_CHNPTR register contains the address of the next descriptor in memory when the EMDMA_CTL.CHEN bit =1. This register also has bits to change DMA directions for each descriptor and the PCI bit. Note that the lower 30-bits of this register are to be written with 30 MSB's of the Word Aligned Byte addresses of the corresponding next descriptor address.

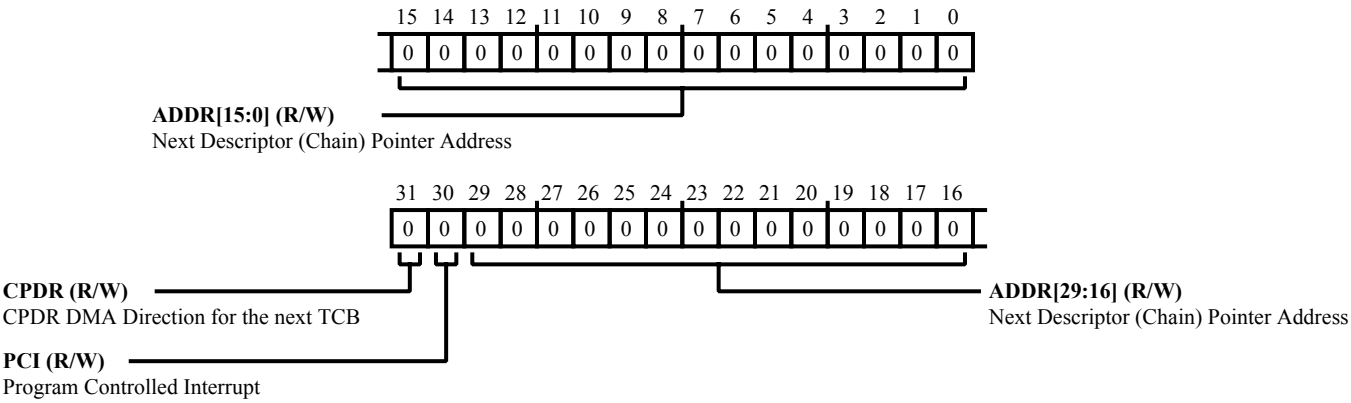


Figure 37-12: EMDMA_CHNPTR Register Diagram

Table 37-16: EMDMA_CHNPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	CPDR	CPDR DMA Direction for the next TCB. The EMDMA_CHNPTR.CPDR bit configures whether the DMA is a write to internal memory or a read from internal memory. Note that this setting is applicable only if EMDMA_CTL.OFCEN =1 and is not applicable for Delay Line DMA.	
		0	Write to Channel 0 (Channel 1 reads)
		1	Read from Channel 0 (Channel 1 Writes)
30 (R/W)	PCI	Program Controlled Interrupt. The EMDMA_CHNPTR.PCI bit PCI sets whether an interrupt is generated after the current TCB or if no interrupt is generated. (Only affects DMA if chaining is enabled)	
		0	Enable DMA Channel interrupt to occur at the completion of the entire DMA chained transfer
		1	Enable DMA Channel interrupt to occur at the completion of current DMA sequence
29:0 (R/W)	ADDR	Next Descriptor (Chain) Pointer Address. The EMDMA_CHNPTR.ADDR bit field provides the address of the next descriptor in memory.	

Internal Count Register

The EMDMA_CNT0 register contains the number of words to be transferred for channel 0 DMA. Note: If Delay Line DMA is enabled then the EMDMA_CNT0 register serves as the count register for the Delay Line writes.

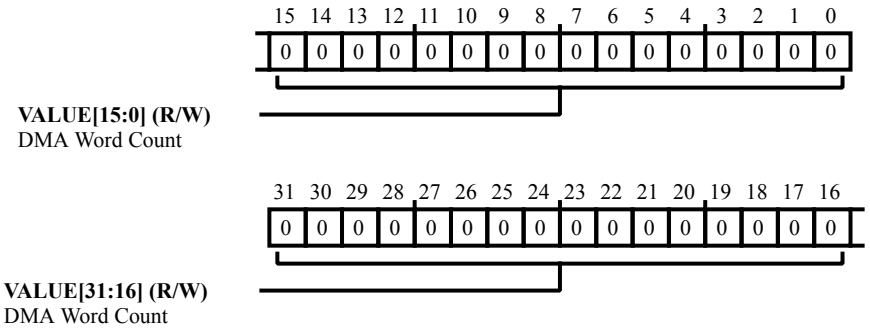


Figure 37-13: EMDMA_CNT0 Register Diagram

Table 37-17: EMDMA_CNT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	DMA Word Count. The EMDMA_CNT0.VALUE bit field is the DMA word count.

External Count Register

The EMDMA_CNT1 register contains the number of words to be transferred for channel 1 DMA. Note: If Delay Line DMA is enabled then the EMDMA_CNT1 register serves as the count register for the Delay Line writes.

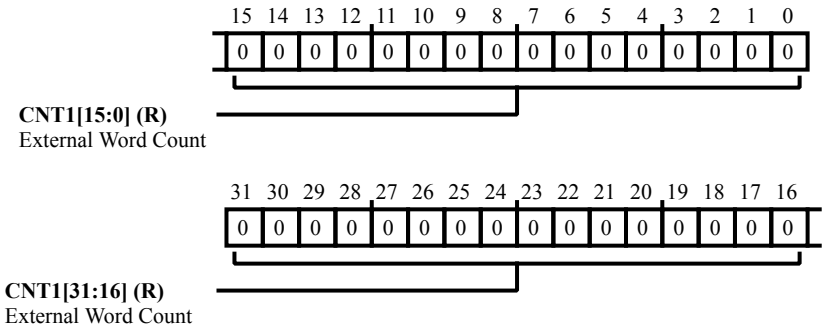


Figure 37-14: EMDMA_CNT1 Register Diagram

Table 37-18: EMDMA_CNT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	CNT1	External Word Count. The EMDMA_CNT1.CNT1 bit field is the external word count.

External Memory DMA Control Register

The EMDMA_CTL register contains bits that enable and configure EMDMA and indicate DMA transfer status.

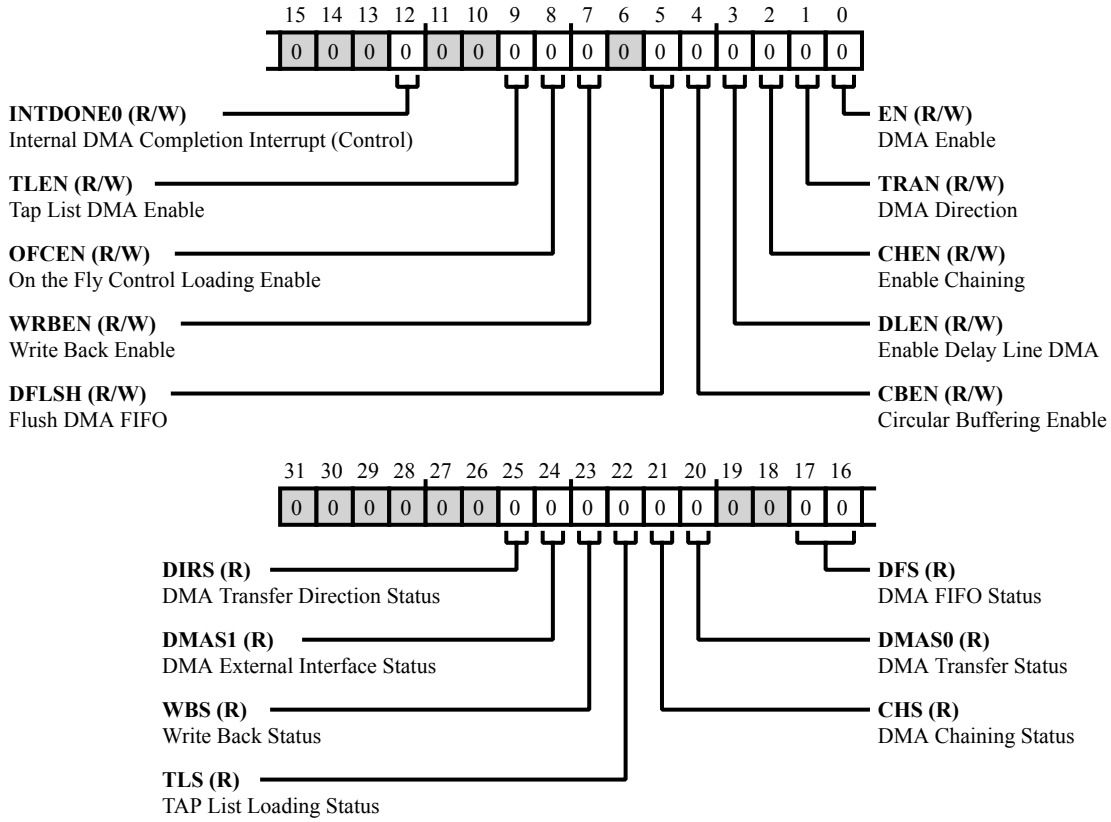


Figure 37-15: EMDMA_CTL Register Diagram

Table 37-19: EMDMA_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
25 (R/NW)	DIRS	DMA Transfer Direction Status. The EMDMA_CTL.DIRS bit provides the DMA transfer status direction. This is useful for delay line DMA where the transfer direction changes with the state of the DMA state machine. For standard DMA, the EMDMA_CTL.DIRS bit reflects the state of the EMDMA_CTL.TRAN bit.	
		0	DMA direction is Channel 1 Reads
		1	DMA direction is Channel 1 Writes

Table 37-19: EMDMA_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/NW)	DMAS1	DMA External Interface Status. The `EMDMA_CTL.DMAS1` bit provides the DMA channel 1 transfer status.	
		0	Channel 1 DMA does not have any access pending
		1	Channel 1 DMA access are pending
23 (R/NW)	WBS	Write Back Status. The `EMDMA_CTL.WBS` bit provides the delay line write pointer write back status.	
		0	Write pointer write back is not active
		1	Write pointer write back is active
22 (R/NW)	TLS	TAP List Loading Status. The `EMDMA_CTL.TLS` bit provides the DMA tap list loading status.	
		0	TAP list loading is not active
		1	TAP list loading is active
21 (R/NW)	CHS	DMA Chaining Status. The `EMDMA_CTL.CHS` bit provides the DMA chaining status.	
		0	DMA chain loading is not active
		1	DMA chain loading is active
20 (R/NW)	DMAS0	DMA Transfer Status. The `EMDMA_CTL.DMAS0` bit provides the DMA channel 0 transfer status.	
		0	DMA idle
		1	DMA in progress
17:16 (R/NW)	DFS	DMA FIFO Status. The `EMDMA_CTL.DFS` bit field provides the DMA FIFO status.	
		0	FIFO EMPTY
		1	FIFO Partially Full
		2	Reserved
		3	FIFO Full

Table 37-19: EMDMA_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	INTDONE0	Internal DMA Completion Interrupt (Control). The EMDMA_CTL.INTDONE0 bit configures when the DMA complete interrupt is generated. The EMDMA_CTL.INTDONE0 =1 setting is provided for backward compatibility with older SHARC processors.	
		0	Interrupt on access completion (Channel 0 or Channel 1 DMA completion)
		1	Interrupt on Channel 0 DMA completion.
9 (R/W)	TLEN	Tap List DMA Enable. The EMDMA_CTL.TLEN bit enables scatter/gather tap list DMA.	
		0	Disables the tap list based scatter/gather DMA
		1	Enables the tap list based scatter/gather DMA
8 (R/W)	OFCEN	On the Fly Control Loading Enable. The control bits in EMDMA_CHNPTR register are used to describe the next TCB behavior if the EMDMA_CTL.OFCEN bit is set and therefore the DMA controls can be changed from TCB to TCB. 0 = disables the control bits in the EMDMA_CHNPTR register 1 = Enables the control bits in the EMDMA_CHNPTR register. If chaining is enabled with EMDMA_CTL.OFCEN bit set then the EMDMA_CTL.TRAN bit has no effect, and direction is determined by EMDMA_CHNPTR.CPDR bit.	
7 (R/W)	WRBEN	Write Back Enable. The EMDMA_CTL.WRBEN bit enables write back of the EIEP register after reads and or writes.Write back is automatically enabled for delay line DMA. WRBEN is applicable only if chaining is enabled (EMDMA_CTL.CHEN =1).	
5 (R/W)	DFLSH	Flush DMA FIFO. The EMDMA_CTL.DFLSH bit flushes the DMA FIFO. The buffer is only flushed if this bit is set. It can be set with the enable bit. Setting this bit also clears the EMDMA_CTL.DFS bit.	
4 (R/W)	CBEN	Circular Buffering Enable. The EMDMA_CTL.CBEN bit enables circular buffering. Circular buffering can be used with normal DMA as well, if circular buffering is enabled with chaining for normal DMA then ELEP and EBEP should be part of the TCB.	
		0	Disables circular buffering with delay line DMA
		1	Enables circular buffering with delay line DMA

Table 37-19: EMDMA_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	DLEN	Enable Delay Line DMA. The `EMDMA_CTL.DLEN` bit enables delay line DMA. This bit is applicable only if the `EMDMA_CTL.CHEN` bit =1.	
		0	Delay-line DMA disabled
		1	Delay-line DMA enabled
2 (R/W)	CHEN	Enable Chaining. The `EMDMA_CTL.CHEN` bit enables DMA chaining.	
		0	Chaining disabled
		1	Chaining enabled
1 (R/W)	TRAN	DMA Direction. The `EMDMA_CTL.TRAN` bit determines the DMA data direction. Note: If delay line DMA is enabled then this bit does not have any effect. For delay line DMA, transfer direction depends on the state of delay line transfers. For Internal-Internal or External-External DMA this bit has to be set.	
		0	Write through Channel 0 (Channel 1 reads)
		1	Read from Channel 0(Channel 1 writes)
0 (R/W)	EN	DMA Enable. The `EMDMA_CTL.EN` bit enables DMA.	
		0	Disable DMA
		1	Enable DMA

Internal Index Register

The EMDMA_INDX0 register contains the start address of the buffer for Channel 0 DMA. Note: For delay line DMA the EMDMA_INDX0 register serves as the delay line write index which is the start address of the channel 0 DMA buffer for the channel 1 write data.

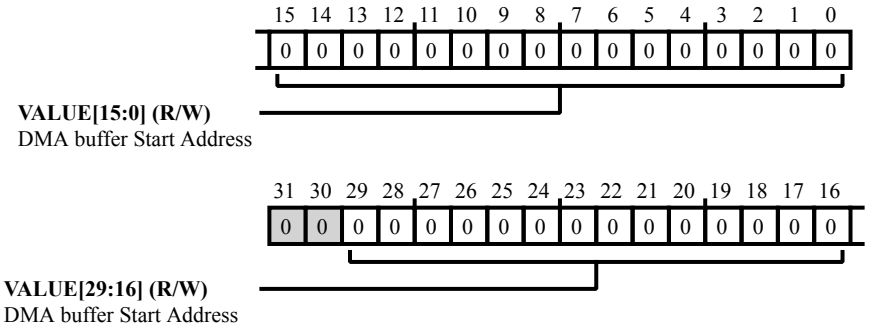


Figure 37-16: EMDMA_INDX0 Register Diagram

Table 37-20: EMDMA_INDX0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	DMA buffer Start Address. The EMDMA_INDX0.VALUE bit field is written with the 30 MSBs of the Word Aligned Byte addresses.

External Index Register

The EMDMA_INDX1 register contains the start address of the buffer for channel 1 DMA.

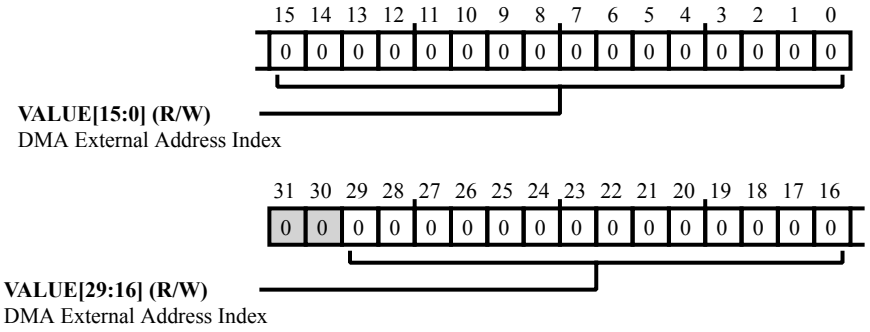


Figure 37-17: EMDMA_INDX1 Register Diagram

Table 37-21: EMDMA_INDX1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	DMA External Address Index. The EMDMA_INDX1.VALUE bit field is the DMA external address index.

Internal Modifier Register

The EMDMA_MOD0 register contains the channel 0 DMA address modifier.

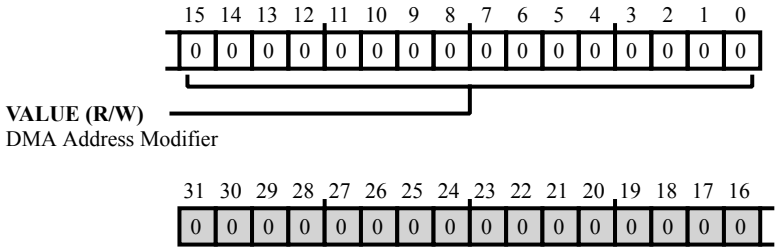


Figure 37-18: EMDMA_MOD0 Register Diagram

Table 37-22: EMDMA_MOD0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	DMA Address Modifier. The EMDMA_MOD0.VALUE bit field is the DMA address modifier.

External Modifier Register

The EMDMA_MOD1 register contains the external (channel 1 DMA) address modifier.

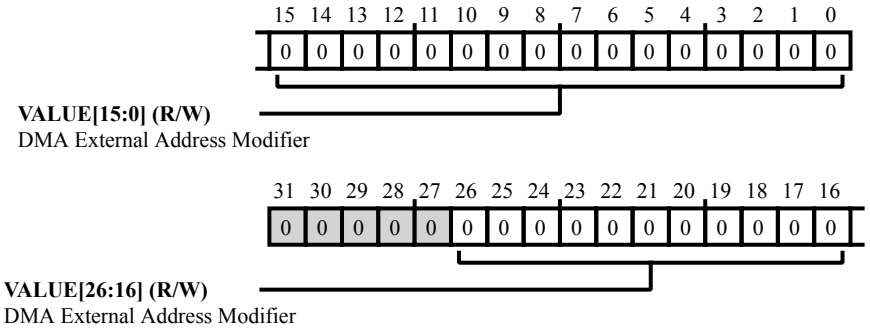


Figure 37-19: EMDMA_MOD1 Register Diagram

Table 37-23: EMDMA_MOD1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
26:0 (R/W)	VALUE	DMA External Address Modifier. The EMDMA_MOD1.VALUE bit field is the DMA external address modifier.

Delay Line Tap Count Register

The EMDMA_TCNT register is the tap count register for Delay Line DMA. This register holds the length of the tap list (the number of taps). The total number of words read from the delay line is equal to the EMDMA_TCNT (tap count) multiplied by the EMDMA_CNT1 (read block size).

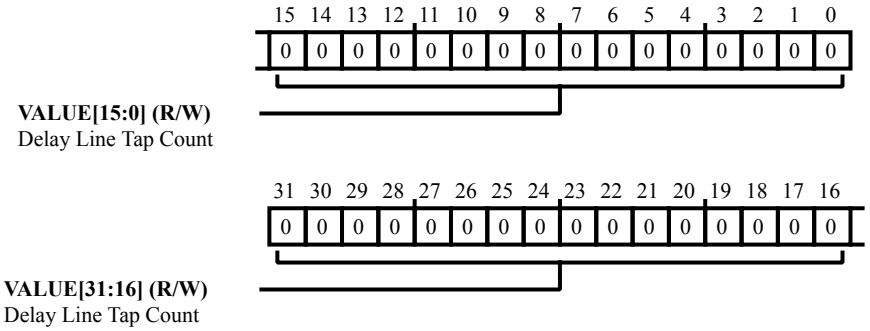


Figure 37-20: EMDMA_TCNT Register Diagram

Table 37-24: EMDMA_TCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Delay Line Tap Count. The EMDMA_TCNT.VALUE bit field is the tap count register for Delay Line DMA.

Tap List Pointer Register

The EMDMA_TPTR register holds the address of an array in memory which holds offsets to be used when accessing a Delay-line in system memory. The offset represents the first address of each read block. Note that the lower 30-bits of this register are to be written with 30 MSBs of the Word Aligned Byte address of the Array.

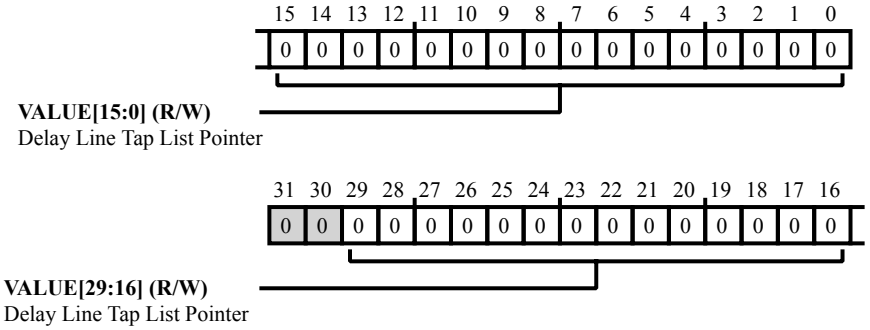


Figure 37-21: EMDMA_TPTR Register Diagram

Table 37-25: EMDMA_TPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Delay Line Tap List Pointer. The EMDMA_TPTR.VALUE bit field contains the offsets to be used when accessing a Delay-line in system memory.

38 Cyclic Redundancy Check (CRC)

The CRC peripheral performs the cyclic redundancy check (CRC) of the block of data that is presented to the peripheral. The peripheral provides a means to verify periodically the integrity of the system memory, the contents of memory-mapped registers (MMRs), or communication message objects. It is based on a CRC32 engine that computes the signature of 32-bit data presented to the peripheral.

The dedicated hardware compares the calculated signature of the operation to a pre-loaded expected signature. If the two signatures fail to match, the peripheral generates an error.

The source channel of the memory-to-memory DMA channels can provide data. The CRC optionally forwards data to memory through the destination DMA channel. Alternatively, the peripheral supports data presented by any qualified controller of the CRC peripheral bus.

The CRC peripheral implements a reduced table-look-up algorithm to compute the signature of the data. The CRC uses a programmable 32-bit CRC polynomial to generate the look-up table (LUT) contents automatically.

More CRC peripheral modes allow for initializing large memory sections with a constant value, or for verifying that sections of memory are equal to a constant value.

NOTE: CRC is supported by MDMA0, MDMA1, MDMA4, and MDMA5 channels only.

CRC Features

The CRC peripheral supports a number of key features.

- Memory scan modes for memory verification
- Memory transfer modes for on-the-fly CRC calculations while transferring data from one memory to another
- A programmable 32-bit CRC polynomial with automatic LUT generation
- Data mirroring options

The CRC module also includes the following features.

- CRC checksum computation and comparison modes
- 32-bit programmable CRC polynomial with bit reverse option
- Automatic look-up table (LUT) generation

- Data mirroring options for endian and reflected polynomial cases
- Automatic clear and preset of results
- Fault and error interrupt reporting
- DMA and MMR based operation

Because the CRC module is closely tied to memory-to-memory DMA (MDMA) channel pairs, the use cases include the following features.

- Memory scan mode with CRC compute or compare
- Memory transfer mode with CRC compute or compare
- Memory fill operation with 32-bit data patterns
- Memory verify operation
- MMR write access to FIFO of destination DMA
- MMR read access to FIFO of source DMA
- Profiting from advanced DMA features, like descriptor mode and bandwidth control or monitor

CRC Functional Description

The CRC peripheral supports a number of modes of operation that allow for the initialization and verification of regions of memory. The peripheral supports efficient memory-fill and verification operations on regions of memory with or against a constant value. These modes of operation do not require the CRC engine to calculate a signature. Other modes of operation allow for the calculation of CRC signature and verification for a memory region. The modes allow for on-the-fly CRC calculation when performing memory-to-memory DMA transfers from one memory region to another.

To minimize the need for core accesses, the peripheral interfaces with one or more (depending on processor features) memory-to memory DMA (MDMA) channels. This connectivity permits flexible configuration, in which data can be written-to or read-from the peripheral using DMA transactions, core transactions, or a combination of both.

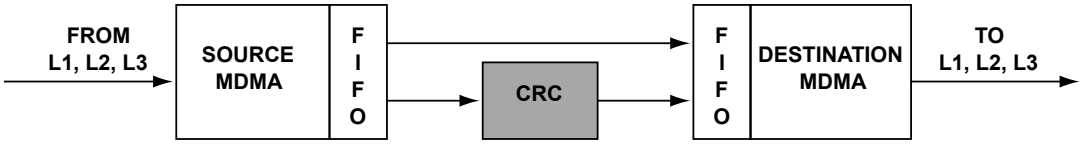


Figure 38-1: Memory Flow

ADSP-SC59x CRC Register List

The Cyclic Redundancy Check (CRC) unit includes the data comparison, polynomial operation, and look up table generation features needed for CRC operation. The CRC provides CRC protection as specified by many functional safety requirements. This unit facilitates the system software's ability to periodically check the correctness of the

code/data available in memory. A set of registers govern CRC operations. For more information on CRC functionality, see the CRC register descriptions.

Table 38-1: ADSP-SC59x CRC Register List

Name	Description
CRC_COMP	Data Compare Register
CRC_CTL	Control Register
CRC_DCNT	Data Word Count Register
CRC_DCNTCAP	Data Count Capture Register
CRC_DCNTRLD	Data Word Count Reload Register
CRC_DFIFO	Data FIFO Register
CRC_FILLVAL	Fill Value Register
CRC_INEN	Interrupt Enable Register
CRC_INEN_CLR	Interrupt Enable Clear Register
CRC_INEN_SET	Interrupt Enable Set Register
CRC_POLY	Polynomial Register
CRC_RESULT_CUR	CRC Current Result Register
CRC_RESULT_FIN	CRC Final Result Register
CRC_STAT	Status Register

ADSP-SC59x CRC Interrupt List

Table 38-2: ADSP-SC59x CRC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
190	CRC0_ERR	CRC0 Error	Level	
191	CRC1_ERR	CRC1 Error	Level	
196	CRC0_DCNTEXP	CRC0 Data Count Expiration	Level	
197	CRC1_DCNTEXP	CRC1 Data Count Expiration	Level	
198	CRC2_ERR	CRC2 Error	Level	
199	CRC3_ERR	CRC3 Error	Level	
204	CRC2_DCNTEXP	CRC2 Data Count Expiration	Level	
205	CRC3_DCNTEXP	CRC3 Data Count Expiration	Level	

CRC Definitions

To make the best use of the CRC, it is useful to understand the following terms.

CRC

Acronym for Cyclic Redundancy Check. An error detection code that can detect changes within a block of data.

CRC Polynomial

The 32-bit polynomial used by the CRC engine to generate the look-up table required for the CRC implementation

LUT

Acronym for the Look-up Table. The look-up table is automatically generated from the supplied 32-bit CRC polynomial.

DMA

Acronym for Direct Memory Access. Used to describe a data transfer that takes place through a DMA channel allowing data distribution around a system without intervention from the core.

MDMA

Acronym for Memory-To-Memory DMA transfer that often requires the use of two DMA channels to transfer data from one memory region to another memory region. One DMA channel is configured as a source channel and the second as a destination channel.

CRC Block Diagram

The ***CRC Block Diagram*** shows the functional block diagram of the CRC. The following sections describe the blocks.

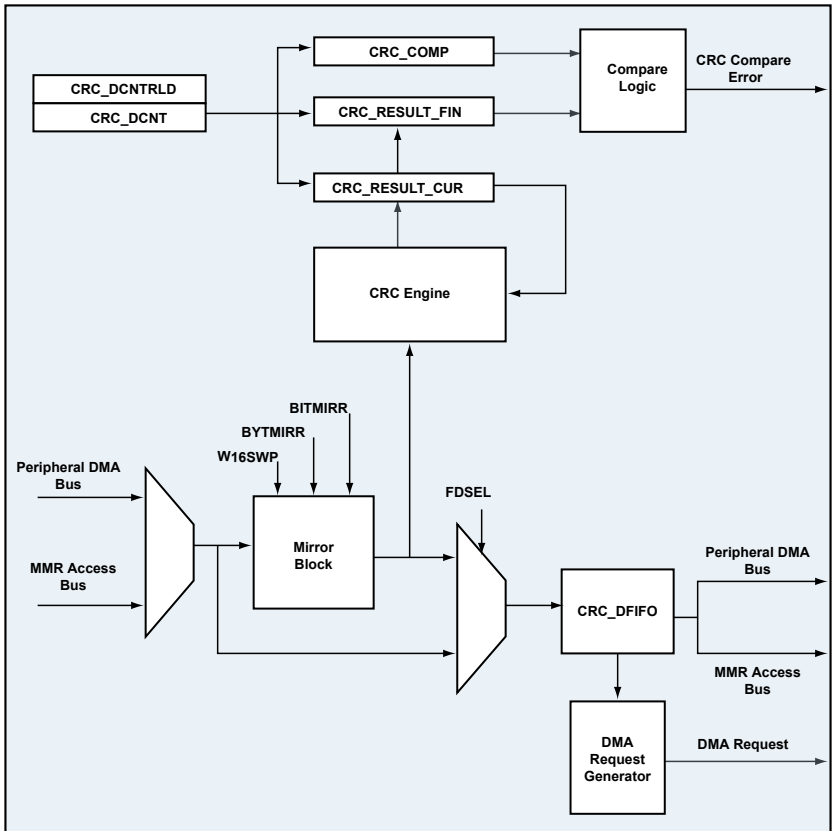


Figure 38-2: CRC Block Diagram

Peripheral DMA Bus

The CRC peripheral provides both an incoming and outgoing datapath to the peripheral DMA bus. The MDMA source channel is interfaced to the incoming datapath providing data to the CRC peripheral. For memory transfer and data fill modes, the CRC uses the MDMA destination channel to either output the data from the CRC FIFO or use the data for the fill operation.

MMR Access Bus

The core uses the MMR access bus to access all the memory-mapped registers of the peripheral for configuration, status, and debug purposes. The core can also use the MMR access bus to feed data to the CRC peripheral or read data from the FIFO of the CRC peripheral. The CRC operation is an alternative to the DMA channel operation to read data from the FIFO.

Data received by MMR writes can transfer to destination DMA. Similarly, data received by source DMA can be output through the MMR interface. Optionally, intermediate results can be made available to the MMR interface.

Mirror Block

The mirror block individually controls bit-reversing of the polynomial, the computation results, and the expected result. Bit mirroring, byte mirroring, word swapping, and any combination of these operations can control endian and the reflection of processed data.

Data FIFO

The CRC data FIFO is a 32-bit-wide 4-entry FIFO. The FIFO is accessible to both the peripheral DMA bus and the MMR access bus. The FIFO status is accessible from the CRC_STAT register.

DMA Request Generator

The DMA request generator is responsible for granting incoming DMA requests from the source DMA channel and issuing outgoing DMA requests to the destination DMA channel.

CRC Engine

The CRC engine is a 32-bit CRC engine that implements the reduced table look-up scheme. The CRC engine provides support for a user-programmable 32-bit polynomial that the CRC uses to load the look-up table parameters required for the CRC calculation. The CRC engine is a single cycle implementation operating on 32 bits of data per cycle.

Compare Logic

The compare logic takes the final CRC signature and compares it to the expected CRC signature, generating a CRC compare error when the signatures do not match. A compare error can flag a system fault.

CRC Architectural Concepts

A 32-bit polynomial is required before calculation of the CRC signature can occur. The CRC uses the polynomial to generate the contents of an internal look-up table that the reduced table look-up implementation requires. The look-up table is automatically generated when the polynomial is written. It must be initialized prior to any operation that requires the use of the CRC engine.

The mirror block logic can manipulate the data presented to the CRC engine before the CRC uses the data in the calculation of the CRC signature. The data mirror operation is configurable to allow for bit reversing, byte reversing, and 16-bit word swapping operations on the incoming data. For memory transfer compute-and-compare operations, programs can configure the peripheral to output the data in the same form in which it was received. Or, the operation can output the mirrored data in the same manner that it is presented to the CRC engine.

While the CRC peripheral is in operation, the status of the FIFO is continually updated and reflected in the CRC_STAT register. The FIFO status is required for core-based accesses to the CRC peripheral. The status indicates when:

- The CRC peripheral can receive data

- Data is available for reading from the FIFO
- The result of the CRC_RESULT_CUR register has been updated

The status of the CRC_RESULT_CUR register indicates that the current CRC calculation has completed and the result is available.

Look-up Table

The look-up table consists of a set of sixteen 32-bit registers that hardware populates automatically when a write access takes place to the CRC_POLY register. 16 clock cycles are required to generate all 16 look-up table entries. The status of the process for generating the look-up table is reflected in CRC_STAT.LUTDONE allowing for software to poll on the completion of the event or for generation of an interrupt.

NOTE: Hardware must populate the look-up table before any operation using the CRC peripheral can take place, even if the operation does not use the CRC engine. The peripheral does not issue any data requests until the table generation process is complete. In addition, the CRC_STAT.IBR field, that indicates the input buffer status as required for core-based transfers, is only valid upon completion of the process for generating the look-up table.

Data Mirroring

The data mirror block can be configured to manipulate the incoming data before the data passes to the CRC engine and, optionally, to the FIFO. This configuration allows the peripheral to handle various forms of endianness and to function with reflected polynomials.

There are three configuration bits that control the data mirroring process: CRC_CTL.BITMIRR, CRC_CTL.BYTMIRR, and CRC_CTL.W16SWP. The ***Data Mirroring Options*** table details how these options affect the incoming data and the output generated by the mirror block.

Table 38-3: Data Mirroring Options

W16SWP	BYTMIRR	BITMIRR	Output Data
0	0	0	Dout[31:0] = Din[31:0]
0	0	1	Dout[31:0] = Din[24:31],Din[16:23],Din[8:15],Din[0:7]
0	1	0	Dout[31:0] = Din[7:0],Din[15:8],Din[23:16],Din[31:24]
0	1	1	Dout[31:0] = Din[0:7],Din[8:15],Din[16:23],Din[24:31]
1	0	0	Dout[31:0] = Din[15:0], D[31:16]
1	0	1	Dout[31:0] = Din[8:15],Din[0:7], Din[24:31],Din[16:23]
1	1	0	Dout[31:0] = Din[23:16],Din[31:24], Din[7:0],Din[15:8]
1	1	1	Dout[31:0] = Din[16:23],Din[24:31], Din[0:7],Din[8:15]

When the CRC is configured to operate in the memory transfer compute-and-compare mode, the bit-reversed output data can be written to the FIFO. This feature is controlled through the CRC_CTL.FDSEL field.

In addition to providing bit swapping and mirror options to the incoming data, the CRC peripheral also supports bit mirroring on the following registers.

- CRC_RESULT_CUR and CRC_RESULT_FIN, controlled through the CRC_CTL.RSLTMIRR field. When mirroring is enabled, the values to be written to these registers are fully bit-reversed before the write operation occurs.
- CRC_POLY, controlled through the CRC_CTL.POLYMIRR field. When mirroring is enabled, the 32-bit polynomial is fully bit-reversed before the write operation to the register occurs.
- CRC_COMP, controlled through the CRC_CTL.CMPMIRR field. When mirroring is enabled, the contents to be loaded to this register are fully bit-reversed before the write operation occurs.

FIFO Status and Data Requests

The CRC peripheral provides indication of the input and output buffer status through CRC_STAT.IBR and CRC_STAT.OBR respectively. For core-based operations, software must monitor these status fields prior to writing to or reading from the CRC FIFO. No write to the CRC FIFO can occur while CRC_STAT.IBR indicates that the buffer is not ready to accept data. Similarly, the CRC FIFO cannot be read until CRC_STAT.OBR indicates that data is available.

The memory scan modes of operation only require the monitoring of the input buffer status. The memory transfer, compute-and-compare mode uses both input and output buffer status. If the current result of the CRC computation is required, then software must verify that the current operation has completed and that the intermediate result is ready. The CRC_STAT.IRR indicates the status.

NOTE: The memory transfer fill mode of operation requires the use of a DMA channel. The CRC does not support core reads from the CRC FIFO for this mode of operation.

Memory transfer, compute-and-compare mode uses burst transactions to make the most efficient use of the available resources. In this mode, when the FIFO is initially empty and the peripheral is enabled, the CRC_STAT.IBR bit indicates that the CRC is ready to accept data. The peripheral generates data requests to the source DMA channel (if the CRC uses DMA). While the number of words remaining in the CRC_DCNT register is greater than the FIFO depth, the peripheral issues data requests or accepts incoming data in bursts. The peripheral continues until the CRC FIFO becomes full.

Once full, the CRC_STAT.IBR and CRC_STAT.OBR bits are updated, and then the CRC issues outgoing data requests. Only when the FIFO is empty can the peripheral accept further incoming data, and the CRC_STAT.IBR and CRC_STAT.OBR bits are updated once again.

Once CRC_DCNT is decremented such that the number of words waiting for processing is less than the number required to fill the FIFO, the burst mode of operation is disabled. Incoming data is accepted when the FIFO is not full. Outgoing data is available when the FIFO is not empty. Therefore, there are no restrictions requiring the word count to be a multiple of the FIFO depth.

All other CRC modes of operation indicate that incoming data can be accepted when the FIFO is not full. Outgoing data is available when the FIFO is not empty.

The CRC_CTL.OBRSTALL and CRC_CTL.IRRSTALL bit configurations also influence how the CRC generates data requests and status bits. The following list describes the bits.

- The CRC_CTL.OBRSTALL bit can be configured such that the CRC peripheral stalls as soon as there is output data available in the FIFO. Use this mode of operation only in memory transfer, compute-and-compare mode. This mode results in the processing of one 32-bit word at a time. The peripheral does not request or accept incoming data until the current value being processed is read from the peripheral.
- The CRC_CTL.IRRSTALL bit can be configured so that the CRC peripheral stalls all further incoming data requests until the CRC_RESULT_CUR register is read after being updated. Use this mode of operation for CRC signature generation. It is not applicable to memory transfer data-fill mode or memory scan data-verify mode of operation.

CRC Operating Modes

The following sections describe the various operating modes of the CRC interface.

Data Transfer Modes

The CRC peripheral supports two main categories of operation involving data transfers:

- Memory scan mode
- Memory transfer mode

Memory scan modes are read-only operations that allow the contents of memory to be read into the peripheral and verified for correctness. There are two forms of memory scan mode:

- CRC compute-and-compare performs a CRC calculation on data presented to the peripheral and compares the CRC result with a pre-determined and pre-loaded result. An error is generated when the results differ.
- Data verify compares each 32-bit data word presented to the CRC peripheral to a pre-loaded 32-bit value and generates an error when the data differs.

Both modes of operation require at the most, a single DMA channel to read the data from memory into the peripheral. No data is forwarded to the data output or destination DMA. The CRC can also use core-driven transfers for either of these modes of operation.

The memory transfer modes involve memory write or memory read-and-write operations allowing for memory to be initialized or transferred from one region of memory to another. There are two forms of memory transfer mode:

- CRC compute-and-compare performs a full data transfer from one memory region to another memory region. The CRC generates a signature on the data presented and compares it with a pre-determined and pre-loaded result. An error is generated when the results differ.
- Data fill initializes a region of memory with a pre-loaded 32-bit constant value.

The CRC compute-and-compare mode of operation requires both incoming and outgoing data channels. The operation occurs either using DMA channels, using core-driven write or read operations to and from the FIFO or using

a combination of both. The data fill mode of operation requires only a memory write DMA destination channel—this mode does not support core driven operations.

Memory Scan Compute-and-Compare Mode

In this mode of operation, the CRC engine of the peripheral is enabled. The mode is configured through the CRC_CTL.OPMODE field and the CRC engine performs a 32-bit CRC operation on the incoming data stream.

The length of the data stream is configured through the CRC_DCNT register. The accumulated result of the CRC operation is contained in the CRC_RESULT_CUR register. As the CRC engine processes each 32-bit word, the CRC_DCNT register is decremented and CRC_RESULT_CUR is updated.

Once CRC_DCNT decrements to zero, the contents of the CRC_RESULT_CUR register are copied to CRC_RESULT_FIN and CRC_STAT.DCNTEXP is updated accordingly. The CRC uses the CRC_COMP register to store the expected result of the operation. After the CRC calculation, CRC_COMP is compared with CRC_RESULT_FIN and CRC_STAT.CMPERR is updated to reflect the status of the compare operation. CRC_STAT.CMPERR must be cleared before the next CRC operation is performed.

The CRC peripheral also contains the CRC_DCNTRLD register. The CRC uses this register to reload CRC_DCNT upon completion of the CRC operation in preparation for the next transfer.

The initial seed of the CRC computation can be configured through CRC_CTL.AUTOCLRZ and CRC_CTL.AUTOCLRF. This configuration provides a way to reset CRC_RESULT_CUR to 0x00000000, 0xFFFFFFFF or to leave the current register contents untouched for the next operation.

The peripheral can be configured to allow for the compare error and data expiration events to generate an interrupt.

Memory Scan Data Verify

In this mode of operation, the CRC engine of the peripheral is not required. The mode is enabled through the CRC_CTL.OPMODE field. Each 32-bit word of the data stream is compared with a constant value that is stored in the CRC_COMP register. The CRC_DCNT register contains the number of words for comparison. The CRC_DCNT register is decremented upon receiving a new 32-bit word from the data stream. If the compare operation fails, the CRC_STAT.CMPERR bit is updated and the contents of CRC_DCNT are captured in the CRC_DCNTCAP register. This information can be used to identify the location in the data stream where the error occurred. Clear the CRC_STAT.CMPERR field to reenable capturing of further errors.

Once CRC_DCNT decrements to zero, CRC_STAT.DCNTEXP is updated accordingly to signal the end of the operation. The peripheral can be configured to allow for the compare error and data expiration events to generate an interrupt.

Memory Transfer Compute-and-Compare Mode

In this mode of operation, the CRC engine of the peripheral is enabled. The mode is configured through the CRC_CTL.OPMODE field and the CRC engine performs a 32-bit CRC operation on the incoming data stream.

The length of the data stream is configured through the CRC_DCNT register. The accumulated result of the CRC operation is contained in the CRC_RESULT_CUR register. As the CRC engine processes each 32-bit word, the CRC_DCNT register is decremented and CRC_RESULT_CUR is updated.

Once CRC_DCNT decrements to zero, the contents of the CRC_RESULT_CUR register are copied to CRC_RESULT_FIN and CRC_STAT.DCNTEXP is updated accordingly. The CRC uses the CRC_COMP register to store the expected result of the operation. Upon completion of the CRC calculation, CRC_COMP is compared with CRC_RESULT_FIN and CRC_STAT.CMPERR is updated to reflect the status of the compare operation. Clear CRC_STAT.CMPERR before the next CRC operation is performed.

The CRC peripheral also contains CRC_DCNTRLD register. The CRC uses this register to reload CRC_DCNT upon completion of the CRC operation in preparation for the next transfer.

The initial seed of the CRC computation can be configured through CRC_CTL.AUTOCLRZ and CRC_CTL.AUTOCLRF. This configuration provides a means to reset CRC_RESULT_CUR to 0x00000000, 0xFFFFFFFF or to leave the current register contents untouched for the next operation.

The peripheral can be configured to allow for the compare error and data expiration events to generate an interrupt.

Memory Transfer Data Fill Mode

In this mode of operation, the CRC engine of the peripheral is not required. The mode is enabled through the CRC_CTL.OPMODE field. The CRC_FILLVAL register is written with a 32-bit value. The CRC uses this value to initialize a block memory through the memory-to-memory DMA destination channel. When the CRC peripheral and the DMA destination channel are enabled, the contents of the CRC_FILLVAL register is written to the DMA channel to initialize the memory region. The CRC_DCNT register contains the number of words for the write operation.

Once CRC_DCNT decrements to zero, CRC_STAT.DCNTEXP is updated accordingly to signal the end of the operation. The peripheral can be configured to allow for the data expiration event to generate an interrupt.

CRC Event Control

The CRC peripheral can enable certain CRC status operations to generate an interrupt event to the system event controller. There, a CRC error can be qualified as a system fault.

Interrupt Signals

The CRC peripheral can generate two interrupt requests that are optionally enabled as interrupts with the Arm core, or as events to SEC1 for fault operations. One is a CRC status interrupt and the other is a CRC error interrupt.

The CRC_STAT.CMPERR status bit can be configured as an interrupt and is signaled through the CRC error interrupt signal. The CRC_STAT.CMPERR status field is set whenever the CRC peripheral performs a compare operation that fails. This status can be the result of a failed memory scan data-verify operation that compares the contents of a memory range with a constant 32-bit value. Or, it can be the result of a CRC signature calculated for a memory region that does not match the expected pre-programmed result for a memory-compare operation.

The CRC_STAT.DCNTEXP status bit is set when the CRC_DCNT register has decremented to zero. The status indicates that the CRC peripheral has now processed all the data requested for the current CRC operation. The CRC can also use this signal to generate an interrupt. The interrupt is signaled on the CRC status interrupt signal.

Both these status bits can be configured to generate and interrupt through the CRC_INEN register. The CRC_INEN register also has bit set, CRC_INEN_SET, and bit clear CRC_INEN_CLR equivalent registers that the CRC uses for the enabling and disabling of these interrupt sources.

The CRC_STAT register has two write one to clear (W1C) fields for clearing the two interrupt sources.

NOTE: Disabling the CRC peripheral through the CRC_CTL.BLKEN bit does not result in the clearing of interrupt sources. Clear the interrupt sources using a W1C operation to the CRC_STAT register.

CRC Programming Model

It is important to note the following restrictions when using the CRC peripheral with the DMA channels:

1. When enabling the CRC peripheral and the DMA channels, enable the CRC peripheral prior to enabling the DMA channels.
2. When disabling the CRC peripheral and the DMA channels, disable the DMA channels prior to disabling the CRC peripheral.

CRC Mode Configuration

Describes a number of tasks showing the various operation modes of the CRC peripheral.

- Look-up Table Generation
- Core Driven Memory Scan Compute-and-Compare Mode
- DMA Driven Memory Scan Compute-and-Compare Mode
- Core Driven Memory Scan Data Verify Mode
- DMA Driven Memory Scan Data Verify Mode
- Core Driven Memory Transfer Compute-and-Compare Mode
- DMA Driven Memory Transfer Compute-and-Compare Mode
- DMA Driven Memory Transfer Data Fill Mode

Look-up Table Generation

Describes the steps required to initialize the CRC peripheral LUT.

1. Write the 32-bit CRC polynomial of choice to the CRC_POLY register.

   ADDITIONAL INFORMATION: This operation results in the CRC peripheral starting the LUT initialization process. The CRC_STAT.LUTDONE bit is updated to reflect the operation is in progress.

2. Poll the CRC_STAT.LUTDONE bit until the status bit indicates that the operation is completed.

The CRC peripheral has completed initialization of all the LUT registers and is now ready for data operations. The CRC_STAT.LUTDONE bit remains in the current state until the CRC_POLY register is written again, or the peripheral or processor are reset.

Core Driven Memory Scan Compute-and-Compare Mode

Performs CRC signature calculation and verification for a region of memory using core transactions. The CRC peripheral is configured such that it operates in burst mode due to the stalling options configured through disabling the CRC_CTL register.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per CRC_CTL.BLKEN

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_RESULT_CUR register.

   ADDITIONAL INFORMATION: This register can be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the pre-calculated final CRC signature result for the memory region that the software uses in the final compare operation.

5. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts. If enabled, ensure that the corresponding interrupt handlers are also configured.

6. Initialize CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan compute-and-compare mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.
   - Disable the CRC_CTL.OBRSTALL and CRC_CTL.IRRSTALL bit options for this task example.
   - Configure all mirroring and bit reversal options.

- Configure CRC auto-clear options.

The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

7. Write memory region data to the CRC peripheral.
   a. While CRC_STAT.IBR bit indicates that the input buffer is ready, write the CRC_DFIFO register with 32-bit data.

      ADDITIONAL INFORMATION: Repeat this step until all data has been written.
8. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures that all the data has been processed.
9. Poll the CRC_STAT.CMPERR bit if the interrupt was disabled to check for a compare error.

   ADDITIONAL INFORMATION: Perform this step only if the compare error interrupt is not enabled.
10. Write to the CRC_STAT register to clear both the CRC_STAT.DCNTEXP and CRC_STAT.CMPERR bits.

    ADDITIONAL INFORMATION: If interrupts were enabled, then clear of these status bits within the interrupt handlers for the respective interrupts.

    The CRC compute-and-compare operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation.

The integrity check of the memory through the expected CRC signature has completed. The final result is indicated through the CRC_STAT.CMPERR bit and the corresponding interrupt when enabled.

Clear any W1C CRC status bits before performing more CRC operations.

DMA Driven Memory Scan Compute-and-Compare Mode

Performs CRC signature calculation and verification for a region of memory using DMA transactions. The CRC peripheral is configured such that it operates in the burst mode of operation due to the stalling options configured through disabling CRC_CTL.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per the CRC_CTL.BLKEN bit.

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_RESULT_CUR register.

   ADDITIONAL INFORMATION: This register can be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the pre-calculated final CRC signature result for the memory region that the software uses in the final operation.

5. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts, as needed. If enabled, ensure that the corresponding interrupt handlers are also configured.

6. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan compute compare mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.

   - Disable the CRC_CTL.OBRSTALL and CRC_CTL.IRRSTALL bit options for this task example.
   - Configure all mirroring and bit reversal options.
   - Configure all CRC auto clear options.

   The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

7. Configure and enable the memory-to-memory source DMA channel for memory read STOP mode.

   ADDITIONAL INFORMATION: This step starts the data transfer from the memory region and writes the data to the CRC peripheral.

8. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if the counter expired interrupt is disabled. Polling ensures all the data has been processed.

9. Poll the CRC_STAT.CMPERR bit if the interrupt was disabled to check for a compare error.

   ADDITIONAL INFORMATION: Perform this step only if the compare error interrupt is not enabled.

10. Write the CRC_STAT register to clear both the CRC_STAT.DCNTEXP and CRC_STAT.CMPERR bits.

    ADDITIONAL INFORMATION: If interrupts were enabled, then clear these status bits within the interrupt handlers for the respective interrupts.

    The CRC compute-and-compare operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation.

The integrity check of the memory through the expected CRC signature has completed and the final result indicated is through CRC_STAT.CMPERR and the corresponding interrupt, when enabled.

Clear any W1C CRC status bits before performing a further CRC operation. Clear any W1C status bits of the memory-to-memory source DMA channel before the next CRC operation.

Core Driven Memory Scan Data Verify Mode

Reads a region of memory using core transactions and performs a compare operation on each 32-bit word against a single pre-loaded 32-bit constant. The compare error interrupt is enabled to capture and log the location of any compare errors.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per CRC_CTL.BLKEN

The interrupt service routine for the compare error interrupt reads and stores the contents of CRC_DCNTCAP register to a buffer before clearing the compare error interrupt.

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the 32-bit constant that the memory region is expected to be filled with. Each 32 bit of data presented to the peripheral is compared with this value.

4. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts. If enabled, ensure that the corresponding interrupt handlers are also configured.

5. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan data verify mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.

   The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

6. Write memory region data to the CRC peripheral.

a. Poll the CRC_STAT.IBR bit until input buffer is ready.

b. Write the CRC_DFIFO register with 32-bit data.

ADDITIONAL INFORMATION: Repeat these two steps until the entire memory region has been written to the CRC peripheral.

7. Poll the CRC_INEN_SET.DCNTEXP bit if the interrupt was disabled.

ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures all the data has been processed.

8. Check if the buffer used to capture the CRC_DCNTCAP register upon a compare error has any new entries.

ADDITIONAL INFORMATION: The values captures in the buffer provide a means to locate where in the memory region the failures occurred.

9. Write to the CRC_STAT to clear both the CRC_INEN_SET.DCNTEXP and CRC_INEN.CMPERR bits.

ADDITIONAL INFORMATION: If interrupts were enabled, the clear these status bits within the interrupt handlers for the respective interrupts.

The CRC memory scan-verify operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation.

The result of the integrity check of the memory with the 32-bit constant is indicated through the CRC_INEN.CMPERR bit and the corresponding interrupt, when enabled. Each comparison error is traceable due to the logging of CRC_DCNTCAP from within the compare error interrupt handler.

Clear any W1C CRC status bits before performing a further CRC operation.

DMA Driven Memory Scan Data Verify Mode

The memory scan data verify mode reads a region of memory using DMA transactions and performs a compare operation on each 32-bit word against a single pre-loaded 32-bit constant. The compare error interrupt is enabled to capture and log the location of any compare errors.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per the CRC_CTL.BLKEN bit

The interrupt service routine for the compare error interrupt reads and stores the contents of the CRC_DCNTCAP register to a buffer before clearing the compare error interrupt.

1. Initialize the CRC_DCNT register.

ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: The CRC module uses this register to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the 32-bit constant that the memory region is expected to be filled with. Each 32 bit of data presented to the peripheral is compared with this value.

4. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts, as needed. If enabled, ensure that the corresponding interrupt handlers are also configured.

5. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan data verify mode and CRC_CTL.BLKEN configured to enable the CRC peripheral.

   The CRC peripheral is now enabled and ready for the core or DMA channel to write the data.

6. Configure and enable the memory-to-memory source DMA channel for memory read STOP mode.

   ADDITIONAL INFORMATION: This step starts the data transfer from the memory region and writes the data to the CRC peripheral.

7. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures all the data has been processed.

8. Check if the buffer used to capture the CRC_DCNTCAP register upon a compare error has any new entries.

   ADDITIONAL INFORMATION: The values captures in the buffer provide a means to locate where in the memory region the failures occurred.

9. Write the CRC_STAT register to clear both the CRC_STAT.DCNTEXP and CRC_STAT.CMPERR bits.

   ADDITIONAL INFORMATION: If interrupts were enabled, then clear these status bits within the interrupt handlers for the respective interrupts.

   The CRC memory scan-verify operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation.

The result of the integrity check of the memory with the 32-bit constant is indicated through the CRC_STAT.CMPERR bit and the corresponding interrupt when enabled. Each comparison error is traceable due to the logging of the CRC_DCNTCAP register from within the compare error interrupt handler.

Clear any W1C CRC status bits and DMA status bits before performing a further CRC operation.

Core Driven Memory Transfer Compute-and-Compare Mode

The memory transfer compute-and-compare mode performs CRC signature calculation and verification for a region of memory using core transactions while copying the contents to another memory region. The CRC peripheral is configured such that it operates in the burst mode of operation due to the stalling options configured through disabling the CRC_CTL register.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per the CRC_CTL.BLKEN bit

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of the current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_RESULT_CUR register.

   ADDITIONAL INFORMATION: This register can be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the pre-calculated final CRC signature result for the memory region that the software uses in the final compare operation.

5. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts, as needed. If enabled, ensure that the corresponding interrupt handlers are also configured.

6. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan compute-and-compare mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.

   a. Disable the CRC_CTL.OBRSTALL bit and the CRC_CTL.IRRSTALL bit options for this task example.

   b. Configure all mirroring and bit reversal options.

c. Configure CRC auto clear options

The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

7. Write memory region data to the CRC peripheral and read it back to the new destination.
   a. While the CRC_STAT.IBR bit indicates that the input buffer is ready, write the CRC_DFIFO register with 32-bit data.
   b. While the CRC_STAT.OBR bit indicates that the output buffer is ready, read the CRC_DFIFO register and store data to new destination.

      ADDITIONAL INFORMATION: Repeat these two steps until all required data has been processed through the CRC peripheral and copied to the new destination.
8. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if the counter expired interrupt is disabled. Polling ensures all the data has been processed.
9. Poll the CRC_STAT.CMPERR bit if the interrupt was disabled to check for a compare error.

   ADDITIONAL INFORMATION: Perform this step only if the compare error interrupt is not enabled.
10. Write the CRC_STAT register to clear both CRC_STAT.DCNTEXP and CRC_STAT.CMPERR bits.

    ADDITIONAL INFORMATION: If interrupts were enabled, then clear these status bits within the interrupt handlers for the respective interrupts.

    The CRC compute-and-compare operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation. The memory region has also been copied to its new destination.

The memory region has been copied to a new location and an integrity check of the memory through the expected CRC signature has also completed. The final result is indicated through the CRC_STAT.CMPERR bit and the corresponding interrupt when enabled.

Clear any W1C CRC status bits before performing a further CRC operation.

DMA Driven Memory Transfer Compute-and-Compare Mode

The memory transfer compute-and-compare mode performs CRC signature calculation and verification for a region of memory using DMA transactions. The memory region is also copied to another memory region using memory-to-memory DMA transfers. The CRC peripheral is configured such that it operates in burst mode due to the stalling options configured through disabling CRC_CTL.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)

- The CRC block is disabled per the CRC_CTL.BLKEN register.

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_RESULT_CUR register.

   ADDITIONAL INFORMATION: This register can be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize the CRC_COMP register.

   ADDITIONAL INFORMATION: This register contains the pre-calculated final CRC signature result for the memory region that the software uses in the final compare operation.

5. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts, as needed. If enabled, ensure that the corresponding interrupt handlers are also configured.

6. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory scan compute compare mode and CRC_CTL.BLKEN configured to enable the CRC peripheral.

   a. Disable the CRC_CTL.OBRSTALL and the CRC_CTL.IRRSTALL bit options for this task example.

   b. Configure all mirroring and bit reversal options

   c. Configure CRC auto clear options

   The CRC peripheral is now enabled and ready for the core or DMA channel to write data.

7. Configure and enable the memory-to-memory source DMA channel for memory read STOP mode and destination DMA channel for memory write STOP mode.

   ADDITIONAL INFORMATION: This step starts the data transfer from one memory region to another through the memory-to-memory DMA channels and the CRC peripheral.

8. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures all the data has been processed.

9. Poll the CRC_STAT.CMPERR bit if the interrupt was disabled to check for a compare error.

ADDITIONAL INFORMATION: Perform this step only if the compare error interrupt is not enabled.

10. Write the CRC_STAT register to clear both the CRC_STAT.DCNTEXP and the CRC_STAT.CMPERR bits.

    ADDITIONAL INFORMATION: If interrupts were enabled, then clear these status bits within the interrupt handlers for the respective interrupts.

    The CRC compute-and-compare operation is now complete. The CRC peripheral is ready to be configured for the next CRC operation. The memory region has also been copied to its new destination.

The integrity check of the memory through the expected CRC signature has completed and the final result is indicated through the CRC_STAT.CMPERR bit and the corresponding interrupt when enabled. The memory region has also been copied to its final destination.

Clear any W1C CRC status bits before performing a further CRC operation. Also, clear any W1C status bits of the memory-to-memory source and destination DMA channels before the next CRC operation.

DMA Driven Memory Transfer Data Fill Mode

This mode initializes a region of memory to a constant 32-bit value using DMA transactions.

The task assumes the following:

- The polynomial has been loaded and the look-up table is fully initialized
- All CRC interrupts have been serviced (none pending)
- The CRC block is disabled per the CRC_CTL.BLKEN bit

1. Initialize the CRC_DCNT register.

   ADDITIONAL INFORMATION: The value loaded must represent the number of 32-bit words in the memory region for which the software calculates and verifies the signature.

2. Initialize the CRC_DCNTRLD register.

   ADDITIONAL INFORMATION: This value is used to reload the CRC_DCNT register upon completion of current CRC operation. If no further operation is needed, then this register can be initialized to zero.

3. Initialize the CRC_FILLVAL register.

   ADDITIONAL INFORMATION: This register contains the 32-bit constant that the CRC module uses to fill the memory region.

4. Initialize the CRC_INEN register.

   ADDITIONAL INFORMATION: The CRC module uses this register to enable the generation of the CRC interrupts for notification of block completion. Configure these interrupts as required. If enabled, ensure that the corresponding interrupt handlers are also configured.

5. Initialize the CRC_CTL register with the CRC_CTL.OPMODE bit set to memory transfer fill mode and the CRC_CTL.BLKEN bit configured to enable the CRC peripheral.

   The CRC peripheral is now enabled and is ready for the DMA channel to write data.

6. Configure and enable the memory-to-memory destination DMA channel for memory write STOP mode.

   ADDITIONAL INFORMATION: This step starts the data transfer taking the constant 32-bit value from the CRC peripheral and writing the data to the DMA channel.

7. Poll the CRC_STAT.DCNTEXP bit if the interrupt was disabled.

   ADDITIONAL INFORMATION: Perform this step only if counter expired interrupt is disabled. Polling ensures that all the data has been processed.

8. Write the CRC_STAT register to clear the CRC_STAT.DCNTEXP bit.

   ADDITIONAL INFORMATION: If interrupts were enabled, then clear this status bit within the interrupt handlers for the respective interrupts.

   The CRC memory transfer fill operation is complete, and the CRC peripheral is ready to be configured for the next CRC operation.

The memory region is filled with the constant data and the CRC peripheral is ready to be configured for a new operation.

Clear any W1C CRC status bits and DMA status bits before performing a further CRC operation.

ADSP-SC59x CRC Register Descriptions

Cyclic Redundancy Check Unit (CRC) contains the following registers.

Table 38-4: ADSP-SC59x CRC Register List

Name	Description
CRC_COMP	Data Compare Register
CRC_CTL	Control Register
CRC_DCNT	Data Word Count Register
CRC_DCNTCAP	Data Count Capture Register
CRC_DCNTRLD	Data Word Count Reload Register
CRC_DFIFO	Data FIFO Register
CRC_FILLVAL	Fill Value Register
CRC_INEN	Interrupt Enable Register
CRC_INEN_CLR	Interrupt Enable Clear Register
CRC_INEN_SET	Interrupt Enable Set Register

Table 38-4: ADSP-SC59x CRC Register List (Continued)

Name	Description
CRC_POLY	Polynomial Register
CRC_RESULT_CUR	CRC Current Result Register
CRC_RESULT_FIN	CRC Final Result Register
CRC_STAT	Status Register

Data Compare Register

The CRC_COMP register contains the value corresponding to the expected CRC result or signature for the current data stream. At the end of the operation, the content of this register is used to compare against the result produced by the CRC operation. In data verify mode, each incoming data value is compared with the content of this register.

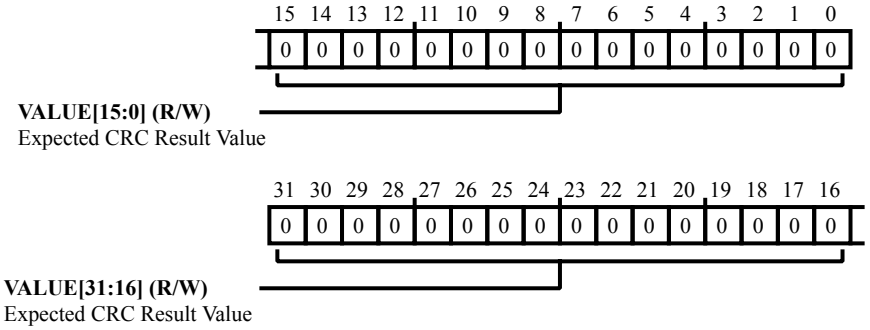


Figure 38-3: CRC_COMP Register Diagram

Table 38-5: CRC_COMP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Expected CRC Result Value. The CRC_COMP.VALUE bit field contains the value corresponding to the expected CRC result or signature for the current data stream.

Control Register

The CRC_CTL register configures the operation modes and settings for the CRC.

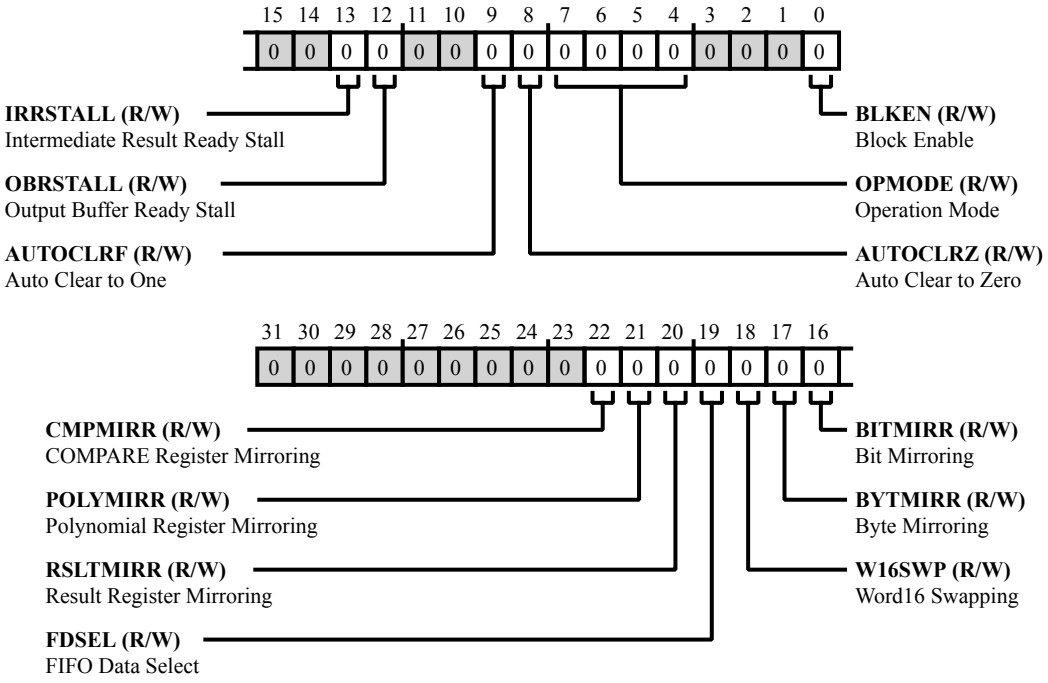


Figure 38-4: CRC_CTL Register Diagram

Table 38-6: CRC_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22 (R/W)	CMPMIRR	COMPARE Register Mirroring. The CRC_CTL.CMPMIRR bit enables data mirroring for the CRC_COMP compare register. When enabled, the 32-bit value in this register is fully bit mirrored (reversed). The bit-reversed value is used for comparison with the CRC_RESULT_FIN register.	
		0	Disable compare mirroring
		1	Enable compare mirroring
21 (R/W)	POLYMIRR	Polynomial Register Mirroring. The CRC_CTL.POLYMIRR bit enables data mirroring for the CRC_POLY polynomial register. When enabled, the 32-bit value in this register is fully bit mirrored (reversed). The bit-reversed value is used for CRC computations.	
		0	Disable polynomial mirroring
		1	Enable polynomial mirroring

Table 38-6: CRC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/W)	RSLTMIRR	Result Register Mirroring. The CRC_CTL.RSLTMIRR bit enables data mirroring for the CRC_RESULT_CUR and CRC_RESULT_FIN result registers. When enabled, the 32-bit values in these registers are fully bit mirrored (reversed).	
		0	Disable result mirroring
		1	Enable result mirroring
19 (R/W)	FDSEL	FIFO Data Select. The CRC_CTL.FDSEL bit selects whether the CRC writes modified or unmodified data to the FIFO in memory transfer mode. If enabled, the data written is affected by the state of the data mirroring selections (CRC_CTL.BITMIRR, CRC_CTL.BYTMIRR, and CRC_CTL.W16SWP) before being written to the FIFO.	
		0	Write unmodified data to FIFO
		1	Write modified data to FIFO
18 (R/W)	W16SWP	Word16 Swapping. The CRC_CTL.W16SWP bit enables the CRC's data mirror block to swap the upper and lower 16-bit words within the 32-bit input data, before further processing.	
		0	Disable word16 swapping
		1	Enable word16 swapping
17 (R/W)	BYTMIRR	Byte Mirroring. The CRC_CTL.BYTMIRR bit enables the CRC's data mirror block to mirror the bytes within the 32-bit input data, before further processing.	
		0	Disable byte mirroring
		1	Enable byte mirroring
16 (R/W)	BITMIRR	Bit Mirroring. The CRC_CTL.BITMIRR bit enables the CRC's data mirror block to mirror the bits within each byte of the 32-bit input data, before further processing.	
		0	Disable bit mirroring
		1	Enable bit mirroring
13 (R/W)	IRRSTALL	Intermediate Result Ready Stall. The CRC_CTL.IRRSTALL bit enables stalling the state machine for input data when there is a valid intermediate result to be read in the CRC_RESULT_CUR register. This feature should be used only in CRC computation modes (for example, CRC_CTL.OPMODE =1 or =3).	
		0	Do not stall
		1	Stall on IRR

Table 38-6: CRC_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	OBRSTALL	Output Buffer Ready Stall. The CRC_CTL.OBRSTALL bit enables stalling the state machine for input data when there is valid data in the output buffer. This feature should be used only in memory-to-memory transfer modes (for example, CRC_CTL.OPMODE =1).	
		0	Do not stall
		1	Stall on OBR
9 (R/W)	AUTOCLRF	Auto Clear to One. The CRC_CTL.AUTOCLRF bit enables auto clear to one when the CRC is in intermediate results ready stall mode (CRC_CTL.IRRSTALL=1) and the CRC data count expires (CRC_DCNT=0). Note that the CRC_CTL.AUTOCLRZ bit must be disabled, or the CRC_CTL.AUTOCLRF bit has no effect.	
		0	No auto clear
		1	Auto clear
8 (R/W)	AUTOCLRZ	Auto Clear to Zero. The CRC_CTL.AUTOCLRZ bit enables auto clear to zero when the CRC is in intermediate results ready stall mode (CRC_CTL.IRRSTALL=1) and the CRC data count expires (CRC_DCNT=0). Note that CRC_CTL.AUTOCLRF must be disabled, or the CRC_CTL.AUTOCLRZ has no effect.	
		0	No auto clear
		1	Auto clear
7:4 (R/W)	OPMODE	Operation Mode. The CRC_CTL.OPMODE bit field selects the memory transfer or scan mode.	
		0	Reserved
		1	CRC compute/compare memory transfer
		2	Data fill memory transfer
		3	CRC compute/compare memory scan
		4	Data verify memory scan
0 (R/W)	BLKEN	Block Enable. The CRC_CTL.BLKEN bit enables and disables the CRC operation.	
		0	Disable
		1	Enable

Data Word Count Register

The CRC_DCNT register holds the word count that is used for the CRC operation. On transfer of every 32-bit word, the CRC decrements by 1 the content of this register. When the count decrements to zero, this event triggers a CRC compare action, and the CRC_DCNT register is automatically loaded from the CRC_DCNTRLD register for the next CRC operation.

Note that the initial value programmed into the CRC_DCNT register may be different from what is programmed in the CRC_DCNTRLD register.

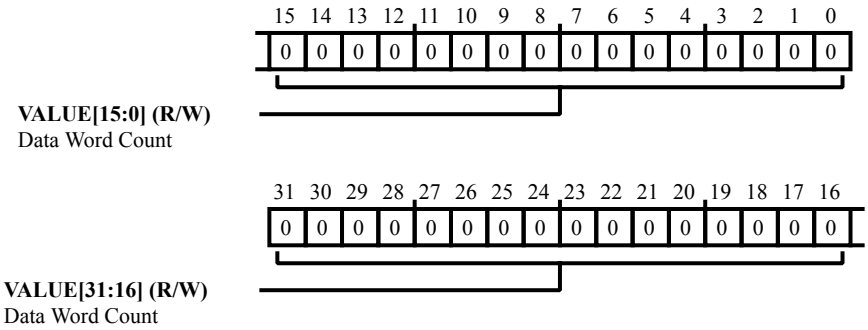


Figure 38-5: CRC_DCNT Register Diagram

Table 38-7: CRC_DCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Data Word Count. The CRC_DCNT.VALUE bit field holds the word count that is used for the CRC operation.

Data Count Capture Register

The CRC_DCNTCAP register captures the CRC_DCNT value when a compare operation fails in data verify mode. This capture can be used to track the position of an error in the data stream. The capture operation is enabled only if the CRC_STAT.CMPERR bit indicates no compare error. After an error occurs and the data count is captured, no further errors are logged until the CRC_STAT.CMPERR bit is cleared. To obtain the position of an error in the data stream, subtract the CRC_DCNTCAP register value from the initial CRC_DCNT.

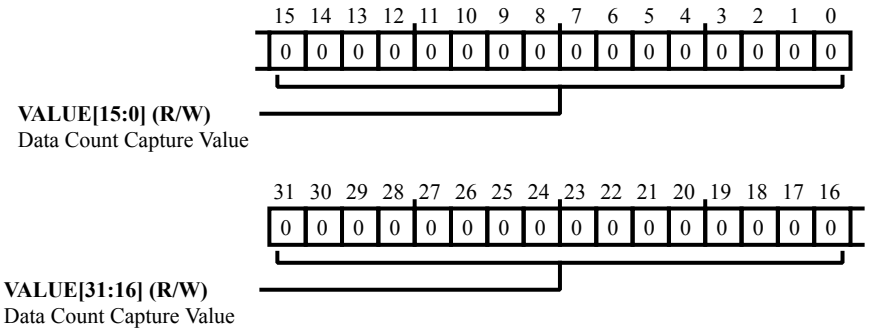


Figure 38-6: CRC_DCNTCAP Register Diagram

Table 38-8: CRC_DCNTCAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Data Count Capture Value. The CRC_DCNTCAP.VALUE bit field contains the CRC_DCNT value when a compare operation fails in data verify mode.

Data Word Count Reload Register

The CRC_DCNTRLD register holds the value that the CRC automatically loads into CRC_DCNT when the CRC_DCNT decrements to 0. At startup, the value programmed in CRC_DCNT and the CRC_DCNTRLD register could be different. So, for the first iteration, the CRC operation happens for the count initially programmed in the CRC_DCNT register. While for subsequent CRC operations, the count is taken from the CRC_DCNTRLD register.

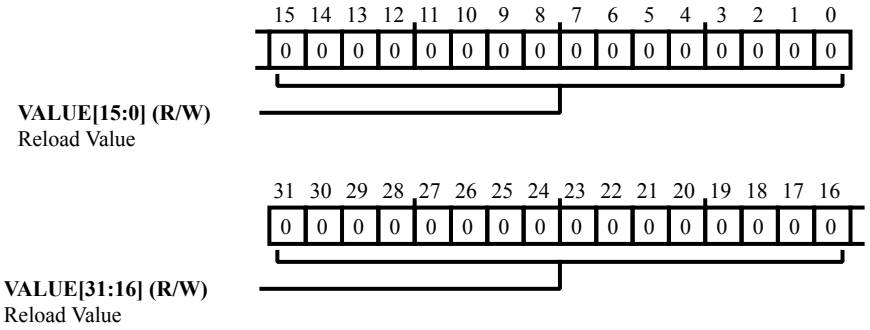


Figure 38-7: CRC_DCNTRLD Register Diagram

Table 38-9: CRC_DCNTRLD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Reload Value. The CRC_DCNTRLD.VALUE bit field holds the value that automatically loads into CRC_DCNT when the CRC_DCNT decrements to 0.

Data FIFO Register

In memory transfer mode (non-data fill mode), the data from the DMA or processor core buses is written into the CRC_DFIFO on each input data grant (DMA grant or core write). Data is read from this FIFO on each output data grant (DMA grant or core read). FIFO status information is available in the CRC_STAT register. Whenever, the FIFO has valid data, output data requests are generated.

Note that in non-memory transfer mode and in data fill mode, the input data does not get written into this FIFO. So, this register should not be read in these modes.

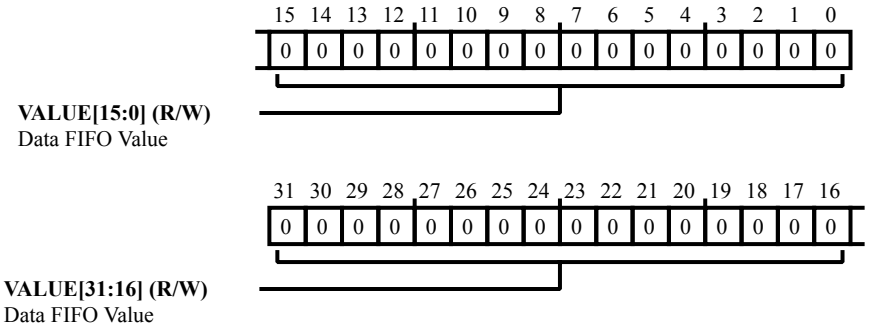


Figure 38-8: CRC_DFIFO Register Diagram

Table 38-10: CRC_DFIFO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Data FIFO Value. The CRC_DFIFO.VALUE bit field is the data from the DMA or processor core buses.

Fill Value Register

The CRC_FILLVAL register holds the value that the CRC uses for the memory fill operation. In data fill mode, the value programmed in this register is used for the memory fill operation.

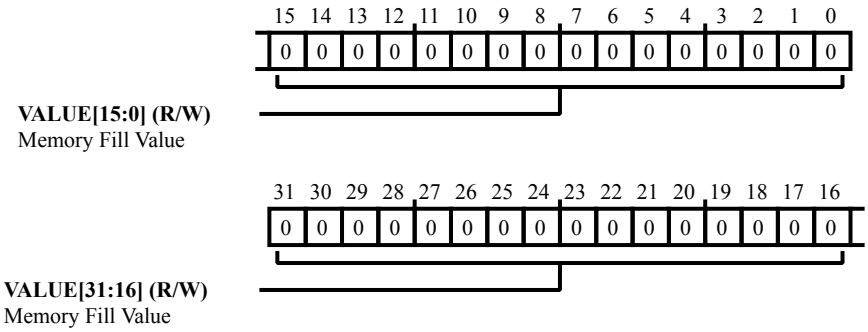


Figure 38-9: CRC_FILLVAL Register Diagram

Table 38-11: CRC_FILLVAL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Memory Fill Value. The CRC_FILLVAL.VALUE bit field holds the value that the CRC uses for the memory fill operation.

Interrupt Enable Register

The CRC_INEN register unmasks (enables) or masks (disables) interrupt requests generated in the CRC from going to the processor core.

Note that CRC interrupts are not disabled when the CRC is disabled (CRC_CTL.BLKEN =0).

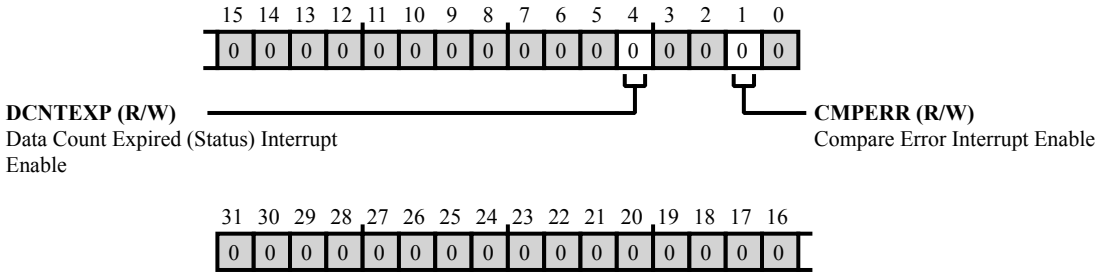


Figure 38-10: CRC_INEN Register Diagram

Table 38-12: CRC_INEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	DCNTEXP	Data Count Expired (Status) Interrupt Enable. The CRC_INEN.DCNTEXP enables (unmasks) the data count expired (CRC status) interrupt.	
		0	Disable (mask) interrupt
		1	Enable (unmask) interrupt
1 (R/W)	CMPERR	Compare Error Interrupt Enable. The CRC_INEN.CMPERR enables (unmasks) the data compare interrupt, which is generated when CRC data comparison fails.	
		0	Disable (mask) interrupt
		1	Enable (unmask) interrupt

Interrupt Enable Clear Register

The CRC_INEN_CLR register permits clearing individual bits in the CRC_INEN register without affecting other bits in the register.

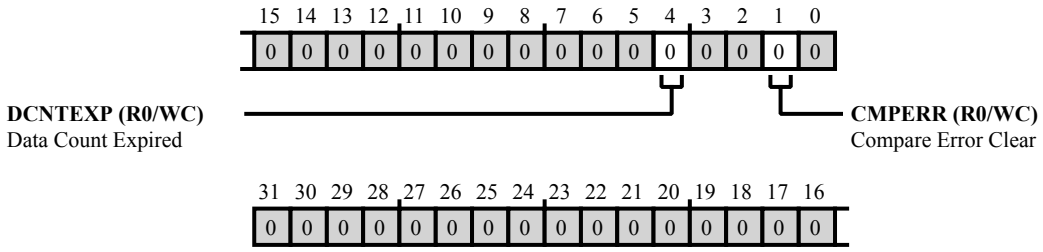


Figure 38-11: CRC_INEN_CLR Register Diagram

Table 38-13: CRC_INEN_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R0/WC)	DCNTEXP	Data Count Expired. The CRC_INEN_CLR.DCNTEXP bit clears the data count expired (status) interrupt.	
		0	No effect
		1	Clear bit
1 (R0/WC)	CMPERR	Compare Error Clear. The CRC_INEN_CLR.CMPERR bit clears the compare error interrupt.	
		0	No effect
		1	Clear bit

Interrupt Enable Set Register

The CRC_INEN_SET register permits setting individual bits in the CRC_INEN register without affecting other bits in the register.

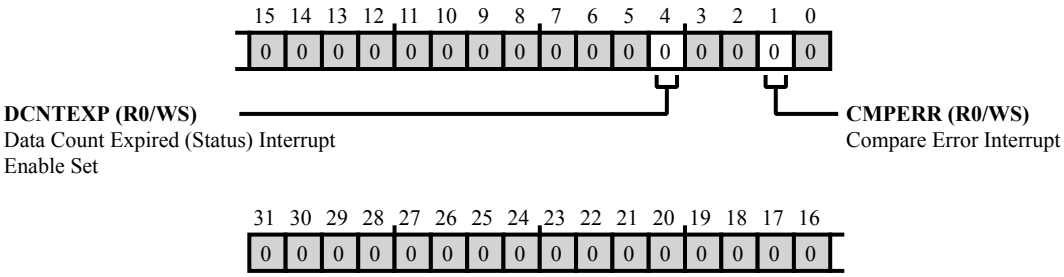


Figure 38-12: CRC_INEN_SET Register Diagram

Table 38-14: CRC_INEN_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R0/WS)	DCNTEXP	Data Count Expired (Status) Interrupt Enable Set. The CRC_INEN_SET.DCNTEXP bit sets the data count expired (status) interrupt.	
		0	No effect
		1	Set bit
1 (R0/WS)	CMPERR	Compare Error Interrupt. The CRC_INEN_SET.CMPERR bit sets the compare error interrupt.	
		0	No effect
		1	Set bit

Polynomial Register

The CRC_POLY register holds a 32-bit polynomial for CRC operations. Bit 31 corresponds to the coefficient of x31 of the CRC polynomial, bit 30 corresponds to the coefficient of x30, and so on through bit 0. A coefficient of x32 is assumed to be "1" for any polynomial that is selected. Based on the polynomial in the CRC_POLY register, the CRC generates a look-up table (LUT), which is used to compute the CRC of the incoming data stream.

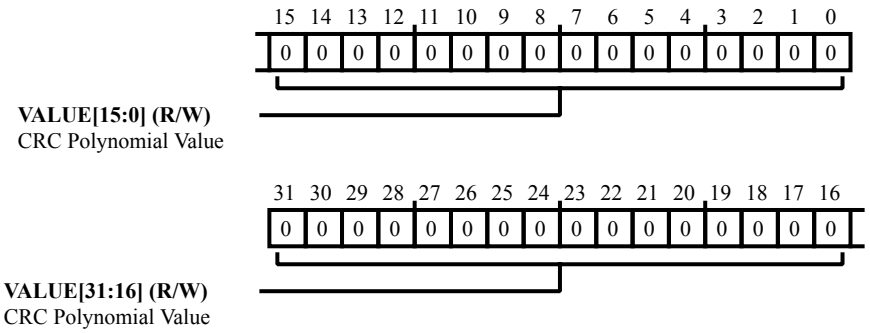


Figure 38-13: CRC_POLY Register Diagram

Table 38-15: CRC_POLY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	CRC Polynomial Value. The CRC_POLY.VALUE bit field holds the 32-bit polynomial for CRC operations.

CRC Current Result Register

The CRC_RESULT_CUR register holds the current or intermediate CRC result. It is updated when new data is written into the CRC. Each time the CRC_DCNT expires, the CRC loads the value from this register into the CRC_RESULT_FIN register. The CRC_RESULT_CUR register may be set to auto clear to zero or auto clear to ones when CRC_DCNT expires by configuring the CRC_CTL.AUTOCLRZ and CRC_CTL.AUTOCLRF bits. Before starting a CRC operation, the CRC_RESULT_CUR register should be programmed to the desired value.

Note that this register can be read by the processor core at any time.

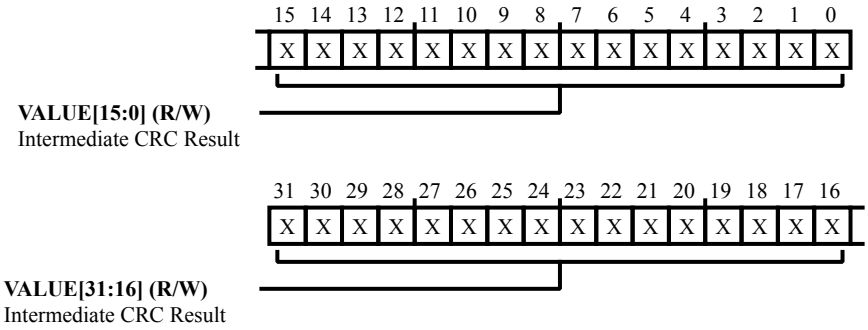


Figure 38-14: CRC_RESULT_CUR Register Diagram

Table 38-16: CRC_RESULT_CUR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Intermediate CRC Result. The CRC_RESULT_CUR.VALUE bit field holds the current or intermediate CRC result.

CRC Final Result Register

The CRC_RESULT_FIN register holds the final CRC computed for a data stream. A data stream is a DMA of CRC_DCNT number of words into the CRC. When CRC_DCNT decrements to zero for each data stream, the CRC loads the CRC_RESULT_FIN register with the value from the CRC_RESULT_CUR register.

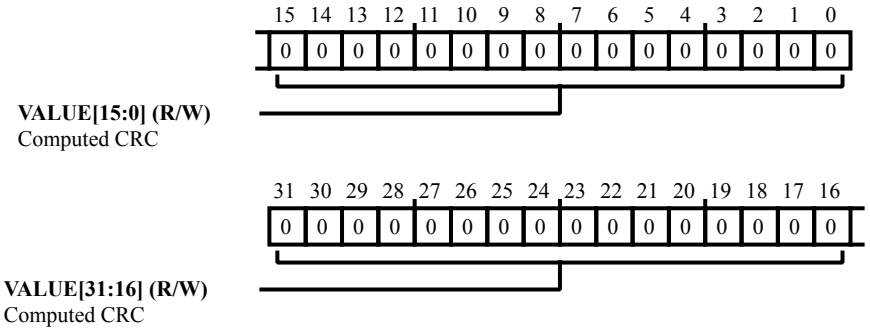


Figure 38-15: CRC_RESULT_FIN Register Diagram

Table 38-17: CRC_RESULT_FIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Computed CRC. The CRC_RESULT_FIN.VALUE bit field holds the final CRC computed for a data stream.

Status Register

The CRC_STAT register indicates the status for CRC operations and interrupt generation.

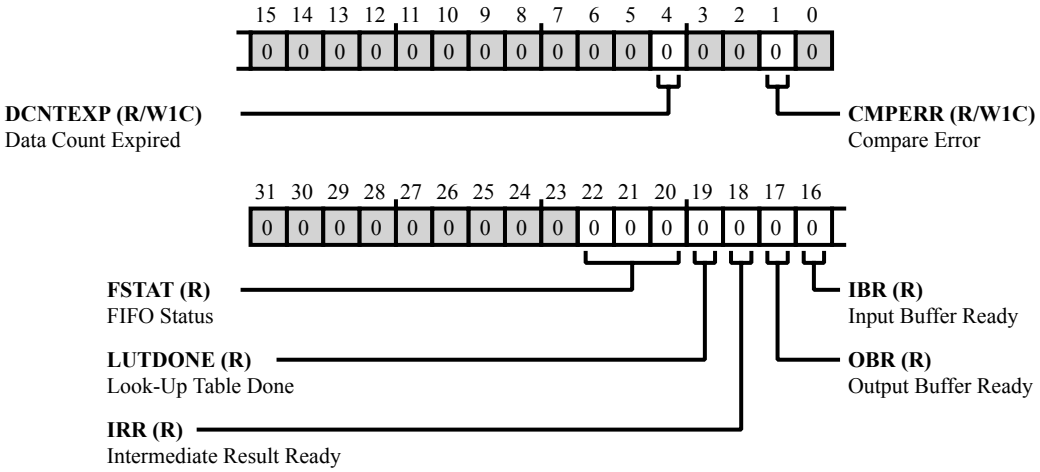


Figure 38-16: CRC_STAT Register Diagram

Table 38-18: CRC_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
22:20 (R/NW)	FSTAT	FIFO Status. The CRC_STAT.FSTAT indicates the current FIFO status. This field is read-only.	
		0	FIFO empty
		1	FIFO has 1 data
		2	FIFO has 2 data
		3	FIFO has 3 data
		4	FIFO has 4 data (full)
19 (R/NW)	LUTDONE	Look-Up Table Done. The CRC_STAT.LUTDONE bit indicates that the CRC has generated the look-up table for the current polynomial. This read-only bit is cleared at reset and cleared when the CRC_POLY is written.	
		0	No status
		1	LUT generation done

Table 38-18: CRC_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/NW)	IRR	Intermediate Result Ready. The CRC_STAT.IRR bit indicates that the CRC has updated the CRC_RESULT_CUR register with intermediate CRC results for the new data written to the CRC. The processor core should read from the CRC_RESULT_CUR register only after detecting CRC_STAT.IRR =1. This read-only bit is cleared by CRC hardware and is valid when the CRC_CTL.IRRSTALL bit is enabled.	
		0	No status
		1	Intermediate results ready
17 (R/NW)	OBR	Output Buffer Ready. The CRC_STAT.OBR bit indicates that the CRC has data ready for the processor core to read. The processor core should read from the CRC only after detecting CRC_STAT.OBR =1. This read-only bit is cleared by CRC hardware.	
		0	No status
		1	Output buffer ready
16 (R/NW)	IBR	Input Buffer Ready. The CRC_STAT.IBR bit indicates that the CRC is ready to accept a processor core write. The processor core should write to the input register only after detecting that CRC_STAT.IBR =1. This read-only bit is cleared by CRC hardware.	
		0	No status
		1	Input buffer ready
4 (R/W1C)	DCNTEXP	Data Count Expired. The CRC_STAT.DCNTEXP bit indicates that the CRC_DCNT has expired. This W1C bit is not automatically cleared when the CRC is disabled (CRC_CTL.BLKEN =0). When the CRC sets this bit on CRC_DCNT expiry, the CRC generates the CRC_INEN.DCNTEXP interrupt.	
		0	No status
		1	Data counter expired
1 (R/W1C)	CMPERR	Compare Error. The CRC_STAT.CMPERR bit indicates that a CRC mismatch or data mismatch has been detected. This W1C bit is not automatically cleared when the CRC is disabled (CRC_CTL.BLKEN =0). When the CRC sets this bit on detecting a mismatch, the CRC generates the CRC_INEN.CMPERR interrupt. While this bit is set, the CRC_DCNTCAP register is disabled from capturing the data count values.	
		0	No status
		1	Compare error

39 System Security

The requirement to protect content, keys, intellectual property, and other sensitive information have become increasingly prevalent. The processor contains several modules and system elements that contribute to creating a secure operating environment for trusted code to execute.

The modules and system elements that can be used are:

- Boot Kernel
- Secure Core
- System Protection Unit (SPU)
- System Memory Protection Unit (SMPU)
- DEBUG through the Test Access Port Controller (TAPC)
- One-Time Programmable (OTP) memory
- Cryptographic Accelerators (optionally)

NOTE: This product includes security features that can be used to protect embedded non-volatile memory contents and prevent execution of non-authorized code. When security is enabled on this device (either by the ordering party or the subsequent receiving parties), Analog Devices' ability to conduct Failure Analysis on returned devices will be limited. Contact Analog Devices, Inc. for details on the Failure Analysis limitations for this device.

Security Features

The security infrastructure in the system provides the following features:

- Secure operating environment for secure code execution
- Protect sensitive IP from theft by malicious users or competitors
- Protect sensitive data (for example cipher keys)
- Allow debugging while still maintaining security

Security Functional Description

In order to provide secure operating environment in a system, it requires the involvement of multiple elements.

Boot Kernel

The Boot Kernel is the root of trust for a secure system. Since the boot kernel is developed by Analog Devices, Inc and is stored in ROM and can't be changed, it can be trusted. The boot kernel validates the authenticity of the application binary image that needs to be booted in. It also handles decrypting the binary image if it's encrypted.

Verifying the authenticity of the application binary asserts that

1. The image is not tampered with or altered
2. The image came from a trusted developer

If the image is encrypted, it ensures confidentiality since the boot image is stored on an external storage device and can be more easily read or stolen than in the part.

Once the boot kernel can verify and optionally decrypt the boot image, it can be loaded into the processor for execution. At this point, the *chain of trust* is continued with the verified application.

Secure Booting

Secure booting is when the boot kernel uses cryptographic algorithms to perform checks on the application binary and to decrypt it. When security is enabled, signature verification and optional decryption on the application binary are both performed by secure boot. See Security Mode Configuration.

Secure Core

In a system with multiple requester and completer resources, not everything is considered secure. There are secure and non-secure peripherals and secure and non-secure segments of memory.

For a system to be considered secure, a secure core that can access and execute instructions (verified application) from secure memory must be configured. Typically, in a single core processor, either the core security is hardwired, or the core can switch between secure and non-secure modes.

System Protection Unit (SPU)

The SPU in the system serves two functions. First, it acts as a gatekeeper, guarding against non-secure accesses to secure resources (peripherals). Second, it is used to define which resources in the system are secure or non-secure requesters and which resources in the system are secure or non-secure completers.

NOTE: Though the SPU can be configured by secure or non-secure requester, the first steps that the verified secure application must perform are:

1. Configure the SPU as a secure completer itself so only other secure requesters can configure it
2. Define and configure the secure requesters and completers using the SPU

This way, once the SPU is secure and other secure requesters and completers are configured, non-secure requesters cannot tamper with the security privileges of secure requesters and completers nor can non-secure resources be changed to a secure resource.

System Memory Protection Unit (SMPU)

Similar to the SPU protecting the MMR address range for peripherals, the SMPU can guard memory ranges or pages. Memory pages can be configured as secure or non-secure. Again, like in the case of the SPU, the SMPU can guard against non-secure transaction attempts to secure memory.

NOTE: A verified application should reside in memory configured as secure. By default the SMPU has all memory configured as secure. When the program needs to update the memory protections using the SMPU, the application and its data should remain in secure memory.

Debug

The typical way of accessing a system is through the debug port via a JTAG or serial wire interface. When these access points are not secure, sensitive IP such as cipher keys can be exposed and code can be changed to disable other security settings. To guard against this type of attack or security hole and still provide debugging capabilities for a developer, a debug unit with security features is used.

When the developer first receives the part, they can define a JTAG/DEBUG key that is programmed into OTP memory. Once security is enabled, the debug unit compares the key sent from the host debugger with the key inside the system. If a match occurs, debug access to secure resources are allowed. There is also a provision to bypass the loading of the JTAG/DEBUG key from the OTP memory. See Boot ROM and Booting the Processor

One-Time-Programmable (OTP) Memory

Customer programmable OTP memory is used to safely and securely store sensitive information such as cipher keys.

In a public key algorithm (for example, ECDSA which is used in secure boot), the public key is used to verify the digital signature that accompanies the application binary. The private key is used by the developer on the host development machine to create the digital signature. The public and private key pair is unique, plus the public key needs to be stored in non-volatile, one-time-programmable memory. When the public key can be changed then users can generate their own public/private key pairs and successfully boot in malicious code. This can either re-purpose the part or change security configurations to allow easier access to sensitive information stored elsewhere in the OTP memory.

When security is not enabled, OTP can be accessed freely. When a user decides to set the LOCK bit in OTP memory to enable security, portions of the OTP, specifically the first 6K bits and another 1K bits for the customer boot information are also locked. From then on, only secure requesters can access those memory regions.

Cryptographic Accelerators

Cryptographic algorithms are mathematical tools to help provide security. Hardware engines provide some advantages but are not necessarily required. For computationally expensive operations like those used in Elliptic Curve

Cryptography, like ECDSA used in secure boot, the operations can be accelerated while the core performs other tasks. Also, it's less likely that the hardware engine can be hacked to change the results.

Security Mode Configuration

Security as a feature does not necessarily be employed by the user. There are no steps to disable security, if security protection is not required. The part does not have security enabled.

To use the security features, perform the following steps:

- Generate the public/private key pair on the host development machine.[1]
- Program OTP memory with the public key.[2]
- Program OTP memory with the decryption key if the application binary needs to be encrypted.[2]
- Program OTP memory with the debug/JTAG key.[2]
- Develop the application and sign it, creating the digital signature with the private key.[1]
- If confidentiality is required, encrypt the application binary before signing it.
- Set the LOCK bit in OTP memory to enable security. After this, subsequent boots are secure boots.[2, 3]

[1] Software tools are provided with developments tools to generate keys, sign boot streams, and also perform encryption. Refer to the development tools manuals for details on how to use.

[2]Refer to the OTP Chapter for programming the OTP and other related information.

[3]Refer to the Booting Chapter for more information on Secure Booting and other related information.

Status and Error Signals

In a fully functional secure system, non-secure resources should not even attempt to access a secure resource. When this happens, either the code has been altered or replaced with malicious code or the system contains a coding error.

Errors or error events are dependent on the configuration of the SPU and SMPU, the protection units which guard against security violations. In the case of the SMPU, the error can simply be captured, captured and interrupt generated, or the access prevented without capturing any error. It is the developer's responsibility to:

1. Determine when there was an error because of a blocked access.
2. Determine how to handle a blocked access. For example fix the bug (offline) or try to use a different resource (run time).

40 System Protection Unit (SPU)

In a system with multiple system MMR requesters, peripheral configurations can be changed unintentionally, leading to bad data or even system malfunctions. The peripherals are shared resources in the system. The SPU restricts access to certain MMRs, similar to the functionality of a semaphore.

The SPU also protects peripherals based on security settings. It is part of the overall security infrastructure of the processor.

SPU Features

The SPU has the following features:

- Write-protect system MMR from certain system requesters and core requesters.
- Simultaneously lock multiple peripheral configuration registers through a global lock mechanism.
- Write-protect and block access to its own write-protection registers from other system requesters.
- Defined security privileges to peripherals and system resources.
- Security protection to guard secure peripheral MMRs against non-secure accesses.

SPU Functional Description

The following sections provide information on the function of the SPU.

ADSP-SC59x SPU Register List

The System Protection Unit (SPU) provides a set of registers that can protect system resources from errant writes. The protection categories are global lock (protects configuration registers) and write protect register lock (protects the write protect register). For more information on SPU functionality, see the SPU register descriptions.

Table 40-1: ADSP-SC59x SPU Register List

Name	Description
SPU_CTL	Control Register
SPU_SECURECHK	Secure Check Register

Table 40-1: ADSP-SC59x SPU Register List (Continued)

Name	Description
SPU_SECURECTL	Secure Control Register
SPU_SECUREC[n]	Secure Core Registers
SPU_SECUREP[n]	Secure Peripheral Register
SPU_STAT	Status Register
SPU_WP[n]	Write Protect Register n

ADSP-SC59x SPU Interrupt List

Table 40-2: ADSP-SC59x SPU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
216	SPU0_INT	SPU0 Interrupt	Level	

Peripheral Register Write Protection

The SPU has a write-protection register (SPU_WP[n]) associated with each peripheral. Each of these write-protection registers has the exact same bits that correspond to a particular SMMR requester (for example, Core 0, MDMA). When the bits are set, the SPU locks the corresponding SMMR requesters from accessing the register address space of the associated peripheral. The bits in the register can be cleared to allow access to the registers of the peripheral again. When the SPU initializes the write-protection register, any writes that are in-progress complete before the SPU blocks subsequent writes.

In the ***SPU Write Protect Registers*** figure, each write-protect register in the SPU is associated with a particular peripheral.

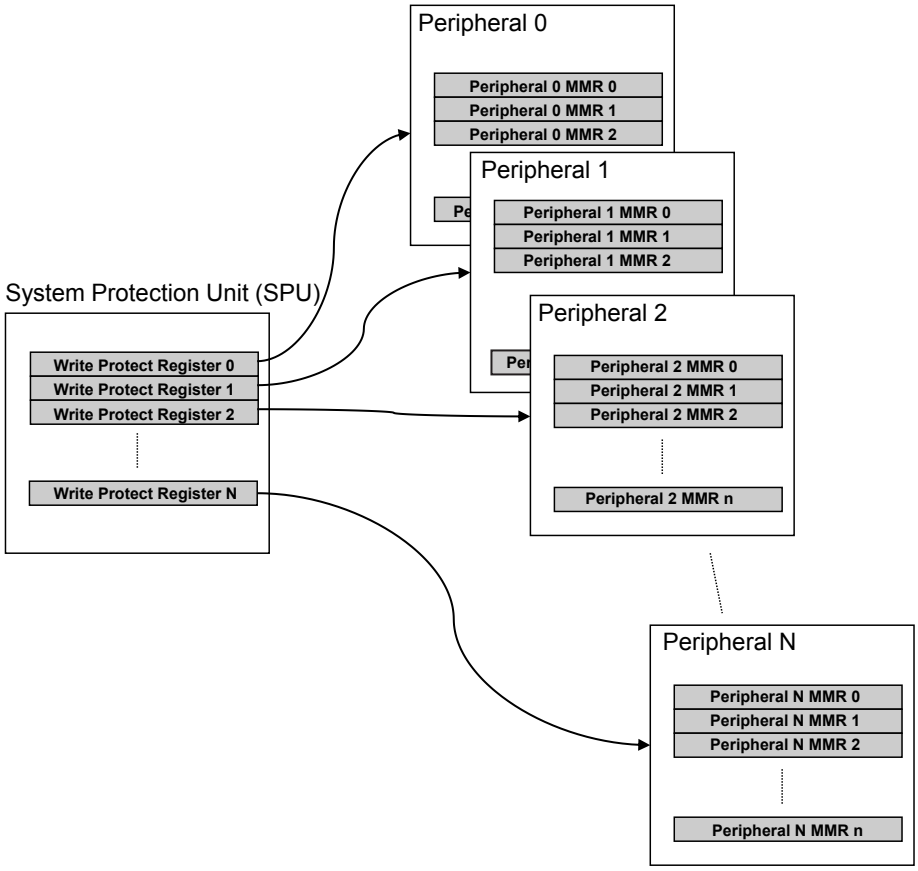


Figure 40-1: SPU Write Protect Registers

In the figure, a write-protect register in the SPU module blocks write-attempts to the MMR space of the associated peripheral. The bits in the write-protect register specify from which requesters to block write-access.

NOTE: A SPU write protection register (`SPU_WP[n]`) exists for the SPU alone. If all defined bits are set in this register for the SPU, any configurations in the SPU are locked and cannot be changed. Only a system reset can restore access to the SPU.

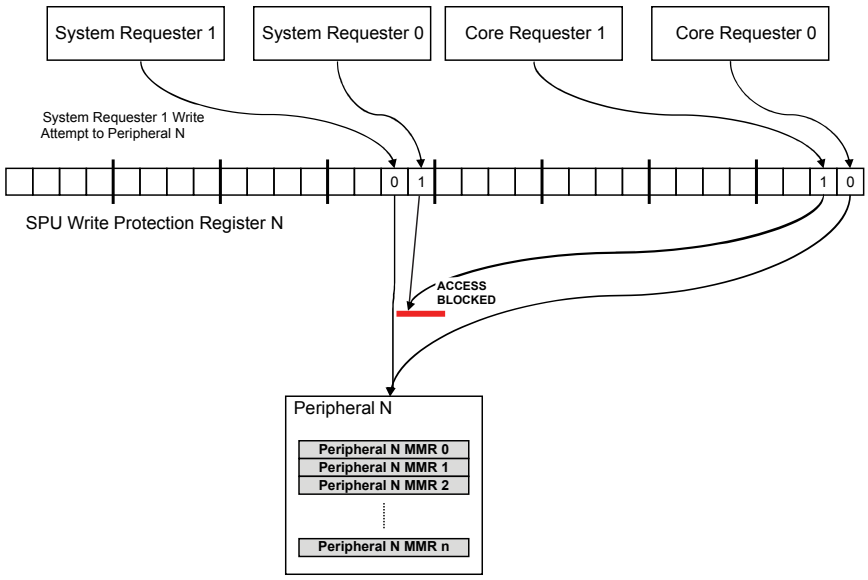


Figure 40-2: SPU Write-Protect Register Blocking Access from System Requester 0 and Core Requester 1

Global Locking

The SPU also has global locking capability. When enabled by setting `SPU_CTL.GLCK` bit field to a value other than 0xAD, a system-wide global lock signal is active. Some peripherals have a lock enable bit in their control register. When this bit is set, the peripheral recognizes the global lock signal and blocks further write-accesses to its own control register. Access to the configuration register of the peripheral is enabled when the global lock is turned off in the SPU.

The ***Global Locking*** figure is a conceptual diagram. The diagram shows how the SPU module (or any peripheral) blocks any write attempts to its control register when:

- The global lock signal from the SPU is active, and
- The global lock enable bit is set in the control register of the peripheral

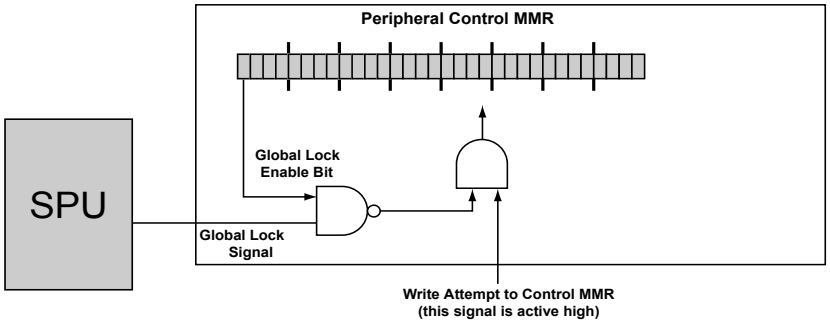


Figure 40-3: Global Locking

The SPU can write-protect its own registers. When the `SPU_CTL.WPLCK` bit is set and global locking is enabled, the SPU blocks accesses to the SPU write-protection registers. To enable write access to the write-protection registers in the SPU, disable the global locking.

SPU Block Diagram

The ***SPU System-Level Block Diagram*** shows a system-level block diagram of where the SPU is located in the system. It resides between the SMMR interface and the system crossbar. Depending on the configuration of the SPU write-protect registers, it can block access to some peripherals from certain SMMR requesters.

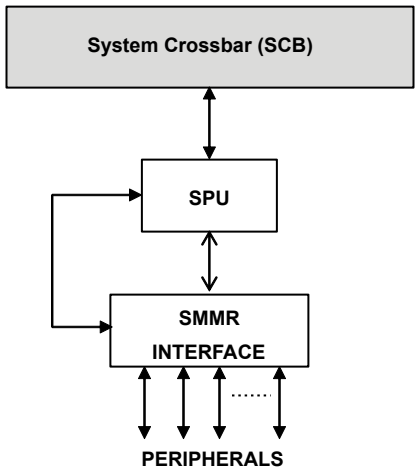


Figure 40-4: SPU System-Level Block Diagram

SPU Architectural Concepts

As shown in the block diagram, the SPU sits between the system crossbar (SCB) and the SMMR interface to the peripherals. The SPU gates any MMR access to any peripheral from any requester that comes through the SCB. Depending on the configuration of the write-protection registers in the SPU, the SPU does or does not allow the MMR write to go through.

The SPU also checks whether the transaction is a secure or non-secure transaction and blocks it according to the configured security setting for the target destination. A secure requester can generate secure read or secure write transactions which can access secure or non-secure completers. A non-secure requester can generate non-secure read or non-secure write transactions and can only access non-secure completers.

SPU Event Control

The system protection unit provides write-protection against MMRs peripherals and its own write-protect registers. If a write attempt is made to any locked MMR peripheral the SPU has write-protected, it blocks the write. The SPU generates a bus error to the requester that attempted the write. That requester does or does not generate an event, based on the returned error.

The SPU can be configured to generate an interrupt for the write-protection violation by setting the `SPU_CTL.PINTEN` bit. The SPU can also be configured to generate an interrupt for a security violation by setting the `SPU_SECURECTL.SINTEN` bit. If either one or both bits is triggered, the `SPU_STAT.VIRQ` bit is set.

The SPU can also lock its own registers from write attempts. If a write-attempt is made to a locked register in the SPU, the SPU blocks it and records it as an error in the SPU_STAT.LWERR bit. Again, the SPU generates a bus error to the requester that attempted the write.

The requester does or does not generate an event, based on the returned error.

The SPU does not generate an event for a blocked write access to an SPU register. If the SPU_CTL.PINTEN bit is set, the SPU triggers an interrupt for this blocked access attempt.

The global lock is enabled by setting the SPU_CTL.GLCK bit to something other than 0xAD. If the lock bit is set in that same configuration register, a peripheral can block write access to its configuration register. When the SPU blocks a write attempt, the peripheral logs and reports the failed attempt. The SPU is unaware and therefore does not provide any indication of a failed write attempt to the configuration register of the peripheral.

SPU Programming Model

The system protection unit (SPU) consists of write-protect and access-protect registers. Each one corresponds to a different peripheral instance. Bits in the write-protect registers correspond to system requesters that can modify the MMR contents of the peripherals. By writing to these write-protect registers, the corresponding memory-mapped registers of the peripheral are write-protected against requesters whose bits in the write-protect register are set.

The SPU globally locks the control register of the peripheral. Peripherals that support this feature have a lock enable bit in their control register. The peripheral blocks any additional write attempts to its control register from any requester when:

- The global lock signal is active from the SPU, and
- The lock enable bit of the peripheral is set

If the lock enable bit of a peripheral is not set and the global lock signal is active, access to that control register of the peripheral is still allowed. To grant access again, disable the global lock signal from the SPU by writing the value 0xAD into the SPU_CTL.GLCK bit field.

Another protection mechanism that the SPU offers is write-protection against the write-protection registers. If the write protect register lock bit (SPU_CTL.WPLCK) is set and the global lock signal is active, writes to the write-protect registers of the SPU are blocked. To reenable access to the write-protect registers in the SPU, deactivate the global lock signal by writing 0xAD into the SPU_CTL.GLCK bit field.

For security, the SPU provides a set of SPU_SECUREC[n] registers (one for each processor core from Analog Devices) to configure their security settings. The SPU also provides a set of SPU_SECUREP[n] registers (one for each peripheral instance) to configure their security settings.

Enabling and Disabling the SPU

The SPU is always operating. There are no bits to enable or disable the SPU. The SPU configuration can be updated at any time. Any ongoing transactions finish before a new configuration is in effect. By default, the SPU does not write-protect any of the MMRs.

Write-Protecting the SPU

The SPU is treated like any other peripheral in the system. As such, the SPU also has an associated write-protection register. If this write-protection register is configured to block all writes from all requesters, any SPU configuration remains the same until the next system reset.

Checking the Security State

In some cases while running a peripheral, an application system requester does not know whether they are a secure requester generating secure transactions or not. The SPU provides a means for checking the security state of the requester through the SPU_SECURECHK register. When read by a secure requester, the register reads 0xFFFFFFFF and when read by a non-secure requester, the value is 0x00000000.

SPU Mode Configuration

The SPU can provide address range-wide protection by write-protecting the peripherals MMR address range from system MMR requesters. It can also provide register wide protection using global locking. Peripherals that support this feature can enable it in their respective configuration register. When the SPU enables the global lock signal, all subsequent writes to the configuration register of the peripheral are blocked until the global lock signal is deasserted. Similarly, the write-protection registers of the SPU can be write-protected using the global lock signal as well. The SPU uses all these modes of operation together.

Locking Write-Protect Registers

Use the following steps to lock (write-protect) a register.

1. Set the SPU_CTL.WPLCK bit and configure the SPU_CTL.GLCK field to something other than 0xAD.

The SPU write-protect registers are blocked from further write accesses.

Protecting a Peripheral

Use the following procedure to protect a peripheral.

1. Determine which peripheral needs protection and locate the corresponding write-protect register (SPU_WP[n]) in the SPU. See the section.
2. Determine the SMMR requesters from which the peripheral needs protection. Then, set the corresponding bit or bits in the write-protect register (SPU_WP[n]) for the peripheral. See the section.

After setting the write-protect register for the particular peripheral, the identified SMMR requesters are blocked from writing to any MMR in the address space of the peripheral. This block remains in place until the bits in the write-protect register are cleared.

Configuring Security Privileges of a Peripheral

Use the following procedure to configure the security privileges of a peripheral.

1. Determine the peripheral and its corresponding secure peripheral register (SPU_SECUREP[n]) in the SPU. See the section.
2. If the peripheral is to be a secure completer that only accepts secure transactions, set bit 0 (SPU_SECUREP[n].SSEC).
3. If the peripheral is to be a secure requester that generates secure transactions (keeping in mind not all peripherals can be requesters), set bit 1 (SPU_SECUREP[n].MSEC).

This procedure sets the security privileges of a peripheral.

NOTE: Only a secure requester can set security privileges, keeping the chain of trust intact. If a non-secure requester configures the security privileges, it can undermine security protection.

ADSP-SC59x SPU Register Descriptions

System Protection Unit (SPU) contains the following registers.

Table 40-3: ADSP-SC59x SPU Register List

Name	Description
SPU_CTL	Control Register
SPU_SECURECHK	Secure Check Register
SPU_SECURECTL	Secure Control Register
SPU_SECUREC[n]	Secure Core Registers
SPU_SECUREP[n]	Secure Peripheral Register
SPU_STAT	Status Register
SPU_WP[n]	Write Protect Register n

Control Register

The SPU control register (SPU_CTL) provides a global lock for configuration registers as well as control for locking the write protect (SPU_WP[n]) registers. It also controls the generation of an interrupt to report blocked accesses.

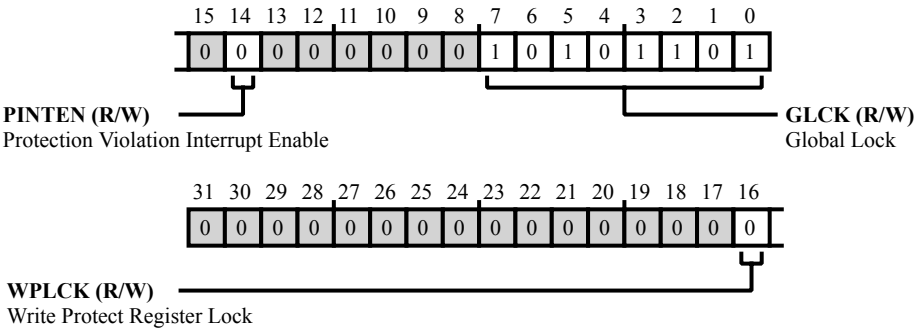


Figure 40-5: SPU_CTL Register Diagram

Table 40-4: SPU_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	WPLCK	Write Protect Register Lock. When the SPU_CTL.WPLCK bit is set in combination with the SPU_CTL.GLCK bit, writes to the SPU's write protect registers are blocked and return an error.	
		0	Disable
		1	Enable
14 (R/W)	PINTEN	Protection Violation Interrupt Enable. When the SPU_CTL.PINTEN bit is set (=1), a block of any transaction according to the configured settings produces an interrupt.	
		0	Disable
		1	Enable
7:0 (R/W)	GLCK	Global Lock. The SPU_CTL.GLCK controls the global lock signal. The global lock signal provides register-based write protection. Writing 0xAD to this field disables the lock, and writing any other value enables the lock.	

Secure Check Register

The SPU_SECURECHK register reads by secure requesters return 0xFFFFFFFF. Reads by non-secure requesters return 0x00000000.

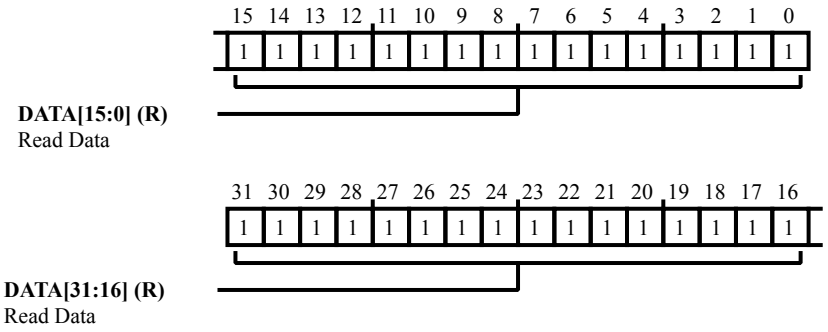


Figure 40-6: SPU_SECURECHK Register Diagram

Table 40-5: SPU_SECURECHK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	DATA	Read Data. The SPU_SECURECHK.DATA bit field performs reads. Reads by secure requesters return 0xFFFFFFFF. Reads by non-secure requesters return 0x00000000.

Secure Control Register

The SPU Secure Control Register (SPU_SECURECTL) allows the user to lock write access to all the SPU_SECUREC[n] and SPU_SECUREP[n] registers as well as configure the interrupt generation in an event of a security error. It also allows bulk clear of the SSEC bits and/or MSEC bits in the SPU_SECUREP[n] registers.

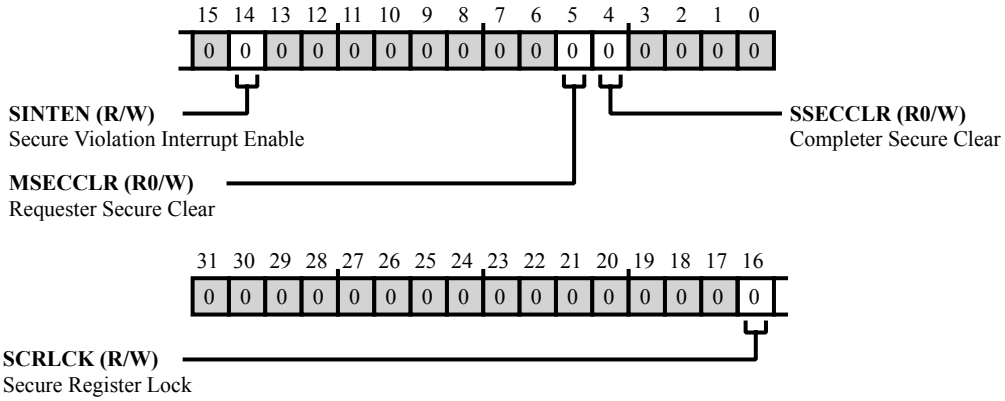


Figure 40-7: SPU_SECURECTL Register Diagram

Table 40-6: SPU_SECURECTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	SCRLCK	Secure Register Lock. When the SPU_SECURECTL.SCRLCK bit is set in combination with the SPU_CTL.GLCK bit, writes to the Security Configuration registers (SPU_SECUREC[n] and SPU_SECUREP[n]) are blocked and return an error which is captured in the SPU_STAT.LWERR bit.	
		0	Disable
		1	Enable
14 (R/W)	SINTEN	Secure Violation Interrupt Enable. The SPU_SECURECTL.SINTEN bit generates an interrupt if a security violation was captured. Interrupt status is provided in the SPU_STAT.VIRQ bit.	
		0	Disable
		1	Enable
5 (R0/W)	MSECCLR	Requester Secure Clear. When the SPU_SECURECTL.MSECCLR bit is set, the SPU_SECUREP[n].MSEC bits in all SPU_SECUREP[n] registers are cleared. The SPU_SECURECTL.MSECCLR bit always reads back as a 0.	
		0	No Action
		1	Clear All Requester Secure Control Bits

Table 40-6: SPU_SECURECTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R0/W)	SSECCLR	Completer Secure Clear. When the SPU_SECURECTL.SSECCLR bit is set, the SPU_SECUREP[n].SSEC bits in all SPU_SECUREP[n] registers are cleared. The SPU_SECURECTL.SSECCLR bit always reads back as a 0.	
		0	No Action
		1	Clear All Completer Secure Control Bits

Secure Core Registers

A SPU register exists for every DSP core in the system. The bits enable or disable security for features in the core.

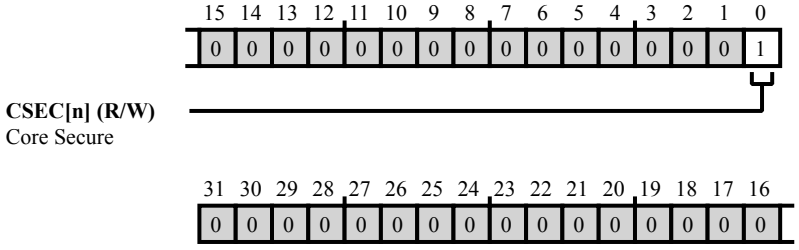


Figure 40-8: SPU_SECUREC[n] Register Diagram

Table 40-7: SPU_SECUREC[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	CSEC[n]	Core Secure. The `SPU_SECUREC[n].CSEC[n]` bit controls whether non-secure accesses are allowed to L1 memory of the processor core. When =1, the core (as a completer) is set as secure meaning only secure transactions are allowed to L1.	
		0	Disable
		1	Enable

Secure Peripheral Register

In the system, each SPU_SECUREP[n] register is assigned to a specific MMR address range associated with one peripheral. Each SPU_SECUREP[n] has a Completer Secure (SSEC) bit and a Requester Secure (MSEC) bit. When the SSEC bit is set, the SPU will only allow secure requesters generating secure transactions to access the peripheral's MMR address space. When the MSEC bit is set, the associated peripheral will be secure and will generate secure transactions.

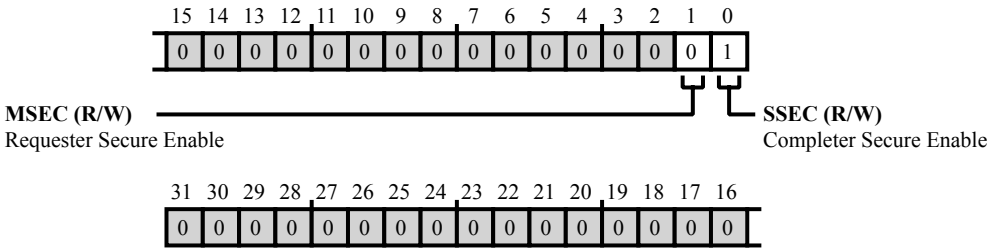


Figure 40-9: SPU_SECUREP[n] Register Diagram

Table 40-8: SPU_SECUREP[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	MSEC	Requester Secure Enable. The SPU_SECUREP[n].MSEC bit controls whether the peripheral generates secure transactions as a requester. When clear (=0), the peripheral generates non-secure transactions as a requester (if applicable). When set (=1), the peripheral generates secure transactions as a requester.	
		0	Disable
		1	Enable
0 (R/W)	SSEC	Completer Secure Enable. The SPU_SECUREP[n].SSEC bit controls whether the peripheral is protected from non-secure transactions. When clear (=0), the security status of the transaction is ignored. When set (=1), only secure transactions are allowed to access the address space of the peripheral and non-secure transactions are blocked.	
		0	Disable
		1	Enable

Status Register

The SPU_STAT register indicates if there have been any errors, active interrupts and global lock status.

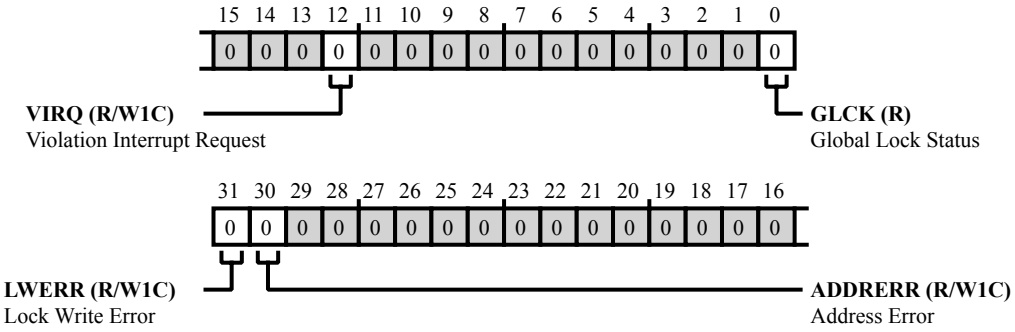


Figure 40-10: SPU_STAT Register Diagram

Table 40-9: SPU_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Lock Write Error. The SPU_STAT.LWERR indicates whether there has been an attempted write to a register in the SPU with its lock bit (SPU_CTL.WPLCK or SCRLCK) set while SPU_CTL.GLCK was asserted. This bit is W1C.	
		0	Inactive
		1	Active
30 (R/W1C)	ADDRERR	Address Error. The SPU_STAT.ADDRERR indicates whether there has been an attempted write to a read-only register or an access an invalid address in the SPU MMR address range. This bit is W1C.	
		0	Inactive
		1	Active
12 (R/W1C)	VIRQ	Violation Interrupt Request. The SPU_STAT.VIRQ bit indicates that a security and/or protection violation has been detected and interrupt asserted. This is a W1C bit.	
		0	Inactive
		1	Active
0 (R/NW)	GLCK	Global Lock Status. The SPU_STAT.GLCK indicates whether the global lock is enabled or disabled.	
		0	Disabled (global_lock=0)
		1	Enabled (global_lock=1)

Write Protect Register n

In the system, each SPU_WP[n] register is assigned to a specific MMR address range associated with one peripheral. When the appropriate bits are set, writes to the peripheral from a specific requester are blocked and an error is returned to the requester. For more information, see the processor specific additional information for the SPU_WP[n] register.

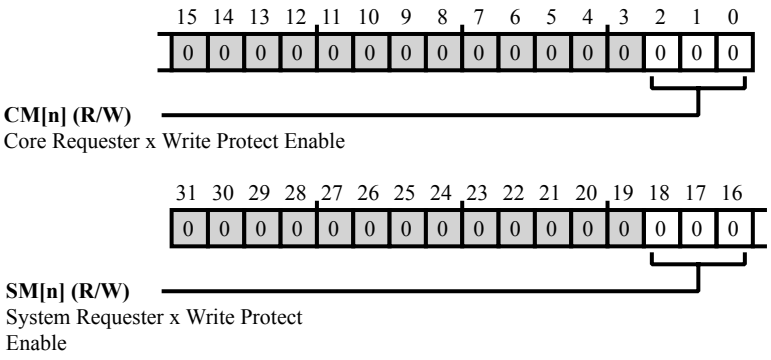


Figure 40-11: SPU_WP[n] Register Diagram

Table 40-10: SPU_WP[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18:16 (R/W)	SM[n]	System Requester x Write Protect Enable. The SPU_WP[n].SM[n] bits correspond to different system requesters in the system. When a particular bit is set in this field, the corresponding system requester cannot write to the corresponding peripheral's MMR address space. The write attempt is blocked by the SPU.
2:0 (R/W)	CM[n]	Core Requester x Write Protect Enable. The SPU_WP[n].CM[n] bits correspond to different cores in the system. When a particular bit is set in this field, the corresponding core cannot write to the corresponding peripheral's MMR address space. The write attempt is blocked by the SPU.

Write-Protect, Secure Peripheral, and Secure Core Registers

The SPU consists of a collection of write-protect registers each of which are associated with a specific peripheral or target. The SPU also has a collection of secure peripheral registers which are also associated with specific peripherals. The table gives the write-protect register and secure peripheral number for each of the peripherals that are provided with write protection and security through the SPU. The SPU for the processor is configured with 136 write-protect registers and also 136 secure peripheral registers. The number corresponding to a peripheral correlates to both the Write-Protect register and the Secure Peripheral register.

For each processor, there are different numbers of controllers that are able to access the SMMR space

The SPU_WP[n] register shows which bits enable the protection against which controller.

For each peripheral, there is a corresponding write-protect register, SPU_WP[n], and secure peripheral register, SPU_SECUREP[n]. The table shows the write-protect register and secure peripheral number for each peripheral.

Table 40-11: Write-Protect Register and Secure Peripheral Number(n)

No.	Peripheral/Block Name	SPU WPn Mapping	SECUREPn Mapping	Controller Capable
0	Reserved	WP0	SECUREP0	N/A
1	DMC0_PRG_CDC_BRIDGE	WP1	SECUREP1	N/A
2	SPIF GPV SPACE	WP2	SECUREP2	N/A
3	LP0	WP3	SECUREP3	N/A
4	LP1	WP4	SECUREP4	N/A
5	LP0 DDE	WP5	SECUREP5	Yes
6	LP1 DDE	WP6	SECUREP6	Yes
7	TWI3	WP7	SECUREP7	N/A
8	TWI4	WP8	SECUREP8	N/A
9	TWI5	WP9	SECUREP9	N/A
10	TWI0	WP10	SECUREP10	N/A
11	TWI1	WP11	SECUREP11	N/A
12	TWI2	WP12	SECUREP12	N/A
13	SPORT0A	WP13	SECUREP13	N/A
14	SPORT0B	WP14	SECUREP14	N/A
15	SPORT1A	WP15	SECUREP15	N/A
16	SPORT1B	WP16	SECUREP16	N/A
17	SPORT2A	WP17	SECUREP17	N/A
18	SPORT2B	WP18	SECUREP18	N/A
19	SPORT3A	WP19	SECUREP19	N/A
20	SPORT3B	WP20	SECUREP20	N/A
21	SPORT4A	WP21	SECUREP21	N/A
22	SPORT4B	WP22	SECUREP22	N/A
23	SPORT5A	WP23	SECUREP23	N/A
24	SPORT5B	WP24	SECUREP24	N/A
25	SPORT6A	WP25	SECUREP25	N/A
26	SPORT6B	WP26	SECUREP26	N/A
27	SPORT7A	WP27	SECUREP27	N/A
28	SPORT7B	WP28	SECUREP28	N/A

Table 40-11: Write-Protect Register and Secure Peripheral Number(n) (Continued)

No.	Peripheral/Block Name	SPU WPn Mapping	SECUREPn Mapping	Controller Capable
29	UART0	WP29	SECUREP29	N/A
30	UART1	WP30	SECUREP30	N/A
31	UART2	WP31	SECUREP31	N/A
32	UART3	WP32	SECUREP32	N/A
33	PORTA	WP33	SECUREP33	N/A
34	PORTB	WP34	SECUREP34	N/A
35	PORTC	WP35	SECUREP35	N/A
36	PORTD	WP36	SECUREP36	N/A
37	PORTE	WP37	SECUREP37	N/A
38	PORTF	WP38	SECUREP38	N/A
39	PORTG	WP39	SECUREP39	N/A
40	PORTH	WP40	SECUREP40	N/A
41	PORTI	WP41	SECUREP41	N/A
42	PADS	WP42	SECUREP42	N/A
43	PINT0	WP43	SECUREP43	N/A
44	PINT1	WP44	SECUREP44	N/A
45	PINT2	WP45	SECUREP45	N/A
46	PINT3	WP46	SECUREP46	N/A
47	PINT4	WP47	SECUREP47	N/A
48	PINT5	WP48	SECUREP48	N/A
49	PINT6	WP49	SECUREP49	N/A
50	PINT7	WP50	SECUREP50	N/A
51	Reserved	WP51	SECUREP51	N/A
52	Reserved	WP52	SECUREP52	N/A
53	WDT0	WP53	SECUREP53	N/A
54	WDT1	WP54	SECUREP54	N/A
55	WDT2	WP55	SECUREP55	N/A
56	CNT0	WP56	SECUREP56	N/A
57	OTP MMR	WP57	SECUREP57	N/A
58	SMPU-OTP	WP58	SECUREP58	N/A
59	Reserved	WP59	SECUREP59	N/A

Table 40-11: Write-Protect Register and Secure Peripheral Number(n) (Continued)

No.	Peripheral/Block Name	SPU WPn Mapping	SECUREPn Mapping	Controller Capable
60	HADC0	WP60	SECUREP60	N/A
61	TMU0	WP61	SECUREP61	N/A
62	TMR	WP62	SECUREP62	N/A
63	SPORT0A DDE	WP63	SECUREP63	Yes
64	SPORT0B DDE	WP64	SECUREP64	Yes
65	SPORT1A DDE	WP65	SECUREP65	Yes
66	SPORT1B DDE	WP66	SECUREP66	Yes
67	SPORT2A DDE	WP67	SECUREP67	Yes
68	SPORT2B DDE	WP68	SECUREP68	Yes
69	SPORT3A DDE	WP69	SECUREP69	Yes
70	SPORT3B DDE	WP70	SECUREP70	Yes
71	SPORT4A DDE	WP71	SECUREP71	Yes
72	SPORT4B DDE	WP72	SECUREP72	Yes
73	SPORT5A DDE	WP73	SECUREP73	Yes
74	SPORT5B DDE	WP74	SECUREP74	Yes
75	SPORT6A DDE	WP75	SECUREP75	Yes
76	SPORT6B DDE	WP76	SECUREP76	Yes
77	SPORT7A DDE	WP77	SECUREP77	Yes
78	SPORT7B DDE	WP78	SECUREP78	Yes
79	UART0RX DDE	WP79	SECUREP79	Yes
80	UART0TX DDE	WP80	SECUREP80	Yes
81	UART1RX DDE	WP81	SECUREP81	Yes
82	UART1TX DDE	WP82	SECUREP82	Yes
83	UART2RX DDE	WP83	SECUREP83	Yes
84	UART2TX DDE	WP84	SECUREP84	Yes
85	UART3RX DDE	WP85	SECUREP85	Yes
86	UART3TX DDE	WP86	SECUREP86	Yes
87	PPI0 DDE0	WP87	SECUREP87	Yes
88	PPI0 DDE1	WP88	SECUREP88	Yes
89	OSPI	WP89	SECUREP89	N/A
90	Reserved	WP90	SECUREP90	N/A

Table 40-11: Write-Protect Register and Secure Peripheral Number(n) (Continued)

No.	Peripheral/Block Name	SPU WPn Mapping	SECUREPn Mapping	Controller Capable
91	EPPI0	WP91	SECUREP91	N/A
92	Reserved	WP92	SECUREP92	N/A
93	SPI0TX DDE	WP93	SECUREP93	Yes
94	SPI0RX DDE	WP94	SECUREP94	Yes
95	SPI1TX DDE	WP95	SECUREP95	Yes
96	SPI1RX DDE	WP96	SECUREP96	Yes
97	SPI2TX DDE	WP97	SECUREP97	Yes
98	SPI2RX DDE	WP98	SECUREP98	Yes
99	SPI3TX DDE	WP99	SECUREP99	Yes
100	SPI3RX DDE	WP100	SECUREP100	Yes
101	SPI0	WP101	SECUREP101	N/A
102	SPI1	WP102	SECUREP102	N/A
103	SPI2	WP103	SECUREP103	N/A
104	SPI3	WP104	SECUREP104	N/A
105	EMAC0	WP105	SECUREP105	Yes
106	EMAC1	WP106	SECUREP106	Yes
107	CAN0	WP107	SECUREP107	N/A
108	CAN1	WP108	SECUREP108	N/A
109	DMC0	WP109	SECUREP109	N/A
110	Reserved	WP110	SECUREP110	N/A
111	Reserved	WP111	SECUREP111	N/A
112	Reserved	WP112	SECUREP112	N/A
113	L2CTL0	WP113	SECUREP113	N/A
114	SMPU L2CTL0 CL2 0	WP114	SECUREP114	N/A
115	SMPU L2CTL0 DL2 0	WP115	SECUREP115	N/A
116	SMPU L2CTL0 CL2 1	WP116	SECUREP116	N/A
117	SMPU L2CTL0 DL2 1	WP117	SECUREP117	N/A
118	SMPU L2CTL0 CL2 2	WP118	SECUREP118	N/A
119	SEC0	WP119	SECUREP119	N/A
120	TRU0	WP120	SECUREP120	N/A
121	SPU0	WP121	SECUREP121	N/A

Table 40-11: Write-Protect Register and Secure Peripheral Number(n) (Continued)

No.	Peripheral/Block Name	SPU WPn Mapping	SECUREPn Mapping	Controller Capable
122	RCU0	WP122	SECUREP122	N/A
123	CGU0	WP123	SECUREP123	N/A
124	CGU1	WP124	SECUREP124	N/A
125	CDU0	WP125	SECUREP125	N/A
126	DPM0	WP126	SECUREP126	N/A
127	PLL0	WP127	SECUREP127	N/A
128	PLL1	WP128	SECUREP128	N/A
129	SWU_L2CTL0_CL2_0	WP129	SECUREP129	N/A
130	SWU_L2CTL0_DL2_0	WP130	SECUREP130	N/A
131	SWU_L2CTL0_CL2_1	WP131	SECUREP131	N/A
132	SWU_L2CTL0_DL2_1	WP132	SECUREP132	N/A
133	SWU SMMR	WP133	SECUREP133	N/A
134	SWU_L2CTL0_CL2_2	WP134	SECUREP134	N/A
135	MDMA2	WP135	SECUREP135	Yes
136	MDMA3	WP136	SECUREP136	Yes
137	MDMA7	WP137	SECUREP137	Yes
138	MLB0	WP138	SECUREP138	Yes
139	SWU DMC0	WP139	SECUREP139	N/A
140	SMPU DMC0	WP140	SECUREP140	N/A
141	SMPU SPI2	WP141	SECUREP141	N/A
142	MEC0	WP142	SECUREP142	N/A
143	MEC1	WP143	SECUREP143	N/A
144	MEC2	WP144	SECUREP144	N/A
145	CRC0	WP145	SECUREP145	N/A
146	CRC1	WP146	SECUREP146	N/A
147	MDMA0 DDE0 CRC0	WP147	SECUREP147	Yes
148	MDMA0 DDE1 CRC0	WP148	SECUREP148	Yes
149	MDMA1 DDE0 CRC1	WP149	SECUREP149	Yes
150	MDMA1 DDE1 CRC1	WP150	SECUREP150	Yes
151	MDMA4 DDE0 CRC2	WP151	SECUREP151	Yes
152	MDMA4 DDE1 CRC2	WP152	SECUREP152	Yes

Table 40-11: Write-Protect Register and Secure Peripheral Number(n) (Continued)

No.	Peripheral/Block Name	SPU WPn Mapping	SECUREPn Mapping	Controller Capable
153	MDMA5 DDE0 CRC3	WP153	SECUREP153	Yes
154	MDMA5 DDE1 CRC3	WP154	SECUREP154	Yes
155	SWU SPI2	WP155	SECUREP155	N/A
156	MISC REG	WP156	SECUREP156	N/A
157	CRC2	WP157	SECUREP157	N/A
158	CRC3	WP158	SECUREP158	N/A
159	MDMA6	WP159	SECUREP159	Yes
160	A55 MISC	WP160	SECUREP160	N/A
161	A55 SYS TIMER WRITE	WP161	SECUREP161	N/A
162	A55 SYS TIMER READ	WP162	SECUREP162	N/A
163	STM0	WP163	SECUREP163	N/A
164	SH1_IIR1	WP164	SECUREP164	N/A
165	SH1_IIR2	WP165	SECUREP165	N/A
166	SH1_IIR3	WP166	SECUREP166	N/A
167	SH1_FIR	WP167	SECUREP167	N/A
168	SH1_IIR0	WP168	SECUREP168	N/A
169	SH0_IIR1	WP169	SECUREP169	N/A
170	SH0_IIR2	WP170	SECUREP170	N/A
171	SH0_IIR3	WP171	SECUREP171	N/A
172	SH0_FIR	WP172	SECUREP172	N/A
173	SH0_IIR0	WP173	SECUREP173	N/A
174	USB0	WP174	SECUREP174	Yes
175	SDIO0	WP175	SECUREP175	Yes
176	DAI0	WP176	SECUREP176	N/A
177	DAI1	WP177	SECUREP177	N/A
178	CRYPTO SPE	WP178	SECUREP178	Yes
179	CRYPTO PKP	WP179	SECUREP179	N/A
180	DLMDMA0	WP180	SECUREP180	Yes
181	DAP ROM	WP181	SECUREP181	N/A
182	SHARC0 DBG	WP182	SECUREP182	N/A
183	SHARC0 CTI	WP183	SECUREP183	N/A

Table 40-11: Write-Protect Register and Secure Peripheral Number(n) (Continued)

No.	Peripheral/Block Name	SPU WPn Mapping	SECUREPn Mapping	Controller Capable
184	SHARC0 PTM	WP184	SECUREP184	N/A
185	STM	WP185	SECUREP185	N/A
186	SHARC1 DBG	WP186	SECUREP186	N/A
187	SHARC1 CTI	WP187	SECUREP187	N/A
188	SHARC1 PTM	WP188	SECUREP188	N/A
189	CSTF	WP189	SECUREP189	N/A
190	ETF	WP190	SECUREP190	N/A
191	ETR	WP191	SECUREP191	N/A
192	TPIU	WP192	SECUREP192	N/A
193	CTI Trace	WP193	SECUREP193	N/A
194	CTI System	WP194	SECUREP194	N/A
195	CoreSight Timestamp Generator	WP195	SECUREP195	N/A
196	A55 Integration ROM	WP196	SECUREP196	N/A
197	A55 DBG	WP197	SECUREP197	N/A
198	A55 PMU	WP198	SECUREP198	N/A
199	A55 CTI	WP199	SECUREP199	N/A
200	A55 ETM	WP200	SECUREP200	N/A
201	TAPC MMR	WP201	SECUREP201	N/A
202	Debug Control	WP202	SECUREP202	N/A
203	SWU C0 S1	WP203	SECUREP203	N/A
204	SWU C0 S2	WP204	SECUREP204	N/A
205	SWU C1 S1	WP205	SECUREP205	N/A
206	SWU C1 S2	WP206	SECUREP206	N/A
207	GIC Port0 0	WP207	SECUREP207	N/A
209	GIC Port0 2	WP209	SECUREP209	N/A
210	GIC Port0 3	WP210	SECUREP210	N/A
212	GIC Port0 5	WP212	SECUREP212	N/A
213	GIC Port0 6	WP213	SECUREP213	N/A

ADSP-SC59x Processor Specific Information

Global Locking

The global lock signal from the SPU along with the peripheral lock bit can be used to provide lock functionality for the control MMR of the peripheral. The Global Lock (`SPU_CTL.GLCK`) field determines whether global lock is enabled or not. Global Lock is disabled if the `SPU_CTL.GLCK` field is 0xAD (default value), otherwise it is enabled. The following is a list of peripherals that have the global lock bit in their control MMR.

- General-Purpose IO (GPIO)
- System Event Controller (SEC0)
- Trigger Routing Unit (TRU0
- Clock Generation Unit (CGU0)
- Clock Generation Unit1 (CGU1)
- Clock Distribution Unit (CDU0)
- Dynamic Power Management (DPM)
- Reset Control Unit (RCU0)
- System Protection Unit (SPU0)
- L2 Memory Controller (L2CTL0)

41 Security Packet Engine (PKTE)

The PKTE is a security packet engine designed to off-load the host processor to improve the speed of applications requiring cryptographic processing. The packet engine contains a set of modules for encryption and decryption, hashing, and pseudo-random number generation.

PKTE Features

The PKTE has the following features.

- Hardware assisted processing for the cryptographic ciphers, hashes, and pseudo-random number generation
- Header and trailer processing for Internet security protocols
- DMA capability to move data in and out of the engine efficiently and to allow the engine to run autonomously while moving data
- Interrupt controller to signal module status and errors
- Clock manager for enabling or disabling different features to save power

NOTE: Not all algorithms, decrypt, and hash functions and extra features are available on all product models. For complete information on included features, see the product-specific data sheet.

NOTE: This packet engine provides support for various network security protocols by processing headers and trailers as well as accelerating cryptographic functions. Not all processors have direct support for Ethernet. As such, the packet engine can still be used if Ethernet is indirectly used.

PKTE Functional Description

The packet engine contains a set of modules for encryption and decryption, hashing, and pseudo-random number generation. The following sections describe these functional blocks.

ADSP-SC59x PKTE Register List

The Security Packet Engine (PKTE) provides security-related features. A set of registers governs PKTE operations. For more information on PKTE functionality, see the PKTE register descriptions.

Table 41-1: ADSP-SC59x PKTE Register List

Name	Description
PKTE_ARC4STATE_ADDR	Packet Engine ARC4 State Record Address
PKTE_ARC4STATE_BUF	Starting Entry of 256-byte ARC4 State Buffer
PKTE_BUF_PTR	Packet Engine Buffer Pointer Register
PKTE_BUF_THRESH	Packet Engine Buffer Threshold Register
PKTE_CDRBASE_ADDR	Packet Engine Command Descriptor Ring Base Address
PKTE_CDSC_CNT	Packet Engine Command Descriptor Count Register
PKTE_CDSC_INCR	Packet Engine Command Descriptor Count Increment Register
PKTE_CFG	Packet Engine Configuration Register
PKTE_CLK_CTL	PE Clock Control Register
PKTE_CONT	PKTE Continue Register
PKTE_CTL_STAT	Packet Engine Control Register
PKTE_DATAIO_BUF	Starting Entry of 256-byte Data Input/Output Buffer
PKTE_DEST_ADDR	Packet Engine Destination Address
PKTE_DMA_CFG	Packet Engine DMA Configuration Register
PKTE_ENDIAN_CFG	Packet Engine Endian Configuration Register
PKTE_HLT_CTL	Packet Engine Halt Control Register
PKTE_HLT_STAT	Packet Engine Halt Status Register
PKTE_IMSK_DIS	Interrupt Mask Disable Register
PKTE_IMSK_EN	Interrupt Mask Enable Register
PKTE_IMSK_STAT	Interrupt Masked Status Register
PKTE_INBUF_CNT	Packet Engine Input Buffer Count Register
PKTE_INBUF_INCR	Packet Engine Input Buffer Count Increment Register
PKTE_INT_CFG	Interrupt Configuration Register
PKTE_INT_CLR	Interrupt Clear Register
PKTE_INT_EN	Interrupt Enable Register
PKTE_IUMSK_STAT	Interrupt Unmasked Status Register
PKTE_LEN	Packet Engine Length Register
PKTE_OUTBUF_CNT	Packet Engine Output Buffer Count Register
PKTE_OUTBUF_DECR	Packet Engine Output Buffer Count Decrement Register
PKTE_RDRBASE_ADDR	Packet Engine Result Descriptor Ring Base Address
PKTE_RDSC_CNT	Packet Engine Result Descriptor Count Registers

Table 41-1: ADSP-SC59x PKTE Register List (Continued)

Name	Description
PKTE_RDSC_DECR	Packet Engine Result Descriptor Count Decrement Registers
PKTE_RING_CFG	Packet Engine Ring Configuration
PKTE_RING_PTR	Packet Engine Ring Pointer Status
PKTE_RING_STAT	Packet Engine Ring Status
PKTE_RING_THRESH	Packet Engine Ring Threshold Registers
PKTE_SA_ADDR	Packet Engine SA Address
PKTE_SA_ARC4IJPTR	ARC4 i and j Pointer Register
PKTE_SA_CMD0	SA Command 0
PKTE_SA_CMD1	SA Command 1
PKTE_SA_IDIGEST[n]	SA Inner Hash Digest Registers
PKTE_SA_KEY[n]	SA Key Registers
PKTE_SA_NONCE	SA Initialization Vector Register
PKTE_SA_ODIGEST[n]	SA Outer Hash Digest Registers
PKTE_SA_RDY	SA Ready Indicator
PKTE_SA_SEQNUM[n]	SA Sequence Number Register
PKTE_SA_SEQNUM_MSK[n]	SA Sequence Number Mask Registers
PKTE_SA_SPI	SA SPI Register
PKTE_SRC_ADDR	Packet Engine Source Address
PKTE_STAT	Packet Engine Status Register
PKTE_STATE_ADDR	Packet Engine State Record Address
PKTE_STATE_BYTE_CNT[n]	State Hash Byte Count Registers
PKTE_STATE_IDIGEST[n]	State Inner Digest Registers
PKTE_STATE_IV[n]	State Initialization Vector Registers
PKTE_USERID	Packet Engine User ID

ADSP-SC59x PKTE Interrupt List

Table 41-2: ADSP-SC59x PKTE Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
161	PKTE0_IRQ	PKTE0 Interrupt	Level	

PKTE Definitions

Command Descriptor

An 8-word structure that is either written directly into the packet command MMR set or is placed in a Command Descriptor Ring (CDR) in the processor memory. The packet engine sequentially processes the structure. The command descriptor contains the information that varies for every packet. This information includes pointers to the SA record, the state information, the source packet, and the destination packet.

Command Descriptor Ring (CDR)

A circular contiguous portion of memory which is used to manage one or more command descriptions for the packet engine.

Result Descriptor

When the packet engine completes the processing of a packet, it writes a result descriptor with the state information. The result descriptor can be read directly from the result register set or from the Result Descriptor Ring (RDR) in the processor memory.

Result Descriptor Ring (RDR)

A circular contiguous portion of memory which holds the mirror or copy of the CDR but contains the result descriptors. The RDR and CDR can be overlaid on top of each other.

Security Association (SA) Record

A structure that contains the remainder of the information the packet engine requires to process a packet. Most of the information fields in the SA record such as the key and encryption mode are static for the lifetime of the association. The fields do not require frequent manipulation by the processor core. The SA record non-static fields are the sequence number and sequence number mask. The SA record can have a corresponding state record for saving results from the current operations that are useful for future operations. The state record can hold the IV, the hash byte count, and the intermediate hash digest.

Cipher

A method or algorithm to encrypt or decrypt information

Hash

A cryptographic hash is a function that takes an arbitrary block of data and returns a fixed-size bit string. Four main properties define the function:

- It is easy to compute a hash value for any given input

- It is infeasible to generate the original input from a given hash
- It is infeasible to modify the input without changing the resulting hash
- It is infeasible to find two different inputs that result in the same hash

Autonomous Ring Mode (ARM)

Mode of operation in which most of the parameters as well as the data are set up in memory and moved to the engine for configuration and processing through DMA.

Target Command Mode (TCM)

Mode of operation where some parameters are set up in memory and moved into the packet engine through DMA while the other parameters are directly written to the registers. DMA moves the input and output data in and out of the engine.

Direct Host Mode (DHM)

Mode of operation that does not use DMA. All parameters are directly written to and read from the MMRs. The input and output are written to and read from the FIFO buffers.

Cipher Module

The cipher module does the symmetric encrypt or decrypt operations for:

- Data Encryption Standard (DES)
- Triple-DES
- ARC4
- Advanced Encryption Standard (AES) algorithms

The cipher module supports standard modes for DES and AES that include Electronic Code Book (ECB) and Cipher Block Chaining (CBC). The key size for DES is 56 bits, for Triple-DES is 168 bits. The AES module also provides support for AES counter-modes for IPsec and SRTP. All AES modes can use key sizes of 128 bits and 192/256 bits. Key scheduling is automatic and done in parallel with the encrypt or decrypt operation.

Hash Module

The hash module is tightly coupled with the encrypt or decrypt module and provides hardware accelerated one-way hash functions. Operations that combine both hash and encrypt or decrypt functions are provided to reduce processing time for data that needs both applied. For hash-then-decrypt operations, the packet engine performs parallel execution of both functions from the input buffer. For encrypt-then-hash operations, the processing is pipelined from the input buffer to provide minimum latency. An offset can be specified between the start of hashing and the start of encryption to support protocols such as IPsec or SRTP. The HMAC keyed hashing mechanism is supported

for MD5, SHA-1, SHA-2-224 and SHA-2-256. The SSL-MAC is supported for MD5 and SHA-1. A second AES-CBC module for the hash function enables parallel processing for AES-CCM, a combined hash and encrypt algorithm.

Pseudo-Random Number Generator

Cipher algorithms that operate in CBC mode or counter-mode, require an IV. This IV must not be secret; however the IV must be unpredictable and unique for each execution of the encryption process. Pseudo-random number generators are deterministic algorithms that output statistically independent and unbiased numbers. True random number generators are non-deterministic and use the randomness that occurs in a physical process. The packet engine incorporates an ANSI X9.31 compliant Pseudo Random Number Generator (PRNG) that it can use to generate unique IVs using strong encryption. The ANSI X9.31 PRNG is defined as part of the ANSI X9 standards that are used to secure financial transactions. The function can also be used for pseudo-random number generation as part of an implementation of the digital signature standard described in NIST FIPS PUB 186-2.

The PRNG function, as defined by ANSI X9.31, is based on the AES cipher. This section describes the function to promote understanding of the different inputs and outputs of the PRNG function itself.

NOTE: The PRNG in the packet engine is only based on the AES cipher with 128-bit keys. Other ciphers and key lengths are not supported for the PRNG based on ANSI X9.31.

Let $e \times K(Y)$ represent the AES encryption of Y under the key K.

The PRNG function uses three inputs:

- K, a 128-bit key
- V, a 128-bit seed value
- DT, a 128-bit date/ time vector which is updated on each iteration

The intermediate value I is the result of an AES encryption of the data and time vector under key K.

$I = e \times K(DT)$

That value I is then XOR-ed with the seed V and AES encrypted under key K. The result R is the output of the PRNG function.

$R = e \times K(I \text{ XOR } V)$

A new seed value V is generated from the AES encryption of the result R XOR'ed with the intermediate value I under the key K.

$V = e \times K(R \text{ XOR } I)$

The PRNG function is deeply integrated inside the datapath of the packet engine. The function is controlled indirectly through the PRNG mode bits in the `PKTE_CTL_STAT.PRNGMD` bit field of the command descriptor and the IV source selection bits in the `PKTE_SA_CMD0.IVSRC` bit field of the SA record.

The PKTE module supports four different modes.

1. Load IV from PRNG for the current operation: `PKTE_CTL_STAT.PRNGMD` = 0b00 and `PKTE_SA_CMD0.IVSRC` = 0b11.
2. PRNG init mode initializes the PRNG with a key, seed, and date/time value: `PKTE_CTL_STAT.PRNGMD` = 0b01.

   Before the PRNG function can be used, it must be initialized with a key, seed, and date/time value. At initialization, the key, seed, and date/time values are programmed. Other PRNG operations do not change the key, however the seed, and date/time values are updated (not re-programmed). The date/time is updated every 128 system clock cycles.

   The date/time is a 128-bit value with randomly distributed number of ones and zeros. It must not be all zeros.

   The ***SA Record for PRNG Init Operation*** table shows how the key, seed and date/time values are loaded into the PKTE registers for initialization.

Table 41-3: SA Record for PRNG Init Operation

Parameter	SA Field	Description
K	SA_KEY0	PRNG key [127:96]
	SA_KEY1	PRNG key [95:64]
	SA_KEY2	PRNG key [63:32]
	SA_KEY3	PRNG key [31:0]
V	SA_IDIGEST0	PRNG seed [127:96]
	SA_IDIGEST1	PRNG seed [95:64]
	SA_IDIGEST2	PRNG seed [63:32]
	SA_IDIGEST3	PRNG seed [31:0]
DT	SA_ODIGEST0	PRNG date/time [127:96]
	SA_ODIGEST1	PRNG date/time [95:64]
	SA_ODIGEST2	PRNG date/time [63:32]
	SA_ODIGEST3	PRNG date/time [31:0]

3. PRNG generate mode generates pseudo-random data on initialized key, seed, and date/time value: `PKTE_CTL_STAT.PRNGMD` = 0b10.

   The PRNG function can be used to generated pseudo-random data for other purposes than IVs. For this mode, the PRNG must have been initialized once with the PRNG init mode.

   The PRNG generate mode uses the initialized key and unique changing date/time value as inputs. The pseudo-random data is output to the output buffer of the packet engine.

   The `LEN` field in the command descriptor indicates the amount of pseudo random data that is generated in multiples of 16 bytes. The maximum is 255 × 16 = 4080 bytes.

In autonomous ring mode, the output data is copied to the host memory at the destination address in the command descriptor. In direct host mode, the host must read the data directly from the output buffer. No SA record is used for this function.

Directly after the PRNG generate mode, a new pseudo-random number is generated and available for the next operation that uses the option `PKTE_SA_CMD0.IVSRC = PRNG`.

4. PRNG test mode generates pseudo-random data on initialized key, seed, and input (test) data: `PKTE_CTL_STAT.PRNGMD` = 0b11.

   The PRNG test mode can be used to test the correctness of the PRNG function. This mode is similar to the PRNG generate mode, except that the data is read from the input buffer of the packet engine, instead of the date/time value.

   For this mode, the PRNG must have been initialized once with the PRNG Init mode.

   The `LEN` field in the command descriptor indicates the amount of pseudo-random data to be generated in multiples of 16 bytes. The maximum is limited to the `LEN` field in bytes.

   In autonomous ring mode, the output data is copied to the host memory at the destination address in the command descriptor. In the direct host mode, the host must read the data directly from the output buffer. No SA record is used for this function.

   Directly after the PRNG test mode, a new pseudo-random number is generated and available for the next operation that uses the option `PKTE_SA_CMD0.IVSRC = PRNG`.

Packet Engine Processing Details

This section describes data processing through the packet engine. It describes padding and supported algorithms for each protocol.

A valid Security Association (SA) must be created before packet processing can start. A formatted SA record must reside in memory and be accessible to the packet engine. The host processor application is responsible for these tasks.

Crypto Padding

Padding is the process of adding data to fill-out a fixed-size plain text data structure. Three factors determine when to use a pad field:

1. If a block cipher encryption algorithm is used, a pad field is used to expand the plain text to a multiple of the block size.
2. Padding can be used to ensure that the cipher text terminates on an n-byte boundary.
3. Padding can conceal the actual length of the payload.

To facilitate peak encrypt or decrypt performance, the packet engine supports the following most commonly used padding functions in hardware:

1. Pad generation and insertion of pad bytes to the end of plain text prior to encryption, for outbound operations.

2. Pad verification to check for correct padding after decrypting a packet for inbound operations.
3. Pad consumption to strip the pad bytes from the plain text data after decrypting a packet, for inbound operations.

Pad Generation and Insertion

The pad type and quantity of bytes the packet engine inserts depends on the plain text length and the value of the following fields:

- `PKTE_SA_CMD0.PADTYPE` and `PKTE_SA_CMD0.EXTPAD` defines the type of padding
- `PKTE_CTL_STAT.PADVAL` defines a value that is inserted in the pad
- `PKTE_SA_CMD0.CIPHER` enforces a certain pad alignment
- `PKTE_SA_CMD1.CIPHERMD` enforces a certain pad alignment
- `PKTE_SA_CMD0.SCPAD` allows stream ciphers to be padded
- `PKTE_CTL_STAT.PADCTLSTAT` controls the pad alignment

The `PKTE_CTL_STAT.PADCTLSTAT` bit field of the result descriptor returns the total number of inserted pad bytes.

Pad Types

The pad type bit field (`PKTE_SA_CMD0.PADTYPE`) and the extended pad bit (`PKTE_SA_CMD0.EXTPAD`) select the pad type for the extended protocol group. The packet engine can generate different pad types in hardware as described in the ***Pad Examples*** table.

The `PKTE_CTL_STAT.PADVAL` bit field, together with the number of pad bytes, defines the value that is inserted in the pad. The format of the pad and the use of this field is best explained in an example (see the ***Pad Examples*** table).

For the IPsec pad type, this field holds the value that is inserted into the next header field (in the ESP trailer) of the innermost operation's header. For the Constant pad type or the Constant SSL pad type, this field holds the inserted fixed constant pad value. For all other pad types, this field is not used and must be zero.

Table 41-4: Pad Types

Pad Type	Value	Description
IPSec	0b000	Append 0 to 255 pad bytes, followed by a pad length byte and a next header byte. The first pad byte appended to the plain text is numbered 1, with subsequent pad bytes making up a monotonically increasing sequence: 1, 2, 3 and up. Append the pad length field that indicates the number of pad bytes (0–255), where a value of zero indicates no pad bytes present. Append the next header byte as specified in the `PKTE_CTL_STAT.PADVAL` field of the command descriptor. A minimum of 2 bytes are appended; zero pad bytes plus the pad length byte plus the next header byte, in which case the `PKTE_CTL_STAT.PADCTLSTAT` field in the result descriptor returns 0x02. A maximum of 257 bytes can be appended, in which case the `PKTE_CTL_STAT.PADCTLSTAT` field in the descriptor returns 0x01.
PKCS#7	0b001	Appends 1–128 pad bytes with a pad byte value equal to the pad length (1–128).
Constant	0b010	Appends 0–255 pad bytes of a user-specified character to the plain text data. This character is specified in the `PKTE_CTL_STAT.PADVAL` field of the command descriptor.
Zero	0b011	Appends 0–255 pad bytes of 0x00 to the plain text data.
Constant SSL	0b110	Appends 0–255 pad bytes of a user-specified character to the plain text data, followed by a 'pad length' byte (0–255). This character is specified in the `PKTE_CTL_STAT.PADVAL` field of the command descriptor. A total of 256 bytes can be appended, in which case the pad field returns 0x00.

For example, the ***Pad Examples*** table shows the appended pad for any of the pad types for an outbound (encrypt) operation. The table shows a plain text input of 2 bytes using the 8-byte block cipher crypto-algorithm DES-ECB and a `PKTE_CTL_STAT.PADVAL` field value of 0xAA.

Table 41-5: Pad Examples

Pad Type	Pad field (extended to Plain Text)	`PKTE_CTL_STAT`
IPSec	0x01, 0x02, 0x03, 0x04, 0x04, 0xAA	0x06
PKCS#7	0x06, 0x06, 0x06, 0x06, 0x06, 0x06	0x06
Constant	0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA	0x06
Zero	0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x06
Constant SSL	0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0x05	0x06

Pad Length

The ***Pad Alignment*** table lists the alignment (boundaries) to which the packet engine will pad, based on:

- The selected crypto algorithm
- Crypto mode
- The value of the pad stream cipher bit
- The value of pad control

The minimum number of inserted pad bytes depends on the cipher algorithm, selected using the PKTE_SA_CMD0.CIPHER bit field, and the cipher mode, selected using the PKTE_SA_CMD1.CIPHERMD bit field.

For block ciphers, the plain text data is always (as a minimum) padded to the next block boundary. More pad bytes beyond the algorithm or protocol alignment requirements can be inserted using the pad control (PKTE_CTL_STAT.PADCTLSTAT) in the command descriptor. This feature can be used for *traffic flow security* to conceal the number of plain text bytes in an encrypted packet.

Encrypt operations that use block ciphers have minimum pad requirements based on their block size. The packet engine enforces a minimum pad alignment for block ciphers according to the ***Pad Alignment*** table. For ESP outbound operations, the minimum pad alignment is forced to 4 bytes.

For stream ciphers and null crypto the data is never padded when the stream cipher pad bit (PKTE_SA_CMD0.SCPAD) =0. When PKTE_SA_CMD0.SCPAD =1 the plain text data is padded to the length as defined by the PKTE_CTL_STAT.PADCTLSTAT bits in the command descriptor.

NOTE: The SSL protocol does not allow padding to exceed the ciphers block length. This length is 8 bytes for DES/Triple-DES and 16 bytes for AES. For SSL, the packet engine does not enforce this pad alignment value. The host processor must ensure that the PKTE_CTL_STAT.PADCTLSTAT bit field is configured correctly.

Table 41-6: Pad Alignment

Pad Control **PADCTLSTAT = PKTE_CTL_STAT[31:24]**			**0x00**	**0x01**[*1]	**0x02**	**0x04**	**0x08**	**0x10**	**0x20**	**0x40**	**0x80**[*2]
Crypto Algorithm	**Crypto Mode**	**Pad Stream Ciphers**	**%8**	**%1**	**%4**	**%8**	**%16**	**%32**	**%64**	**%128**	**%256**
DES	ECB, CBC	N/A	8	8	8	8	16	32	64	128	256
AES	ECB, CBC	N/A	16	16	16	16	16	32	64	128	256
	CTR, ICM with ESP	N/A	8	4	4	8	16	32	64	128	256
	CTR, ICM no ESP	no	0	0	0	0	0	0	0	0	0
		yes	8	0	4	8	16	32	64	128	256
NULL	with ESP	N/A	8	4	4	8	16	32	64	128	256
	no ESP	no	8	0	0	0	0	0	0	0	0
		yes	8	0	4	8	16	32	64	128	256
ARC4	with ESP	N/A	8	4	4	8	16	32	64	128	256
	no ESP	no	0	0	0	0	0	0	0	0	0
		yes	8	0	4	8	16	32	64	128	256

*1 When PKTE_CTL_STAT.PADCTLSTAT is configured for no padding (0x01), it does not mean that no padding bytes are inserted. When PKCS#7 padding is selected, a pad length field with a value =1 is inserted. When SSL or TLS padding is selected, a pad length field with a value =0 is inserted. When IPsec padding is selected, a pad length field is forced to (PKTE_CTL_STAT.PADCTLSTAT=0x20). When zero pad and constant pad are selected, no pad bytes are inserted.

*2 Pad type PKCS#7 supports a maximum length of 128 pad bytes, so the packet engine overrules a 256-byte alignment (PKTE_CTL_STAT.PADCTLSTAT=0x80) to a 128-byte boundary (PKTE_CTL_STAT.PADCTLSTAT=0x40).

The packet engine does not constrain the pad type that is used for an operation; any pad type can be used for each operation. The user must be aware that some protocol specifications only allow specific pad types. The SRTP specification does not have padding defined as padding performed by RTP. The host must pad the RTP packet.

The host software can implement padding that is not supported in hardware. In this case, the host must select the *zero pad* type and set the PKTE_CTL_STAT.PADCTLSTAT bit field in the packet descriptor to zero (no padding). The hardware padding engine does not add any bytes. When using a block cipher, the host must insert pad bytes. Then, the data to be encrypted (plain text and pad bytes) are a multiple of the block ciphers boundary. For stream ciphers, any number of pad bytes can be added.

Pad Verification and Consumption

The packet engine can validate a pad type against the expected values. The value of the following bits controls the pad verify function:

- PKTE_SA_CMD0.PADTYPE and PKTE_SA_CMD0.EXTPAD define the type of padding
- PKTE_SA_CMD0.CIPHER enforces a certain pad alignment
- PKTE_SA_CMD1.CIPHERMD enforces a certain pad alignment
- PKTE_SA_CMD0.SCPAD allows stream ciphers to be padded

When packet processing is complete, the status byte in the first word of the result descriptor reports the pad verification status. Refer to the Table 41-28 Extended Error Codes - Status Encoding table.

The PKTE_CTL_STAT.PADCTLSTAT bits in the first word of the result descriptor return the total number of detected pad bytes and returns zero for a pad verify error.

When the IPsec pad type is selected, the PKTE_CTL_STAT.PADVAL bit field of the result descriptor returns the next header field. For IPsec ESP outbound operations, this field returns the decimal value 50. For IPsec inbound operations and basic inbound operations that use the IPsec pad mode, the packet engine returns the next header field it detects on the header of the innermost operation. This value is typical for the payload protocol, such as TCP or UDP. However, in bundling scenarios or in IPv6 with destination option headers, another header value could be seen. For all other inbound operations, the returned pad value is zero.

Pad verification is performed for inbound (decrypt) operations that use IPsec, TLS/DTLS or PKCS#7 pad type:

- In combination with a block cipher algorithm: DES-ECB, DES-CBC, AES-ECB, AES-CBC,
- In combination with a stream cipher and stream cipher padding PKTE_SA_CMD0.SCPAD enabled: AES-CTR, AES-ICM and ARC4
- In combination with null-crypto (no encryption).

Pad Types

The PKTE_SA_CMD0.PADTYPE bit field and the extended pad PKTE_SA_CMD0.EXTPAD bit selects the pad type for the extended protocol group pad type. The packet engine can verify the different pad types in hardware as described in the ***Pad Types*** table.

The constant and zero pad types are not verified since they do not include a pad length field. The SSL pad type is not verified since it does not have a defined pattern.

Table 41-7: Pad Types

Pad Type	SA_CMD0[18, 7:6]	Description
IPSec	0b000	Verify that the pad field includes 0–255 pad bytes, followed by a correct pad length and a next header byte. Verify that pad bytes appended to the plain text are an incremental count, starting at one. Verify that the pad length field is the number of pad bytes (0–255), where a value of zero indicates no pad bytes present. Verify that a next header byte is present as the last byte of the packet, the value is not verified. This is after removal of the ICV. The total number of detected pad bytes is returned in the PKTE_CTL_STAT.PADCTLSTAT field in the result descriptor. **NOTE:** A minimum of 2 bytes must be present, zero pad bytes plus the pad length byte plus the next header byte. In this case the PKTE_CTL_STAT.PADCTLSTAT field in the descriptor returns 0x02. A maximum of 257 bytes can be present, in which case the PKTE_CTL_STAT.PADCTLSTAT field in the result descriptor returns 0x01. The value of the next header byte is returned in the PKTE_CTL_STAT.PADVAL field in the result descriptor.
PKCS#7	0b001	Verify that the pad field includes 1–128 pad bytes, with a pad byte value equal to the pad length (1–128).

When a block cipher is used and the payload is not padded to the nearest block size boundary, as required by the protocol, a *block size error* is generated for all pad types.

Pad Consumption

The packet engine can optionally consume the decrypted pad bytes for an inbound operation that uses the IPsec, SSL, TLS/DTLS, or PKCS#7 pad types. The pad types constant and zero are not consumed since they do not include a pad length field.

Pad consumption (or stripping) is selected on a flow-by-flow basis in the SA-record with the PKTE_SA_CMD1.CPYPAD bit. When this bit is set, the length returned in the LEN field of the result descriptor is the total length of the plain text including the pad. When the PKTE_SA_CMD1.CPYPAD is disabled, the detected pad length, as returned in the PKTE_CTL_STAT.PADCTLSTAT bits, is subtracted from the total length and then returned in the LEN field of the result descriptor.

The pad is always written to the result packet buffer in memory. When the PKTE_SA_CMD1.CPYPAD bit is disabled, only the result length is corrected.

Crypto and Hash Algorithms

The packet engine supports a wide range of crypto and hash algorithms to accelerate basic operations and protocol operations. These algorithms are:

- Basic Encrypt and Basic Decrypt Operations
- Basic Hash Operations
- Basic Encrypt-Hash and Basic Hash-Decrypt Operations
- IPSec ESP Operations
- SRTP Operations

The following tables provide allowed algorithm combinations. Those algorithms not listed in the tables are invalid and can give unexpected results.

NOTE: Not all crypto and hash algorithms are available on all product models. For information on algorithm availability, see the product-specific data sheet.

Table 41-8: Algorithms for Basic Encrypt and Basic Decrypt Operations

Crypto Algorithm	Crypto Mode
DES, Triple-DES	ECB, CBC
AES	ECB, CBC, CRT, ICM
NULL	—

Table 41-9: Algorithms for Basic Encrypt and Basic Decrypt Operations

Crypto Algorithm	Crypto Mode
SHA-1	Basic hash, HMAC, SSL-MAC
SHA-224	Basic hash, HMAC
SHA-256	Basic hash, HMAC
NULL	—

Table 41-10: Algorithms for Basic Encrypt-Hash and Basic Hash-Decrypt Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	ECB, CBC	SHA-1	Basic hash, HMAC, SSL-MAC
		SHA-224	Basic hash, HMAC
		SHA-256	Basic hash, HMAC
		MD5	Basic hash, HMAC, SSL-MAC
		NULL	—

Table 41-10: Algorithms for Basic Encrypt-Hash and Basic Hash-Decrypt Operations (Continued)

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
AES	ECB, CBC, CRT, ICM	SHA-1	Basic hash, HMAC, SSL-MAC
		SHA-224	Basic hash, HMAC
		SHA-256	Basic hash, HMAC
		MD5	Basic hash, HMAC, SSL-MAC
		NULL	—
NULL	—	SHA-1	Basic hash, HMAC, SSL-MAC
		SHA-224	Basic hash, HMAC
		SHA-256	Basic hash, HMAC
		MD5	Basic hash, HMAC, SSL-MAC

Table 41-11: Algorithms for IPsec ESP Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	CBC	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
AES	CBC, CTR	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
NULL	CBC	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	—

Table 41-12: Algorithms for Basic SSL and Extended SSL Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	CBC	SHA-1	SSL-MAC
		MD5	SSL-MAC
		NULL	—

Table 41-12: Algorithms for Basic SSL and Extended SSL Operations (Continued)

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
AES	CBC	SHA-1	SSL-MAC
		MD5	SSL-MAC
		NULL	—
ARC4	Stateful	SHA-1	SSL-MAC
		MD5	SSL-MAC
		NULL	—
NULL	—	SHA-1	SSL-MAC
	—	MD5	SSL-MAC
	—	NULL	—

Table 41-13: Algorithms for Basic TLS, Extended TLS and DTLS Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	CBC	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
AES	CBC, CTR	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
ARC4[1]	Stateful	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
NULL	—	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—

1 Only for Basic TLS and Extended TLS

Table 41-14: Algorithms for SRTP Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
AES	ICM	SHA-1	HMAC
NULL	—	SHA-1	HMAC

IV Processing

An initialization vector (IV) is necessary to start a cipher stream or a block cipher in any of the streaming modes of operation. The IV must be unique for each packet. The IV ensures that all cipher texts are unique even if produced by the same encryption key. This functionality prevents every packet from needing a unique encryption key.

Depending on the packet engine operation, the IV can be loaded from different sources. The IV format depends on the algorithm and the source of the IV. The ***Format of the IV*** table provides an overview of all IV formats.

Table 41-15: Format of the IV

Algorithm	IV Source (`PKTE_SA_CMD0.IVSRC`)	Format	IV Offset (`PKTE_SA_CMD1.HSHCOFFST`)
DES/Triple-DES (CBC)	Previous Result of IV	Internal IV register [63:0]	0x00
	Input Buffer	Input buffer [63:0]	0x02
	Saved IV	State Record Saved IV [63:0]	0x00
	Automatic	PRNG output [63:0]	0x00
AES (CBC)	Previous Result of IV	Internal IV register [127:0]	0x00
	Input Buffer	Input Buffer [127:0]	0x04
	Saved IV	State Record Saved IV [127:0]	0x00
	Automatic	PRNG output [127:0]	0x00
AES (ICM) for Basic and SRTP	Input Buffer	Input Buffer [127:0]	0x04
	Saved IV	State Record Saved IV [127:0]	0x00
AES (CTR) for Basic and IP-sec	Input Buffer	SA_NONCE || Input Buffer[63:0] || 0x00000001	0x02
	Saved IV	State Record Saved IV [127:0]	0x00
	Automatic	SA_NONCE || PRNG output [95:32] || 0x00000001	0x00

Notes:

1. The `PKTE_SA_CMD1.HSHCOFFST` bit field provides the IV offset. The offset is only applicable for basic hash or encrypt operations. For protocol operations, the offset is automatically enforced.

2. AES-CTR: The Nonce value as described in RFC 3686, is mapped to the PKTE_SA_NONCE register. This Nonce value remains constant for the lifetime of the security association.
3. The host processor controls the IV update using the save-IV bit (PKTE_SA_CMD0.SVIV). When part of a packet is processed using a stream cipher and the encrypt or decrypt data is not an integer multiple of the block size, the saved IV is invalid. It must not be used to resume processing the packet.
4. The packet engine supports automatic IV generation for outbound operations. A new IV is generated for every packet with the internal PRNG. This automatic IV generation can be used for all DES, Triple-DES, and AES modes that use an IV, except for AES-ICM mode.
5. For outbound operations, automatic IV generation is the most efficient. No additional I/O is required, and the host processor does not need to provide the IV. When the saved IV option supplies the IV, the IV must be changed for each packet sent. This activity happens when the packet engine writes back the IV to the state record after processing.
6. Outbound IPsec operations put the IV explicitly at the front of the packet. For an inbound IPsec operation, loading from the input buffer is most efficient and always used.
7. CBC processing must not use a predictable IV. Do not use the saved IV and previous result IV options for CBC processing. Refer to RFC 3602 for more details.

ARC4 Processing

The ARC4 algorithm supports two modes of operation: stateless and stateful. For SSL, TLS and DTLS operations that use the ARC4 algorithm, the mode must be set to stateful.

Stateless Mode. Each packet is processed with a newly initialized ARC4 key taken from the key field of the SA record. In this mode, the state information from the SA is never to be read.

CAUTION: When an ARC4 operation in stateless mode is interrupted by an *Interface Error* and the ARC4 state building process is aborted, the next packet fails. The ARC4 is not reset between two packets. For the next packet the ARC4 proceeds from the internal state that it was aborted. A soft reset *does not* reset the ARC4 internal state, a hardware reset is required.

Stateful Mode. When the PKTE_CTL_STAT.INITARC4 bit =1, the ARC4 algorithm initializes using the key specified in the SA record. When the PKTE_CTL_STAT.INITARC4 bit =0, the ARC4 context is read from the ARC4 state field of the SA record and the*i* and *j* pointer field of the SA record. The encrypt and decrypt processing continues from this algorithm state.

The packet engine supports ARC4 key sizes of 40 bits to 128 bits. Longer keys can be used, but they cannot be made inside the packet engine.

The host processor applies ARC4 key scheduling function to the s-box and puts the 256-byte result into the ARC4 state record. It writes the *i* and *j* pointers in the SA (initial $i = 1$, $j = 0$). The host programs the packet engine and specifies stateful operation to continue the ARC4 algorithm.

Hash State Loading

The hash state can be loaded from various sources, depending on the selected protocol and hash algorithm. The ***Different Sources for Loading the Hash State*** table provides a list of all the options.

Table 41-16: Different Sources for Loading the Hash State

Hash Algorithm **PKTE_SA_CMD0.HASHSRC =**	From SA 0b00	RESERVED 0b01	From State 0b10	No Load 0b11
SHA-1	yes	-	yes	Yes
SHA-224	Yes	-	Yes	Yes
SHA-256	Yes	-	Yes	Yes
Null hash	x	x	x	x

Sequence Number Processing

The packet engine supports sequence number generation and verification for IPsec and extended SSL, TLS, and DTLS protocol operations.

A Sequence Number (SN) is an unsigned number that a sender must implement, and a receiver can use to support anti-replay service (replay attacks) for a specific SA. This processing includes detection of the same packet that arrives more than once and detection of packets that arrive in an incorrect sequence and is outside an accepted level of order relative to the last received packet.

The packet engine supports the following options.

- Sequence number loaded from SA for outbound operations and is retrieved from the input packet for inbound operations.

The sequence number and sequence number mask fields are part of the SA record.

Sequence Number Processing in Extended SSL/TLS

SSL and TLS use an implicit sequence number of 64 bits that is not send in the packet but part of the hash.

For inbound operations, no sequence number verification is performed; instead, an incorrect sequence number results in an authentication error (`PKTE_CTL_STAT.AUTHERR`).

For outbound operations, with `PKTE_SA_CMD0.HDRPROC` enabled and `PKTE_SA_CMD1.ENSQNCHK` enabled, the packet engine generates a sequence number error when the 64-bit sequence number counter overflows (counter is 2^{64}-1 and increments to 0). The host processor must not send the packet.

For both outbound and inbound operations, with `PKTE_SA_CMD0.HDRPROC` enabled and `PKTE_SA_CMD1.ENSQNCHK` enabled, the packet engine reads the sequence number from the `PKTE_SA_SEQNUM[n]` registers in the SA. When the operation is finished, the packet engine stores the incremented sequence number in the same SA fields. The sequence number mask `PKTE_SA_SEQNUM_MSK[n]` registers in the SA are not used.

For both outbound and inbound operations, with PKTE_SA_CMD0.HDRPROC disabled, the packet engine does not increment the sequence number. It expects the host to provide the sequence number through the input buffer as part of the header.

Sequence Number Processing in DTLS

DTLS uses an explicit sequence number of 64 bits that is sent in the packet. The DTLS sequence number is composed of the epoch (16 bits) and packet sequence number (48 bits) that together form the 64-bit number, like the TLS sequence number. The epoch is incremented after each *Change Cipher Spec* message. The packet sequence number is incremented per packet starting from zero after each change cipher spec message.

For outbound operations, with PKTE_SA_CMD0.HDRPROC enabled, the packet engine reads the sequence number from the PKTE_SA_SEQNUM[n] fields in the SA, then inserts the sequence number in the packet. Then the packet engine stores the incremented sequence number in the PKTE_SA_SEQNUM[n] fields in the SA. The sequence number mask PKTE_SA_SEQNUM_MSK[n] fields in the SA are not used.

For outbound operations, with PKTE_SA_CMD0.HDRPROC enabled and PKTE_SA_CMD1.ENSQNCHK enabled, the packet engine generates a sequence number error when the 48-bit sequence number counter overflows (counter is $2^{48} - 1$ and increments to 0). The host must not send the packet.

For outbound operations, with PKTE_SA_CMD0.HDRPROC disabled or PKTE_SA_CMD1.ENSQNCHK disabled, the packet engine does not increment and verify a sequence number counter overflow, and therefore never generates a sequence number error. With PKTE_SA_CMD0.HDRPROC disabled the Packet Engine does not update the sequence number and sequence number mask fields in the SA and expects the host to provide the sequence number through the input buffer as part of the header.

For inbound operations, with PKTE_SA_CMD0.HDRPROC enabled and PKTE_SA_CMD1.ENSQNCHK enabled, the packet engine verifies the PKTE_SA_SEQNUM[n] fields against the sequence number in the packet using the PKTE_SA_SEQNUM_MSK[n] from the SA. Three situations can occur:

1. If the received sequence number falls outside and above the 64-bit sequence number mask, the mask is shifted. The packet engine updates the SA with the received sequence number and the shifted sequence number mask.
2. If the received sequence number falls inside the 64-bit sequence number mask and is not a duplicate sequence number (the same as received before), the corresponding bit in the mask is set. The packet engine updates the SA with the received sequence number and the updated sequence number mask.
3. If the received sequence number falls outside the 64-bit sequence number mask or matches and earlier received number a sequence number error is generated. The packet engine does not update the sequence number and sequence number mask fields in the SA. The host must discard the packet.

For inbound operations, with PKTE_SA_CMD0.HDRPROC disabled or PKTE_SA_CMD1.ENSQNCHK disabled, the packet engine does not verify the sequence number against the sequence number in the packet and therefore never generates a sequence number error.

With PKTE_SA_CMD0.HDRPROC disabled the packet engine does not update the sequence number and sequence number mask fields in the SA and expects the host to provide the sequence number through the input buffer as part of the header.

The following tables provide details of sequence number processing.

Table 41-17: Sequence Number Generation and Verification Control (Outbound)

Header Processing SA_CMD0[19]	Anti-Replay Service SA_CMD1[29]	Description
Outbound		
1	1	Sequence number generation (increment)
1	1	Sequence number overflow check (2^{32}–1) to zero for IPsec, (2^{48}–1) to zero for DTLS and (2^{64}–1) to zero for Extended SSL/TLS.
1	0/1	Sequence number update in SA
0/1	0/1	Extended sequence number update in SA

Table 41-18: Sequence Number Generation and Verification Control (Inbound)

Header Processing SA_CMD0[19]	Anti-Replay Service SA_CMD1[29]	Description
Inbound		
1	1	Sequence number verification (check against sequence number and sequence number mask in SA). Applicable only for IPsec and DTLS.
1	0/1	Sequence number update in SA, except on authentication error
1	1	Sequence number mask update in SA. Applicable only for IPsec and DTLS.
0/1	0/1	Extended sequence number update in SA

Table 41-19: Header Processing Enabled, Anti-replay Service Enabled

	IPsec ESP	Ext. SSL	Ext. TLS	DTLS
Header processing enabled, anti-replay service enabled				
Inbound				
Initial value in the SA SA_SEQNUM1[31:0] SA_SEQNUM0[31:0] SA_SEQNUM_MSK	B3 B0 0x00000000 0x00000000 not used	B3 B0 0x00000000 0x00000000 not used	B3 B0 0x00000000 0x00000000 not used	B3 B0 0xAA550000 0x00000000 not used
Value in the first packet or hashbyte stream, highest byte is B0	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0x00000000 B4 B7 0x00000000	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0xAA550000 B4 B7 0x00000001

Table 41-19: Header Processing Enabled, Anti-replay Service Enabled (Continued)

	IPsec ESP	Ext. SSL	Ext. TLS	DTLS
Header processing enabled, anti-replay service enabled				
Inbound				
Initial value in the SA after first packet SA_SEQNUM1[31:0] SA_SEQNUM0[31:0] SA_SEQNUM_MSK	B3 B0 0x00000000 0x00000001 not used	B3 B0 0x00000000 0x00000001 not used	B3 B0 0x00000000 0x00000001 not used	B3 B0 0xAA550000 0x00000001 not used
Initial value in the SA SA_SEQNUM1[31:0] SA_SEQNUM0[31:0] SA_SEQNUM_MSK	B3 B0 0xFEDCBA98 0x76543210 not used	B3 B0 0xFEDCBA98 0x76543210 not used	B3 B0 0xFEDCBA98 0x76543210 not used	B3 B0 0xAA55BA98 0x76543210 not used
Value in the first packet or hashbyte stream, highest byte is B0	B0 B3 0xFEDCBA98 B4 B7 0x76543211	B0 B3 0xFEDCBA98 B4 B7 0x76543211	B0 B3 0xFEDCBA98 B4 B7 0x76543211	B0 B3 0xAA55BA98 B4 B7 0x76543211
Initial value in the SA after the packet SA_SEQNUM_1[31:0] SA_SEQNUM_0[31:0] SA_SEQNUM_MASK	B3 B0 0xFEDCBA98 0x76543211 not used	B3 B0 0xFEDCBA98 0x76543211 not used	B3 B0 0xFEDCBA98 0x76543211 not used	B3 B0 0xAA55BA98 0x76543211 not used
Highest value before overflow SA_SEQNUM_1[31:0] SA_SEQNUM_0[31:0] SA_SEQNUM_MASK	B0 B3 0xFFFFFFFF 0xFFFFFFFF not used	B0 B3 0xFFFFFFFF 0xFFFFFFFF not used	B0 B3 0xFFFFFFFF 0xFFFFFFFF not used	B0 B3 0xAA55FFFF 0xFFFFFFFF not used
Value in the packet or hashbyte stream, highest byte is B0	B0 B3 0x00000000B4 B7 0x00000000	B0 B3 0xFFFFFFFF B4 B7 0xFFFFFFFF	B0 B3 0xFFFFFFFF B4 B7 0xFFFFFFFF	B0 B3 0xAA55FFFF B4 B7 0xFFFFFFFF
Value after overflow in SA SA_SEQNUM_1[31:0] SA_SEQNUM_0[31:0] SA_SEQNUM_MASK	B0 B3 0x00000000 0x00000000 not used	B0 B3 0x00000000 0x00000000 not used	B0 B3 0x00000000 0x00000000 not used	B0 B3 0xAA550000 0x00000000 not used

Table 41-19: Header Processing Enabled, Anti-replay Service Enabled (Continued)

	IPsec ESP	Ext. SSL	Ext. TLS	DTLS
Header processing enabled, anti-replay service enabled				
Inbound				
Initial value in SA	B3 B0	B3 B0	B3 B0	B3 B0
SA_SEQNUM1[31:0]	0x00000000	0x00000000	0x00000000	0xAA550000
SA_SEQNUM0[31:0]	0x00000000	0x00000000	0x00000000	0x00000000
SA_SEQNUM_MSK1[31:0]	0x00000000	0x00000000	0x00000000	0x00000000
SA_SEQNUM_MSK0[31:0]	0x00000000	0x00000000	0x00000000	0x00000000
Expected value in the first packet or hash-byte stream, highest byte is B0	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0x00000000 B4 B7 0x00000001	B0 B3 0xAA550000 B4 B7 0x00000001
Value in SA after first packet	B3 B0	B3 B0	B3 B0	B3 B0
SA_SEQNUM1[31:0]	0x00000000	0x00000000	0x00000000	0xAA550000
SA_SEQNUM0[31:0]	0x00000001	0x00000001	0x00000001	0x00000001
SA_SEQNUM_MSK1[31:0]	0x00000000	0x00000000	0x00000000	0x00000000
SA_SEQNUM_MSK0[31:0]	0x00000001	0x00000001	0x00000001	0x00000001

PKTE Block Diagram

The ***PKTE Block Diagram*** shows the functional blocks within the PKTE.

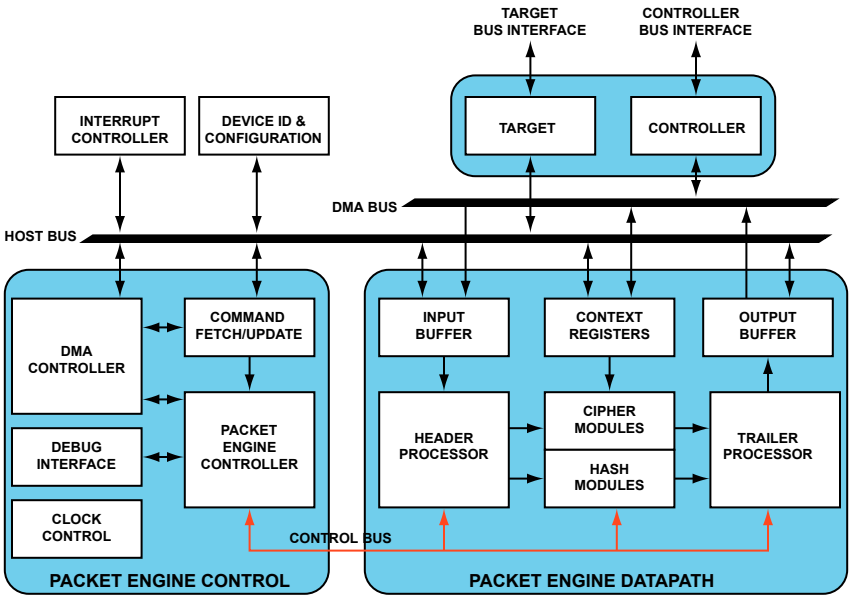


Figure 41-1: PKTE Block Diagram

PKTE Architectural Concepts

The following descriptions provide details on the functional blocks within the security packet engine.

Packet Engine

The packet engine contains symmetric cipher and hash engines. It is optimized to off-load intensive cryptographic operations from the host processor. It can perform parallel and pipelined encryption and hashing operations, reducing the latency, and processing time for packets that need both operations applied. The processor core provides the command information and packet data for the packet engine. The packet engine can run autonomously, using its local DMA controller to perform DMA transfers across the main bus to access the memory of the processor core. The DMA process incorporates flow-control to guarantee proper data flow. Two elements provide the command information that defines the processing for each packet:

- Command descriptor
- SA record

When the packet engine finishes the operation, it updates the SA record, when needed, and provides a result descriptor.

The packet engine has four different modes of operation. The modes give the processor core various levels of control over the command information and packet data transfers to and from the packet engine.

- Autonomous Ring Mode (ARM)
- Target Command Mode (with and without result descriptor ring) (TCM)
- Direct Host Mode (DHM)

Input/Output FIFO Buffers

The data for the packet engine is buffered at both input and output. These buffers decouple the DMA I/O process from the cipher and hash modules inside the packet engine. This functionality enables large DMA burst sizes and allows the crypto engines to process data during I/O latency periods. Data moves automatically from the input buffer through the encryption and hash engines to the output buffer. When the output buffer is full, the process stops until the data is read and space is available in the output buffer. Each buffer is a 256-byte dual-port RAM.

Parallel Operations

The hash functionality and encrypt or decrypt functionality are tightly coupled. Operations that combine both hash and encrypt or decrypt functions are available to reduce processing time for data that must apply both. For hash-then-decrypt operations, the packet engine performs parallel execution of both functions from the input buffer. For encrypt-then-hash operations, the processing is pipelined from the input buffer to provide minimum latency. An offset can be specified between the start of the hashing and the start of the encryption to support protocols such as IPSec and SRTP.

DMA Controller

The packet engine uses a high-performance DMA controller for autonomous data transfers for:

- Command descriptor reads
- SA record and state record reads
- Packet data read
- Result packet writes
- SA record and state record writes
- Result descriptor writes

Interrupt Controller

The packet engine includes an interrupt controller that, under programmable configuration control, can generate an interrupt on completion of certain operations. Individual interrupts can be masked and cleared. The interrupt registers show both the raw and masked interrupt status of the internal interrupts. The processor core can use interrupts, together with their associated threshold settings, to optimize the overall packet processing in the system. One interrupt can inform the processor core that the input side of the packet engine is almost empty to avoid a stall-on-empty condition. One interrupt can inform the host that the output side is almost full to avoid a stall-on-full condition. The controller uses several interrupts to inform the processor core about errors inside the packet engine. All available interrupts are combined into a single output port as either a level- or edge-active programmable interrupt output.

Clock Controller

The packet engine includes a clock controller that generates clock enable signals. A clock manager external to the packet engine uses the clock enable signals to switch the clocks to modules in the packet engine, reducing power consumption. The power saving can be significant, depending on the crypto-operation and the idle time of the packet engine. The clock controller generates the clock enable signals dynamically depending on the current crypto-operation. A clock control register provides the processor core the possibility to override this dynamic process.

PKTE Operating Modes

The packet engine can be configured in one of three command modes. For all modes the packet engine can generate an interrupt at completion of packet processing:

- Autonomous Ring Mode (ARM). The core prepares descriptors in the CDR and then initiates a descriptor fetch by triggering the packet engine. When a packet operation is complete, the packet engine writes the result descriptor out into a ring in host memory using the system requester bus interface.
- Target Command Mode (TCM). The core directly writes the command descriptors to the packet command register set to initiate a packet operation. This process eliminates an extra DMA transfer to fetch a descriptor, but requires the core to synchronously initiate packet processing. This mode can be configured both without RDR or with RDR. In the latter case, the packet engine writes the result descriptor out into the RDR in host memory using the system requester bus interface.
- Direct Host Mode (DHM). The core has full control over the packet engine and uses the system completer bus interface. The core provides all the command descriptors, SA record and state record, and packet input data.

When the packet engine completes processing, the core must read the packet output data, the SA record, the state record, and the result descriptors.

Autonomous Ring Mode (ARM)

The *Autonomous Ring Mode* allows the packet engine and the host processor to operate asynchronously. A queue of multiple packets in the host processor memory can be processed continuously to provide the highest possible throughput. The packet engine autonomously fetches the command descriptor, the SA record, and optionally the state record and the input data from host processor memory. After the packet engine finishes processing, it autonomously writes the output data, updates the SA and state record, and writes the result descriptor in the host processor memory. It accesses the host processor memory through DMA read transfers across the system bus requester.

This mode uses both command descriptor ring (CDR) and result descriptor ring (RDR).

Physically the CDR, RDR, SA record, source packet, and result packet can all be in different memories depending on the system memory architecture. The host processor writes command descriptors to the CDR in host processor memory. Then, it writes to the PKTE_CDSC_INCR register with the number of command descriptors that it prepared in the CDR. This write to the PKTE_CDSC_INCR register is the trigger for the packet engine to fetch the command descriptors sequentially from the CDR. When a command descriptor is fetched and written to the internal packet command register set, the descriptor is validated. If the ownership bits are set for the packet engine and the command is valid, processing starts. If not, that command is discarded, and a result descriptor is written to the RDR with the error code *invalid command descriptor*. In this mode, the host processor can set a threshold on the CDR and enable an associated interrupt. The packet engine generates an interrupt when the number of command descriptors in the CDR is equal or below the threshold value.

The SA record and state record that contains the crypto context information are stored in a memory area. The packet engine autonomously accesses the memory area through DMA transfers across the system bus requester. Also, the source packet and result packet are stored in a memory area that the packet engine autonomously accesses through the same bus requester interface.

After decoding the command descriptor, the packet engine fetches the SA record and then, optionally, the state record.

Then, the source packet is fetched and stored in the input buffer. Packets less than the size of the input buffers are fetched entirely at once. Larger packets are fetched in parts that completely fill the input buffer. The packet engine initiates a new fetch each time the number of empty spaces in the input buffer reaches its threshold value. When the first packet data is available in the input buffer, the crypto engines start processing the data. After processing, the crypto engines write the result packet to the output buffer.

The packet engine writes the result packet from the output buffer to host processor memory when the number of bytes in the output buffer reaches its threshold value. Packets less than the threshold value are written entirely at once. Larger packets are written in parts that completely empty the output buffer

The source packet data fetching, data processing, and result packet data writing are parallel processes that continue until the last result packet is written to host processor memory. Then, the packet engine optionally writes the SA record and the state record to update the crypto context information. As a final step, the packet engine writes the

result descriptor to the RDR. The host processor must either poll the RDR or wait for an interrupt from the packet engine to determine when packet processing is complete.

Target Command Mode (TCM)

This mode provides a synchronous interface between the processor core and the packet engine. The Command Descriptor Ring (CDR) is disabled and the processor core initiates packet processing by writing the command descriptor directly to the internal command descriptor MMRs of the packet engine. The Result Descriptor Ring (RDR) is optional.

- For `PKTE_CFG.MODE` =01, the RDR is disabled and the processor core reads the result descriptor directly from the internal result descriptor register set for the packet engine.
- For `PKTE_CFG.MODE` =10, the RDR is enabled and stored in a memory area that the packet engine can access through its requester bus interface as in autonomous ring mode.

In target command mode, the packet engine autonomously fetches the SA record, state record, source packet data as in autonomous ring mode. Also, as in ARM, after processing, the packet engine updates state fields of the SA record and state record in the host processor memory.

Direct Host Mode (DHM)

This mode provides a synchronous interface between the processor core and the packet engine. The packet engine is under full control of the processor core. The host processor writes the command descriptors, SA record, and state record directly to the packet engine registers. Then, the processor core writes the source packet data into the input buffer. When processing is complete, the processor core reads back the result packet data from the output buffer. Finally, it reads the result descriptor, the updated SA record, and state record directly from the packet engine registers.

PKTE Event Control

The following section provides information about interrupts in the PKTE module.

PKTE Interrupt Signals

The Packet Engine has an internal Interrupt Controller with 9 interrupt sources. There are 7 registers associated with the interrupt controller:

1. Interrupt Unmasked Status - `PKTE_IUMSK_STAT`
2. Interrupt Mask Status - `PKTE_IMSK_STAT`
3. Interrupt Clear Register - `PKTE_INT_CLR`
4. Interrupt Enable Register - `PKTE_INT_EN`
5. Interrupt Mask Disable - `PKTE_IMSK_DIS`
6. Interrupt Mask Enable - `PKTE_IMSK_EN`

7. Interrupt Configuration - PKTE_INT_CFG

The ***Packet Engine Interrupt Controller Block Diagram*** shows the blocks of the interrupt controller.

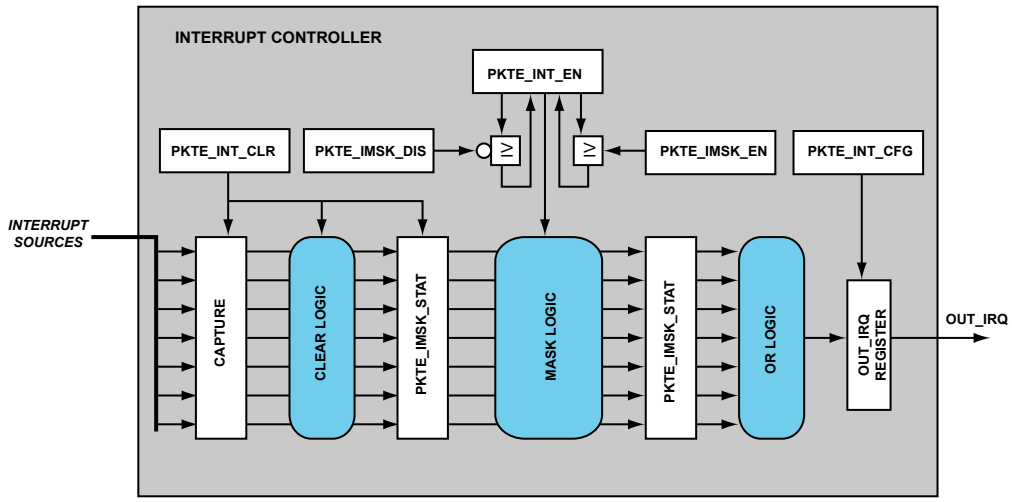


Figure 41-2: Packet Engine Interrupt Controller Block Diagram

All of the interrupt sources are pulse or level events in their native form.

These interrupts are captured and stored at their unmasked and masked status in their respective PKTE_IUMSK_STAT and PKTE_IMSK_STAT registers. This allows the host processor to read the status of any interrupt source either before or after the mask is applied.

The PKTE_INT_EN register provides a mask to select what interrupt source are enabled to the output interrupt request. Writing a one to the PKTE_INT_CLR register resets both the masked and unmasked interrupt.

The PKTE_IMSK_EN and PKTE_IMSK_DIS registers can be set to enable and disable individual interrupts respectively in the PKTE_INT_EN register. This avoids the need for read-modify-write operations from the host processor.

Ring Interrupts

Two interrupts are provided for efficient ring management: The CDR threshold interrupt (*cdrthrs*h) and the RDR threshold interrupt (*rdrthrsh*).

Command Descriptor Ring

The CDR threshold interrupt (cdrthrsh) is a level-based interrupt and connects to the threshold value in the PKTE_RING_THRESH.CDRTHRSH bit field. It enables the host processor to efficiently fill the CDR. The host processor writes command descriptors to the CDR with this interrupt masked until the CDR is full. Then the host processor enables the CDR threshold interrupt. When the interrupt is activated, the host processor clears the interrupt and it is guaranteed that it can put CDR threshold number of descriptors in the CDR.

Example Configuration:

```
PKTE_RING_CFG.RINGSZ=256,
PKTE_RING_THRESH.CDRTHRSH=224,
PKTE_INT_EN.CDRTHRSH=0 /*(IRQ disabled)*/
```

1. The host writes 8 Command Descriptors at once, then writes the PKTE_CDSC_INCR register to 8.
2. This is repeated until there are less than 8 empty entries in the CDR.
3. The fill level is now equal to the threshold (224)
4. The host enables the CDR threshold IRQ (PKTE_INT_EN.CDRTHRSH=1)
5. The packet engine processes packets and the fill level (PKTE_CDSC_CNT register) decreases.
6. Then the CDR threshold IRQ is activated as the fill level equals 224.
7. The host handles the interrupt, clears it, and continues with step #1.

Result Descriptor Ring

The RDR threshold interrupt (rdrthrsh) is a level-based interrupt and connects to the threshold value in the PKTE_RING_THRESH.RDRTHRSH bit field and the timeout value in the RD timeout (PKTE_RING_THRESH.RDTO) bit field. It enables the host processor to efficiently empty the RDR. The timeout reminds the host processor that when the threshold kicks in result descriptors stay long in the RDR and must be processed to reduce latency. The timeout counts when the ring is not empty, regardless of the fill level and restarts when the host processor writes the PKTE_RDSC_CNT register. Initially the host processor enables the RDR threshold interrupt. When the interrupt is activated the host processor reads result descriptors until the RDR is empty or contains less than the PKTE_RING_THRESH.RDRTHRSH number of descriptors.

Example Configuration:

```
PKTE_RING_CFG.RINGSZ=256,
PKTE_RING_THRESH.RDRTHRSH=32, timeout=1ms
PKTE_INT_EN.RDRTHRSH=1     /*(IRQ enabled)*/
```

1. The Packet Engine writes the Result Descriptor, timeout counter starts.
2. The Packet Engine writes 32 more Result Descriptors, fill level (PKTE_RDSC_CNT register) increases to 33.
3. The fill level exceeds threshold within 1ms, the RDR threshold IRQ is activated.
4. The host handles the interrupt, reads 8 Result Descriptors at once, then writes the PKTE_RDSC_DECR register with 8. The write to the PKTE_RDSC_DECR register restarts the timeout counter. The fill level is now under the threshold but there are still 25 descriptors left.
5. The Packet Engine writes 8 more Result Descriptors, fill level increases to 33.
6. The fill level exceeds threshold within 1 ms, the RDR threshold IRQ is activated.
7. The host handles the interrupt, reads 8 Result Descriptors at once, then writes the PKTE_RDSC_DECR register with 8. The write to the PKTE_RDSC_DECR register restarts the timeout counter. The fill level is now under the threshold but there are still 25 descriptors left.

8. After 1 ms, the timeout counter interrupt is activated.
9. The host handles the interrupt, reads 8 Result Descriptors at once, then writes the PKTE_RDSC_DECR register with 8. This is repeated until there are less than 8 full entries in the RDR. The fill level is now under the threshold and the RDR threshold IRQ interrupt is inactive. Each write to the PKTE_RDSC_DECR register restarts the timeout counter.

PKTE Programming Model

The host processor must always follow a pre-defined sequence of five phases required by the packet engine on a per packet basis when using direct host mode. The following sections describe the five phases.

Phase One. Write the Command Descriptor

1. Write the first command descriptor word with status and control information to the PKTE_CTL_STAT register.
2. Optionally, write the user ID to the PKTE_USERID register.
3. Write the last descriptor word to the PKTE_LEN register.
4. Write the value 0x1 to the PKTE_CDSC_CNT register. This operation triggers the packet engine to validate the command descriptor. When the command descriptor is invalid, an error is generated. (See the PKTE_CTL_STAT section in the Register Descriptions). When the command descriptor is valid, the packet engine waits for an PKTE_SA_RDY register write.

Phase Two. Write the State Registers, (ARC4 Buffer) and SA Registers

All required fields of the SA record and state record must be written. The fields required depend on the operation. The last field to be written is the PKTE_SA_RDY register. This register triggers the packet engine to start processing.

1. Write the required state record fields.
   - ARC4 state
   - IV
   - Digest count
   - State digest
2. Write the required SA record data.
3. To complete the SA record and state record, write the PKTE_SA_RDY register.

Phase Three. Write the Source Packet Data and Read Result Packet Data

The packet engine has input and output buffers. If a source packet is smaller than the size of the input buffer, then the packet can be written in one part. Otherwise, it must be written in multiple parts. The same applies to the output data. If the result packet size is smaller than the size of the output buffer, then the packet can be read in one part. Otherwise, it must be read in multiple parts.

NOTE: An outbound packet that is smaller than the size of the input buffer can increase in size due to padding and does not always fit in the output buffer. Conversely, an inbound packet that is larger than the size of the input buffer can decrease in size, and due to de-padding, can fit in the output buffer. If the input buffer becomes empty or the output buffer becomes full, the engine stalls.

Two following steps describe different situations:

- Source packet smaller than the size of the input buffer, start at step #1.
- Source packet larger than the size of the input buffer, start at step #3.

The host processor must follow these steps:

1. Write the source packet data. Write the full source packet to the input buffer. Go to step #4.
2. Write the input buffer count register (PKTE_INBUF_CNT) with the number of valid bytes that are written to the input buffer. This value must correspond to the value in the PKTE_LEN.TOTLEN field of the command descriptor rounded up to the next 4-byte multiple. Go to step #5 to check the packet engine status.
3. Write part of the source packet data. The PKTE_STAT.IBUFEMPTYCNT field indicates the amount of free space in the input buffer. Programs write the number of bytes determined by the setting in the PKTE_BUF_THRESH register. Write the (partial) source packet to the input buffer. The host processor must resume where it ended the previous write operation. Do not write more than the buffer size at once. Go to step 4.
4. Write the PKTE_INBUF_CNT register with the number of valid bytes written to the input buffer. Go to step 5 to check the packet engine status.
5. Check packet engine status. Wait for an interrupt or poll the PKTE_STAT register for any of the following conditions:
   - Condition 1–an error interrupt or any of the bits [7:5] in the PKTE_STAT register becomes active to indicate a packet processing error. Depending on the type of error, the host processor must take appropriate action. Usually, the result packet is not valid after a processing error has occurred. Go to phase 5 to read the result descriptor.
   - Condition 2–an operation done interrupt or the PKTE_STAT.OPDN bit becomes active (the packet engine completed processing). Go to step 8 to read the remaining output data.
   - Condition 3–an output buffer threshold interrupt or the PKTE_STAT.OBUFREQ bit becomes active. Go to step #6 to read a block of output data.

- Condition 4–an input buffer threshold interrupt or the PKTE_STAT.IBUFREQ bit becomes active. Go to step #3 to write a block of input data.

6. Read part of the output data. The PKTE_STAT.OBUFFULLCNT bit field indicates the number of bytes in the output buffer. This value is rounded up to full words. Programs read the number of bytes indicated in the PKTE_BUF_THRESH register. Read the (partial) output packet from the output buffer. The host processor must resume where it ended the previous read operation. Do not read more than the input buffer size at once. Go to step #7.
7. Write the PKTE_OUTBUF_CNT register with the number of valid bytes read from the output buffer. Go to step #5 to check the packet engine status.
8. Read the remaining output data. The PKTE_STAT.OBUFFULLCNT bit field indicates the number of bytes in the output buffer. This value is rounded up to words. Read the (partial) packet output data from the output buffer. The host processor must resume where it ended the previous read operation. Go to step #9.
9. Write the PKTE_OUTBUF_CNT register with the number of valid bytes read from the output buffer. Go to phase four.

Phase Four. Read the Result Descriptor

1. Read the first result descriptor word from the PKTE_CTL_STAT.
2. Optionally read the user ID from the PKTE_USERID register.
3. Read the last result descriptor word from the PKTE_LEN register.
4. Write the value 0x1 to the PKTE_RDSC_DECR register. This operation allows the packet engine to accept new command descriptors. Go to phase five.

Phase Five. Read the SA Record and State Record

Depending on the operation, the SA record or state record is updated. Check the bit fields [23:16] in the PKTE_CTL_STAT register for the following conditions:

- Condition 1 - At least one error bit in the bit fields [23:16] of the PKTE_CTL_STAT register is set. Do not update the local host processor maintained version of the SA record and state record but take any required action.
- Condition 2 - None of the error bits in the bit fields [23:16] of the PKTE_CTL_STAT register is set, the packet is processed normally (without errors). Update the host processor maintained version of the SA record and state record with the result read from the packet engine registers:
  - ARC4 state
  - Sequence number
  - Sequence number mask
  - Result IV

- Result digest count
- Result digest

PKTE Mode Configuration

Before using the packet engine, it must be configured. The mode of the packet engine must be defined and the PRNG (if used) must be initialized.

Configure the packet engine in one of three command modes:

- Autonomous Ring Mode: `PKTE_CFG.MODE` =b'11
- Target Command Mode: `PKTE_CFG.MODE` =b'10
- Direct Host Mode: `PKTE_CFG.MODE` =b'00

PKTE Programming Concepts

The following sections provide conceptual information for programming the PKTE.

Packet Engine Descriptor

IMPORTANT: Depending on the mode, ARM, TCM, or DHM, the descriptor is either:

- in the memory of the host processor in the command descriptor ring, or
- written directly to the descriptor registers in the packet engine

References to descriptor registers are for either the register that is mirrored in the descriptor structure in memory or for the actual register itself.

Command descriptors are host-supplied commands that control the real-time operation of the packet engine. The packet engine returns result descriptors at the end of an operation that provide the status information to the host. The ***Command Descriptor Structure*** and the ***Result Descriptor Structure*** tables show these descriptors.

Table 41-20: Command Descriptor Structure

Word Offset	31:24	23:20	19:16	15:8	7:0	Address Offset
0	Pad Control	—		Next Header/ Pad Value	Control	0x000
1	Source Address					0x004
2	Destination Address					0x008
3	SA Address					0x00C
4	SA State Address					0x010
5	Reserved/ARC4 State Address					0x014

Table 41-20: Command Descriptor Structure (Continued)

Word Offset	31:24	23:20	19:16	15:8	7:0	Address Offset
6	User ID					0x018
7	Bypass (words)	Control	Reserved	Input Packet Length (bytes)		0x01C

Table 41-21: Result Descriptor Structure

Word Offset	31:24	23:20	19:16	15:8	7:0	Address Offset
0	Pad Status	Status		Next Header/ Pad Value	Control	0x000
1	Source Address					0x004
2	Destination Address					0x008
3	SA Address					0x00C
4	SA State Address					0x010
5	Reserved/ARC4 State Address					0x014
6	User ID					0x018
7	Bypass (words)	Control	Reserved	Input Packet Length (bytes)		0x01C

When the packet engine is configured for autonomous ring mode, command descriptors and result descriptors reside in a ring in host memory. Command descriptors are automatically fetched from the Command Descriptor Ring (CDR) through DMA into the command descriptor registers. When an operation is complete, the result descriptors are automatically read from the packet engine and through DMA to the Result Descriptor Ring (RDR).

When the packet engine is configured for direct host mode, the host processor manually writes the command descriptor directly to the internal command descriptor MMR set. When an operation is complete, the host processor manually reads the result descriptor directly from the result descriptor MMR set.

The target command mode is a combination of the direct host mode and the autonomous ring mode. The host processor writes the command descriptor directly to the internal command descriptor register set. When an operation is complete there are two options.

1. The host processor can read the result descriptor directly from the result descriptor registers.
2. The result descriptors reside in a ring in host memory and the result descriptors are automatically DMA'd from the packet engine to the RDR.

When the host processor writes a command descriptor to the command descriptor registers, the packet engine is triggered when the host processor updates the `PKTE_CDSC_CNT` register. This functionality guarantees that all fields in the command descriptor are valid before the command is executed.

Descriptor Processing

This section describes the functional steps of the packet engine while processing the command descriptors.

Descriptor Ring Configuration

At initialization, the host processor specifies the size of the Command Descriptor Ring (CDR). The Result Descriptor Ring (RDR) has the same size.

NOTE: In some configurations, these two rings overlay each other with the results written on top of the command descriptors. This configuration is called overlaid ring mode.

When the packet engine is configured and enabled, it fetches the descriptors from the CDR using system bus requester reads.

Descriptor Ring Processing

To validate the descriptor exchange between the host processor and the packet engine, the ownership bits `PKTE_CTL_STAT.PERDY` and `PKTE_CTL_STAT.HOSTRDY` are used. One pair of ownership bits is in the first word of the descriptor (`PKTE_CTL_STAT`), and one pair is in the last word (`PKTE_LEN`). The 'consumer' of a descriptor must verify that both ownership pairs match to ensure that a race condition did not occur between one party writing and the other party reading the descriptor. A race condition can occur when a memory locking scheme is not used.

Each pair [`PKTE_CTL_STAT.PERDY`, `PKTE_CTL_STAT.HOSTRDY`] of ownership bits provide 3 states:

- b'00 = idle or null descriptor
- b'01 = host processor has written a descriptor in the CDR and passed ownership to the packet engine
- b'10 = packet engine processing complete: packet engine has written the descriptor in the RDR and passed ownership back to the host.
- b'11 = Reserved

At initialization, the host sets the entire CDR memory area to zero, when the CDR is used.

1. The host processor writes one or more command descriptors to the CDR. The host processor must set the `PKTE_CTL_STAT.HOSTRDY` bit to 1 and the `PKTE_CTL_STAT.PERDY` bit to 0 to indicate that ownership has passed to the packet engine. These bits are mirrored in the `PKTE_LEN` descriptor word.
2. The host processor must write the `PKTE_CDSC_INCR` register with the number of new valid command descriptors in the CDR.
3. The packet engine reads and validates one command descriptor.
4. The packet engine reads the SA record and state record, processes the packet, and updates the SA record and state record.

5. If the rings are not overlaid and the PKTE_CFG.ENCDRUPDT bit is 1, the packet engine writes the result descriptor to the CDR with the PKTE_CTL_STAT.HOSTRDY bit set to 0 and PKTE_CTL_STAT.PERDY bit set to 1. These bits are mirrored in the PKTE_LEN descriptor word.
6. The packet engine writes the result descriptor to the RDR. The packet engine sets the PKTE_CTL_STAT.PERDY bit to 1 and the PKTE_CTL_STAT.HOSTRDY bit to 0 to indicate that the ownership has passed to the host. These bits are mirrored in the PKTE_LEN descriptor word.
7. The packet engine decrements the value in the PKTE_CDSC_CNT register. If the value is not zero, the packet engine reads with the next command descriptor (step 3).
8. The packet engine increments the value in the PKTE_RDSC_CNT register.
9. The host processor reads one or more result descriptors from the RDR and processes the results.
10. The host processor must write the PKTE_RDSC_DECR register with the number of processed result descriptors in the RDR.

Descriptor Ownership

The ownership of the command descriptor and result descriptor is set by ownership bits in the first and last word of the respective descriptor, in the PKTE_CTL_STAT register and the PKTE_LEN register. For the command descriptor, it is the processor core that sets the ownership to the packet engine. For the result descriptor, it is the packet engine that sets the ownership to the host processor.

The packet engine reads the ownership bits before processing, irrespective of the mode of the packet engine. The ownership bits are used to validate and identify the descriptor. When two separate rings are used, the packet engine can be programmed to clear the ownership bits of the command descriptors in the CDR, so the host processor knows which descriptors have been processed.

NOTE: This update of the ownership bits can be disabled in the PKTE_CFG register when the host processor actively counts the number of descriptors in the CDR to prevent ring wrapping.

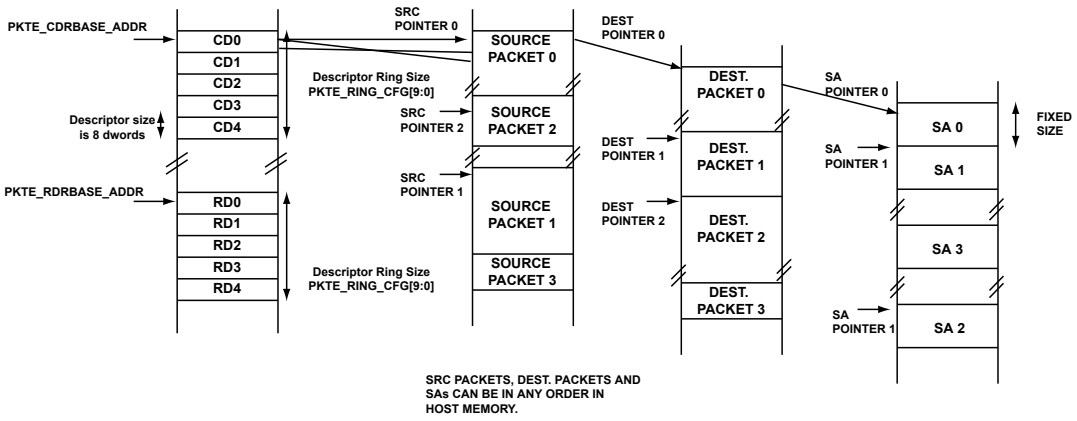


Figure 41-3: Descriptor Rings in Autonomous Ring Mode

SA Record and State Record Structure

The SA record is a packed structure that contains the remainder of the information needed by the packet engine to process a packet. Most of the information fields in the SA record, such as the key and encryption mode, are static for the lifetime of the association. The fields do not require frequent manipulation by the host processor. The SA record non-static fields are the sequence number and sequence number mask.

The SA record can have a corresponding state record that is used to save results from the current operations that can be used for future operations. The state record can hold the IV, the hash byte count, and the intermediate hash digest.

If an SA record is used for operations that use ARC4 processing in stateful mode, it has a corresponding ARC4 state record that holds the ARC4 State.

In this manual, the state record and the ARC4 state record are referred to as state record.

There is no practical limit to how many SA records and corresponding state records the packet engine can support.

In the autonomous ring mode and target command mode, once the packet engine has validated a command descriptor, it automatically fetches the SA record and optional state record. After processing, the packet engine updates the stateful fields in the SA record and state record in the host processor memory.

In direct host mode, after the descriptor is validated, the host must write the SA record directly into the internal registers of the packet engine. After processing, the host reads the stateful fields from the SA registers of the packet engine and saves them back to the SA record in the host processor memory.

SA Record Structure

The ***SA Record Structure*** table shows the structure for an SA Record. When using direct host mode, the corresponding elements are accessed directly with the registers. When using autonomous ring mode or target command mode, the SA Record is defined, configured, and accessed in host memory.

Table 41-22: SA Record Structure

Word Offset	Description (name)	Use
0	PKTE_SA_CMD0[31:0]	SA Control word 0 (all operations)
1	PKTE_SA_CMD1[31:0]	SA Control word 1 (all operations)
2	PKTE_SA_KEY0[31:0]	Key word (DES, Triple-DES, AES-128/192/256, ARC4,)
3	PKTE_SA_KEY1[63:32]	Key word (DES, Triple-DES, AES-128/192/256, ARC4,)
4	PKTE_SA_KEY2[95:64]	Key word (Triple-DES, AES-128/192/256, ARC4,)

Table 41-22: SA Record Structure (Continued)

Word Offset	Description (name)	Use
5	PKTE_SA_KEY3[127:96]	Key word (Triple-DES, AES-128/192/256, ARC4,)
6	PKTE_SA_KEY4[159:128]	Key word (Triple-DES, AES-192/256)
7	PKTE_SA_KEY5[191:160]	Key word (Triple-DES, AES-192/256)
8	PKTE_SA_KEY6[223:192]	Key word (AES-256)
9	PKTE_SA_KEY7[255:224]	Key word (AES-256)
10	PKTE_SA_IDIGEST0[31:0]	Inner Hash digest (Basic Hash and HMAC with MD5,SHA-1, SHA-224, SHA-256)
11	PKTE_SA_IDIGEST1[63:32]	Inner Hash digest (Basic Hash and HMAC with MD5,SHA-1, SHA-224, SHA-256)
12	PKTE_SA_IDIGEST2[95:64]	Inner Hash digest (Basic Hash and HMAC with MD5,SHA-1, SHA-224, SHA-256)
13	PKTE_SA_IDIGEST3[127:96]	Inner Hash digest (Basic Hash and HMAC with MD5,SHA-1, SHA-224, SHA-256)
14	PKTE_SA_IDIGEST4[159:128]	Inner Hash digest (Basic Hash and HMAC with SHA-1, SHA-224, SHA-256)
15	PKTE_SA_IDIGEST5[191:160]	Inner Hash digest (Basic Hash and HMAC with SHA-224, SHA-256)
16	PKTE_SA_IDIGEST6[223:192]	Inner Hash digest (Basic Hash and HMAC with SHA-224, SHA-256)
17	PKTE_SA_IDIGEST7[255:224]	Inner Hash digest (Basic Hash and HMAC with SHA-256)
18	PKTE_SA_ODIGEST0[31:0]	Outer Hash digest (HMAC with MD5, SHA-1, SHA-224, SHA-256)

Table 41-22: SA Record Structure (Continued)

Word Offset	Description (name)	Use
19	PKTE_SA_ODIGEST1[63:32]	Outer Hash digest (HMAC with MD5, SHA-1, SHA-224, SHA-256)
20	PKTE_SA_ODIGEST2[95:64]	Outer Hash digest (HMAC with MD5, SHA-1, SHA-224, SHA-256)
21	PKTE_SA_ODIGEST3[127:96]	Outer Hash digest (HMAC with MD5, SHA-1, SHA-224, SHA-256)
22	PKTE_SA_ODIGEST4[159:128]	Outer Hash digest (HMAC with SHA-1, SHA-224, SHA-256)
23	PKTE_SA_ODIGEST5[191:160]	Outer Hash digest (HMAC with SHA-224, SHA-256)
24	PKTE_SA_ODIGEST6[223:192]	Outer Hash digest (HMAC with SHA-224, SHA-256)
25	PKTE_SA_ODIGEST7[255:224]	Outer Hash digest (HMAC with SHA-256)
26	PKTE_SA_SPI[31:0]	SPI (IPsec), Type[23:16] / Version [15:0] (SSL, TLS, DTLS)
27	PKTE_SA_SEQNUM0[31:0]	Sequence Number (IPsec, SSL, TLS, DTLS with Header Processing)
28	PKTE_SA_SEQNUM1[63:32]	
29	PKTE_SA_SEQNUM_MSK0[31:0]	Sequence Number Mask (IPsec, DTLS inbound with Header Processing)
30	PKTE_SA_SEQNUM_MSK1[63:32]	
31	PKTE_SA_NONCE[31:0] / PKTE_SA_READY	Nonce value (AES-CTR, AES-ICM)/ARC4, i and j pointers (ARC4,)/SA ready indicator (Direct Host Mode)

Some of these fields may be updated by the packet engine. These include:

- PKTE_SA_SEQNUM0
- PKTE_SA_SEQNUM1
- PKTE_SA_SEQNUM_MSK0
- PKTE_SA_SEQNUM_MSK1

All the other fields remain unchanged.

SA State Structure

The security association state structure contains information that may be updated after each packet, such as the IV and the intermediate hash result. The *SA State Structure* table shows the SA state structure and usage. In direct host mode, the elements are accessed directly using the PKTE registers. In target command mode and autonomous ring mode, this structure is defined and updated in host memory.

Table 41-23: SA State Structure

Word Offset	Description (name)	Use
0	PKTE_STATE_IV0[31:0]	Initialization Vector (DES, Triple DES, AES)
1	PKTE_STATE_IV1[63:32]	Initialization Vector (DES, Triple DES, AES)
2	PKTE_STATE_IV2[95:64]	Initialization Vector (AES)
3	PKTE_STATE_IV3[127:96]	Initialization Vector (AES)
4	PKTE_STATE_BYTE_CNT0[31:0]	Current hash byte count (MD5, SHA-1, SHA-224, SHA-256)
5	PKTE_STATE_BYTE_CNT1[63:32]	Current hash byte count (MD5, SHA-1, SHA-224, SHA-256)
6	PKTE_STATE_IDIGEST0[31:0]	Inner Hash digest (mirror of PKTE_SA_IDIGEST0)
7	PKTE_STATE_IDIGEST1[63:32]	Inner Hash digest (mirror of PKTE_SA_IDIGEST1)
8	PKTE_STATE_IDIGEST2[95:64]	Inner Hash digest (mirror of PKTE_SA_IDIGEST2)
9	PKTE_STATE_IDIGEST3[127:96]	Inner Hash digest (mirror of PKTE_SA_IDIGEST3)
10	PKTE_STATE_IDIGEST4[159:128]	Inner Hash digest (mirror of PKTE_SA_IDIGEST4)
11	PKTE_STATE_IDIGEST5[191:160]	Inner Hash digest (mirror of PKTE_SA_IDIGEST5)
12	PKTE_STATE_IDIGEST6[223:192]	Inner Hash digest (mirror of PKTE_SA_IDIGEST6)
13	PKTE_STATE_IDIGEST7[255:224]	Inner Hash digest (mirror of PKTE_SA_IDIGEST7)

ARC4 State Structure

The *ARC4 State Structure* table describes the state structure used with ARC4. When using the PKTE in direct host mode, these fields are accessed with the registers, starting at the value in the `PKTE_ARC4STATE_BUF` register. When using the PKTE in autonomous ring mode or target command mode, these fields are defined and accessed in a structure in host memory.

Table 41-24: ARC4 State Structure

Word Offset	Description (name)*1	Use
0	PKTE_ARC4_STATE0[3:0]*2	ARC4 (Basic, SSL and TLS)
1	PKTE_ARC4_STATE1[7:4]	
...	...	
62	PKTE_ARC4_STATE2[251:248]	
63	PKTE_ARC4_STATE3[255:252]	

*1 The indices in these fields indicate bytes.

*2 There are no corresponding named registers for these fields. PKTE_ARC4_STATE0 corresponds to `PKTE_ARC4STATE_BUF` and PKTE_ARC4_STATE1 corresponds to the 32-bit register following `PKTE_ARC4STATE_BUF` and so on.

Configuring Operations in the PKTE

The operation (cipher, hash function, and others) that the PKTE performs is configured primarily in the `PKTE_SA_CMD0` register. The following sections include a series of tables to help configure the least significant 16 bits of the `PKTE_SA_CMD0` register. These fields include:

- The operation code field (`PKTE_SA_CMD0.OPCD`)
- The direction field (`PKTE_SA_CMD0.DIR`)
- The operation group field (`PKTE_SA_CMD0.OPGRP`)
- The padding type (`PKTE_SA_CMD0.PADTYPE`)
- The cipher selection (`PKTE_SA_CMD0.CIPHER`)
- The hash selection (`PKTE_SA_CMD0.HASH`)

Basic Operations and Decoding

Table 41-25: Basic Operation Decoding

Outbound				Inbound			
OpGroup	Dir	OpCode	Operation	OpGroup	Dir	OpCode	Operation
0b00	0	0b000	Encrypt	0b00	1	0b000	Decrypt
0b00	0	0b001	Encrypt - Hash	0b00	1	0b001	Hash - Decrypt
0b00	0	0b010	Reserved	0b00	1	0b010	Reserved
0b00	0	0b011	Hash	0b00	1	0b011	Hash
0b00	0	0b100... 0b110	Reserved	0b00	1	0b100... 0b110	Reserved
0b00	0	0b111	PRNG	0b00	1	0b111	Reserved

Table 41-26: Protocol Operation Decoding

Outbound				Inbound			
OpGroup	**Dir**	**OpCode**	**Operation**	**OpGroup**	**Dir**	**OpCode**	**Operation**
0b01	0	0b000	ESP Outbound	0b01	1	0b000	ESP Inbound
0b01	0	0b001... 0b011	Reserved	0b01	1	0b001	Reserved
0b01	0	0b100	Basic SSL Outbound	0b01	1	0b010	Basic SSL Inbound
0b01	0	0b101	Basic TLS Outbound	0b01	1	0b011	Basic TLS Inbound
0b01	0	0b110	Reserved	0b01	1	0b100... 0b110	Reserved
0b01	0	0b111	SRTP Outbound	0b01	1	0b111	SRTP Inbound

NOTE: For SSL/TLS and SRTP, no header processing is performed in hardware.

Table 41-27: Extended Protocol Operation Decoding

Outbound				Inbound			
OpGroup	**Dir**	**OpCode**	**Operation**	**OpGroup**	**Dir**	**OpCode**	**Operation**
0b11 0	0	0b000	Reserved	0b11	1	0b000	Reserved
0b11 0	0	0b001	DTLS Outbound	0b11	1	0b001	DTLS Inbound
0b11 0	0	0b010... 0b011	Reserved	0b11	1	0b010... 0b011	Reserved
0b11 0	0	0b100	Ext. SSL Outbound	0b11	1	0b100	Ext. SSL Inbound
0b11 0	0	0b101	Ext. TLS v1.0 Outbound	0b11	1	0b101	Ext. TLS v1.0 Inbound
0b11 0	0	0b110	Ext. TLS v1.1 Outbound	0b11	1	0b110	Ext. TLS v1.1 Inbound
0b11 0	0	0b111	Reserved	0b11	1	0b111	Reserved

Error Code Description

The PKTE_CTL_STAT register is used to configure the packet engine for processing in Direct Host Mode (DHM) or Target Command Mode (TCM). The PKTE_CTL_STAT structure element in memory is used when the packet

engine is configured for Autonomous Ring Mode (ARM). In both cases, when an operation is started, errors are reported in the status field (bits [23:16]) of this register or structure element. The ***Extended Error Codes - Status Encoding*** table provides a guide on how to decipher the meaning of the bits that are set when an error occurs.

Extended Error Codes

The following table provides information about the extended errors associated with the PKTE module.

Table 41-28: Extended Error Codes - Status Encoding

STATUS bits [23:16]	Hex Value	Priority	Description	Processing Result
0b0000_0000	0x00	NA	Successful completion. No errors occurred during processing of the packet.	Packet fully processed
0b----_---1	0x-1	NA	Authentication Error. For an inbound IPsec ESP operation, the Integrity Check Value (ICV) does not match the computed value. For an inbound SRTP operation, the authentication tag does not match the computed value. For a basic SSL/TLS, Extended SSL/TLS or DTLS operation the Message Authentication Code (MAC) does not match the computed value.	Packet fully processed
0b----_--1-	0x-2	NA	Pad Verify Error. For inbound operations that use pad type Constant TLS, IPsec or PKCS#7, the decrypted pad does not match the expected values for the selected pad type.	Packet fully processed
0b----_-1--	0x-4	NA	Sequence Number Error. For an inbound IPsec or DTLS operation, there was a fault in the Anti-Replay Sequence Number. For an outbound IPsec packet, the sequence number overflows; count is 2^{32}– 1 and increments to 0. For an outbound DTLS operation, the sequence number overflows; count is 2^{48}– 1 and increments to 0. For an outbound SSL or TLS operation, the sequence number overflows; count is 2^{64}– 1 and increments to 0.	Packet fully processed
0b0000_1---	0x08	1	System Bus error. The requester bus interface generates an error due to ERROR response from system completer. The completer bus interface generates an error due to request for non-word (32-bit) access.	Packet is stopped. The host must reject the packet and apply a hardware reset to the system.

Table 41-28: Extended Error Codes - Status Encoding (Continued)

STATUS bits [23:16]	Hex Value	Priority	Description	Processing Result
0b0001_1---	0x18	2	Invalid Command Descriptor Error. The ownership bits in the command descriptor are not set to the packet engine, after the PKTE_CDSC_CNT register is incremented.	Command descriptor is ignored, no packet is processed. The packet must be re-queued or discarded.
0b0010_1---	0x28	3	Invalid Crypto Operation Error. A reserved operation is selected.	The SA record is ignored, no packet is processed. The packet must be re-queued or discarded.
0b0011_1---	0x38	4	Invalid Crypto Algorithm Error. A reserved cipher is selected, refer to PKTE_SA_CMD0.CIPHER. A reserved hash is selected, refer to PKTE_SA_CMD0.HASH.	The SA record is ignored, no packet is processed. The packet must be re-queued or discarded.
0b0100_1---	0x48	5	SPI Error. On an inbound packet, the 32-bit SPI value in the packet does not match the value in the SA while header processing is enabled. Note: A failure caused by an SPI mismatch, in general should not occur because the host checks the SPI and does not send an incorrect SPI to the packet engine.	Packet is fully processed. The host must reject the packet.
0b0101_1---	0x58	3	Zero Length Error. The packet length defined in the command descriptor PKTE_LEN.TOTLEN is zero, which is illegal.	Packet command is ignored, no packet processed. The host must reject the packet.

Table 41-28: Extended Error Codes - Status Encoding (Continued)

STATUS bits [23:16]	Hex Value	Priority	Description	Processing Result
0b0110_1xxx	0x68	6	Invalid Packet Length Error. For Basic Encrypt-Hash and Hash-Decrypt operations: `PKTE_LEN.TOTLEN` < Hash/Encrypt Offset For IPsec ESP inbound operations: `PKTE_LEN.TOTLEN` < ICV length or `PKTE_LEN.TOTLEN` is non-4 byte aligned For SRTP inbound operations: `PKTE_LEN.TOTLEN` ≤ IV (opt.) + Bypass Offset + ROC For SSL inbound operations: `PKTE_LEN.TOTLEN` ≤ 1 or packet length > 65535 bytes (SSL packet-bypass length) For TLS and DTLS inbound operations: `PKTE_LEN.TOTLEN` ≤ 13 or payload length > 65535 bytes (data to be hashed) Note: For IPsec ESP the ICV is stripped before the length is checked.	Packet processing is stopped Result packet length is zero. The host must reject the packet and apply a software reset of the packet engine.
0b0111_1xxx	0x78	7	Block Size Error. The length of the inbound packet defined in the Command Descriptor `PKTE_LEN.TOTLEN` is not a multiple of the DES or AES block cipher length. For outbound packets the size is always automatically aligned (padded) to the correct block size. The hashed packet length is not a multiple of the hash block size for intermediate hash operation. For a final hash operation no error is generated. Note: For IPsec ESP operations the ICV is stripped before the block size is checked.	Packet is fully processed. The host must reject the packet.
0b1000_1xxx	0x88	8	Processing Error. The number of bytes in the input buffer is more than defined in the `PKTE_LEN.TOTLEN` field. The number of bytes written to the output buffer is less than processed in the datapath.	Packet processing stops. Result packet length is zero. The host must reject the packet and apply a software reset.
0b1010_1xxx 0b1111_1xxx	Reserved			

Number Format

When dealing with cryptographic functions, data and keys are large vectors. For instance, AES supports keys of sizes 128, 192, and 256 bits. When a key needs to be loaded or read, multiple 32-bit key registers are used, namely

PKTE_SA_KEY[n] registers. The first key register PKTE_SA_KEY[n]0 holds bits 31:0, and PKTE_SA_KEY[n]1 holds the next thirty two bits 63:32, and so on.

Generally, for large vectors defined as Byte0, Byte1, Byte2, Byte3 and so on, the values are stored in the PKTE registers as PKTE_REG0 = Byte3Byte2Byte1Byte0, followed by PKTE_REG1 = Byte7Byte6Byte5Byte4 and so on.

PKTE Programming Examples

Use these examples to extend your understanding of PKTE features, operating modes, event control, and programming modes.

Calculating SHA in Direct Host Mode

This section describes how to configure the packet engine to calculate a hash digest using one of supported SHA algorithms in direct host mode. This configuration follows the procedure outlined in the *PKTE Programming Model* section.

1. Configure the packet engine for direct host mode by setting the PKTE_CFG.MODE bit =0
2. Set the ownership back to the packet engine to process a command descriptor by setting the PKTE_CTL_STAT.PERDY bit =0 and the PKTE_CTL_STAT.HOSTRDY bit =1.
3. Set the PKTE_CTL_STAT.HASHFINAL bit to indicate this command descriptor handles all the input data for the hash calculation. This configuration is needed for the packet engine because the last block requires special handling (see FIPS 180-4 for details).
4. Set size of the input data in bytes in the PKTE_LEN.TOTLEN bit field.
5. Also set the PKTE_LEN.PEDONE bit =0 and the PKTE_LEN.HSTRDY bit =1. These bits must be the same as the PKTE_CTL_STAT.PERDY and PKTE_CTL_STAT.HOSTRDY bits to guarantee ownership.
6. Set the PKTE_CDSC_CNT register =1 to trigger the packet engine to start validating the command descriptor. In this case, the PKTE_CTL_STAT, PKTE_LEN and PKTE_CDSC_CNT registers are the only command descriptor registers modified.
7. Configure the PKTE_SA_CMD0 and PKTE_SA_CMD1 registers to define the operation. For an SHA, set the PKTE_SA_CMD0.OPCD bit field =0b011 for hash operation and the PKTE_SA_CMD0.OPGRP bit field =0b00 for basic operation.
8. Select the specific SHA function using the PKTE_SA_CMD0.HASH bit field as follows.
    - For SHA-1, PKTE_SA_CMD0.HASH =0b0001
    - SHA-224, PKTE_SA_CMD0.HASH =0b0010
    - for SHA-256, PKTE_SA_CMD0.HASH =0b0011
9. Depending on the SHA selected, the appropriate digest length must be chosen for the PKTE_SA_CMD0.DIGESTLEN bit field as follows.

- For SHA-1, PKTE_SA_CMD0.DIGESTLEN =0b0101 (5 words)
- For SHA-224, PKTE_SA_CMD0.DIGESTLEN =0b0111 (7 words)
- For SHA-256, PKTE_SA_CMD0.DIGESTLEN =0b1000 (8 words)

10. The SHA specifies initial constants. These constants can be pre-loaded or read from memory. In this example, by setting the PKTE_SA_CMD0.HASHSRC bit field =0b11, the packet engine provides the correct initial constants depending on the SHA chosen.
11. Next, set the PKTE_SA_CMD1.CPYDGST bit =1 and PKTE_SA_CMD1.CPYPAD bit =1 to move the result to the output buffer of the packet engine at the PKTE_DATAIO_BUF location.
12. At this point, write to the PKTE_SA_RDY register with any value to trigger the operation.
13. Start writing the input to the data buffer of the packet engine starting at the PKTE_DATAIO_BUF location.
14. Write the PKTE_INBUF_CNT register with the length of the input rounded up to the next multiple of 4. For example, if the input length is 30 bytes, set this register to 32.
15. Poll the PKTE_STAT register to see if any errors occurred or if the operation completed without errors.

Once the operation is done, the digest is available in the packet engine data I/O buffer.

NOTE: The input data or message is input into the packet engine data buffer in big endian format while the result or digest is little endian format.

Performing AES Decryption in Direct Host Mode

This section describes how to configure the packet engine to decrypt using AES-128 in direct host mode. This configuration follows the procedure outlined in the *PKTE Programming Model* section.

1. Configure the packet engine for direct host mode by setting the PKTE_CFG.MODE bit =0
2. Start configuring the command descriptor registers. Set the ownership back to the packet engine to process a command descriptor by setting the PKTE_CTL_STAT.PERDY bit =0 and the PKTE_CTL_STAT.HOSTRDY bit =1.
3. Next, configure the PKTE_LEN.TOTLEN bit field with the size of the packet or message to decrypt. If the entire input message (cipher text) fits into the 256-byte data I/O buffer of the packet engine, the process can be done in one shot.
4. Set the PKTE_LEN.PEDONE bit =0 and the PKTE_LEN.HSTRDY bit =1. These bits must have the same setting as the PKTE_CTL_STAT.PERDY and PKTE_CTL_STAT.HOSTRDY bits to guarantee ownership.
5. Set the PKTE_CDSC_CNT register =1 to trigger the packet engine to start validating the command descriptor. In this case, the PKTE_CTL_STAT, PKTE_LEN, and PKTE_CDSC_CNT registers are the only command descriptor registers modified.
6. Next, configure the PKTE_SA_CMD0 and PKTE_SA_CMD1 registers to define the operation.

- For a AES decrypt inbound cipher operation, set the PKTE_SA_CMD0.OPCD bit field =0b000 and the PKTE_SA_CMD0.DIR bit field =0b1.
- Set thePKTE_SA_CMD0.OPGRP bit field =0b00 for basic operation.
- To choose the AES cipher, set the PKTE_SA_CMD0.CIPHER bit field =0b0011. Set the PKTE_SA_CMD0.HASH bit field to 0b1111 to choose the NULL function.

7. Next, set the PKTE_SA_CMD1.AESKEYLEN bit field to select the appropriate key length. In this case, setting it to 0b10 select 128 bits. Also, set PKTE_SA_CMD1.CIPHERMD bit field to select the mode. In this case, setting it to 0b01 select CBC mode.
8. Continue configuring the Security Association (SA) record by loading the key in the PKTE_SA_KEY[n] registers.
9. Next load the initialization vector in the SA state registers (PKTE_STATE_IV[n]).
10. Finally, write anything in to the PKTE_SA_RDY register to trigger the operation.
11. The input data can now be written into the data I/O buffer starting at PKTE_DATAIO_BUF.
12. After the data is written, write the length (or next multiple of 4) into PKTE_INBUF_CNT register.
13. Poll PKTE_STAT to see if any errors occurred or if the operation completed without errors.

Once the operation is done, the result can be found in the same data I/O buffer.

ADSP-SC59x PKTE Register Descriptions

Security Packet Engine (PKTE) contains the following registers.

Table 41-29: ADSP-SC59x PKTE Register List

Name	Description
PKTE_ARC4STATE_ADDR	Packet Engine ARC4 State Record Address
PKTE_ARC4STATE_BUF	Starting Entry of 256-byte ARC4 State Buffer
PKTE_BUF_PTR	Packet Engine Buffer Pointer Register
PKTE_BUF_THRESH	Packet Engine Buffer Threshold Register
PKTE_CDRBASE_ADDR	Packet Engine Command Descriptor Ring Base Address
PKTE_CDSC_CNT	Packet Engine Command Descriptor Count Register
PKTE_CDSC_INCR	Packet Engine Command Descriptor Count Increment Register
PKTE_CFG	Packet Engine Configuration Register
PKTE_CLK_CTL	PE Clock Control Register
PKTE_CONT	PKTE Continue Register
PKTE_CTL_STAT	Packet Engine Control Register

Table 41-29: ADSP-SC59x PKTE Register List (Continued)

Name	Description
PKTE_DATAIO_BUF	Starting Entry of 256-byte Data Input/Output Buffer
PKTE_DEST_ADDR	Packet Engine Destination Address
PKTE_DMA_CFG	Packet Engine DMA Configuration Register
PKTE_ENDIAN_CFG	Packet Engine Endian Configuration Register
PKTE_HLT_CTL	Packet Engine Halt Control Register
PKTE_HLT_STAT	Packet Engine Halt Status Register
PKTE_IMSK_DIS	Interrupt Mask Disable Register
PKTE_IMSK_EN	Interrupt Mask Enable Register
PKTE_IMSK_STAT	Interrupt Masked Status Register
PKTE_INBUF_CNT	Packet Engine Input Buffer Count Register
PKTE_INBUF_INCR	Packet Engine Input Buffer Count Increment Register
PKTE_INT_CFG	Interrupt Configuration Register
PKTE_INT_CLR	Interrupt Clear Register
PKTE_INT_EN	Interrupt Enable Register
PKTE_IUMSK_STAT	Interrupt Unmasked Status Register
PKTE_LEN	Packet Engine Length Register
PKTE_OUTBUF_CNT	Packet Engine Output Buffer Count Register
PKTE_OUTBUF_DECR	Packet Engine Output Buffer Count Decrement Register
PKTE_RDRBASE_ADDR	Packet Engine Result Descriptor Ring Base Address
PKTE_RDSC_CNT	Packet Engine Result Descriptor Count Registers
PKTE_RDSC_DECR	Packet Engine Result Descriptor Count Decrement Registers
PKTE_RING_CFG	Packet Engine Ring Configuration
PKTE_RING_PTR	Packet Engine Ring Pointer Status
PKTE_RING_STAT	Packet Engine Ring Status
PKTE_RING_THRESH	Packet Engine Ring Threshold Registers
PKTE_SA_ADDR	Packet Engine SA Address
PKTE_SA_ARC4IJPTR	ARC4 i and j Pointer Register
PKTE_SA_CMD0	SA Command 0
PKTE_SA_CMD1	SA Command 1
PKTE_SA_IDIGEST[n]	SA Inner Hash Digest Registers
PKTE_SA_KEY[n]	SA Key Registers

Table 41-29: ADSP-SC59x PKTE Register List (Continued)

Name	Description
PKTE_SA_NONCE	SA Initialization Vector Register
PKTE_SA_ODIGEST[n]	SA Outer Hash Digest Registers
PKTE_SA_RDY	SA Ready Indicator
PKTE_SA_SEQNUM[n]	SA Sequence Number Register
PKTE_SA_SEQNUM_MSK[n]	SA Sequence Number Mask Registers
PKTE_SA_SPI	SA SPI Register
PKTE_SRC_ADDR	Packet Engine Source Address
PKTE_STAT	Packet Engine Status Register
PKTE_STATE_ADDR	Packet Engine State Record Address
PKTE_STATE_BYTE_CNT[n]	State Hash Byte Count Registers
PKTE_STATE_IDIGEST[n]	State Inner Digest Registers
PKTE_STATE_IV[n]	State Initialization Vector Registers
PKTE_USERID	Packet Engine User ID

Packet Engine ARC4 State Record Address

The PKTE_ARC4STATE_ADDR register holds the start address of the SA ARC4 state record.

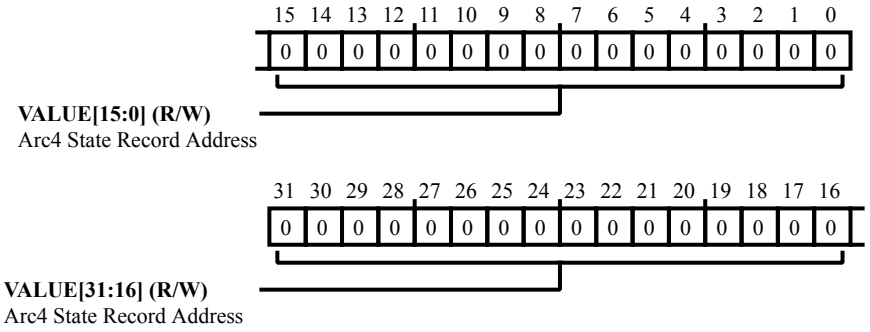


Figure 41-4: PKTE_ARC4STATE_ADDR Register Diagram

Table 41-30: PKTE_ARC4STATE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Arc4 State Record Address.

Starting Entry of 256-byte ARC4 State Buffer

The PKTE_ARC4STATE_BUF register is used to store the pre-processed key that initializes the ARC4 module. In direct host mode, before processing starts, the Host must write the ARC4 state, starting from the base address and increment the address pointer for each write. When processing completes the Host must read the ARC4 state and copy it to the local Host maintained state record. After a reset, a read from any address in the address range of the ARC4 buffer returns an undefined value.

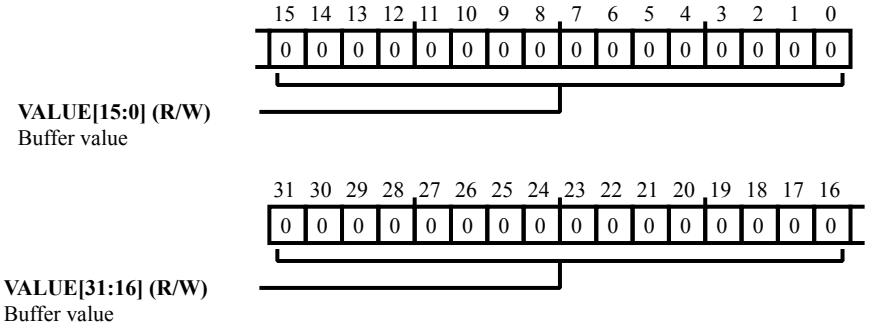


Figure 41-5: PKTE_ARC4STATE_BUF Register Diagram

Table 41-31: PKTE_ARC4STATE_BUF Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Buffer value. The PKTE_ARC4STATE_BUF.VALUE bit field stores the pre-processed key that initializes the ARC4 module.

Packet Engine Buffer Pointer Register

The PKTE_BUF_PTR register contains the offset of the next buffer address (entry) to be read or written by the packet engine. This register is used in direct host mode only.

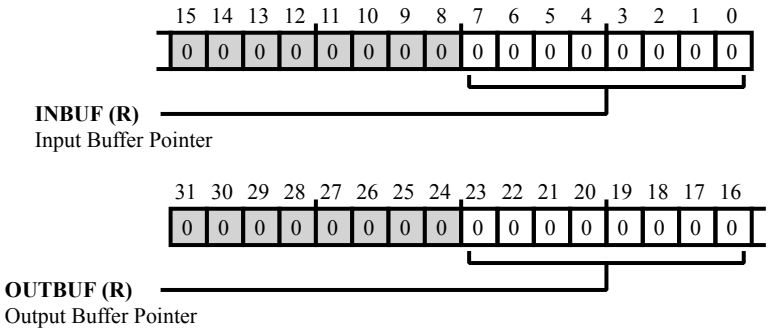


Figure 41-6: PKTE_BUF_PTR Register Diagram

Table 41-32: PKTE_BUF_PTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/NW)	OUTBUF	Output Buffer Pointer. The PKTE_BUF_PTR.OUTBUF bit field indicates the offset of the next address (entry) in the output buffer that will be written next by the packet engine. This bit field is reset to zero after starting up and decremented by 4 at every output buffer write operation. Pointers wrap around; the maximum value this field can have equals the output buffer size minus 4.
7:0 (R/NW)	INBUF	Input Buffer Pointer. The PKTE_BUF_PTR.INBUF bit field indicates the offset of the next address (entry) in the input buffer that will be read next by the packet engine. The bit field is reset to zero after starting up and incremented by 4 at every input buffer read operation. Pointers wrap around; the maximum value this field can have equals the input buffer size minus 4.

Packet Engine Buffer Threshold Register

When in autonomous ring mode or target command mode, the PKTE_BUF_THRESH register defines the high- and low-level value at which the packet engine starts to transfer packet data in or out of the internal packet buffers. These parameters can be used to control the DMA burst size for packet data input and output from the packet engine. In direct host mode, this register contains both threshold values to reduce the amount of packet engine interrupts.

The input buffer threshold (ibufthrsh) interrupt indicates that the input buffer counter is less than or equal to the input buffer threshold value set in this register - this interrupt can be used to wake up a process that stalled on a full input buffer.

The output buffer threshold (obufthrsh) interrupt indicates that the output buffer counter exceeds the output buffer threshold setting. The output buffer interrupt remains active until the output buffer counter is decremented to zero again.

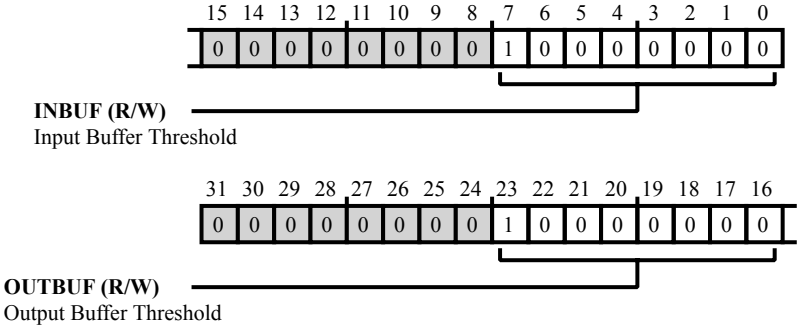


Figure 41-7: PKTE_BUF_THRESH Register Diagram

Table 41-33: PKTE_BUF_THRESH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	OUTBUF	Output Buffer Threshold. The PKTE_BUF_THRESH.OUTBUF bit field specifies how many bytes must be available in the packet engine output buffer before an output transfer starts. Valid values range from 0 to 252, in multiples of 4. In autonomous ring mode, a value of 128 generally gives a good performance, but the optimal value depends on the system and application. In direct host mode, the output buffer threshold (obufthrsh) interrupt activates when the output buffer counter for the output buffer exceeds the value set in this field. A value of 128 generally gives a good performance, but the optimal value depends on the system and application.

Table 41-33: PKTE_BUF_THRESH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	INBUF	Input Buffer Threshold. The `PKTE_BUF_THRESH.INBUF` bit field specifies how many bytes must be free in the packet engine input buffer before an input transfer starts. Valid values range from 0 to 252, in multiples of 4. In autonomous ring mode, a value of 128 generally gives a good performance, but the optimal value depends on the system and application. In direct host mode, the input buffer threshold (ibufthrsh) interrupt activates when the input buffer counter for the input buffer is below or equal the value set in this field. A value of 128 generally gives a good performance, but the optimal value depends on the system and application.

Packet Engine Command Descriptor Ring Base Address

The PKTE_CDRBASE_ADDR register holds the command descriptor ring base address in host memory. It is only applicable in autonomous ring mode. The PKTE_CDRBASE_ADDR register is ignored for all other modes when command descriptors are directly written into the descriptor registers.

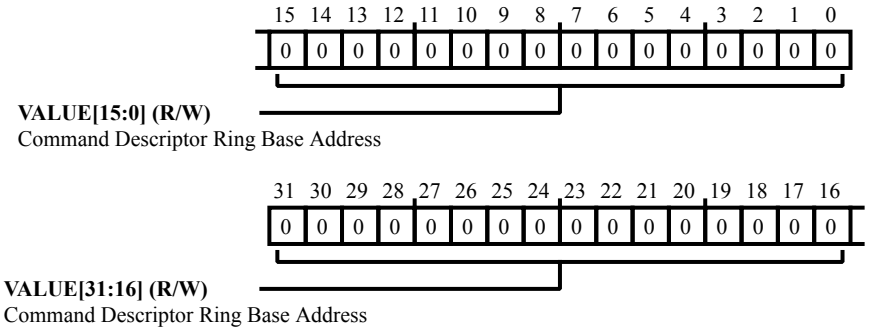


Figure 41-8: PKTE_CDRBASE_ADDR Register Diagram

Table 41-34: PKTE_CDRBASE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command Descriptor Ring Base Address. The PKTE_CDRBASE_ADDR.VALUE bit field specifies the base location of the command descriptor ring in the host memory space.

Packet Engine Command Descriptor Count Register

The PKTE_CDSC_CNT register holds the counter for the number of descriptors in the Command Descriptor Ring (CDR). It is decremented by the packet engine each time a valid descriptor is read from the CDR and processed.

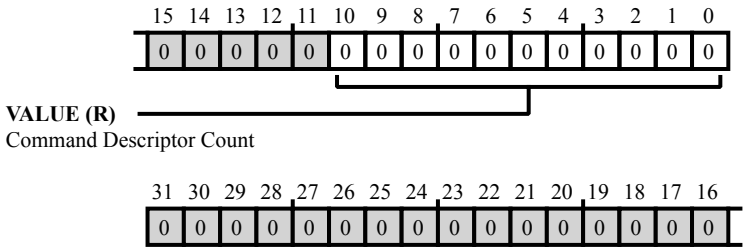


Figure 41-9: PKTE_CDSC_CNT Register Diagram

Table 41-35: PKTE_CDSC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	VALUE	Command Descriptor Count. The PKTE_CDSC_CNT.VALUE bit field provides the number of command descriptors in the command descriptor ring. The packet engine decrements the counter when a valid command descriptor is read from the CDR and processed.

Packet Engine Command Descriptor Count Increment Register

The PKTE_CDSC_INCR register is accessible by the host connected through the system completer bus. The host can increment the command descriptor counter by writing a value between 1 and 255 to the lowest byte of this register.

In autonomous ring mode, the host must prepare 1 to 255 valid command descriptors in the CDR and then write this register with a value between 1 and 255. The write triggers the packet engine to fetch the command descriptors from the CDR. In direct host mode or target command mode, the host must write one valid command descriptor to the internal descriptor registers and then write this register with the value 1, to indicate that one valid descriptor is available.

A CDR threshold interrupt is activated when the command descriptor counter is less than or equal to the threshold value set in the PKTE_RING_THRESH register. This interrupt can be used to wake up a process that stalled on a full CDR.

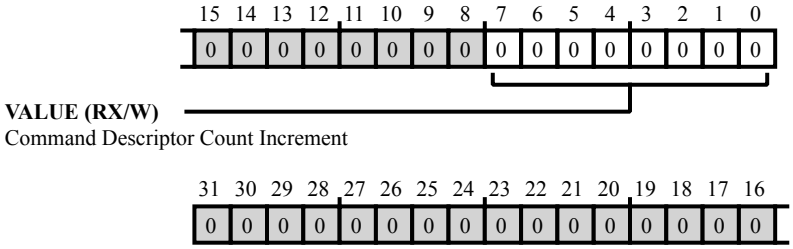


Figure 41-10: PKTE_CDSC_INCR Register Diagram

Table 41-36: PKTE_CDSC_INCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (RX/W)	VALUE	Command Descriptor Count Increment. The value written to the PKTE_CDSC_INCR.VALUE bit field is added to the command descriptor counter. The counter is protected against overflow (see the PKTE_RING_STAT register description). Note that bits[10:8] should be written with zeros.

Packet Engine Configuration Register

The PKTE_CFG register is used to select static settings that control the packet-processing path. This register is typically the last one to be written during the initialization sequence. These settings are typically set at initialization and not changed again.

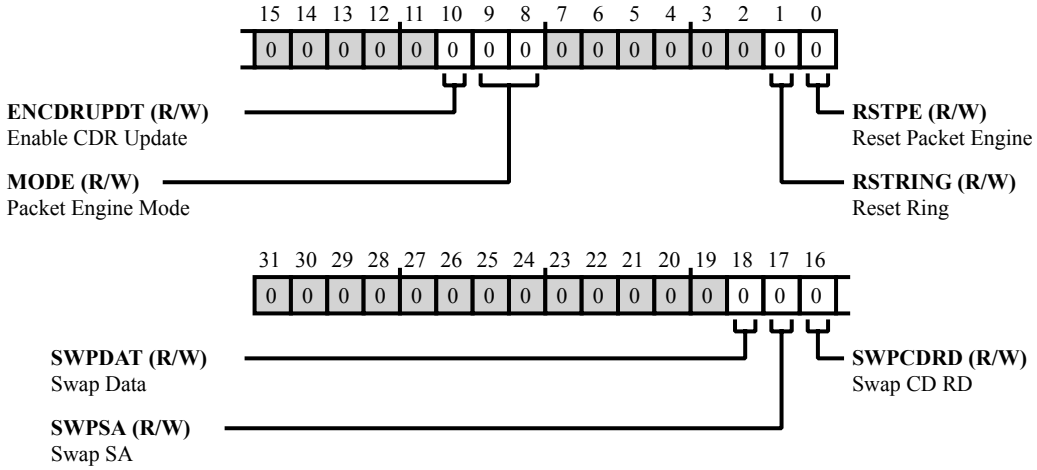


Figure 41-11: PKTE_CFG Register Diagram

Table 41-37: PKTE_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	SWPDAT	Swap Data. The PKTE_CFG.SWPDAT bit enables endian swap for packet data as configured in the PKTE_ENDIAN_CFG.MSTRBSWP bits for the packet data DMA read and write.	
		0	No Endian Swap
		1	Apply Endian Swap
17 (R/W)	SWPSA	Swap SA. The PKTE_CFG.SWPSA bit enables endian swap for a SA record as configured in the PKTE_ENDIAN_CFG.MSTRBSWP bits for the SA record and state record DMA read and write. If the PKTE_ENDIAN_CFG.MSTRBSWP bits specify no endian swap, this bit is ignored.	
		0	No Endian Swap
		1	Apply Endian Swap

Table 41-37: PKTE_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	SWPCDRD	Swap CD RD. The PKTE_CFG.SWPCDRD bit enables endian swap for descriptors as configured in the PKTE_ENDIAN_CFG.MSTRBSWP bits for the command descriptor DMA read and result descriptor DMA write. If the PKTE_ENDIAN_CFG.MSTRBSWP bits specify no endian swap, this bit is ignored.	
		0	No Endian Swap
		1	Apply Endian Swap
10 (R/W)	ENCDRUPDT	Enable CDR Update. The PKTE_CFG.ENCDRUPDT bit enables the packet engine to update, (clear the ownership bits) in the command descriptor in the CDR.	
		0	Do Not Clear Ownership Bits. The packet engine does not clear the ownership bits in the command descriptor when it completes an operation. The host application must clear the ownership bits in "old descriptors" before the packet engine is allowed to wrap around the CDR to re-encounter these "old descriptors". This setting has the advantage of eliminating a separate DMA write to the CDR.
		1	Clear Ownership Bits. The packet engine clears (set to zero) the ownership bits in the current command descriptor in the CDR. This prevents the packet engine from re-processing an "old descriptor" when it wraps around the CDR.
9:8 (R/W)	MODE	Packet Engine Mode. The PKTE_CFG.MODE bit field selects how the packet engine receives commands.	
		0	Direct Host Mode.
		1	Target Command Mode with Result Descriptor Ring Disabled.
		2	Target Command Mode with Result Descriptor Ring Enabled.
		3	Autonomous Ring Mode

Table 41-37: PKTE_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	RSTRING	Reset Ring. The PKTE_CFG.RSTRING bit resets the internal counters for the CDR and RDR, PKTE_CDSC_CNT and PKTE_RDSC_CNT registers) to zero. Resets the PKTE_RING_PTR register to the base address. After the reset the rings are empty. This bit must be written with a '1' to reset the descriptor ring manager and then re-written with a '0' to release the reset. Note that this bit can remain in the reset state if the CDR ring is disabled (PKTE_CFG.MODE is not 0b11). Note that this reset must be coordinated with the 'owner' of the descriptor ring to ensure that the pointers are in sync after the reset.	
		0	Release the Descriptor Ring Manager Reset
		1	Reset the Descriptor Ring Manager
0 (R/W)	RSTPE	Reset Packet Engine. The PKTE_CFG.RSTPE bit resets the packet engine and the state machine logic that drives header processing, DMA, and context management. The PKTE_CFG.RSTPE bit resets the PKTE_CTL_STAT and PKTE_LEN internal registers. This bit must be written with a 1 to reset the packet engine and then re-written with a 0 to release the reset. Note that this bit should not be used by a typical application. It is provided to use during development testing or to recover from critical errors. Note that the PKTE_CTL_STAT.PADVAL and PKTE_CTL_STAT.PADCTLSTAT bit fields are only reset when in autonomous ring mode, but the PKTE_CTL_STAT.PRNGMD bit is not reset. Halt mode is not affected by this reset as well. When exiting out of halt mode, a HW reset is required or a write to the PKTE_CONT register.	
		0	Release the Packet Engine Reset
		1	Reset the Packet Engine

PE Clock Control Register

The PKTE_CLK_CTL register controls the clock enable signals. This register can be used to enable the clock for read and write access to SA registers or to enable the required clock signals for certain crypto functions. The setting of this register overrides the packet engine dynamic clock enable.

In autonomous ring mode and target command modes, this register can be all zeros; the packet engine dynamically requests the external clock manager to activate the module clocks. This register can be used in combination with the debugging interface for internal register access.

In direct host mode, the clock enable bits for the packet engine (PKTE_CLK_CTL.ENPECLK) and for ARC4 (PKTE_CLK_CTL.ENARC4CLK) must be enabled to write and read the SA record and state record registers. All module clocks that are required for the current operation must be enabled during processing.

Note that all the clocks are enabled by default to reset all the registers within the packet engine. After a system reset the host can program this register to disable clocks for power reduction.

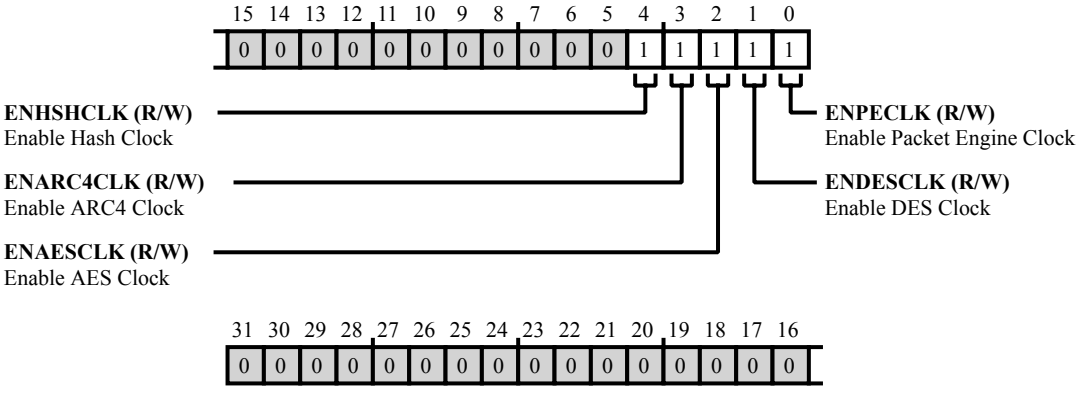


Figure 41-12: PKTE_CLK_CTL Register Diagram

Table 41-38: PKTE_CLK_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	ENHSHCLK	Enable Hash Clock. The PKTE_CLK_CTL.ENHSHCLK bit enables the clock to the hash functions.	
		0	Do not enable the hash clock
		1	Enable the hash clock
3 (R/W)	ENARC4CLK	Enable ARC4 Clock. The PKTE_CLK_CTL.ENARC4CLK bit enables the clock to the ARC4 function.	
		0	Do not enable the ARC4 clock
		1	Enable the ARC4 clock

Table 41-38: PKTE_CLK_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W)	ENAESCLK	Enable AES Clock. The `PKTE_CLK_CTL.ENAESCLK` bit enables the clock to the AES encrypt/decrypt function.	
		0	Do not enable the AES clock
		1	Enable the AES clock
1 (R/W)	ENDESCLK	Enable DES Clock. The `PKTE_CLK_CTL.ENDESCLK` bit enables the clock to the DES function.	
		0	Do not enable the DES clock
		1	Enable the DES clock
0 (R/W)	ENPECLK	Enable Packet Engine Clock. The `PKTE_CLK_CTL.ENPECLK` bit enables the clock in the PKTE data path.	
		0	Do not enable the PKTE data path clock
		1	Enable the PKTE data path clock

PKTE Continue Register

A write to the PKTE_CONT register (with any value) releases the packet engine from a halt state when in halt mode.

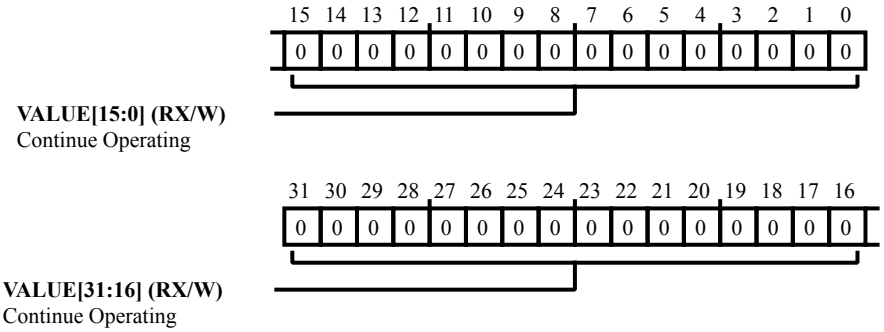


Figure 41-13: PKTE_CONT Register Diagram

Table 41-39: PKTE_CONT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Continue Operating. The PKTE_CONT.VALUE bit field releases the packet engine from a halt state when written with any value in halt mode.

Packet Engine Control Register

The PKTE_CTL_STAT register has a dual function. Together with the data in the SA, this register provides the basic command information for the packet engine to process a packet. When the packet engine successfully or unsuccessfully completes an operation, the packet engine control/status register provides the result status for the host.

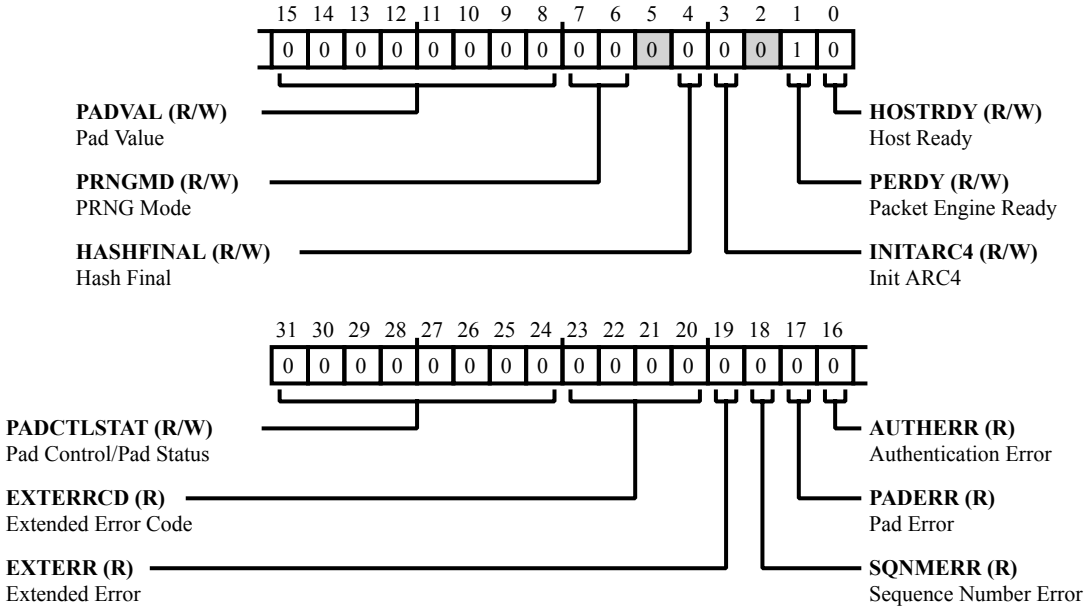


Figure 41-14: PKTE_CTL_STAT Register Diagram

Table 41-40: PKTE_CTL_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	PADCTLSTAT	Pad Control/Pad Status. The PKTE_CTL_STAT.PADCTLSTAT bit field is used to control the pad boundary for pad insertion (outbound) and after processing returns the number of inserted (outbound) or detected (inbound) pad bytes. For the command descriptor, the enumerations below provide the codes for the pad boundary for the outbound operations. This can be used for traffic flow security to conceal the number of payload bytes in an encrypted packet. For the result descriptor inbound operations that use pad types SSL, TLS, IPsec or PKCS#7, it returns the number of detected pad bytes. For all other inbound operations, it returns zero since the other pad modes do not allow implicit determination of pad count. If a pad verify failure occurs, it returns zero. For an outbound operation, it returns the number of inserted pad bytes for all pad types. The pad value includes added bytes such as the pad length and the next header field in an IPsec ESP pad type.	
		0	Align packet end to modulo 8-byte boundary
		1	Align packet end to modulo 1-byte boundary

Table 41-40: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	Align packet end to modulo 4-byte boundary
		4	Align packet end to modulo 8-byte boundary
		8	Align packet end to modulo 16-byte boundary
		16	Align packet end to modulo 32-byte boundary
		32	Align packet end to modulo 64-byte boundary
		64	Align packet end to modulo 128-byte boundary
		128	Align packet end to modulo 256-byte boundary
23:20 (R/NW)	EXTERRCD	Extended Error Code. The PKTE_CTL_STAT.EXTERRCD bit field represents an encoded error condition.	
19 (R/NW)	EXTERR	Extended Error. The PKTE_CTL_STAT.EXTERR bit field provides an extended error code.	
		0	No Extended Error
		1	Extended Error
18 (R/NW)	SQNMERR	Sequence Number Error. The PKTE_CTL_STAT.SQNMERR bit indicates that for an inbound operation, there was a fault in the anti-replay sequence number. For an outbound operation, there was a sequence number overflow condition.	
		0	No Sequence Number Error
		1	Sequence Number Error
17 (R/NW)	PADERR	Pad Error. The PKTE_CTL_STAT.PADERR bit indicates that for an inbound operation the decrypted pad does not match the expected values.	
		0	No Pad Error
		1	Pad Error
16 (R/NW)	AUTHERR	Authentication Error. The PKTE_CTL_STAT.AUTHERR bit indicates that for an inbound operation the authentication value in the packet does not match the computed value.	
		0	No Authentication Error
		1	Authentication Error

Table 41-40: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15:8 (R/W)	PADVAL	Pad Value. The PKTE_CTL_STAT.PADVAL bit field is used to pass the pad value between the host and the packet engine. Command Descriptor: (write-only) For outbound operations that use pad type IPsec, the host must populate this field with the value that is to be inserted into the next header field. For the IPsec ESP operation, this next header is part of the ESP trailer of the innermost operation's header and the value must be 50 decimal. For outbound encrypt operations that use the pad type constant or constant SSL, the host must specify the fixed constant value in this field. For all other outbound and inbound operations, this field is not used. Result Descriptor: (read only) For inbound operations that use pad type IPsec, the packet engine returns the next header field that it detects. For IPsec ESP inbound operations, this is the next header field in the innermost operation's header, which will typically be the value for the payload protocol, such as TCP or UDP. However, in bundling scenarios or in IPv6 with destination option headers, another header value could be seen. For all other outbound operations, the packet engine will not update this field. For all other inbound operations, the returned pad value is zero.	
7:6 (R/W)	PRNGMD	PRNG Mode. The PKTE_CTL_STAT.PRNGMD bits select the pseudo-random number generator mode.	
		0	Operation does not use the PRNG function. Operation does not use the PRNG function.
		1	PRNG Init. PRNG is initialized with a SEED, KEY and an LFSR value as defined in the SA.
		2	PRNG Generate. Pseudo-random data is generated with the LFSR as input value. Before this mode can be used, the PRNG must be initialized with a valid SEED, KEY and LFSR using PRNG Init (PKTE_CTL_STAT.PRNGMD=b'01).
		3	PRNG Test. It can be used to test the PRNG function with custom input data. Before this mode can be used, the PRNG must be initialized once with a valid SEED using PRNG Init (PKTE_CTL_STAT.PRNGMD=b'01).

Table 41-40: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	HASHFINAL	Hash Final. When the PKTE_CTL_STAT.HASHFINAL bit is zero, the data to be hashed must be a multiple of the hash block size, 64 bytes for SHA-1, MD5, SHA-224, SHA-256. This bit is only applicable for Basic Hash, Basic Encrypt-Hash and Basic Hash-Decrypt operations that use the SHA-1, MD5, SHA-224, SHA-256 hash algorithm. The PKTE_CTL_STAT.HASHFINAL bit is overruled for HMAC operations that always completes the hash and always returns the last written value on a read by the host.	
		0	Perform Intermediate Hash Operation. The packet engine performs an intermediate hash operation by generating an intermediate hash digest on the data presented on the input. No hash pad is applied.
		1	Perform Final Hash Operation. The packet engine appends the required final hash pad and generates the final hash digest on the data presented on the input. This completes the hash operation.
3 (R/W)	INITARC4	Init ARC4. The PKTE_CTL_STAT.INITARC4 bit initializes the ARC4 crypto algorithm with a new key. This bit always returns the last written value on a read by the Host. This bit is only applicable for operations that use the ARC4 algorithm and must be zero for all other operations.	
		0	Load ARC4 State and ARC4 i/j pointer from the SA The ARC4 State and ARC4 i/j pointer are loaded from the SA to continue the encrypt/decrypt processing from the previous algorithm state.
		1	Read ARC4 key from the SA-record and initialize ARC4 S-boxes using this key The ARC4 key is read from the SA-record, the ARC4 S-boxes are initialized using this key, prior to the encryption/decryption of data. This bit overrules Stateful mode as defined in bits [9:8] of PKTE_SA_CMD1.
1 (R/W)	PERDY	Packet Engine Ready. The PKTE_CTL_STAT.PERDY bit indicates that the packet engine has completed processing the command descriptor and returns the result descriptor with ownership set to the host. This bit can be reset to 0 by the host and the packet engine, but only the packet engine can set this bit. When the packet engine is idle (not processing), this bit always returns '1' on a read by the host.	

Table 41-40: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	HOSTRDY	Host Ready. The `PKTE_CTL_STAT.HOSTRDY` bit indicates that the host has populated the command descriptor. This bit can be reset to 0 by the host and the packet engine, but only the host can set this bit. When the packet engine is idle (not processing), this bit always returns '0' on a read by the host.

Starting Entry of 256-byte Data Input/Output Buffer

When in direct host mode, the source packet data is written here to be transferred to the packet engine. The host can monitor the available space in the input buffer through the PKTE_STAT register. This is also the location in the packet engine from where output data is read when in direct host mode. The host can monitor the available bytes in the output buffer through the PKTE_STAT register.

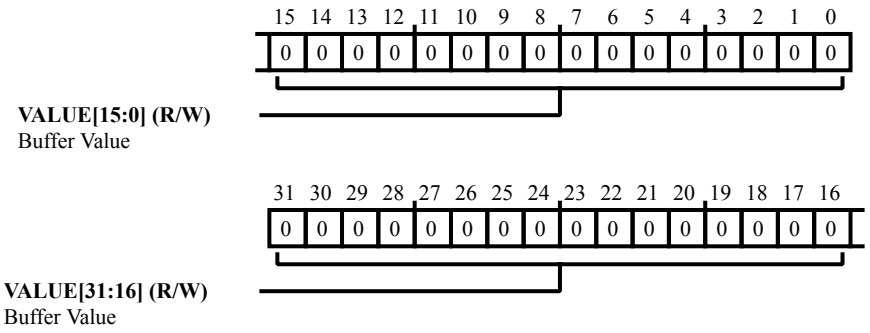


Figure 41-15: PKTE_DATAIO_BUF Register Diagram

Table 41-41: PKTE_DATAIO_BUF Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Buffer Value.

Packet Engine Destination Address

The PKTE_DEST_ADDR register holds the starting (byte) address to write the result packet from the requested operation.

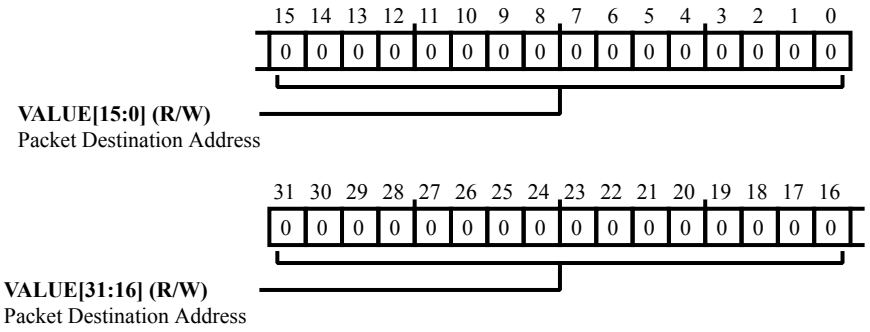


Figure 41-16: PKTE_DEST_ADDR Register Diagram

Table 41-42: PKTE_DEST_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Packet Destination Address.

Packet Engine DMA Configuration Register

The PKTE_DMA_CFG register configures the maximum burst transfer size, enables incremental transfers, and insertions IDLE cycles between two bus transfers.

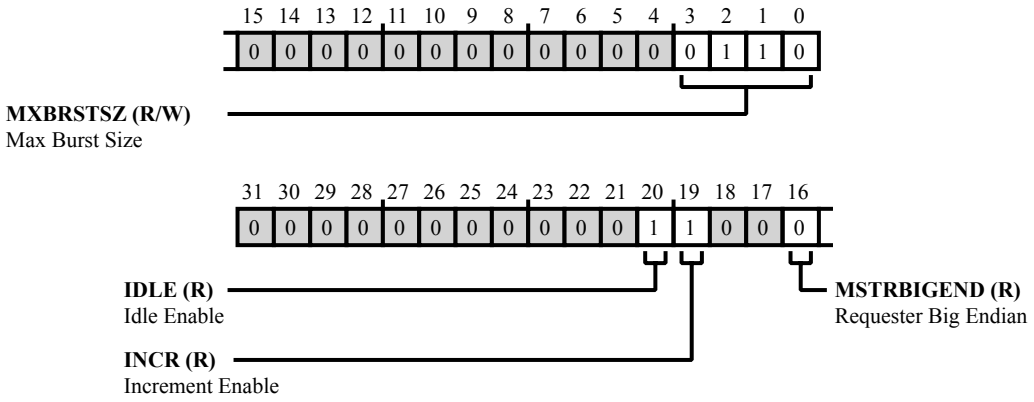


Figure 41-17: PKTE_DMA_CFG Register Diagram

Table 41-43: PKTE_DMA_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/NW)	IDLE	Idle Enable. The PKTE_DMA_CFG.IDLE bit allows the peripheral bus requester to insert one additional IDLE transfer between two successive peripheral bus requester burst operations. This provides the arbiter one additional cycle to hand over the grant to another peripheral bus requester.	
		0	The peripheral bus requester inserts no IDLE cycle between two successive burst operations
		1	The peripheral bus requester inserts one additional IDLE transfer between two successive burst operations

Table 41-43: PKTE_DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	INCR	Increment Enable. The PKTE_DMA_CFG.INCR bit lets the peripheral bus requester generate INC4, INC8 and INC16 type of burst transfers. By default, the peripheral bus requester generates the largest possible incremental burst of unspecified length (INCR) with a maximum length (in bytes) as configured by the PKTE_DMA_CFG.MXBRSTSZ bit field. In case there are less than 4 bytes of data available or the 1kB boundary will be crossed using a burst operation, then a single transfer of size byte is generated. When the PKTE_DMA_CFG.INCR bit is set, the peripheral bus requester generates one or more incremental burst of specified length (INC4, INC8, INC16). In case there is less data available then the smallest possible burst (INC4) or the 1kB boundary will be crossed using a burst operation, then an unspecified length burst or a single transfer of size byte is generated.	
		0	The bus requester will generate only INCR burst types
		1	The bus requester will generate INC4, INC8 and INC16 burst types
16 (R/NW)	MSTRBIGEND	Requester Big Endian. The PKTE_DMA_CFG.MSTRBIGEND bit determines whether the engine is used in a little or big endian system.	
		0	Little endian
		1	Big endian
3:0 (R/W)	MXBRSTSZ	Max Burst Size. The PKTE_DMA_CFG.MXBRSTSZ bit field configures the maximum size of an unspecified length burst (INC) at the bus in bytes. When there is less data available than the PKTE_DMA_CFG.MXBRSTSZ bit field setting or the 1kB boundary will be crossed using a burst operation, then the length of the burst can be less than PKTE_DMA_CFG.MXBRSTSZ. Any requested transfers larger than this size are broken up in to multiple burst transfers of this size or less.	

Packet Engine Endian Configuration Register

The packet engine incorporates a powerful interface specific endian handler. This endian handler allows byte lane swapping in each direction for data passing through the host interface.

The PKTE_ENDIAN_CFG register configures the byte order function for the peripheral bus requester and peripheral bus completer interface. The bits for the peripheral bus requester are combined in four sets of two bits; each group configures a byte swap function for a particular DMA transfer. The same applies for the peripheral bus completer interface.

The PKTE_ENDIAN_CFG register also defines the endian swapping that occurs for host-initiated target transfers and for packet engine requester DMA read and write transfers. Individual endian swap enable bits in the configuration (PKTE_CFG) register can enable the endian swap for various transaction types: command descriptors and result descriptors, SA records and state records, packet data.

In direct host mode, only target operations are supported. Only the target endian configuration of this register is applicable.

Note: This register is typically programmed once during the initialization phase, although software is allowed to dynamically change the setting in this register just before initiating a data transfer. The developer will have to analyze the benefit of the cycles needed to write the endian register dynamically versus handling endian swapping for some data structures in the host system (most modern processors support a byte swap in zero cycles). Certainly the endian swap should be set correctly for the packet data, since this represents the majority of the data transferred.

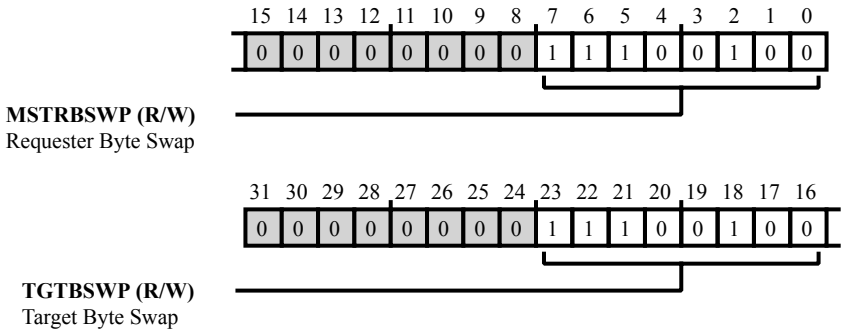


Figure 41-18: PKTE_ENDIAN_CFG Register Diagram

Table 41-44: PKTE_ENDIAN_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	TGTBSWP	Target Byte Swap. The PKTE_ENDIAN_CFG.TGTBSWP bit field configures the byte swap for peripheral bus target transfers. Note that only target word transfers are supported. Each double-bit field in this register specifies the source of the indicated byte lane. The field values are interpreted as follows: 00 = byte 0, 01 = byte 1, 10 = byte 2, 11 = byte 3. Note: Setting the value 0xE4 defines no swap (little endian) and setting value 0x1B defines a full byte swap within a 32-bit word (big endian).
7:0 (R/W)	MSTRBSWP	Requester Byte Swap. The PKTE_ENDIAN_CFG.MSTRBSWP bit field configures the byte swap for peripheral bus requester multi-byte transfers, including command descriptors, result descriptors, SA records, state records and packet data. Separate controls in the PKTE_CFG register can enable this swap individually for each of the 4 types of data. Each double-bit field in this register specifies the source of the indicated peripheral bus byte lane. The field values are interpreted as follows: 00=byte 0, 01=byte 1, 10=byte 2, 11=byte 3. Note: Setting the value 0xE4 defines no swap (little endian) and setting value 0x1B defines a full byte swap within a 32-bit word (big endian).

Packet Engine Halt Control Register

Th PKTE_HLT_CTL register controls the packet engine halt mode. This register can be used for debugging purposes while processing in autonomous ring mode or target command mode. During the halt mode, the host can read all internal registers for examination without side-effects. When halted, the host should not write to any registers. To continue packet engine operation, the host must write to the PKTE_CONT (PKTE continue) register.

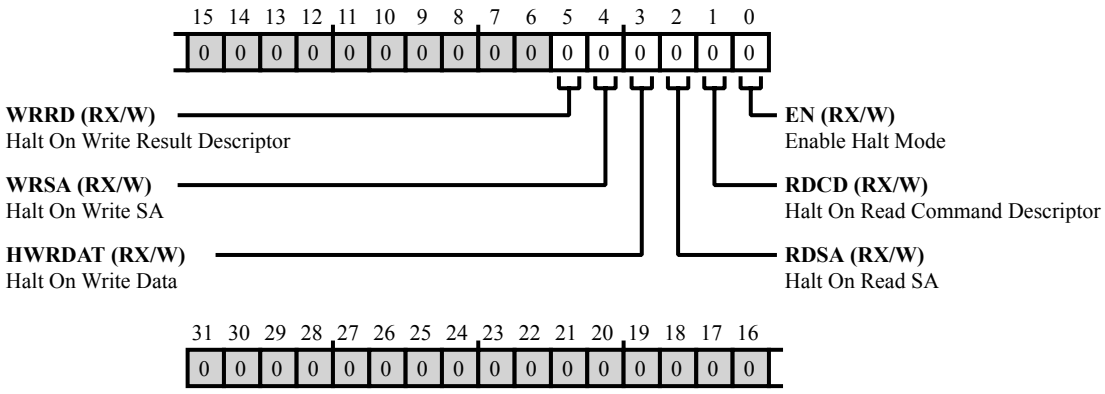


Figure 41-19: PKTE_HLT_CTL Register Diagram

Table 41-45: PKTE_HLT_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (RX/W)	WRRD	Halt On Write Result Descriptor. The PKTE_HLT_CTL.WRRD bit halts the packet engine in the HALT_WRITE_STATUS state after it completes a result descriptor write operation to the result descriptor ring. The host can use this bit to examine the result descriptor that is currently in the host memory.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
4 (RX/W)	WRSA	Halt On Write SA. The PKTE_HLT_CTL.WRSA bit halts the packet engine in the HALT_WRITE_SA state after it completes an SA write operation to the host memory. The host can use this bit to examine the security context that is currently in the host memory.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation

Table 41-45: PKTE_HLT_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (RX/W)	HWRDAT	Halt On Write Data. The PKTE_HLT_CTL.HWRDAT bit halts the packet engine in the HALT_DATA state after it completes writing the result packet data to the host memory. The host can use this bit to examine the result packet that is currently in the host memory.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
2 (RX/W)	RDSA	Halt On Read SA. The PKTE_HLT_CTL.RDSA bit halts the packet engine in the HALT_READ_SA state after it completes an SA read operation from the host memory. The host can use this bit to examine the security context that is currently in the SA registers.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
1 (RX/W)	RDCD	Halt On Read Command Descriptor. The PKTE_HLT_CTL.RDCD bit halts the packet engine in the HALT_READ_DESCR state after it completes a command descriptor read operation from the command descriptor ring. It will halt whether the descriptor is valid or invalid. The host can use this bit to examine the command descriptor that is currently in the internal command descriptor registers.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
0 (RX/W)	EN	Enable Halt Mode. The PKTE_HLT_CTL.EN bit enables halt mode where the packet engine can halt processing at any processing state as indicated by bits [5:1]. When halted, the packet engine continues operation on a write to the PKTE_CONT register.	
		0	Do not enable halt mode
		1	Enable halt mode

Packet Engine Halt Status Register

The PKTE_HLT_STAT register reflects the status of the packet engine in halt mode. This register can be used for debugging purposes while processing in autonomous ring mode or target command mode. When the packet engine is halted, the host can read all internal registers for examination without side effects. The host should not write to any registers.

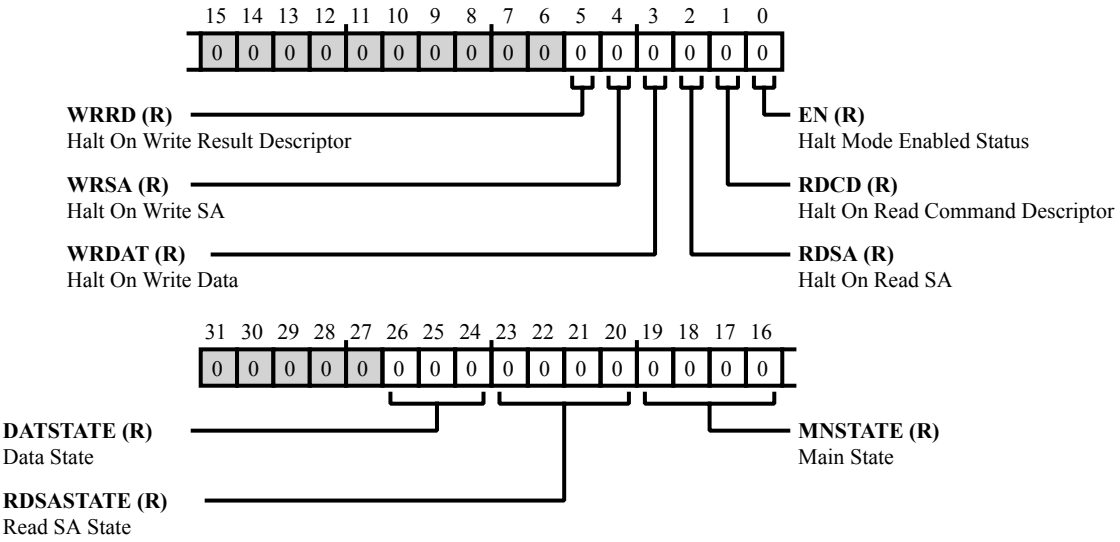


Figure 41-20: PKTE_HLT_STAT Register Diagram

Table 41-46: PKTE_HLT_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
26:24 (R/NW)	DATSTATE	Data State. The PKTE_HLT_STAT.DATSTATE bit field indicates the state of the packet engine read data FSM.	
		0	DATA_IDLE, no operation
		1	DATA_READ
		2	DATA_WRITE
		3	DATA_WAIT
		5	DATA_PAD_READ
		6	DATA_BYP_READ
		7	RESERVED
23:20 (R/NW)	RDSASTATE	Read SA State. The PKTE_HLT_STAT.RDSASTATE bit field indicates the state of the packet engine read SA FSM.	

Table 41-46: PKTE_HLT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	SA_IDLE, no operation
		1	SA_READ_CMD
		2	SA_READ_STATE_IV
		3-5	RESERVED
		6	SA_READ_ARC4_STATE
		7	SA_READ_WAIT
		8	RESERVED
		9	SA_WRITE_PROT_HDR
		10	RESERVED
		11	SA_WRITE_IV
		12	SA_WRITE_DIGEST
		13	SA_WRITE_ARC4_IJ_PNTR
		14	SA_WRITE_ARC4_STATE
		15	SA_WRITE_WAIT
19:16 (R/NW)	MNSTATE	Main State. The `PKTE_HLT_STAT.MNSTATE` bit field indicates the state of the packet engine main FSM.	
		0	MAIN_IDLE, no operation
		1	MAIN_READ_CD, reading command descriptor
		2	MAIN_READ_SA, reading SA
		3	MAIN_DATA, processing data
		4	MAIN_WRITE_SA, writing SA
		5	MAIN_WRITE_STATUS, writing status
		6	MAIN_WRITE_CD, updating command descriptor
		7	MAIN_WRITE_RD, updating result descriptor
		8	MAIN_INIT_WAIT, wait single clock
		9	MAIN_HALT_READ_CD, halt after read command descriptor
		10	MAIN_HALT_READ_SA, halt after read SA
		11	MAIN_HALT_DATA, halt after processing data
		12	MAIN_HALT_WRITE_SA, halt after write SA

Table 41-46: PKTE_HLT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		13	MAIN_WAIT_FOR_CLOCK, wait for clocks to be active
		15	MAIN_HALT_WRITE_RD, halt after write result descriptor
5 (R/NW)	WRRD	Halt On Write Result Descriptor. The PKTE_HLT_STAT.WRRD bit reflects the value in the PKTE_HLT_CTL.WRRD bit.	
4 (R/NW)	WRSA	Halt On Write SA. The PKTE_HLT_STAT.WRSA bit reflects the value in the PKTE_HLT_CTL.WRSA bit.	
3 (R/NW)	WRDAT	Halt On Write Data. The PKTE_HLT_STAT.WRDAT bit reflects the value in the PKTE_HLT_CTL.HWRDAT bit.	
2 (R/NW)	RDSA	Halt On Read SA. The PKTE_HLT_STAT.RDSA bit reflects the value in the PKTE_HLT_CTL.RDSA bit.	
1 (R/NW)	RDCD	Halt On Read Command Descriptor. The PKTE_HLT_STAT.RDCD bit reflects the value in the PKTE_HLT_CTL.RDCD bit.	
0 (R/NW)	EN	Halt Mode Enabled Status.	
		0	Halt mode not enabled
		1	Halt mode enabled

Interrupt Mask Disable Register

The host can use the PKTE_IMSK_DIS register to clear individual bits in the PKTE_INT_EN register for the host interrupt. This register is a bitmap for each of the possible interrupt sources: A 1 clears the interrupt enable bit, a 0 does not affect the interrupt enable bit in the PKTE_INT_EN register. Clearing the enable bits through this register avoids the time-consuming read-modify-write operation on the host.

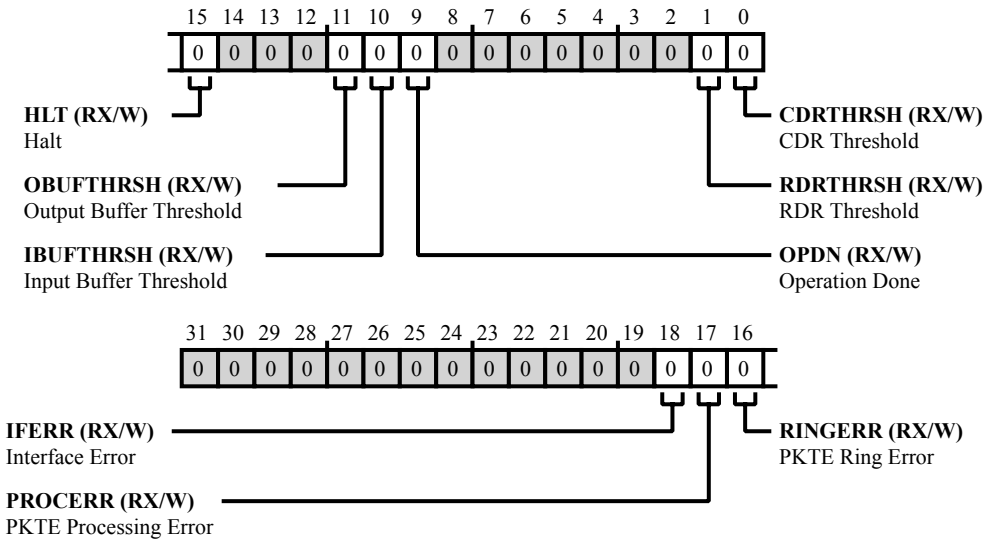


Figure 41-21: PKTE_IMSK_DIS Register Diagram

Table 41-47: PKTE_IMSK_DIS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (RX/W)	IFERR	Interface Error. Write the PKTE_IMSK_DIS.IFERR bit to clear when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (RX/W)	PROCERR	PKTE Processing Error. Write the PKTE_IMSK_DIS.PROCERR bit to clear an extended error that occurred before, during or after processing the current packet in the packet engine.
16 (RX/W)	RINGERR	PKTE Ring Error. Write the PKTE_IMSK_DIS.RINGERR bit to clear a CDR overflow or an RDR underflow.
15 (RX/W)	HLT	Halt. Write the PKTE_IMSK_DIS.HLT bit to clear when the packet engine is in the halt state.

Table 41-47: PKTE_IMSK_DIS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (RX/W)	OBUFTHRSH	Output Buffer Threshold. Write the PKTE_IMSK_DIS.OBUFTHRSH bit to clear the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.
10 (RX/W)	IBUFTHRSH	Input Buffer Threshold. Write the PKTE_IMSK_DIS.IBUFTHRSH bit to clear when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (RX/W)	OPDN	Operation Done.
1 (RX/W)	RDRTHRSH	RDR Threshold. Write the PKTE_IMSK_DIS.RDRTHRSH bit to clear when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in the PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (RX/W)	CDRTHRSH	CDR Threshold. Write the PKTE_IMSK_DIS.CDRTHRSH bit to clear when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Mask Enable Register

The host can use the PKTE_IMSK_EN register to set individual bits in the PKTE_INT_EN register for the host interrupt. This register is a bitmap for each of the possible interrupt sources: A 1 sets the interrupt enable bit, a 0 does not affect the interrupt enable bit in the PKTE_INT_EN register. Setting the enable bits through this register avoids the time-consuming read-modify-write operation on the host.

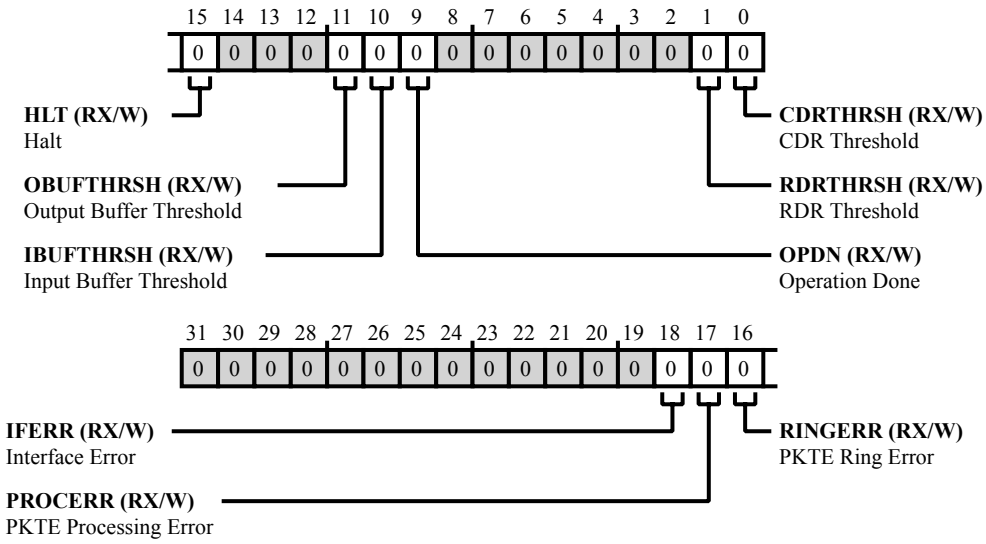


Figure 41-22: PKTE_IMSK_EN Register Diagram

Table 41-48: PKTE_IMSK_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (RX/W)	IFERR	Interface Error. Set the PKTE_IMSK_EN.IFERR bit to indicate a host request for a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (RX/W)	PROCERR	PKTE Processing Error. Set the PKTE_IMSK_EN.PROCERR bit to indicate an extended error occurred before, during or after processing the current packet in the packet engine.
16 (RX/W)	RINGERR	PKTE Ring Error.
15 (RX/W)	HLT	Halt. Set the PKTE_IMSK_EN.HLT bit to indicate when the packet engine is in the halt state.

Table 41-48: PKTE_IMSK_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (RX/W)	OBUFTHRSH	Output Buffer Threshold. Set the PKTE_IMSK_EN.OBUFTHRSH bit to indicate that the output buffer counter exceeds the output buffer threshold value defined in the PKTE_BUF_THRESH.OUTBUF bit.
10 (RX/W)	IBUFTHRSH	Input Buffer Threshold. Set the PKTE_IMSK_EN.IBUFTHRSH bit to indicate the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (RX/W)	OPDN	Operation Done.
1 (RX/W)	RDRTHRSH	RDR Threshold. Set the PKTE_IMSK_EN.RDRTHRSH bit to indicate when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (RX/W)	CDRTHRSH	CDR Threshold. Set the PKTE_IMSK_EN.CDRTHRSH bit to indicate when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Masked Status Register

The PKTE_IMSK_STAT register provides interrupt status visibility to the host, after the interrupt mask is applied. This lets the host view the selected sources of interrupts that are directed to the interrupt output signal, that is connected to the system interrupt controller. As with the unmasked status register, all interrupt bits are latched and must be cleared using the PKTE_INT_CLR register in order to capture a subsequent event. A 1 indicates that the associated interrupt is present.

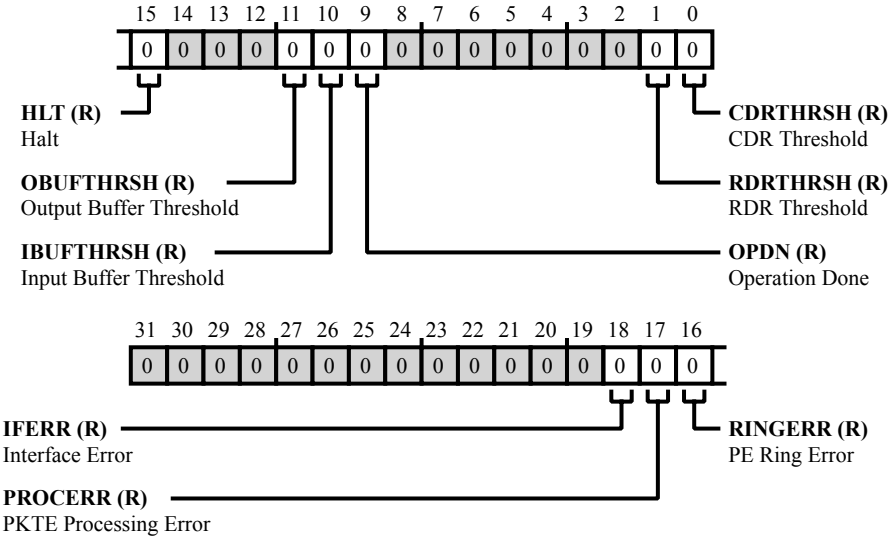


Figure 41-23: PKTE_IMSK_STAT Register Diagram

Table 41-49: PKTE_IMSK_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/NW)	IFERR	Interface Error. The PKTE_IMSK_STAT.IFERR bit is set when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (R/NW)	PROCERR	PKTE Processing Error. The PKTE_IMSK_STAT.PROCERR bit is when an extended error occurred before, during or after processing the current packet in the packet engine.
16 (R/NW)	RINGERR	PE Ring Error. The PKTE_IMSK_STAT.RINGERR bit is set on a CDR overflow or an RDR underflow.
15 (R/NW)	HLT	Halt. The PKTE_IMSK_STAT.HLT bit is set when the packet engine is in the HALT state.

Table 41-49: PKTE_IMSK_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/NW)	OBUFTHRSH	Output Buffer Threshold. The PKTE_IMSK_STAT.OBUFTHRSH bit is set when the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.
10 (R/NW)	IBUFTHRSH	Input Buffer Threshold. The PKTE_IMSK_STAT.IBUFTHRSH bit is set when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (R/NW)	OPDN	Operation Done.
1 (R/NW)	RDRTHRSH	RDR Threshold. The PKTE_IMSK_STAT.RDRTHRSH bit is set when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (R/NW)	CDRTHRSH	CDR Threshold. The PKTE_IMSK_STAT.CDRTHRSH bit is set when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Packet Engine Input Buffer Count Register

The PKTE_INBUF_CNT register provides the number of bytes available in the input buffer. The PKTE_INBUF_CNT register is used in direct host mode only.

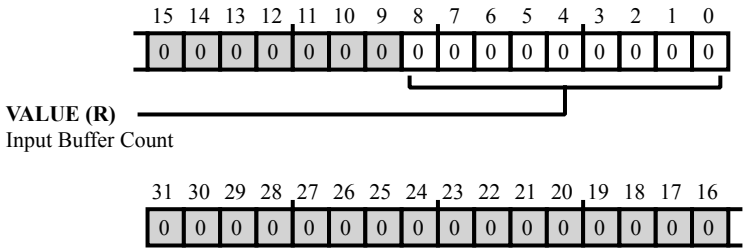


Figure 41-24: PKTE_INBUF_CNT Register Diagram

Table 41-50: PKTE_INBUF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/NW)	VALUE	Input Buffer Count. The PKTE_INBUF_CNT.VALUE bit field provides the number of bytes in the input buffer. The packet engine decrements the counter by 4 when a 32-bit word is read from the input buffer.

Packet Engine Input Buffer Count Increment Register

A host connected through the system completer bus can increment the input buffer counter by writing a value between 4 and 256, in multiples of 4, to the lowest bits of this register. The PKTE_INBUF_INCR register is used in direct host mode only.

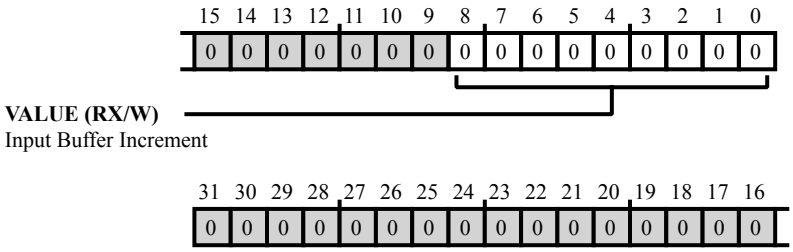


Figure 41-25: PKTE_INBUF_INCR Register Diagram

Table 41-51: PKTE_INBUF_INCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (RX/W)	VALUE	Input Buffer Increment. The value written is added to the input buffer counter. Valid values range from 4 to 256, in multiples of 4.

Interrupt Configuration Register

The PKTE_INT_CFG register configures the interrupt type that is sent to the interrupt line connected to the system interrupt controller. (Note that this only effects the final output of the interrupt subsystem).

Configuring the interrupt output type for pulse causes the interrupt signal to pulse low for two clock cycles when activated. When set for level, the interrupt signal is set low until cleared by the host (it follows the bit in the masked status register). For the host, this is typically set to level.

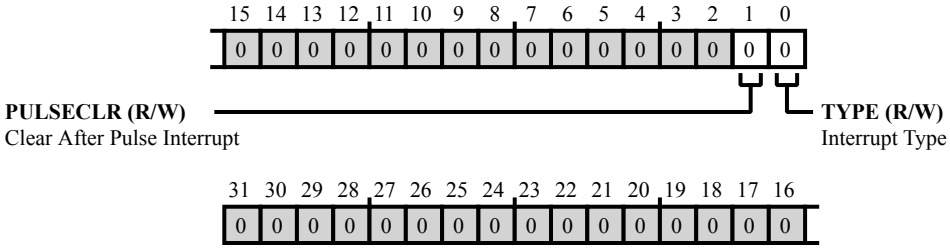


Figure 41-26: PKTE_INT_CFG Register Diagram

Table 41-52: PKTE_INT_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	PULSECLR	Clear After Pulse Interrupt. The PKTE_INT_CFG.PULSECLR bit clears the latched interrupt source after the pulse interrupt.	
		0	Manually clear pulse interrupt source. Do not automatically clear the interrupt sources after pulsing the interrupt output. Clear the source by writing to the PKTE_INT_CLR register.
		1	Automatically clear pulse interrupt source. After pulsing the interrupt output, automatically clear the sources.
0 (R/W)	TYPE	Interrupt Type. The PKTE_INT_CFG.TYPE bit selects the type, pulse or level, for the interrupt output to the system.	
		0	Level. The interrupt output is a level signal that is set low when an enabled interrupt is active until the interrupt is cleared.
		1	Pulse. The interrupt output is a two clock cycle low-active pulse, activated when an enabled interrupt is active.

Interrupt Clear Register

The PKTE_INT_CLR register allows the host processor to clear pending interrupts. A 1 written to a given bit in this register clears the corresponding interrupt. A 0 leaves the interrupt latch unchanged for that position.

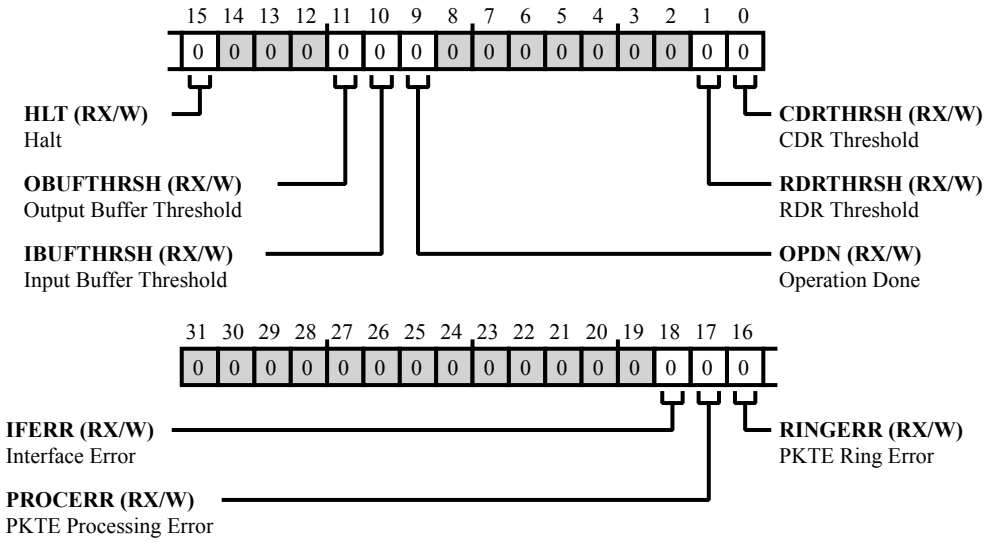


Figure 41-27: PKTE_INT_CLR Register Diagram

Table 41-53: PKTE_INT_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (RX/W)	IFERR	Interface Error. The PKTE_INT_CLR.IFERR bit is set when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (RX/W)	PROCERR	PKTE Processing Error. The PKTE_INT_CLR.PROCERR bit is set when an extended error occurred before, during or after processing the current packet in the packet engine.
16 (RX/W)	RINGERR	PKTE Ring Error. The PKTE_INT_CLR.RINGERR bit is set on a CDR overflow or an RDR underflow.
15 (RX/W)	HLT	Halt. The PKTE_INT_CLR.HLT bit is set when the packet engine is in the HALT state.
11 (RX/W)	OBUFTHRSH	Output Buffer Threshold. The PKTE_INT_CLR.OBUFTHRSH bit is set when the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.

Table 41-53: PKTE_INT_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (RX/W)	IBUFTHRSH	Input Buffer Threshold. The PKTE_INT_CLR.IBUFTHRSH bit is set when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (RX/W)	OPDN	Operation Done.
1 (RX/W)	RDRTHRSH	RDR Threshold. The PKTE_INT_CLR.RDRTHRSH bit is set when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (RX/W)	CDRTHRSH	CDR Threshold. The PKTE_INT_CLR.CDRTHRSH bit is set when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Enable Register

The PKTE_INT_EN register configures the interrupt mask for the host interrupt. This register is a bitmap for each of the possible interrupt sources. A 1 enables the interrupt source and a 0 disables the source. If an interrupt source is disabled, a cleared bit also clears the matching interrupt in the PKTE_IMSK_STAT register.

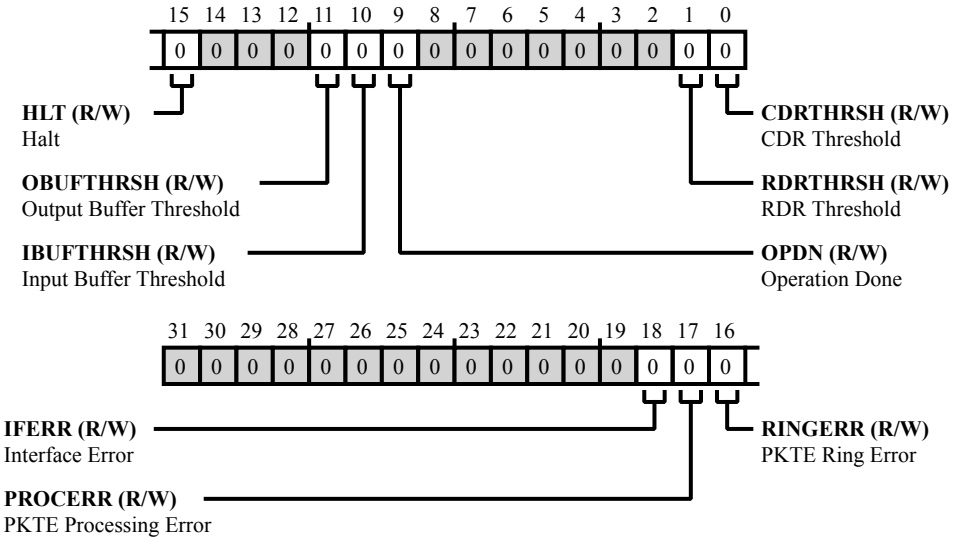


Figure 41-28: PKTE_INT_EN Register Diagram

Table 41-54: PKTE_INT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/W)	IFERR	Interface Error. Set the PKTE_INT_EN.IFERR bit for host requests for a non 32-bit access to the packet engine interrupt or when the packet engine receives an error writing data back out to the host memory system.
17 (R/W)	PROCERR	PKTE Processing Error. Set the PKTE_INT_EN.PROCERR bit to enable the extended error occurred before, during or after processing the current packet in the packet engine interrupt.
16 (R/W)	RINGERR	PKTE Ring Error. Set the PKTE_INT_EN.RINGERR bit to enable the CDR overflow or RDR underflow interrupt.
15 (R/W)	HLT	Halt. Set the PKTE_INT_EN.HLT bit for when the packet engine is in the HALT state.
11 (R/W)	OBUFTHRSH	Output Buffer Threshold. Set the PKTE_INT_EN.OBUFTHRSH bit for to trigger an interrupt when the output buffer counter exceeds the output buffer threshold value defined in the PKTE_BUF_THRESH.OUTBUF bit.

Table 41-54: PKTE_INT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (R/W)	IBUFTHRSH	Input Buffer Threshold. Set the `PKTE_INT_EN.IBUFTHRSH` bit for to trigger an interrupt when the input buffer counter is less than or equal to the input buffer threshold value defined in the `PKTE_BUF_THRESH.INBUF` bit.
9 (R/W)	OPDN	Operation Done.
1 (R/W)	RDRTHRSH	RDR Threshold. Set the `PKTE_INT_EN.RDRTHRSH` bit for to trigger an interrupt when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the `PKTE_RING_THRESH.RDRTHRSH` bit, or the RD counter for the RDR in `PKTE_RDSC_CNT` register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (R/W)	CDRTHRSH	CDR Threshold. Set the `PKTE_INT_EN.CDRTHRSH` bit for to trigger an interrupt when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the `PKTE_RING_THRESH.CDRTHRSH` bit.

Interrupt Unmasked Status Register

The PKTE_IUMSK_STAT register provides interrupt status visibility to the host, prior to the interrupt mask being applied. Using this register, the host can view all potential sources of incoming interrupts. All of these sources, whether masked in or out, are latched in this register and must be cleared using the PKTE_INT_CLR register in order to capture a subsequent event. A 1 indicates that the associated interrupt is present.

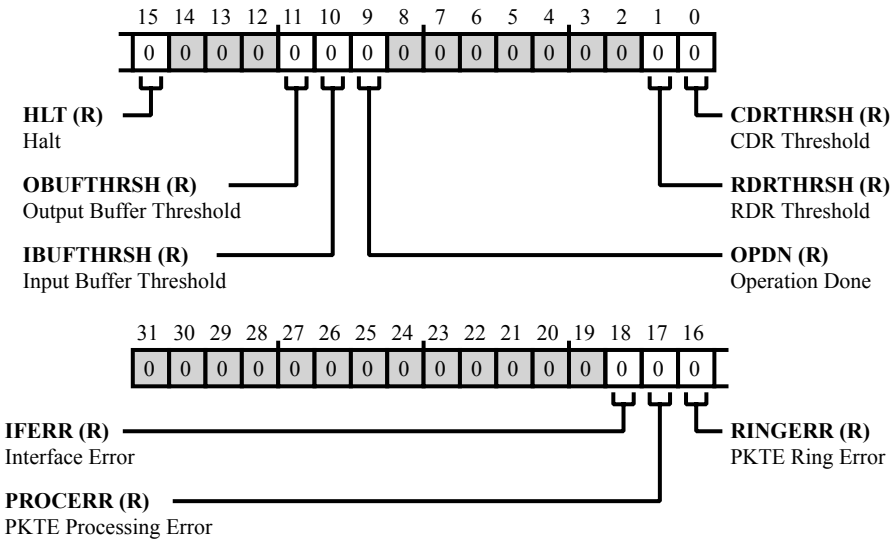


Figure 41-29: PKTE_IUMSK_STAT Register Diagram

Table 41-55: PKTE_IUMSK_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/NW)	IFERR	Interface Error. The PKTE_IUMSK_STAT.IFERR bit is set when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (R/NW)	PROCERR	PKTE Processing Error. The PKTE_IUMSK_STAT.PROCERR bit is set when an extended error occurred before, during or after processing the current packet in the packet engine.
16 (R/NW)	RINGERR	PKTE Ring Error. The PKTE_IUMSK_STAT.RINGERR bit is set on a CDR overflow or an RDR underflow.
15 (R/NW)	HLT	Halt. The PKTE_IUMSK_STAT.HLT bit is set when the packet engine is in the HALT state.

Table 41-55: PKTE_IUMSK_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11 (R/NW)	OBUFTHRSH	Output Buffer Threshold. The PKTE_IUMSK_STAT.OBUFTHRSH interrupt is triggered when the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.
10 (R/NW)	IBUFTHRSH	Input Buffer Threshold. The PKTE_IUMSK_STAT.IBUFTHRSH interrupt is triggered when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (R/NW)	OPDN	Operation Done.
1 (R/NW)	RDRTHRSH	RDR Threshold. The PKTE_IUMSK_STAT.RDRTHRSH bit is set when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (R/NW)	CDRTHRSH	CDR Threshold. The PKTE_IUMSK_STAT.CDRTHRSH bit is set when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Packet Engine Length Register

The PKTE_LEN register gives the length of the packet, the bypass data and a second set of ownership bits.

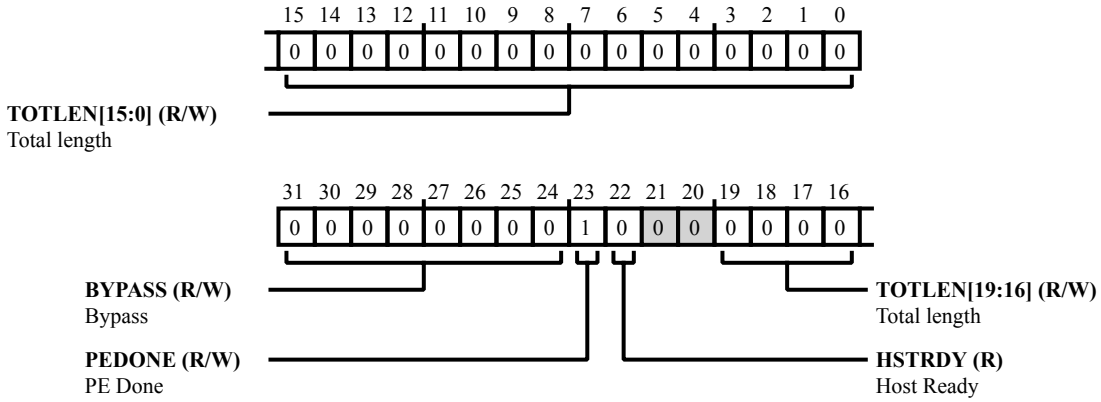


Figure 41-30: PKTE_LEN Register Diagram

Table 41-56: PKTE_LEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYPASS	Bypass. The PKTE_LEN.BYPASS bit field indicates the length of data in words that must bypass the packet engine and are directly copied from the source buffer to the destination buffer. The packet engine does not process this data. Valid bypass offsets range from 0 (0x00) to 255 (0xFF) words. For SRTP operations, this field specifies the offset in words between the hash and encrypt/decrypt data.
23 (R/W)	PEDONE	PE Done. The PKTE_LEN.PEDONE bit is a mirrored bit from the PKTE_CTL_STAT.PERDY bit. The bit is repeated here to guarantee ownership consistency between the first and last word. When the packet engine fetches a descriptor, these bits must match or the descriptor is discarded and fetched again.
22 (R/NW)	HSTRDY	Host Ready. The PKTE_LEN.HSTRDY bit is a mirrored bit of the PKTE_CTL_STAT.HOSTRDY bit. The bit is repeated here to guarantee ownership consistency between the first and last word. It should also be set along with the PKTE_CTL_STAT.HOSTRDY bit when the command descriptor is finished being populated. When the packet engine fetches a descriptor, these bits must match or the descriptor is discarded and fetched again.

Table 41-56: PKTE_LEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
19:0 (R/W)	TOTLEN	Total length. Command Descriptor: The PKTE_LEN.TOTLEN bit field indicates the total length (in bytes) of all data to be passed to the packet engines input buffer for an operation. Exceptions are the PRNG init and PRNG generate operations. The PRNG init operation does not require any input data; this field must be zero. For the PRNG generate operation, this field indicates the number of pseudo-random bytes to be generated. Valid lengths range from 16 (0x00010) to 255*16 = 4080 (0x00FF0) bytes in multiples of 16 bytes. Valid lengths for the basic operation range from 1 (0x00001) to 1,048,575 (0xFFFFF) bytes. This is the length of the data to be encrypted or hashed and includes the bypass data and padding bytes. Valid lengths for IPsec ESP range from 1 (0x00001) to 65535 (0x0FFFF) bytes. This is the length of the IP payload. Valid lengths for SSL v3.0, TLS v1.x and DTLS range from 1 (0x00001) to 16383 (0x03FFF). This is the length of the payload. Valid lengths for SRTP range from 1 (0x00001) to 65535 (0x0FFFF). This is the length of the payload. Note: A length of zero bytes is illegal and will result in an error status code in the result descriptor. Result Descriptor: Upon completion of an operation, the PKTE_LEN.TOTLEN field indicates the result length of the result packet. Valid lengths range from 1 (0x001) to 1,048,575 (0xFFFFF) bytes. This includes the bypass data and padding bytes. Note: When an extended error (PKTE_CTL_STAT[18]=1) is reported in the result descriptor and no packet data is processed, this field returns zero.

Packet Engine Output Buffer Count Register

The PKTE_OUTBUF_CNT register provides the number of data bytes there are in the output buffer. The PKTE_OUTBUF_CNT register is used in direct host mode only.

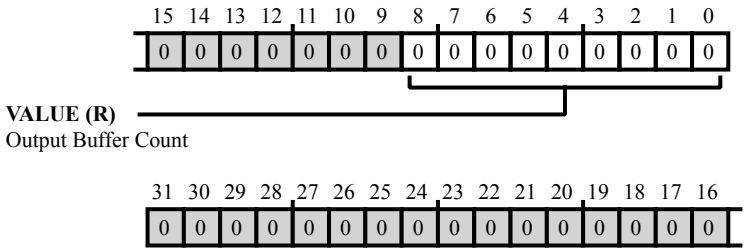


Figure 41-31: PKTE_OUTBUF_CNT Register Diagram

Table 41-57: PKTE_OUTBUF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/NW)	VALUE	Output Buffer Count. The PKTE_OUTBUF_CNT.VALUE bit field provides the number of bytes in the output buffer. The packet engine increments the counter by 4 when a 32-bit word is written to the output buffer.

Packet Engine Output Buffer Count Decrement Register

A host connected via the system completer bus can decrement the output buffer counter by writing a value between 4 and 256, in multiples of 4, to the lowest bits of this register. The PKTE_OUTBUF_DECR register is used in direct host mode only.

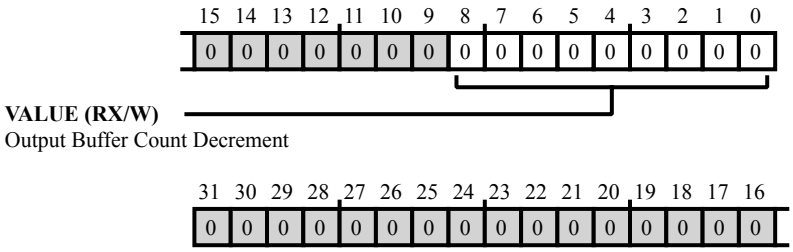


Figure 41-32: PKTE_OUTBUF_DECR Register Diagram

Table 41-58: PKTE_OUTBUF_DECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (RX/W)	VALUE	Output Buffer Count Decrement. The PKTE_OUTBUF_DECR.VALUE bit field is the value written is subtracted to the output buffer counter. Valid values range from 4 to 256, in multiples of 4.

Packet Engine Result Descriptor Ring Base Address

The PKTE_RDRBASE_ADDR register holds the result descriptor ring base address in host memory. It is only applicable in autonomous ring mode and target command mode with RDR enabled. Note that in target command mode, the CDR is not used, but the RDR must be configured when enabled so that the packet engine knows where to write the result descriptors.

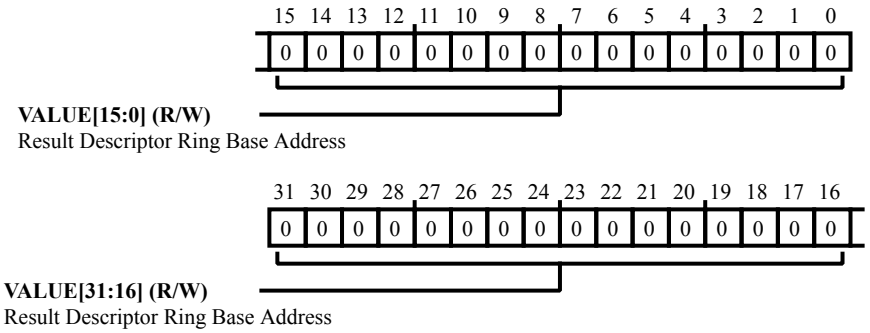


Figure 41-33: PKTE_RDRBASE_ADDR Register Diagram

Table 41-59: PKTE_RDRBASE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Result Descriptor Ring Base Address. The PKTE_RDRBASE_ADDR.VALUE bit field specifies the base location of the result descriptor ring in the host memory space.

Packet Engine Result Descriptor Count Registers

The PKTE_RDSC_CNT register holds the counter for the number of descriptors in the Result Descriptor Ring (RDR). It is incremented by the packet engine each time a valid result descriptor is written to the RDR.

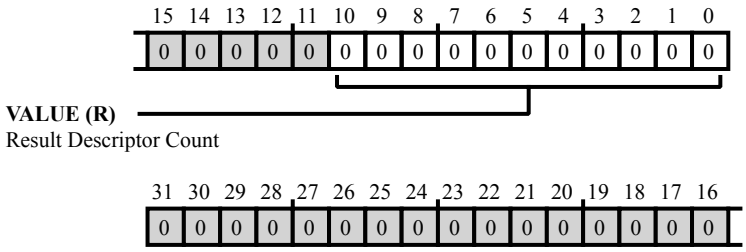


Figure 41-34: PKTE_RDSC_CNT Register Diagram

Table 41-60: PKTE_RDSC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	VALUE	Result Descriptor Count. The PKTE_RDSC_CNT.VALUE bit field provides the number of result descriptors in the result descriptor ring. The packet engine increments the counter when a valid result descriptor is written to the RDR.

Packet Engine Result Descriptor Count Decrement Registers

The PKTE_RDSC_DECR register is accessible by the host connected through the system completer bus can decrement the result descriptor counter by writing a value between 1 and 255 to the lowest byte of this register.

With an RDR enabled, this is the number of result descriptors that have been read by the host. With an RDR disabled, this indicates that the host has read one valid result descriptor.

In autonomous ring mode or target command mode with the RDR enabled, the host must process 1 to 255 result descriptors from the RDR and then write this register with the number of result descriptors that have been processed by the host.

In direct host mode or target command mode with the RDR disabled, the host must read one result descriptor from the internal descriptor registers and then write this register with the value 1, to indicate that one valid descriptor is read. An RDR threshold interrupt is activated when the result descriptor counter exceeds the threshold value set in the PKTE_RING_THRESH register. This interrupt can be used to wake up a process that stalled on an empty RDR.

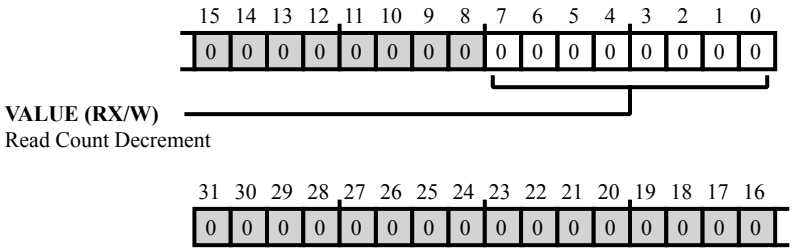


Figure 41-35: PKTE_RDSC_DECR Register Diagram

Table 41-61: PKTE_RDSC_DECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (RX/W)	VALUE	Read Count Decrement. The value written to the PKTE_RDSC_DECR.VALUE bit field is subtracted from the result descriptor counter. The counter is protected against underflow (See the PKTE_RING_STAT register). Note that bits [10:8] should be written with zeros.

Packet Engine Ring Configuration

The PKTE_RING_CFG register configures the size (in number of descriptor ring entries minus 1) for both the command descriptor ring and result descriptor ring in host memory. This register is only applicable for autonomous ring mode and target command mode with RDR enabled.

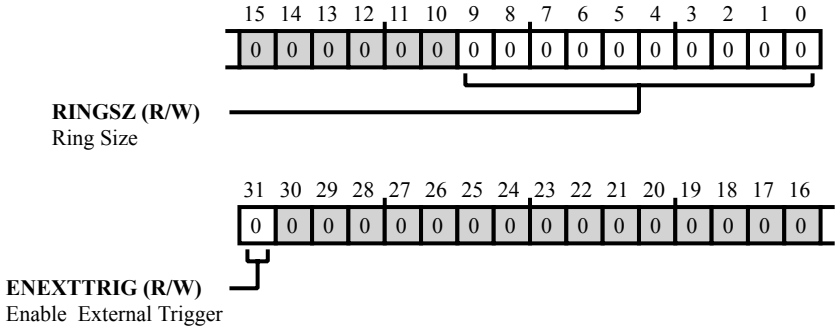


Figure 41-36: PKTE_RING_CFG Register Diagram

Table 41-62: PKTE_RING_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	ENEXTTRIG	Enable External Trigger. The PKTE_RING_CFG.ENEXTTRIG signal enables the increment of the PKTE_CDSC_CNT register through the external input pin ext_cd_cnt_incr and enables the decrement of the PKTE_RDSC_CNT fields through the external input pin ext_rd_cnt_decr.
9:0 (R/W)	RINGSZ	Ring Size. The PKTE_RING_CFG.RINGSZ bit field specifies the size of the command ring in number of descriptors, minus 1. Valid sizes range from 1 (for 2 descriptors) to 1023 (for 1024 descriptors). The accompanying result ring will have the same size.

Packet Engine Ring Pointer Status

The PKTE_RING_PTR register holds the pointers to the current entry of the Command Descriptor Ring (CDR) and Result Descriptor Ring (RDR).

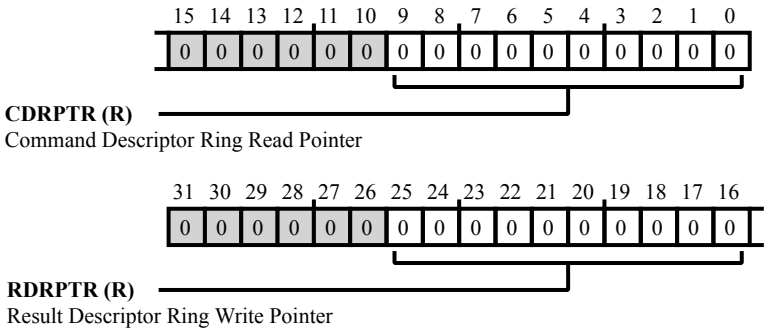


Figure 41-37: PKTE_RING_PTR Register Diagram

Table 41-63: PKTE_RING_PTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:16 (R/NW)	RDRPTR	Result Descriptor Ring Write Pointer. The PKTE_RING_PTR.RDRPTR bit field indicates the entry number in the RDR that will be written next by the packet engine. The PKTE_RING_PTR.RDRPTR bit field is reset to zero after starting up and updated after every result descriptor write DMA operation. Pointers wrap around; the maximum value this field can have equals the contents of the ring size (PKTE_RING_CFG.RINGSZ) bit field.
9:0 (R/NW)	CDRPTR	Command Descriptor Ring Read Pointer. The PKTE_RING_PTR.CDRPTR bit field indicates the entry number in the CDR that will be read next by the packet engine. The PKTE_RING_PTR.CDRPTR bit field is reset to zero after starting up and updated after every command descriptor read DMA operation. Pointers wrap around; the maximum value this field can have equals the contents of the ring size (PKTE_RING_CFG.RINGSZ) field.

Packet Engine Ring Status

The PKTE_RING_STAT register gives indication of either a Command Descriptor Ring (CDR) overflow or a Result Descriptor Ring (RDR) underflow. A ring error (ringerr) interrupt in the interrupt controller is activated on a command descriptor ring overflow or a result descriptor ring underflow. This type of error can occur when the host and the packet engine get out-of-sync. The host can read this register to retrieve information on which ring is corrupted. The corrupted ring must be reset and reinitialized. See the PKTE_CFG.RSTRING bit.

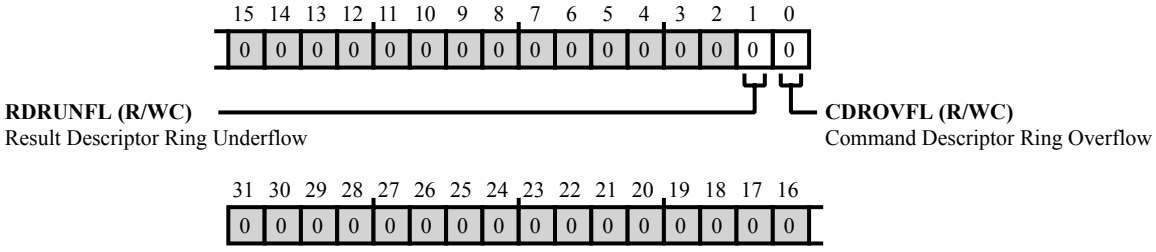


Figure 41-38: PKTE_RING_STAT Register Diagram

Table 41-64: PKTE_RING_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/WC)	RDRUNFL	Result Descriptor Ring Underflow. The PKTE_RING_STAT.RDRUNFL bit is set when the command descriptor count (PKTE_RDSC_CNT) register is decremented below zero. This bit is reset with a write of any value.
0 (R/WC)	CDROVFL	Command Descriptor Ring Overflow. The PKTE_RING_STAT.CDROVFL bit is set when the command descriptor count (PKTE_CDSC_CNT) register is incremented above the ring size (PKTE_RING_CFG.RINGSZ) bits. This bit is reset with a write of any value.

Packet Engine Ring Threshold Registers

To reduce the amount of packet engine result interrupts, the PKTE_RING_THRESH register contains threshold and time-out values.

The CDR threshold (cdrthrsh) interrupt indicates that the command descriptor counter is less than or equal to the CDR threshold (cdrthrsh) value set in this register. This interrupt can be used to wake up a process that stalled on a full CDR.

The RDR threshold (rdrthrsh) interrupt indicates that the result descriptor counter exceeds the result descriptor threshold set here, or that the result descriptor counter is non-zero for a time longer than the result descriptor time-out setting. The RDR result interrupt remains active until the result descriptor counter is decremented below the RDR threshold (rdrthrsh) value. In case the interrupt is the result of a time-out and the result descriptor counter is below the threshold value, the result descriptor counter must be decremented once before the interrupt can be cleared in the interrupt controller.

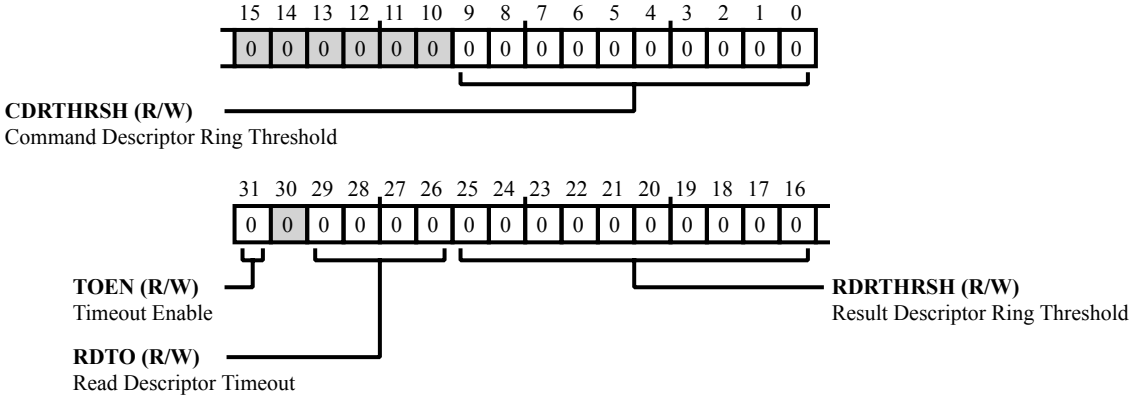


Figure 41-39: PKTE_RING_THRESH Register Diagram

Table 41-65: PKTE_RING_THRESH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	TOEN	Timeout Enable. A 1 in the PKTE_RING_THRESH.TOEN bit indicates the result descriptor timeout counter is enabled. This bit can be used to de-activate the timeout counter to save power.

Table 41-65: PKTE_RING_THRESH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
29:26 (R/W)	RDTO	Read Descriptor Timeout. The timeout enable (PKTE_RING_THRESH.TOEN) bit in this register must be set to activate this PKTE_RING_THRESH.RDTO result descriptor timeout counter. The rdrthrsh interrupt activates when the RD counter for the RDR is non-zero for more than $2^{(N+10)}$ internal system clock cycles, where 'N' is the value set in this field. Valid settings range from 0 to 15. The minimum time-out value for N=0 is 1024 clock cycles and the maximum time-out value for N=15 is 33554432 clock cycles. At 100 MHz, this is 5.12 us for N=0 and ~335.55 ms for N=15. Note: The time-out delay may not be exact - expect a variation on the order of 1024 system clock cycles (just more than one microsecond at 100 MHz system clock frequency).
25:16 (R/W)	RDRTHRSH	Result Descriptor Ring Threshold. The rdrthrsh interrupt activates when the RD counter for the RDR exceeds the value set in the PKTE_RING_THRESH.RDRTHRSH field. Valid settings range from 0 to 1023.
9:0 (R/W)	CDRTHRSH	Command Descriptor Ring Threshold. The cdrthrsh interrupt activates when CD counter for the CDR is below or equal the value set in the PKTE_RING_THRESH.CDRTHRSH field. Valid settings range from 0 to 1023.

Packet Engine SA Address

The PKTE_SA_ADDR register holds the start address of the SA record.

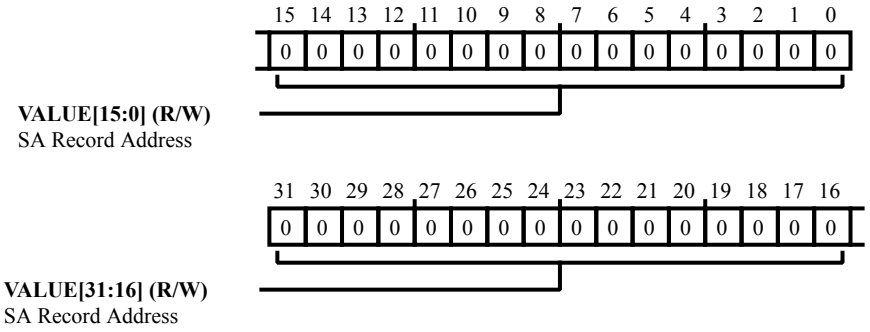


Figure 41-40: PKTE_SA_ADDR Register Diagram

Table 41-66: PKTE_SA_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Record Address. The PKTE_SA_ADDR.VALUE bit field holds the start address of the SA record.

ARC4 i and j Pointer Register

When starting a new ARC4 operation the PKTE_SA_ARC4IJPTR register contains the initialization value, which is zeros. After processing the ARC4 algorithm it contains the latest status of the ARC4_IJ_PNTR.

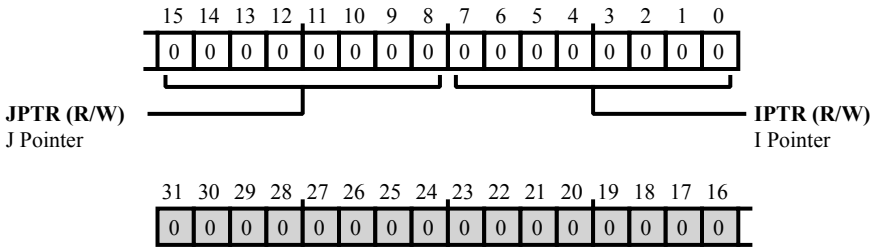


Figure 41-41: PKTE_SA_ARC4IJPTR Register Diagram

Table 41-67: PKTE_SA_ARC4IJPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:8 (R/W)	JPTR	J Pointer. The PKTE_SA_ARC4IJPTR.JPTR bit field contains the j pointer into s-box array for swapping bytes with i pointer.
7:0 (R/W)	IPTR	I Pointer. The PKTE_SA_ARC4IJPTR.IPTR bit field contains the i pointer into s-box array for swapping bytes with j pointer.

SA Command 0

The two SA command registers, PKTE_SA_CMD0 and PKTE_SA_CMD1, are used to control the cryptographic operation of the packet engine. The PKTE_SA_CMD0 register contains the major control bits to define an operation while the PKTE_SA_CMD1 register contains the minor control bits. In direct host mode, this is a write-only register.

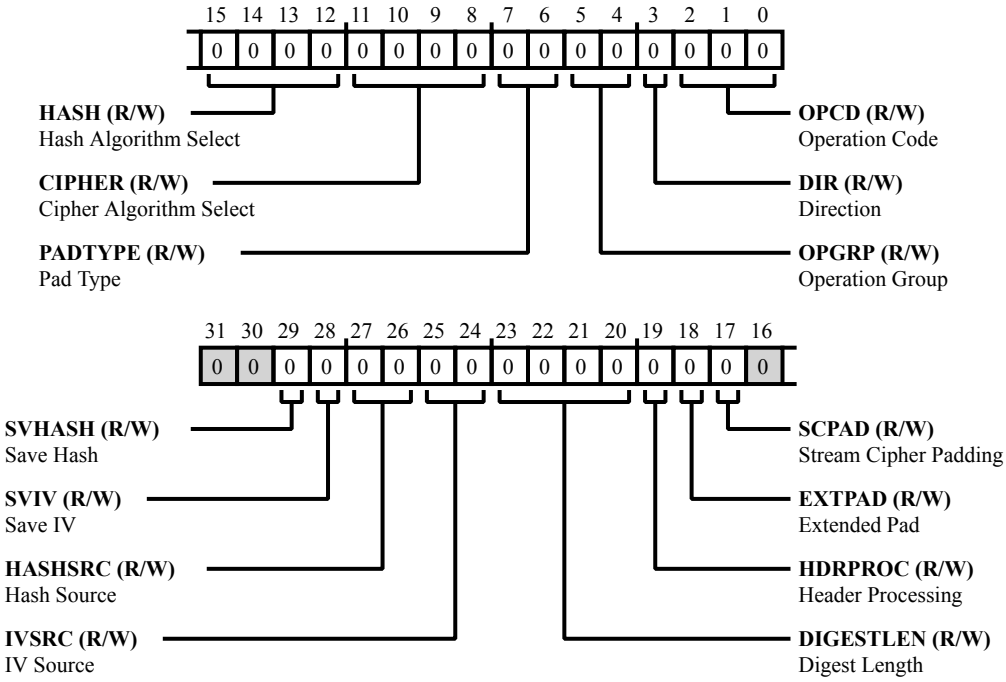


Figure 41-42: PKTE_SA_CMD0 Register Diagram

Table 41-68: PKTE_SA_CMD0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	SVHASH	Save Hash. The PKTE_SA_CMD0.SVHASH bit indicates that the Hash State is saved to the STATE_BYTE_CNT_X and STATE_IDIGEST_X fields in the SA record in memory after completion of a crypto operation.	
		0	Hash state is not saved
		1	Hash state is saved

Table 41-68: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
28 (R/W)	SVIV	Save IV. The `PKTE_SA_CMD0.SVIV` bit field indicates that for DES or the AES the Initialization Vector (IV) is saved to the STATE_IV_X fields in the state record , or the ARC4 state is saved to the ARC4 state record, after completion of the crypto operation.	
		0	ARC4 State is not saved.
		1	ARC4 State is not saved.
27:26 (R/W)	HASHSRC	Hash Source. The `PKTE_SA_CMD0.HASHSRC` bit field selects the source of the hash digest used by the algorithm.	
		0	From SA. Digest only hash byte count is forced to 0x40.
		1	Reserved
		2	From State. Read saved inner hash digest and saved hash byte count.
		3	No Load. Use the hash algorithm defined constants for the initial hash. Hash byte count is 0x00.
25:24 (R/W)	IVSRC	IV Source. The `PKTE_SA_CMD0.IVSRC` bit field selects the source of the initialization vector used by the crypto algorithm.	
		0	No load. Use previous result IV, not applicable for inbound data. This option should never be used for operations with DES-CBC or AES-CBC, (see RFC3602) or any AES counter modes
		1	From input buffer. The IV is provided as part of the input data stream.
		2	From State. Read STATE_IV_X, from the SA structure. Refer to inner hash digest register structure. Useful for resume operations.
		3	From internal PRNG. Not applicable for inbound operations.
23:20 (R/W)	DIGESTLEN	Digest Length. The `PKTE_SA_CMD0.DIGESTLEN` bit field defines the length of the hash digest in words as put in the output buffer.	
		0	3 Words (96-bit output)
		1	1 Word
		2	2 Words

Table 41-68: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	3 Words (IPsec)
		4	4 Words (MD5 and AES-based hash)
		5	5 Words (SHA-1)
		6	6 Words
		7	7 Words (SHA-224)
		8	8 Words (SHA-256)
		9	Reserved
		10	10 bytes (SRTP and TLS)
		11-15	Reserved
19 (R/W)	HDRPROC	Header Processing. The PKTE_SA_CMD0.HDRPROC bit enables header processing for protocol operations. There is no header-processing support for basic SSL, basic TLS and SRTP protocol operations as defined in the protocol group (see the Crypto and Hash Algorithms section). This bit must be zero for these operations; however, the protocol header must be supplied to the packet engine since it is part of the hash calculation. Refer to the protocol specifications for more information about header-processing support for a protocol.	
		0	No header processing
		1	Header processing; insert the protocol header for outbound operations, verify the protocol header for inbound operations.
18 (R/W)	EXTPAD	Extended Pad. The PKTE_SA_CMD0.EXTPAD bit extends the number of padding types. Used in combination with PKTE_SA_CMD0.PADTYPE.	
17 (R/W)	SCPAD	Stream Cipher Padding. The PKTE_SA_CMD0.SCPAD bit enables padding for stream ciphers algorithms.	
15:12 (R/W)	HASH	Hash Algorithm Select. The PKTE_SA_CMD0.HASH bit field selects the hash algorithm.	
		0	MD5
		1	SHA-1
		2	SHA-224
		3	SHA-256
		4-14	Reserved
		15	Null

Table 41-68: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11:8 (R/W)	CIPHER	Cipher Algorithm Select. The `PKTE_SA_CMD0.CIPHER` bit field selects the cipher algorithm to be used for encryption and decryption. Note: Each type of protocol operation supports different sets of crypto algorithms. Refer to the Crypto and Hash Algorithms general processing section for details of the supported algorithms.	
		0	DES
		1	Triple-DES
		2	ARC4
		3	AES
		4-14	Reserved
		15	Null
7:6 (R/W)	PADTYPE	Pad Type. The `PKTE_SA_CMD0.PADTYPE` bit field indicates the type of crypto that must be generated for outbound packets or checked for inbound packets.	
		0	Select IPSec operation (if Bit 18=0); Reserved (if Bit 18=1)
		1	PKCS#7 (if Bit 18=0); Select TLS/DTLS Pad, required for TLS/DTLS operation (if Bit 18=1)
		2	Constant pad (if Bit 18=0); Select Constant SSL Pad, required for SSL operation (if Bit 18=1)
		3	Zero pad (if Bit 18=0), Reserved (if Bit 18=1)
5:4 (R/W)	OPGRP	Operation Group. The `PKTE_SA_CMD0.OPGRP` bit field defines the operation groups. Refer to the Basic Operations and Decoding section for more information.	
		0	Basic operation group
		1	Protocol operation group
		2	Extended protocol operations group
		3	Reserved
3 (R/W)	DIR	Direction. The `PKTE_SA_CMD0.DIR` bit field selects the direction of operation.	
		0	Outbound operations
		1	Inbound operations

Table 41-68: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	OPCD	Operation Code. The `PKTE_SA_CMD0.OPCD` bit field selects the operation within the operation group.

SA Command 1

The PKTE_SA_CMD1 register contains the minor control bits that define an operation. In direct host mode, this is a write-only register.

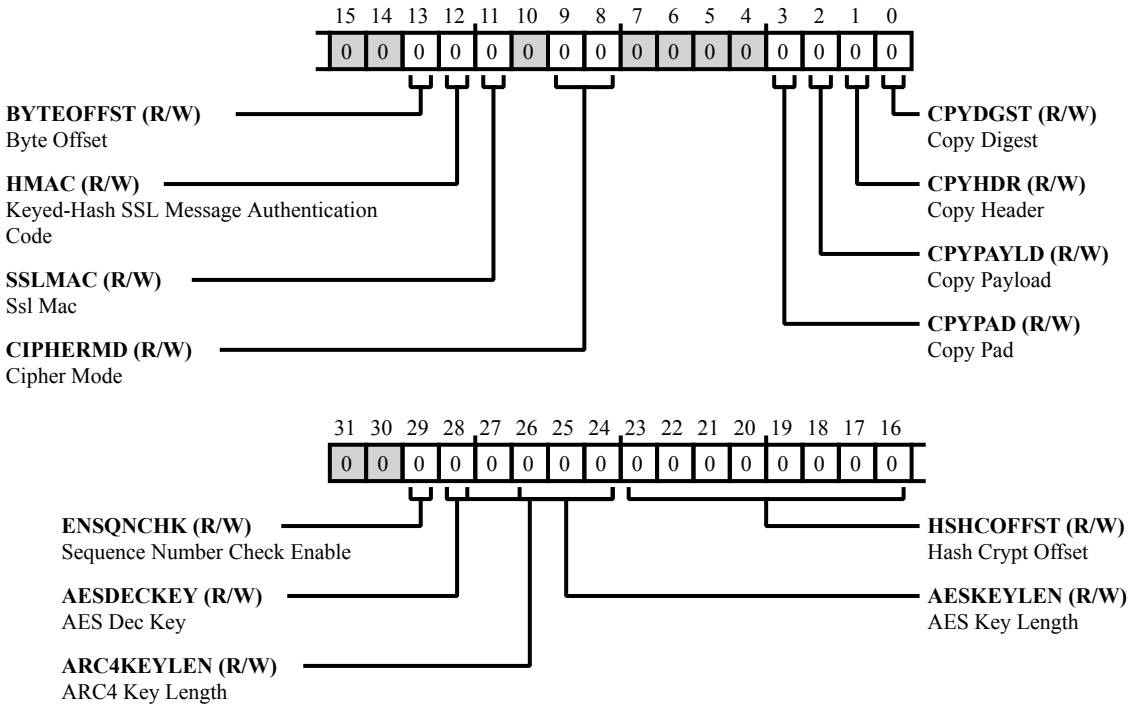


Figure 41-43: PKTE_SA_CMD1 Register Diagram

Table 41-69: PKTE_SA_CMD1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	ENSQNCHK	Sequence Number Check Enable. The PKTE_SA_CMD1.ENSQNCHK bit defines that the key in the SA key field is an AES encrypt key or an AES decrypt key.	
		0	Disable sequence number check
		1	Enable sequence number check
28 (R/W)	AESDECKEY	AES Dec Key. If the PKTE_SA_CMD1.AESDECKEY bit is set, the key in loaded in the PKTE_SA_KEY[n] registers are expected to be the key from the last round from key expansion. If not set, the key loaded in the PKTE_SA_KEY[n] registers are expected to be the same key used during the encryption process.	
		0	AES key is an encrypt key.
		1	AES key is a decrypt key.

Table 41-69: PKTE_SA_CMD1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
26:24 (R/W)	AESKEYLEN	AES Key Length. The PKTE_SA_CMD1.AESKEYLEN bit field select the size of the key used for the AES algorithm in increments of 64 bits.	
		0-1	Reserved
		2	128 Bits
		3	192 Bits
		4	256 Bits
		5-7	Reserved
28:24 (R/W)	ARC4KEYLEN	ARC4 Key Length.	
23:16 (R/W)	HSHCOFFST	Hash Crypt Offset. For Basic Encrypt-Hash and Basic Hash-Decrypt operations, the PKTE_SA_CMD1.HSHCOFFST bit field specifies the offset between the hash data and the encrypt/decrypt data. The data to be hashed is assumed to come first, with an offset to the beginning of encrypt/decrypt data. When PKTE_SA_CMD1.BYTEOFFST, bit 13, is zero, then the offset is defined in 32-bit words. When an initialization vector is loaded through the input buffer, valid values range from IV size to 255. In all other cases, valid values range from 0 to 255. When PKTE_SA_CMD1.BYTEOFFST, bit 13, is one, then the offset is defined in 8-bit bytes. When an initialization vector is loaded through the input buffer, valid values range from IV size to 255. In all other cases, valid values range from 4 to 255. (The IV size is two words for DES, Triple-DES and AES-CTR and four words for AES-CBC and AES-ICM operations). Other operations do not use these bits (a default value is applied by the packet engine).	
13 (R/W)	BYTEOFFST	Byte Offset. The PKTE_SA_CMD1.BYTEOFFST bit defines how the PKTE_SA_CMD1.HSHCOFFST, bits of this register are used.	
		0	HASH_CRYPT_OFFSET is defined in 32-bit words
		1	HASH_CRYPT_OFFSET is defined in 8-bit bytes
12 (R/W)	HMAC	Keyed-Hash SSL Message Authentication Code. For basic operations that include hashing, the PKTE_SA_CMD1.HMAC bit enables the HMAC processing, which calls for an extra outer hash operation.	
		0	Standard Hash
		1	HMAC Processing

Table 41-69: PKTE_SA_CMD1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
11 (R/W)	SSLMAC	Ssl Mac.	
		0	Standard Hash
		1	SSL-MAC processing
9:8 (R/W)	CIPHERMD	Cipher Mode. The `PKTE_SA_CMD1.CIPHERMD` bit field selects the crypto mode to be used for the cipher algorithm.	
		0	Electronic Code Book (ECB) used for DES and AES
		1	Cipher Block Chaining (CBC) used for DES and AES
		2	AES Counter Mode (CTR) for IPsec using a 32-bit counter
		3	AES Integer Counter Mode (ICM) for SRTP using a 16-bit counter. Note: This is implemented as a 32-bit counter, the Host must check for an overflow from the value 0xFF to zero and then refresh the IV.
3 (R/W)	CPYPAD	Copy Pad. The `PKTE_SA_CMD1.CPYPAD` bit indicates that the padding data for an inbound operation is copied to the output buffer and saved in memory.	
		0	Do not copy the padding to output
		1	Copy padding to output
2 (R/W)	CPYPAYLD	Copy Payload. The `PKTE_SA_CMD1.CPYPAYLD` bit indicates that the payload data is copied to the output buffer and saved in memory.	
		0	Do not copy the payload to output
		1	Copy payload to output
1 (R/W)	CPYHDR	Copy Header. The `PKTE_SA_CMD1.CPYHDR` bit indicates that the protocol header is copied to the output buffer and saved in memory. For Basic Encrypt-Hash and Basic Hash-Decrypt operations, the header is defined as the Hash/Crypt Offset data (authenticated only).	
		0	Do not copy the header to output
		1	Copy header to output

Table 41-69: PKTE_SA_CMD1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	CPYDGST	Copy Digest. The `PKTE_SA_CMD1.CPYDGST` bit copies the hash result is to the output buffer and saves in memory. The length of the hash result is defined by the `PKTE_SA_CMD0.DIGESTLEN` field.	
		0	Do not copy hash result to output
		1	Copy hash result to output, when the command descriptor `PKTE_CTL_STAT.HASHFINAL` bit is set.

SA Inner Hash Digest Registers

The PKTE_SA_IDIGEST[n] registers are a set of eight 32-bit read/write registers.

For MD5, SHA-1, SHA-224 and SHA-256, these read/write registers are used to enter a start hash state, and to read the interim or final hash digest.

For IPsec, TLS and DTLS operations that make use of MD5, SHA-1, SHA-224 or SHA-256 with basic hash or HMAC authentication with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the pre-computed inner hash digest. This is the hash of the hash-key padded with 0x36 hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-MD5 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the inner hash pre-compute; this is the hash of the MAC_WRITE_SECRET padded with 0x36 hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-SHA-1 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the MAC_WRITE_SECRET. Note that it is not possible to calculate a hash pre-compute for SHA-1 in combination with SSL-MAC (specification flaw). The packet engine appends the hash-key pad (0x36 hex) and sets the starting hash byte count automatically to 60 decimal / 0x3C hex (to indicate that 60 bytes have already been prepared for the hash).

The reset value for these registers is zero.

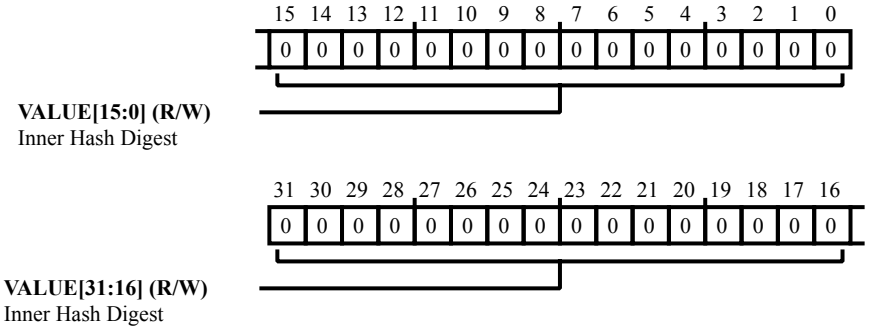


Figure 41-44: PKTE_SA_IDIGEST[n] Register Diagram

Table 41-70: PKTE_SA_IDIGEST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Inner Hash Digest.

SA Key Registers

These are the PKTE_SA_KEY[n] registers for DES, Triple-DES, ARC4 and AES: A set of eight 32-bit write only registers. The reset value of these registers is zero.

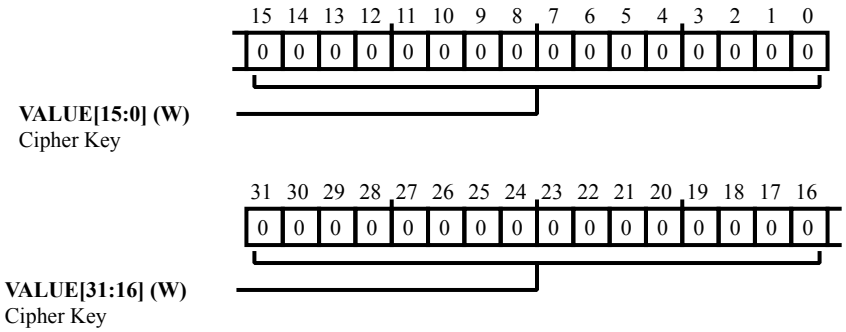


Figure 41-45: PKTE_SA_KEY[n] Register Diagram

Table 41-71: PKTE_SA_KEY[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Cipher Key.

SA Initialization Vector Register

The PKTE_SA_NONCE register is used for operations that make use of the IV value loaded from the SA record. This register is used both to enter a starting IV state, as well as for reading the interim or final IV. For IPsec outbound operations, it is recommended that the automatic IV insertion mode be used, this register is not needed. For IPsec inbound operations, the IV is extracted from the header of the packet.

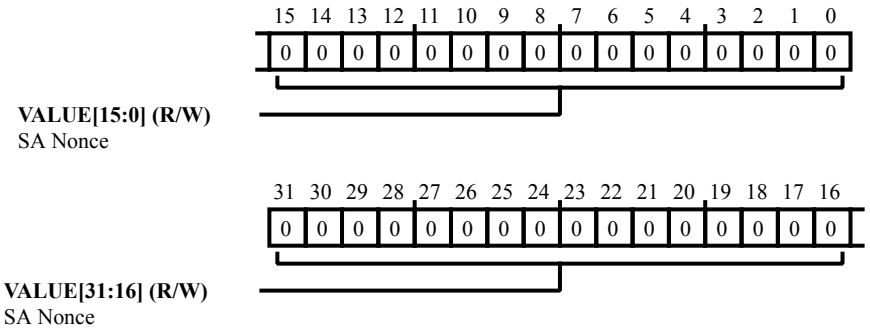


Figure 41-46: PKTE_SA_NONCE Register Diagram

Table 41-72: PKTE_SA_NONCE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Nonce.

SA Outer Hash Digest Registers

The PKTE_SA_ODIGEST[n] registers are a set of five eight 32-bit write-only registers.

For write operations, these registers contain the pre-computed outer hash digest for IPsec operations with basic HMAC operations with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA).

For MD5, SHA-1, SHA-224 and SHA-256, these read/write registers hold a start hash state, or the interim outer hash digest. They are only used for HMAC processing.

For IPsec, SSL, TLS, DTLS and SRTP operations that make use of MD5, SHA-1, SHA-224 or SHA-256 with HMAC authentication with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the pre-computed outer hash digest. This is the hash of the hash-key padded with 0x5C hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-MD5 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the outer hash pre-compute; this is the hash of the MAC_WRITE_SECRET padded with 0x5C hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-SHA-1 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the MAC_WRITE_SECRET. Note that it is not possible to calculate a hash pre-compute for SHA-1 in combination with SSL-MAC (specification flaw). The packet engine appends the required hash-key pad (0x5C hex) and sets the starting hash byte count automatically to 60 decimal / 0x3C hex (to indicate that 60 bytes have already been prepared for the hash).

The reset value for these registers is zero.

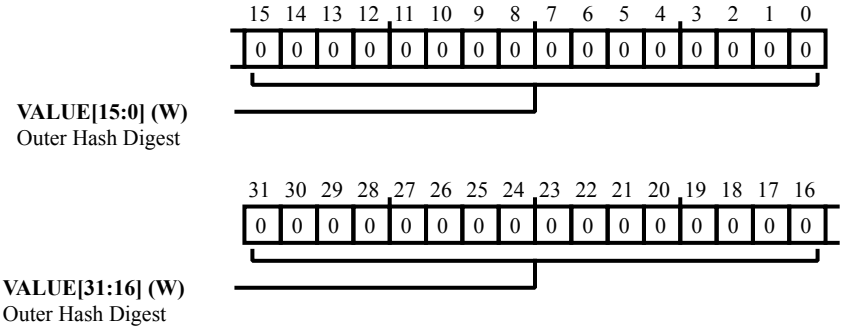


Figure 41-47: PKTE_SA_ODIGEST[n] Register Diagram

Table 41-73: PKTE_SA_ODIGEST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Outer Hash Digest.

SA Ready Indicator

In direct host mode, a write to the PKTE_SA_RDY register triggers the packet engine to start processing using the command descriptor, SA record and state record in the packet engine registers. This register MUST be written for all direct host mode packet operations. It is intended that this register is written in sequence; as the entire SA record is written.

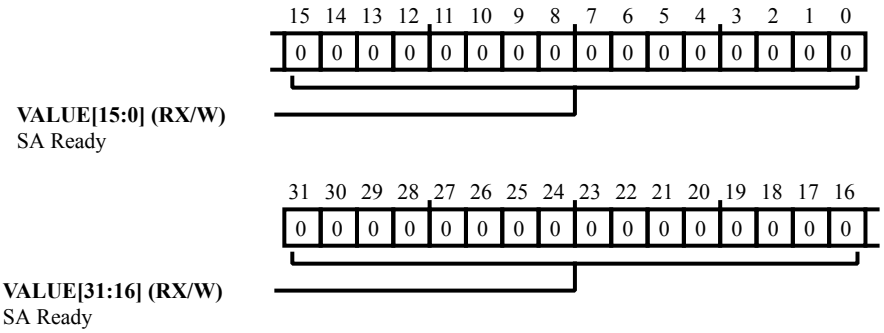


Figure 41-48: PKTE_SA_RDY Register Diagram

Table 41-74: PKTE_SA_RDY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	SA Ready.

SA Sequence Number Register

The PKTE_SA_SEQNUM[n] registers are a set of two read/write registers and are used for IPsec ESP, SSL, TLS, DTLS operations to specify the anti-replay sequence number value that is to be placed in the ESP header (outbound), or to be checked against for inbound packets. The packet engine manages this counter value for both inbound and outbound operations.

Outbound: The host writes the counter value stored in the SA record to this register to start an IPsec, SSL, TLS, DTLS operation. The packet engine automatically increments the count if header processing is selected. Upon successful completion, the host reads back this value and writes it to the SA record.

Inbound: The host writes the counter value stored in the SA record to this register to start an IPsec or DTLS operation. The packet engine automatically performs the specified inbound processing (per RFC 4303) as it processes the packet. As a result, the expected count value may or may not be updated during processing. Upon successful completion, the host should read back this value and write it to the SA record.

Note: The description is only for the direct host mode. The sequence number for autonomous ring mode and target command mode are updated by the packet engine.

The reset value of this register is zero.

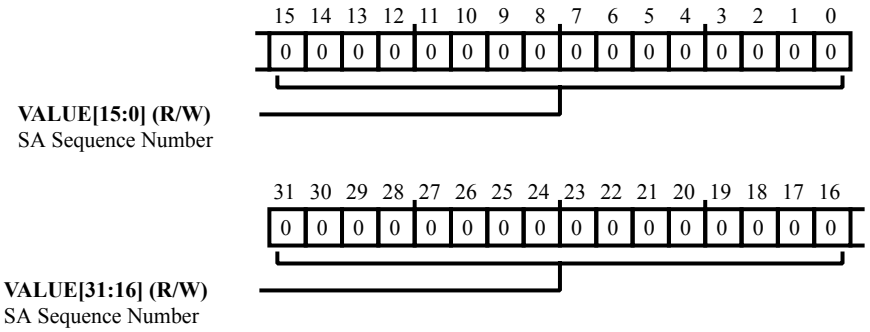


Figure 41-49: PKTE_SA_SEQNUM[n] Register Diagram

Table 41-75: PKTE_SA_SEQNUM[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Sequence Number.

SA Sequence Number Mask Registers

The PKTE_SA_SEQNUM_MSK[n] registers are a set of two read/write registers and are used for IPsec ESP and DTLS operations to specify the anti-replay sequence number mask value for inbound operations. The packet engine manages this counter value automatically.

Inbound: The host writes the counter value stored in the SA record into this register upon starting an IPsec, DTLS operation. The packet engine automatically performs the specified inbound processing (per RFC 4303) as it processes the packet. As a result, the new mask value may or may not be updated during processing. Upon successful completion, the host should read back this value and write it to the SA record.

Outbound: not used.

Note that the above description only applies to the direct host mode, for autonomous ring mode and target command mode the packet engine extracts the sequence number mask from the SA record.

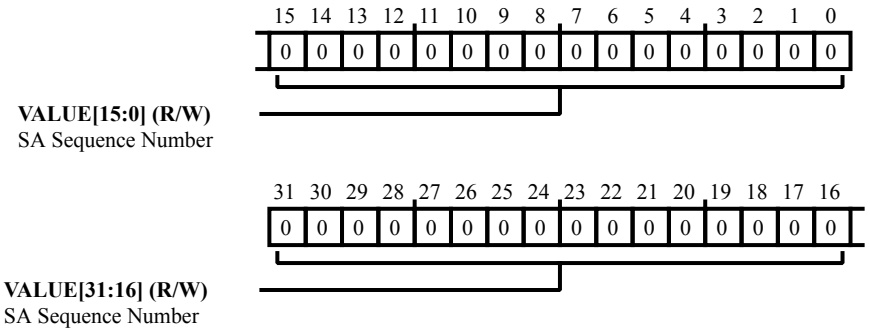


Figure 41-50: PKTE_SA_SEQNUM_MSK[n] Register Diagram

Table 41-76: PKTE_SA_SEQNUM_MSK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Sequence Number.

SA SPI Register

For IPsec operations, the PKTE_SA_SPI register is written with the SPI (Security Parameters Index) associated with the inbound or outbound flow.

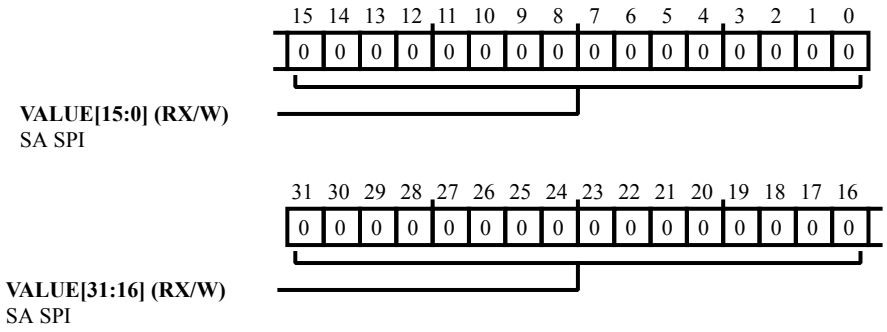


Figure 41-51: PKTE_SA_SPI Register Diagram

Table 41-77: PKTE_SA_SPI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	SA SPI. The PKTE_SA_SPI.VALUE bit field is used for IPsec ESP operations to specify the Security. Parameters Index (SPI) value that is to be placed in the ESP header. There is no need to read back this value at the end of an operation, since the Packet Engine does not change it. For SSL, TLS and DTLS this register stores the 8-bit TYPE field in bits [23:16], and the 16-bit Version field in bits [15:0] that are part of the protocol header.

Packet Engine Source Address

The PKTE_SRC_ADDR register holds the starting (byte) address for the packet to be processed.

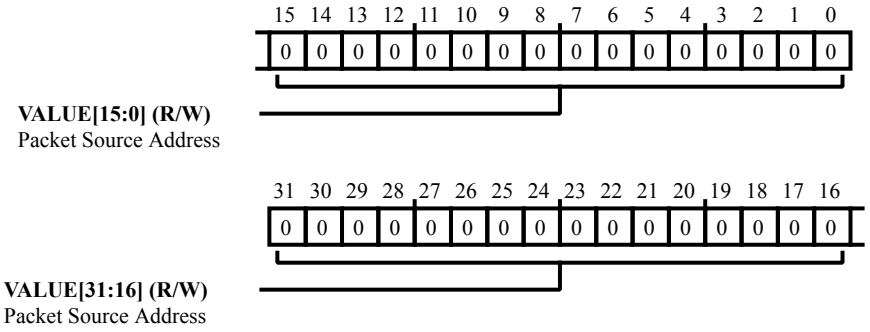


Figure 41-52: PKTE_SRC_ADDR Register Diagram

Table 41-78: PKTE_SRC_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Packet Source Address. The PKTE_SRC_ADDR.VALUE bit field holds the starting (byte) address for the packet to be processed.

Packet Engine Status Register

The PKTE_STAT register is used to provide the status of the packet engine. This register is useful in the direct host mode to determine when data must be written to or read from the packet engine, or for debugging the software when errors occur. This register can be ignored in autonomous ring mode and target command mode where the DMA engine controls the packet data I/O. This is a read-only register. A write to any of the bits has no effect.

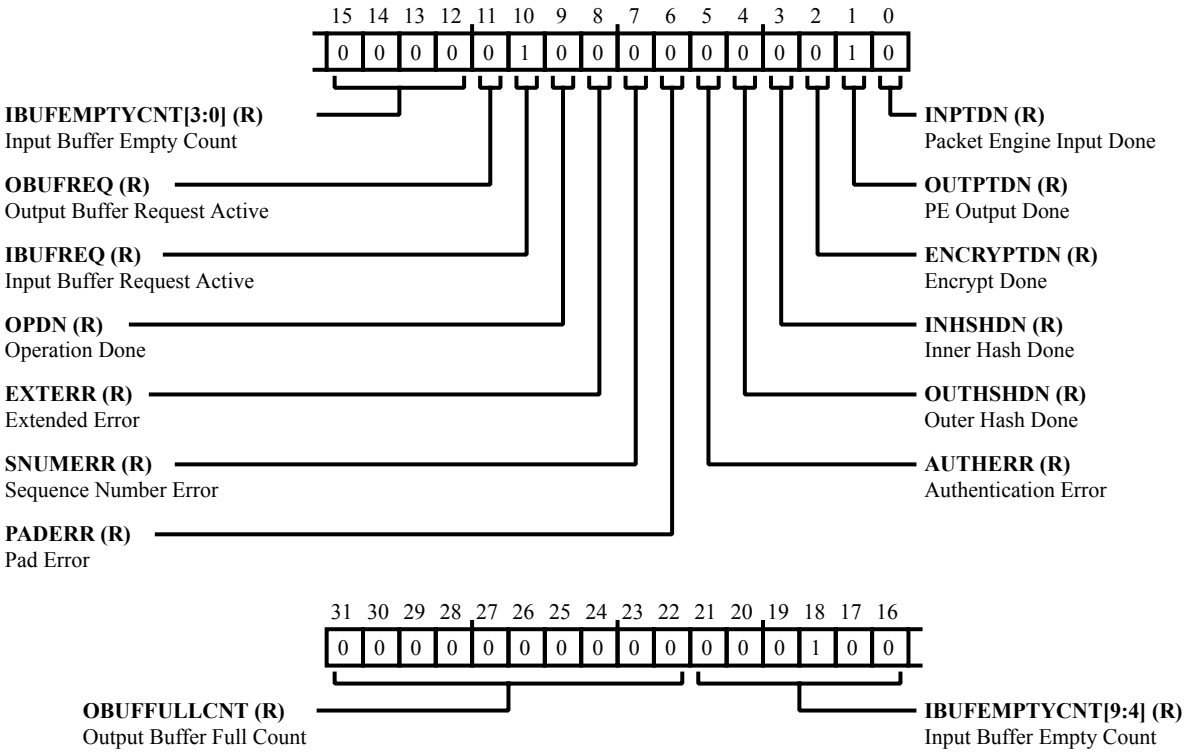


Figure 41-53: PKTE_STAT Register Diagram

Table 41-79: PKTE_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:22 (R/NW)	OBUFFULLCNT	Output Buffer Full Count. The PKTE_STAT.OBUFFULLCNT bit field indicates the number of 32-bit words that are available in the packet engine output buffer. It works in conjunction with bit 11 from this register. When bit 11 is asserted, to indicate a request for output, the word count matches the specified output buffer threshold setting in the PKTE_BUF_THRESH register. For the last output for a given packet, any value from 1 dword to the full output buffer threshold can be seen. Transfers must be a multiple of full dwords. The application must read the PKTE_LEN field in the result descriptor to determine the exact byte-length of the result.

Table 41-79: PKTE_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
21:12 (R/NW)	IBUFEMPTYCNT	Input Buffer Empty Count. The PKTE_STAT.IBUFEMPTYCNT bit field indicates the number of 32-bit empty spaces that are available in the packet engine input buffer. It works in conjunction with the PKTE_STAT.IBUFREQ bit (10) from this register. The value in the register is deducted from the specified packet length, so will never exceed the number of dwords that remain in the packet. For packets smaller than the buffer size, this register typically indicates that buffer space is available for the entire packet (rounded up to the nearest dword). For very large packets, these bits usually have a value around the maximum buffer size, indicating that the full input buffer is available.	
11 (R/NW)	OBUFREQ	Output Buffer Request Active. The PKTE_STAT.OBUFREQ bit indicates that the packet engine requests output data to be read from the output buffer.	
		0	No request for output data
		1	Request for output data
10 (R/NW)	IBUFREQ	Input Buffer Request Active. The PKTE_STAT.IBUFREQ bit indicates that the packet engine requests input data to be written to the input buffer.	
		0	No request for input data
		1	Request for input data
9 (R/NW)	OPDN	Operation Done. The PKTE_STAT.OPDN bit indicates that the packet engine has finished processing a packet when in direct host mode. This bit is zero in autonomous ring mode and target command mode.	
		0	Packet engine is idle
		1	Packet engine has finished processing a packet
8 (R/NW)	EXTERR	Extended Error. The PKTE_STAT.EXTERR bit indicates that an extended error occurred for this packet. For more information, refer to table Extended Error Codes - Status Encoding.	
		0	No extended error
		1	Extended error

Table 41-79: PKTE_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	SNUMERR	Sequence Number Error. For an inbound operation, the PKTE_STAT.SNUMERR bit indicates that there was a fault in the anti-replay sequence number. For an outbound operation, there was a sequence number overflow condition. For more information, refer to table Extended Error Codes - Status Encoding.	
		0	No sequence number error
		1	Input done, all bytes written to input buffer
6 (R/NW)	PADERR	Pad Error. The PKTE_STAT.PADERR bit indicates that an inbound crypto pad fault is detected. For more information about pad verification, refer to the Pad Verification and Consumption section.	
		0	No pad error
		1	Pad error
5 (R/NW)	AUTHERR	Authentication Error. The PKTE_STAT.AUTHERR bit indicates that an inbound ICV (for IPsec) or TAG (for SRTP) or MAC (for SSL/TLS/DTLS) fault is detected; the value carried within the packet did not match the value just computed.	
		0	No authentication error
		1	Authentication error
4 (R/NW)	OUTHSHDN	Outer Hash Done. The PKTE_STAT.OUTHSHDN bit indicates that the outer hash processing for this packet is finished.	
		0	Outer hash busy
		1	Outer hash done
3 (R/NW)	INHSHDN	Inner Hash Done. The PKTE_STAT.INHSHDN bit indicates that the inner hash processing for this packet is finished.	
		0	Inner hash busy
		1	Inner hash done
2 (R/NW)	ENCRYPTDN	Encrypt Done. The PKTE_STAT.ENCRYPTDN bit indicates that the encryption or decryption for this packet is finished.	
		0	Encryption or decryption busy
		1	Encryption or decryption done

Table 41-79: PKTE_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	OUTPTDN	PE Output Done. The PKTE_STAT.OUTPTDN bit indicates that the output data for the current packet is read from the packet engine output buffer.	
		0	Output not done, more output bytes available
		1	Output done, all bytes read from the output buffer
0 (R/NW)	INPTDN	Packet Engine Input Done. The PKTE_STAT.INPTDN bit indicates that the number of bytes specified in the command descriptor PKTE_LEN field is written into the packet engine input buffer.	
		0	Input not done, more input bytes expected
		1	Input done, all bytes written to input buffer

Packet Engine State Record Address

The PKTE_STATE_ADDR register holds the start address of the SA state record.

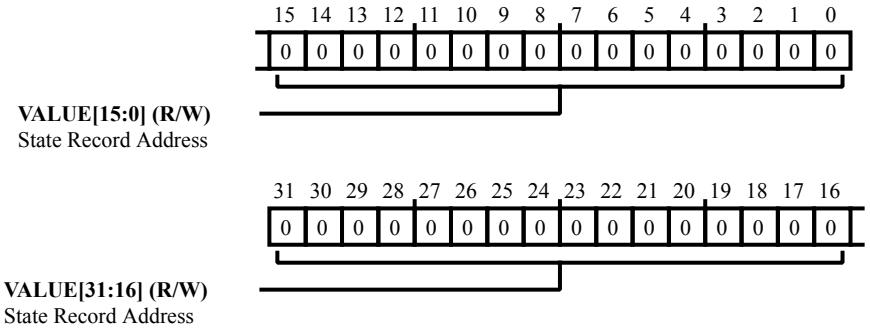


Figure 41-54: PKTE_STATE_ADDR Register Diagram

Table 41-80: PKTE_STATE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	State Record Address. The PKTE_STATE_ADDR.VALUE bit field holds the start address of the SA state record.

State Hash Byte Count Registers

The PKTE_STATE_BYTE_CNT[n] registers are used to enter a starting hash byte count, as well as to read the interim or final byte count.

For some hash operations, these registers are ignored and the byte count is internally set to 64 (0x40 hex) to indicate that the first 64 bytes (512 bits) hash block has been processed using a pre-computed hash state. These operations are:

All IPsec, SSL, TLS, DTLS and SRTP operations that use authentication; the "pre-computed" inner and outer hash digests are loaded from SA words 10 - 19.

Basic operations with PKTE_SA_CMD0.HASHSRC bits = 00 (from SA) specified. For Basic Hash with no HMAC, a pre-computed digest is loaded from SA words 10 - 14. For Basic Hash with HMAC, the inner and outer digests are loaded from SA words 10 - 19.

Note: Protocol operations can not be suspended in mid-packet and resumed later, therefore protocol operations do not use these registers.

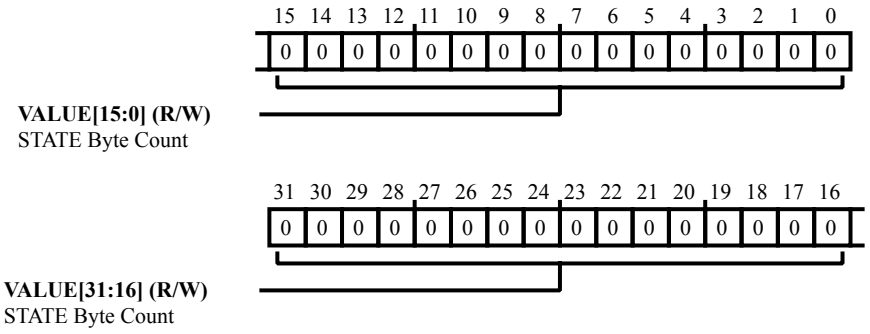


Figure 41-55: PKTE_STATE_BYTE_CNT[n] Register Diagram

Table 41-81: PKTE_STATE_BYTE_CNT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	STATE Byte Count.

State Inner Digest Registers

The PKTE_STATE_IDIGEST[n] registers consist of eight 32-bit registers. These read/write registers are used to read the interim or final hash digest. The PKTE_STATE_IDIGEST[n] registers are only used with basic operations involving basic hash, and are typically used for operations that must be suspended and resumed in the middle of a hash. The interim hash state can be read from these registers along with the hash byte-count from the previous register. Both can be restored when resuming the hash. The appropriate save hash state (PKTE_SA_CMD0.SVHASH=1) and load hash from state (PKTE_SA_CMD0.HASHSRC=0b10) settings must be used. These registers are a mirror of the SA record inner hash digest register.

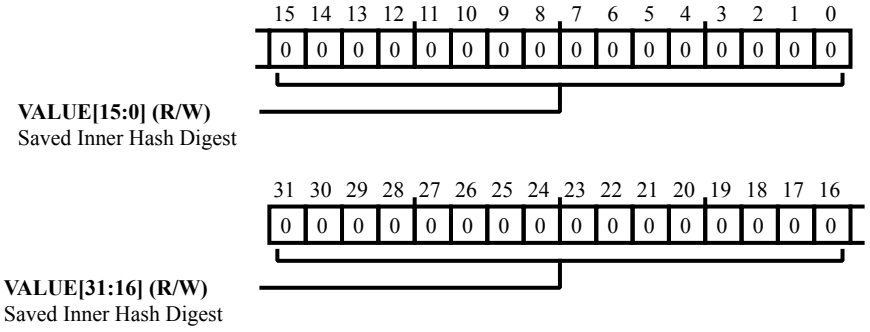


Figure 41-56: PKTE_STATE_IDIGEST[n] Register Diagram

Table 41-82: PKTE_STATE_IDIGEST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Saved Inner Hash Digest.

State Initialization Vector Registers

The PKTE_STATE_IV[n] consists of four 32-bit registers. These registers are used to enter a starting IV state and to read the interim or final IV. PKTE_STATE_IV0 and PKTE_STATE_IV1 are used with DES/3DES cipher while PKTE_STATE_IV0 to PKTE_STATE_IV3 are used with AES cipher. The reset value of these registers is zero.

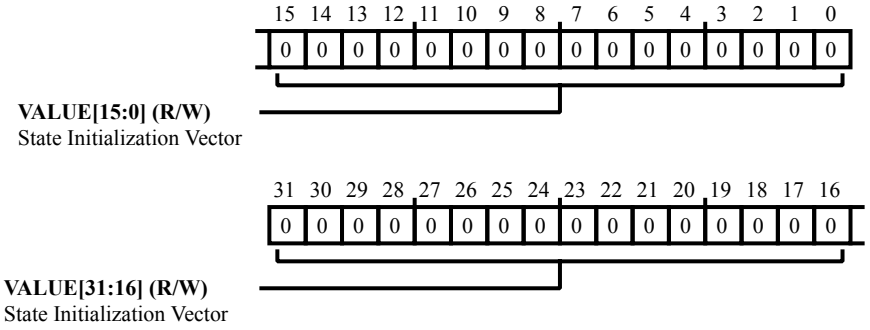


Figure 41-57: PKTE_STATE_IV[n] Register Diagram

Table 41-83: PKTE_STATE_IV[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	State Initialization Vector. The PKTE_STATE_IV[n].VALUE bit field is used to enter a starting IV state and to read the interim or final IV.

Packet Engine User ID

The PKTE_USERID register is a read/write register that gives identification to a command descriptor and the resultant result descriptor. The host is free to use this field for its own purpose. The host can write a unique identifier to the register in direct host mode or includes it as part of the command descriptor in autonomous ring mode. The PKTE_USERID register value passes though the packet engine without alteration to the result descriptor to be read back by the host.

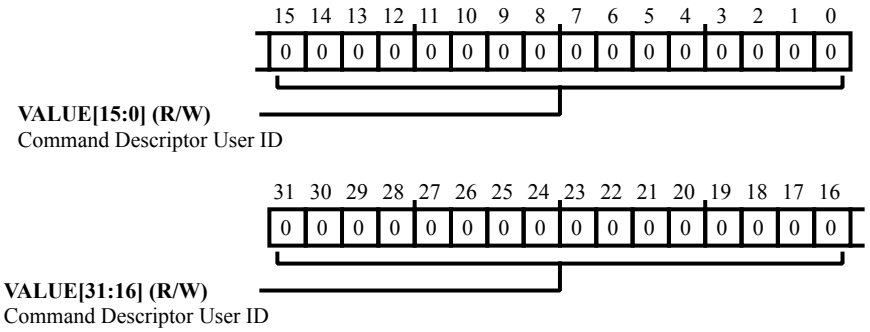


Figure 41-58: PKTE_USERID Register Diagram

Table 41-84: PKTE_USERID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command Descriptor User ID. The PKTE_USERID.VALUE bit field gives identification to a command descriptor and the resultant result descriptor.

42 Public Key Accelerator (PKA)

The PKA helps offload computationally-intensive operations commonly found in public key cryptography algorithms.

PKA Features

The PKA engine provides the following basic operations:

- Large vector addition, subtraction, and combined addition/subtraction
- Large vector shift right or left
- Large vector multiplication, division (with and without quotient)
- Large vector compare and copy

The PKA engine provides the following complex operations:

- Large vector unsigned value modular exponentiation
- Large vector unsigned value modular exponentiation using the 'Chinese Remainders Theorem' (CRT) method with pre-calculated Q inverse vector
- Modular inversion: Given A and M, calculate B such that ((A × B) MOD M) = 1
- ECC point addition/doubling on elliptic curve $y^2 = x^3 + ax + b \pmod p$ with prime number *p* and input values *a* and *b* to the operation. Adding two identical points automatically performs point doubling.
- ECC point multiplication on elliptic curve $y^2 = x^3 + ax + b \pmod p$ with prime number *p* and input values *a* and *b* to the operation. A version of the 'Montgomery ladder' algorithm is used to provide side channel attack resistance.

The PKA also contains hardware logic to automatically zero out the PKA RAM buffer to clear out any information that is considered sensitive or secure.

PKA Functional Description

The following sections provide details on the function of the PKA module.

ADSP-SC59x PKA Register List

The Public Key Accelerator module (PKA) provides security-related features. A set of registers governs PKA operations. For more information on PKA functionality, see the PKA register descriptions.

Table 42-1: ADSP-SC59x PKA Register List

Name	Description
PKA_ALEN	PKA Vector_A Length
PKA_APTR	PKA Vector_A Address
PKA_BLEN	PKA Vector_B Length
PKA_BPTR	PKA Vector_B Address
PKA_COMPARE	PKA Compare Result
PKA_CPTR	PKA Vector_C Address
PKA_DIVMSW	PKA Most-Significant-Word of Divide Remainder
PKA_DPTR	PKA Vector_D Address
PKA_FUNC	PKA Function
PKA_RAM	Start of PKA RAM space
PKA_RESULTMSW	PKA Most-Significant-Word of Result Vector
PKA_SHIFT	PKA Bit Shift Value

PKA Definitions

The following definitions are helpful when using the PKA module.

Elliptic Curve Cryptography (ECC)

A form of public key cryptography based on elliptic curves over finite fields.

RSA

An acronym for Ron Rivest, Adi Shamir, and Leonard Adleman. It is another form of a public key cryptosystem.

Chinese Remainder Theorem (CRT)

A mathematical theorem used for simplifying time-consuming arithmetic used in public key algorithm computations.

Addition Chaining Table (ACT)

A method of speeding up exponentiation by repeatedly squaring the input, storing the result, and reusing the result as input. ACT2 uses a table with 2 address bits (4 entries) and ACT4 uses a table with 4 address bits (16 entries).

PKA Architectural Concepts

The following sections describe the PKA architecture.

Public Key Co-Processor (PKCP)

The Public Key Co-Processor (PKCP) handles the basic large vector processing such as addition, subtraction, multiplication, etc.

Sequencer

The sequencer is small processor that is part of the PKA which handles the more complicated vector processing for public key algorithms. Algorithms include modular exponentiation and the ECC addition and ECC multiply used in Elliptic Curve Cipher algorithms. It executes instructions stored from an internal pre-programmed ROM that handles these operations.

RAM

Input and output vectors are stored in a 4 kB RAM buffer that is part of the MMR space. The address of PKA_RAM is the beginning of the RAM space. This memory is also used as a scratchpad or workspace for the sequencer and PKCP. Programs must place the vectors appropriately following the constraints described in the ***Functional Description*** section of this chapter.

PKA Block Diagram

The ***PKA Block Diagram*** shows the top-level block diagram of the PKA engine. The PKA engine is comprised of five parts:

1. Registers for input, output, status, and control
2. Public Key Co-processor (PKCP) module which performs the basic suite of big number (vector) operations typically found in public key cryptography applications
3. Sequencer which controls modular exponentiation, elliptic curve cryptography, and modular inversion operations.
4. Program ROM associated with the PKA engine exclusively for the sequencer
5. PKA RAM holds the large input and output values as well as the workspace/scratchpad required from the sequencer and PKCP for operations.

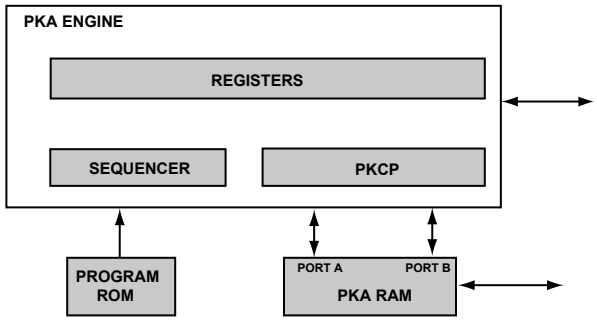


Figure 42-1: PKA Block Diagram

PKCP Vector Operations

The ***Summary of PKCP Vector Operations*** table lists the arguments and results for each PKCP vector operation.

Table 42-2: Summary of PKCP Vector Operations

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
Multiply	A x B → C	Multiplicand	Multiplier	Product	N/A
Add	A + B → C	Addend	Addend	Sum	N/A
Subtract	A - B → C	Minuend	Subtracthend	Difference	N/A
AddSub	A + C - B → D	Addend	Subtracthend	Addend	Result
Right Shift	A >> Shift → C	Input	N/A	Result	N/A
Left Shift	A << Shift → C	Input	N/A	Result	N/A
Divide	A mod B → C, A div B → D	Dividend	Divisor	Remainder	Quotient
Modulo	A mod B → C	Dividend	Divisor	Remainder	N/A
Compare	A = B, A < B, A > B	Input 1	Input 2	N/A	N/A
Copy	A → C	Input	N/A	Result	N/A

To obtain correct result, the input vectors must meet the requirements presented in the ***Operational Restrictions on Input Vectors for PKCP Operations*** table.

Note the following:

- The PKCP does not check input restrictions
- A_Len and B_Len indicate the size of vectors A and B in 32-bit words
- Max_Len equals 128 (32-bit) words, for example, the standard maximum vector size is 4096 bits

Table 42-3: Operational Restrictions on Input Vectors for PKCP Operations

Function	Requirement
Multiply	0 < A_Len, B_Len <= Max_Len
Add	0 < A_Len, B_Len <= Max_Len
Subtract	0 < A_Len, B_Len <= Max_Len Result must be positive (A>=B)
AddSub	0 < A_Len <= Max_Len (B and C operands have A_Len as length, B_Len ignored) Result must be positive ((A + C) >= B)
Right Shift	0 < A_Len <= Max_Len
Left Shift	0 < A_Len <= Max_Len
Divide, Modulo	1 < B_Len <= A_Len <= Max_Len Most significant 32-bit word of B operand cannot be zero
Compare	0 < A_Len <= Max_Len (B operand has A_Len as length, B_Len ignored)
Copy	0 < A_Len <= Max_Len

The host processor is responsible for allocating a block of contiguous memory in PKA RAM for the result vectors. The ***PKCP Result Vector Memory Allocation*** table indicates how much memory is allocated for the result vectors.

Table 42-4: PKCP Result Vector Memory Allocation

Function	Result Vector	Result Vector Length (in 32-bit words)
Multiply	C	A_Len + B_Len + 6 (the 6 'scratchpad' words should be discarded)
Add	C	Max(A_Len, B_Len) + 1
Subtract	C	Max(A_Len, B_Len)
AddSub	D	A_Len + 1
Right Shift	C	A_Len
Left Shift	C	A_Len + 1 (when Shift Value is non-zero) A_Len (when Shift Value is zero)
Divide	C	Remainder → B_Len + 1 (one 'scratchpad' word should be discarded)
	D	Quotient → A_Len – B_Len + 1
Modulo	C	Remainder → B_Len + 1 (one 'scratchpad' word should be discarded)
Compare	None	Compare updates the PKA_COMPARE register
Copy	C	A_Len

Input vectors for an operation are always allowed to overlap in memory (partially or completely). The ***PKCP Result Vector/Input Overlap Restrictions*** table gives restrictions for the overlap of output and input vectors of the operations.

Table 42-5: PKCP Result Vector/Input Overlap Restrictions

Function	Result Vector	Restrictions
Multiply	C	No overlap with A or B vectors allowed
Add, Subtract	C	May overlap with A and/or B vector, provided the start address of the C vector does not lie above the start address of the vectors with which it overlaps
AddSub	D	May overlap with A, B and/or C vector, provided the start address of the D vector does not lie above the start address of the vectors with which it overlaps
Right Shift, Left Shift	C	May overlap with A vector, provided the start address of the C vector does not lie above the start address of the A vector
Divide	C	No overlap with A, B, or D vectors allowed
	D	No overlap with A, B, or C vectors allowed
Modulo	C	No overlap with A or B vectors allowed
Compare	None	Compare does not write a result vector
Copy	None	Same restrictions as for right or left shift, copy of a vector to a lower address is always allowed even if source and destination overlap†

†The copy operation can be used to fill memory by breaking the overlap restrictions, but it requires setting up TWO initial (32-bit) words. To zero a block of memory, set the A vector pointer to the block start, set the C vector pointer two words higher and the A vector length to the block length minus two (words). Fill the first two words of the block with constant zero and perform a PKCP copy operation to zero the remainder of the block.

Modular Exponentiation Operations

The ***Summary of ExpMod Operations*** table summarizes the modular exponentiation operations that the PKA supports.

Table 42-6: Summary of ExpMod Operations

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
ExpMod-ACT2 ExpMod-ACT4 ExpMod-variable	$C^A \bmod B \rightarrow D$	Exponent, length = A_Len	Modulus, length = B_Len	Base, length = B_Len	Result and Workspace
ExpMod-CRT	See below	Exp P followed by Exp Q at next higher even word address†, both A_Len long	Mod P + buffer word followed by Mod Q at next higher even word address‡, both B_Len long	Q inverse, length = B_Len	Input, Result (both 2xB-Len long) and Workspace

† If A_Len is even, Exp Q follows Exp P immediately – if A_Len is odd, there is one empty word between Exp Q and Exp P.

‡ If B_Len is even, there are two empty words between Mod P and Mod Q – if B_Len is odd, there is one empty (buffer) word between Mod Q and Mod P. Note that the engine may zero the words following Mod P and Mod Q.

The ExpMod-CRT operation performs the following computation steps. (These steps implement Garner's recombination algorithm after the basic exponentiations.)

- X <- (Input mod Mod P)Exp P mod Mod P
- Y <- (Input mod Mod Q)Exp Q mod Mod Q
- Z <- ((((X – Y) mod Mod P) · Q inverse) mod Mod P) · Mod Q
- Result <- Y + Z

The ExpMod-ACT2, -ACT4, and -variable functions implement the same mathematical operation but with a differently sized table with pre-calculated *odd powers*. The ExpMod-ACT2 function uses a table with two entries whereas ExpMod-ACT4 uses a table with eight entries. The ACT4 version gives better performance but needs more memory. ExpMod-variable and ExpMod-CRT operations allow the selection of a variable number (from 1 up to and including 16) of odd powers through the register normally used to specify the number of bits to shift for shift operations.

The exponentiation functions are extensions of the set of PKA functions. Input and result vectors are passed the same way as basic PKCP operations. The ***Restrictions on Input Vectors for ExpMod Operations*** table shows the restrictions on the input and result vectors for the exponentiation operations.

Table 42-7: Restrictions on Input Vectors for ExpMod Operations

Function	Requirements
ExpMod-ACT2 ExpMod-ACT4 ExpMod-variable	0 < A_Len <= Max_Len 1 < B_Len <= Max_Len Modulus B must be odd (for example, the least significant bit must be ONE) Modulus B > 2^{32} Base C < Modulus B Vectors B and C must be followed by an empty 32-bit *buffer* word
ExpMod-CRT	0 < A_Len <= Max_Len 1 < B_Len <= Max_Len Mod P and Mod Q must be odd (for example, the least significant bits must be ONE) Mod P > Mod Q > 2^{32}(note that Mod P must be larger than Mod Q) Mod P and Mod Q must be co-prime (their GCD must be 1) 0 < Exp P < (Mod P – 1) 0 < Exp Q < (Mod Q – 1) (Q inverse · Mod Q) 1 (modulo Mod P) Input < (Mod P · Mod Q) Mod P and Mod Q must be followed by an empty 32-bit *buffer* word

The ***ExpMod Result Vector/Scratchpad Area Memory Allocation Starting at PKA_DPTR*** table shows the required scratchpad sizes for the exponentiation operations. These sizes depend on the PKA type. The 'M_Len' used in the table is the 'real' Modulus length in 32-bit words, for example, without trailing zero words at the end. (This description also applies to Mod P in an ExpMod-CRT operation and Modulus B in the other operations.) If the last word of the modulus vector as given is non-zero, 'M_Len' equals B_Len.

Table 42-8: ExpMod Result Vector/Scratchpad Area Memory Allocation Starting at PKA_DPTR

Function	Scratchpad Area Size (in 32-bit words), Result Vector is either M_Len or 2xM_Len 32-bit words long
ExpMod-ACT2	5 x (M_Len + 2)
ExpMod-ACT4	11 x (M_Len + 2)
ExpMod-variable	(# odd powers + 3) x (M_Len + 2)
ExpMod-CRT	(# odd powers + 3) x (M_Len + 2) + (M_Len + 2 – (M_Len MOD 2))

NOTE: During execution of an ExpMod-ACT2, -ACT4 or -variable operation, the last 34 bytes of the PKA RAM are used as the general scratchpad for the sequencer program execution. The ExpMod-CRT operation requires the last 72 bytes of the PKA RAM as the scratchpad. These (fixed location) areas may not overlap with any of the input vectors and/or the D vector scratchpad area. They can be used freely when executing basic PKCP operations.

Table 42-9: ExpMod Scratchpad Area / Input Vector Overlap Restrictions

Function	Result Vector	Restrictions
ExpMod-ACT2 ExpMod-ACT4 ExpMod-variable	D	Scratchpad area starting at D may not overlap with any of the other vectors, except that base C may be co-located with result vector D to save space (for example, `PKA_CPTR`= `PKA_DPTR` is allowed).
ExpMod-CRT	D	Scratchpad area starting at D may not overlap with any of the other vectors. This area is also the location of the main input vector (with length 2 x B_Len)

The ***Maximum Number of Odd Powers*** table indicates the maximum number of odd powers that can be used for different standard PKA RAM sizes and PKA types (non-CRT operations using `PKA_CPTR` = `PKA_DPTR`). As a rule of thumb, for optimal performance, use one odd power for *Verify* operations and 4 (or as many as the implemented PKA RAM size allows) for *Sign* operations. Note the following points about odd powers:

- Using more than eight odd powers is not advisable as the speed advantage for each extra odd power decreases rapidly (and can even become negative for short exponent vector lengths due to the extra pre-processing required).
- The maximum number of odd powers is 16 (limited by the firmware). All '16 odd powers' entries in the table above hit this limit – they are not limited by the PKA RAM size.

Table 42-10: Maximum Number of Odd Powers

Operation	Modulus and Exponent Sizes	Maximum Number of Odd Powers
Non-CRT	1024 bits	16
	2048 bits	10
	4096 bits	2
CRT	2 × 512 bits	16
	2 ×1024 bits	16
	2 × 2048 bits	6

The ***2K-bit Modular Exponentiation PKA RAM Allocation Examples*** table shows example PKA RAM vector allocations for modular exponentiation operations with and without using CRT. The *free space* start address is the first free byte following the vector workspace. The sequencer execution scratchpad of 34 bytes (non-CRT) or 72 bytes (using CRT) must fit between this address and the end of the PKA RAM. Note that the non-CRT operations use `PKA_CPTR` = `PKA_DPTR` to save space.

Table 42-11: 2K-bit Modular Exponentiation PKA RAM Allocation Examples

Operation	(sub-)vector	Start address (Byte Offset)	Size (words)	Buffer (words)
non-CRT (`PKA_ALEN` = 0x040, `PKA_BLEN` = 0x040, 4 odd-powers)	Exponent	0x000 (`PKA_APTR` = 0x000)	64	0
	Modulus	0x100 (`PKA_BPTR` = 0x040)	64	2
	Base	0x208 (`PKA_CPTR` = 0x082)	64	2
	Result	0x208 (`PKA_DPTR` = 0x082)	64	2
	Vector Workspace	0x208 (= Result)	7 × (64+2)=462	0
	Free space	0x940 (2368 bytes used)	-	-
using CRT (`PKA_ALEN` = 0x020, `PKA_BLEN` = 0x020, 4 odd-powers)	Exp P	0x000 (`PKA_APTR` = 0x000)	32	0
	Exp Q	0x080	32	0
	Mod P	0x100 (`PKA_BPTR` = 0x040)	32	2
	Mod Q	0x188	32	2
	Q inverse	0x210 (`PKA_CPTR` = 0x084)	32	0
	Input, Result	0x290 (`PKA_DPTR` = 0x0A4)	64	0
	Vector workspace	0x290 (= Result)	7 × (32+2)+32+2-0 = 272	0
	Free space	0x6D0 (1744 bytes used)	-	-

The following example in pseudo-code describes the execution of a non-CRT modular exponentiation operation using a 512-bit modulus and a 160-bit exponent, using actual test vectors:

```
// Perform a 512/160 bit
modular exponentiation without CRT (using 4 'odd-powers')
// Exponent equals value 0x8FD84098_8A0930CC_9CDC1E8A_B246EB46_2D39F064
// write as vector A to PKA
```

```
RAM Byte offset 0x000:
Write PKA_RAM_BASE+0x000+0x00
0x2D39F064
Write PKA_RAM_BASE+0x000+0x04
0xB246EB46
Write PKA_RAM_BASE+0x000+0x08
0x9CDC1E8A
Write PKA_RAM_BASE+0x000+0x0C
0x8A0930CC
Write PKA_RAM_BASE+0x000+0x10
0x8FD84098
// Modulus equals value 0xF42F559D
1877CA5F_449492B9_42DC7C01_...
// A3C9085B_7236A085_2102B000_A093C6B4_...
// 9D0EDA0C_292DE841_29C23723_4048BDA3_...
// 373C4C9F_45CF15A7_5F049ABF_D8A01B9B
// write as vector B to PKA
RAM Byte offset 0x018 (following exp at next aligned 64-bit word):
Write PKA_RAM_BASE+0x018+0x00
0xD8A01B9B
Write PKA_RAM_BASE+0x018+0x04
0x5F049ABF
Write PKA_RAM_BASE+0x018+0x08
0x45CF15A7
Write PKA_RAM_BASE+0x018+0x0C
0x373C4C9F
Write PKA_RAM_BASE+0x018+0x10
0x4048BDA3
Write PKA_RAM_BASE+0x018+0x14
0x29C23723
Write PKA_RAM_BASE+0x018+0x18
0x292DE841
Write PKA_RAM_BASE+0x018+0x1C
0x9D0EDA0C
Write PKA_RAM_BASE+0x018+0x20
0xA093C6B4
Write PKA_RAM_BASE+0x018+0x24
0x2102B000
Write PKA_RAM_BASE+0x018+0x28
0x7236A085
Write PKA_RAM_BASE+0x018+0x2C
0xA3C9085B
Write PKA_RAM_BASE+0x018+0x30
0x42DC7C01
Write PKA_RAM_BASE+0x018+0x34
0x449492B9
Write PKA_RAM_BASE+0x018+0x38
0x1877CA5F
Write PKA_RAM_BASE+0x018+0x3C
0xF42F559D
```

```
// Base equals value 0x3D291F48_49064887_1149594B_67935110_...
// 14EB8FF0_AB291F3A_54A1B4D1_5E611E44_...
// C989251B_44904B45_0B060482_317F8352_...
// 18CE440E_9BF509F1_6EAF26F2_95F19F12
// write as vector C to PKA
RAM Byte offset 0x060 (following mod after buffer + align words):
Write PKA_RAM_BASE+0x060+0x00
0x95F19F12
Write PKA_RAM_BASE+0x060+0x04
0x6EAF26F2
Write PKA_RAM_BASE+0x060+0x08
0x9BF509F1
Write PKA_RAM_BASE+0x060+0x0C
0x18CE440E
Write PKA_RAM_BASE+0x060+0x10
0x317F8352
Write PKA_RAM_BASE+0x060+0x14
0x0B060482
Write PKA_RAM_BASE+0x060+0x18
0x44904B45
Write PKA_RAM_BASE+0x060+0x1C
0xC989251B
Write PKA_RAM_BASE+0x060+0x20
0x5E611E44
Write PKA_RAM_BASE+0x060+0x24
0x54A1B4D1
Write PKA_RAM_BASE+0x060+0x28
0xAB291F3A
Write PKA_RAM_BASE+0x060+0x2C
0x14EB8FF0
Write PKA_RAM_BASE+0x060+0x30
0x67935110
Write PKA_RAM_BASE+0x060+0x34
0x1149594B
Write PKA_RAM_BASE+0x060+0x38
0x49064887
Write PKA_RAM_BASE+0x060+0x3C
0x3D291F48
// The result value and scratchpad
(vector D) may be co-located with the base vector C for
// a normal modular exponentiation,
so these are located at PKA RAM Byte offset 0x060 too.
// Load pointer and length
registers:
Write PKA_APTR 0x000>>2 //
Exponent pointer
Write PKA_BPTR 0x018>>2 //
Modulus pointer
Write PKA_CPTR 0x060>>2 //
Base pointer
```

```
Write PKA_DPTR 0x060>>2 //
Result/scratchpad pointer
Write PKA_ALENGTH 0x00000005
// Exponent length in 32-bit words
Write PKA_BLENGTH 0x00000010
// Mod/base/result length in 32-bit words
// Start modular exponentiation
and wait until it's done:
Write PKA_SHIFT 0x00000004
// Number of 'odd powers'
Write PKA_FUNCTION 0x0000E000
// 'Run' bit set, 'Sequencer Operations' = 0b110
Wait PKA_FUNCTION[15] == '0'
// 'Run' bit clears itself - Host can also use interrupt!
// Result value equals 0xA497BF8B_DB729088_954005B0_B5CA6691_...
// A3EC491B_091A3D62_03C24214_0863A389_...
// 0C7C03CD_2333E231_35EC10ED_8F91281C_...
// 30F4253B_FE38FAFB_BB4A39DB_C14F2661
// written as vector D at
PKA RAM Byte offset 0x060:
Check PKA_RAM_BASE+0x060+0x00
== 0xC14F2661
Check PKA_RAM_BASE+0x060+0x04
== 0xBB4A39DB
Check PKA_RAM_BASE+0x060+0x08
== 0xFE38FAFB
Check PKA_RAM_BASE+0x060+0x0C
== 0x30F4253B
Check PKA_RAM_BASE+0x060+0x10
== 0x8F91281C
Check PKA_RAM_BASE+0x060+0x14
== 0x35EC10ED
Check PKA_RAM_BASE+0x060+0x18
== 0x2333E231
Check PKA_RAM_BASE+0x060+0x1C
== 0x0C7C03CD
Check PKA_RAM_BASE+0x060+0x20
== 0x0863A389
Check PKA_RAM_BASE+0x060+0x24
== 0x03C24214
Check PKA_RAM_BASE+0x060+0x28
== 0x091A3D62
Check PKA_RAM_BASE+0x060+0x2C
== 0xA3EC491B
Check PKA_RAM_BASE+0x060+0x30
== 0xB5CA6691
Check PKA_RAM_BASE+0x060+0x34
== 0x954005B0
Check PKA_RAM_BASE+0x060+0x38
== 0xDB729088
```

```
Check PKA_RAM_BASE+0x060+0x3C
== 0xA497BF8B
```

Modular Inversion

Besides modular exponentiation, the sequencer also controls modular inversion operations.

Table 42-12: Summary of ModInv Operation

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
ModInv	A^{-1} mod B → D	NumToInvert, length = A_Len	Modulus, length = B_Len	Not Used	Result and Workspace

The above function is an extension of the set of basic PKCP functions with the following exceptions:

- Vector D not only addresses the result, but also a workspace.
- The `PKA_SHIFT` register field is used to return information on the operation's result.

Table 42-13: PKA_SHIFT Result Values for ModInv Operation

Function	PKA_SHIFT Register Field Value At Conclusion
ModInv	0 → success; VectorD holds result 7 → no inverse exists (GCD(A, B) != 1, for example, A and B have common factors); result undefined 31 → error, modulus even; result undefined other values are reserved

The following tables list the restrictions on the input and result vectors for the ModInv operation.

Table 42-14: Operational Restrictions on Input Vectors for the ModInv Operation

Function	Requirements
ModInv	0 < A_Len <= Max_Len 0 < B_Len <= Max_Len Modulus B must be odd (for example, the least significant bit must be ONE) Modulus B may not have value 1 (result is undefined, no error indicated) The highest word of the modulus vector, as indicated by B_Len, may not be zero.

Table 42-15: ModInv Scratchpad Area/Input Vector Overlap Restrictions

Function	Result Vector	Restrictions
ModInv	D	Scratchpad area starting at D may not overlap with any of the other vectors

The following table shows the required scratchpad sizes for the ModInv Operation.

Table 42-16: ModInv Result Vector/Scratchpad Area Memory Allocation (Both Starting at PKA_DPTR)

Function	Scratchpad area size (in 32-bit words), Result Vector is B_Length 32-bit words long
ModInv	5 x (M + ε(M)), with M = Max(A_Length, B_Length) ε(n) = 2 + (n MOD 2), for example, 2 (for n even) or 3 (for n odd)

NOTE: During execution of a ModInv operation, the last 34 bytes of the PKA RAM are used as general scratchpad for the sequencer program execution. This (fixed location) area may not overlap with any of the input vectors and/or the D vector scratchpad area during execution.

Modular Inversion with an Even Modulus

The ModInv operation requires the modulus to be odd. At first, this requirement appears to make the operation useless in the case of RSA key generation where the private key exponent *d* is derived from a chosen public exponent *e* as follows:

d = ModInv(e, φ); where φ = (p-1) × (q-1) and *p* and *q* both prime

Note that φ is even. However, since *e* must be odd (otherwise no inverse exists), *d* can be calculated as:

d = (1 + (φ × (e – ModInv(φ, e))) / e

With four more basic PKCP operations, ModInv can also be used to find inverse values in case the modulus is even.

Modular Inversion with a Prime Modulus

Modular inversion can be performed with a modular exponentiation using the modulus value minus two as exponent, provided that the modulus value is a prime. This is due to the following:

(A^M) mod M = A =>

(A^{M-1}) mod M = 1 =>

(A^{M-2}) mod M = A–1 (mod M)

Under the constraint that M is a prime value.

Especially with the large PKA engines containing an LNME, it is worthwhile to check whether this method is faster than using the ModInv operation directly. The modulus values for the ECC curves supported by this PKA engine must be prime, so this method can be used in ECDSA operations.

ECC Operations

Besides modular exponentiation and modular inversion, the sequencer also controls ECC operations.

Table 42-17: Summary of ECC Operations

Function	Mathematical Operation	Vector A	Vector B	Vector C	Vector D
ECC-ADD	Point addition/ doubling† on elliptic curve: $y^2 = x^3 + ax + b \pmod{p}$ pntA + pntC → pntD	pntA.x followed‡ by pntA.y both B_Len long (A_Len *not* used)	Curve parameter *p* followed‡ by *a* (*b* is not needed) all B_Len long	pntC.x followed by pntC.y both B_Len long	Result, for example, pntD.x followed‡ by pntD.y and workspace
ECC-MUL	Point multiplication on elliptic curve: $y^2 = x^3 + ax + b \pmod{p}$ *k* x pntC → pntD	Scalar *k* A_Len long	Curve parameter *p* followed‡ by *a*and *b* all B_Len long.	pntC.x followed‡ by pntC.y both B_Len long	Result, for example pntD.x followed‡ by pntD.y and workspace

† If pntA = pntC, a point doubling operation is performed automatically.

‡ All input components must be located on a 64-bit boundary and must have extra 'buffer' words (of 32 bits each) after their most significant word. ε must be 3 (B_Len odd) or 2 (B_Len even). Each result component (for example, pntD.x, pntD.y) is followed by ε buffer (zero) words.

The above functions are extensions of the set of PKCP basic functions with the following exceptions:

- Input and result vectors can now be composite (for example, consist of two or three equal-sized subvectors).
- Vector D not only addresses the result, but also a workspace.
- The `PKA_SHIFT` register is used to return information on the result of the operation.

Table 42-18: PKA_SHIFT Result Values for ECC Operations

Function	PKA_SHIFT Register Field Value at Conclusion
ECC-ADD ECC-MUL	0 → success; VectorD holds result point 7 → result is point-at-infinity; VectorD result point undefined 31 → error, (p not odd, p too short, etc); VectorD result point undefined Other values are reserved

The following tables below list the restrictions on the input and result vectors for the ECC operations.

Table 42-19: Operational Restrictions on Input Vectors for ECC Operations

Function	Requirements
ECC-ADD	1 < B_Len <= 24 (maximum vector length is 768 bits) Modulus p must be a prime > 2^{63} Effective modulus size (in bits) must be a multiple of 32† The highest word of the modulus vector, as indicated by B_Len, may not be zero. *a*< p and *b*< p pntA and pntC must be on the curve (this condition is not checked) Neither pntA nor pntC can be the point-at-infinity, although ECC-ADD can return this point as a result
ECC-MUL	0 < A_Len <= 24 (maximum vector length is 768 bits) 1 < B_Len <= 24 (maximum vector length is 768 bits) Modulus p must be a prime > 2^{63} Effective modulus size (in bits) must be a multiple of 32† The highest word of the modulus vector, as indicated by B_Len, may not be zero. *a*< p and *b*< p pntC must be on the curve (this condition is not checked) pntC cannot be the point-at-infinity, although ECC-MUL can return this point as a result

† Modulus lengths of 112 and 521 bits are exceptions to this rule.

Table 42-20: ECC Scratchpad Area/Input Vector Overlap Restrictions

Function	Result Vector	Restrictions
ECC-ADD ECC-MUL	D	Scratchpad area starting at D may not overlap with any of the other vectors

The ***>ECC Result Vector/Scratchpad Area Memory Allocation*** table shows the required scratchpad sizes for the ECC operations:

Table 42-21: ECC Result Vector/Scratchpad Area Memory Allocation (Both Starting at PKA_DPTR)

Function	Scratchpad area size (in 32-bit words), Result Vector is 2x(B_Length+ ε †(B_Length)) 32-bit words long
ECC-ADD	2 × L + 5 x M, where L = B_Length + ε(B_Length) M = B_Length + 1 + ε(B_Length + 1)
ECC-MUL	18 x L + Max(8, L), where L = (B_Length + ε(B_Length))

† ε(*n*) = 2 + (*n* MOD 2), for example, 2 (for *n* even) or 3 (for *n* odd)

NOTE: During execution of an ECC-ADD or ECC-MUL operation, the last 72 bytes of the PKA RAM are used as a general scratchpad for the sequencer program execution. These (fixed location) areas must not overlap with any of the input vectors and the D vector scratchpad area during execution.

The ***ECC Point Multiplication PKA RAM Allocation Examples*** table shows example PKA RAM vector allocations for ECC point multiplication operations. The *free space* start address is the first free byte following the vector scratchpad. The sequencer execution scratchpad of 72 bytes must fit between this address and the end of the PKA RAM. Because of this requirement, a 521-bit ECC point multiplication cannot be performed with 2K byte PKA RAM.

Table 42-22: ECC Point Multiplication PKA RAM Allocation Examples

Modulus Length	(sub-)vector	Start Address (byte offset)	Size (words)	Buffer (words)
192 bits (=6 words, PKA_ALEN=0x006, PKA_BLEN=0x006)	Scalar *k*	0x000 (PKA_APTR = 0x000)	6	0
	p	0x018 (PKA_BPTR = 0x006)	6	2
	a	0x038	6	2
	b	0x058	6	2
	PntC.x (base)	0x078 (PKA_CPTR = 0x01E)	6	2
	PntC.y (base)	0x098	6	2
	PntD.x (result)	0x0B8 (PKA_DPTR = 0x02E)	6	2
	PntD.y (result)	0x0D8	6	0
	Vector scratchpad	0x0B8 (= PntD.x)	(18 × 8) + 8 = 152	0
	Free space	0x318 (792 bytes used)	-	-
384 bits (=12 words, ALENGTH=0x00C, BLENGTH=0x00C)	Scalar *k*	0x000 (PKA_APTR = 0x000)	12	0
	p	0x030 (PKA_BPTR = 0x00C)	12	2
	a	0x068	12	2
	b	0x0A0	12	2
	PntC.x (base)	0x0D8 (PKA_CPTR = 0x036)	12	2
	PntC.y (base)	0x110	12	2
	PntD.x (result)	0x148 (PKA_DPTR = 0x052)	12	2
	PntD.y (result)	0x180	12	0
	Vector scratchpad	0x148 (= PntD.x)	(18 × 14) + 14 = 266	0
	Free space	0x570 (1392 bytes used)	-	-
521 bits (=17 words, ALENGTH=0x011, BLENGTH=0x011)	Scalar *k*	0x000 (PKA_APTR = 0x000)	17	1 (to align *p*)
	p	0x048 (PKA_BPTR = 0x012)	17	3
	a	0x098	17	3
	b	0x0E8	17	3
	PntC.x (base)	0x138 (PKA_CPTR = 0x04E)	17	3
	PntC.y (base)	0x188	17	3
	PntD.x (result)	0x1D8 (PKA_DPTR = 0x076)	17	3

Table 42-22: ECC Point Multiplication PKA RAM Allocation Examples (Continued)

Modulus Length	(sub-)vector	Start Address (byte offset)	Size (words)	Buffer (words)
	PntD.y (result)	0x228	17	0
	Vector scratchpad	0x1D8 (= PntD.x)	(18 × 20) + 20 = 380	0
	Free space	0x7C8 (1992 bytes used)	-	-

The following example in pseudo-code describes the execution of a 192 bits ECC point multiplication, using actual test vectors (the curve parameters and generator point are from standard curve ‘secp192r1’).

```
// Perform a 192 bits ECC point multiplication using PKA RAM layout from table
above.
// Scalar ‘k’ equals value 0x8D98D058_9EFD018A_C9BCF3CF_2C33AEC0_24867D7F_6ADACBFF
// write as vector A to PKA RAM Byte offset 0x000:

    Write PKA_RAM_BASE+0x000+0x00 0x6ADACBFF
    Write PKA_RAM_BASE+0x000+0x04 0x24867D7F
    Write PKA_RAM_BASE+0x000+0x08 0x2C33AEC0
    Write PKA_RAM_BASE+0x000+0x0C 0xC9BCF3CF
    Write PKA_RAM_BASE+0x000+0x10 0x9EFD018A
    Write PKA_RAM_BASE+0x000+0x14 0x8D98D058
// Curve parameter ‘p’ equals value
0xFFFFFFFF_FFFFFFFF_FFFFFFFF_FFFFFFFE_FFFFFFFF_FFFFFFFF
// write as 1st part of vector B immediately following vector A at PKA RAM Byte
offset 0x018
// (no buffer word needed after ‘k’ vector, 64-bit alignment is OK):
    Write PKA_RAM_BASE+0x018+0x00 0xFFFFFFFF
    Write PKA_RAM_BASE+0x018+0x04 0xFFFFFFFF
    Write PKA_RAM_BASE+0x018+0x08 0xFFFFFFFE
    Write PKA_RAM_BASE+0x018+0x0C 0xFFFFFFFF
    Write PKA_RAM_BASE+0x018+0x10 0xFFFFFFFF
    Write PKA_RAM_BASE+0x018+0x14 0xFFFFFFFF
// Curve parameter ‘a’ equals value
0xFFFFFFFF_FFFFFFFF_FFFFFFFF_FFFFFFFE_FFFFFFFF_FFFFFFFC
// write as 2nd part of vector B after one buffer word and one re-alignment word
at 0x038:
    Write PKA_RAM_BASE+0x038+0x00 0xFFFFFFFC
    Write PKA_RAM_BASE+0x038+0x04 0xFFFFFFFF
    Write PKA_RAM_BASE+0x038+0x08 0xFFFFFFFE
    Write PKA_RAM_BASE+0x038+0x0C 0xFFFFFFFF
    Write PKA_RAM_BASE+0x038+0x10 0xFFFFFFFF
    Write PKA_RAM_BASE+0x038+0x14 0xFFFFFFFF
// Curve parameter ‘b’ equals value
0x64210519_E59C80E7_0FA7E9AB_72243049_FEB8DEEC_C146B9B1
// write as 3rd part of vector B after one buffer word and one re-alignment word
at 0x058:
    Write PKA_RAM_BASE+0x058+0x00
0xC146B9B1
```

```
    Write PKA_RAM_BASE+0x058+0x04 0xFEB8DEEC
    Write PKA_RAM_BASE+0x058+0x08 0x72243049
    Write PKA_RAM_BASE+0x058+0x0C 0x0FA7E9AB
    Write PKA_RAM_BASE+0x058+0x10 0xE59C80E7
    Write PKA_RAM_BASE+0x058+0x14 0x64210519
// X-coord of generator point is value
0x188DA80E_B03090F6_7CBF20EB_43A18800_F4FF0AFD_82FF1012
// write as 1st part of vector C following vector B after buffer + alignment
words at 0x078:
    Write PKA_RAM_BASE+0x078+0x00 0x82FF1012
    Write PKA_RAM_BASE+0x078+0x04 0xF4FF0AFD
    Write PKA_RAM_BASE+0x078+0x08 0x43A18800
    Write PKA_RAM_BASE+0x078+0x0C 0x7CBF20EB
    Write PKA_RAM_BASE+0x078+0x10 0xB03090F6
    Write PKA_RAM_BASE+0x078+0x14 0x188DA80E
// Y-coord of generator point is value
0x07192B95_FFC8DA78_631011ED_6B24CDD5_73F977A1_1E794811
// write as 2nd part of vector C after one buffer word and one re-alignment word
at 0x098:
    Write PKA_RAM_BASE+0x098+0x00 0x1E794811
    Write PKA_RAM_BASE+0x098+0x04 0x73F977A1
    Write PKA_RAM_BASE+0x098+0x08 0x6B24CDD5
    Write PKA_RAM_BASE+0x098+0x0C 0x631011ED
    Write PKA_RAM_BASE+0x098+0x10 0xFFC8DA78
    Write PKA_RAM_BASE+0x098+0x14 0x07192B95
// The result point and scratchpad (vector D) follow vector C after one buffer
word and one
// re-alignment word, so these are located at PKA RAM Byte offset 0x0B8.
// Load pointer and length registers:
    Write PKA_APTR 0x000>>2 // Scalar 'k' pointer
    Write PKA_BPTR 0x018>>2 // Curve parameters 'p', 'a' & 'b' pointer
    Write PKA_CPTR 0x078>>2 // Generator point X & Y coordinates pointer
    Write PKA_DPTR 0x0B8>>2 // Result point X & Y coordinates/scratchpad pointer
    Write PKA_ALENGTH 0x00000006 // Scalar 'k' length in 32-bit words
    Write PKA_BLENGTH 0x00000006 // Curve parameters and coordinate lengths in 32-
bit words
// Start ECC point multiplication and wait until it's done:
    Write PKA_FUNCTION 0x0000D000
// 'Run' bit set, 'Sequencer Operations' = 0b101
    Wait PKA_FUNCTION[15] == '0'
// 'Run' bit clears itself - Host can also use interrupt!
    Check PKA_SHIFT == 0x00000000
// Shift field value 0 indicates success - check this
// X-coord of result point is value
0x759B9F39_0E81D268_18C82BB9_CB42BCF5_0E0AE958_85BA3097
// written as 1st part of vector D at PKA RAM Byte offset 0x0B8:
    Check PKA_RAM_BASE+0x0B8+0x00== 0x85BA3097
    Check PKA_RAM_BASE+0x0B8+0x04== 0x0E0AE958
    Check PKA_RAM_BASE+0x0B8+0x08== 0xCB42BCF5
    Check PKA_RAM_BASE+0x0B8+0x0C== 0x18C82BB9
```

```
    Check PKA_RAM_BASE+0x0B8+0x10== 0x0E81D268
    Check PKA_RAM_BASE+0x0B8+0x14== 0x759B9F39
// Y-coord of result point is value
0xECA14640_F92EFF07_CAF2BD55_3FBE28EF_D043F28E_1CC3D238
// written as 2nd part of vector D at PKA RAM Byte offset 0x0D8:
    Check PKA_RAM_BASE+0x0D8+0x00== 0x1CC3D238
    Check PKA_RAM_BASE+0x0D8+0x04== 0xD043F28E
    Check PKA_RAM_BASE+0x0D8+0x08== 0x3FBE28EF
    Check PKA_RAM_BASE+0x0D8+0x0C== 0xCAF2BD55
    Check PKA_RAM_BASE+0x0D8+0x10== 0xF92EFF07
    Check PKA_RAM_BASE+0x0D8+0x14== 0xECA14640
```

ADSP-SC59x PKA Register Descriptions

Public Key Accelerator (PKA) contains the following registers.

Table 42-23: ADSP-SC59x PKA Register List

Name	Description
PKA_ALEN	PKA Vector_A Length
PKA_APTR	PKA Vector_A Address
PKA_BLEN	PKA Vector_B Length
PKA_BPTR	PKA Vector_B Address
PKA_COMPARE	PKA Compare Result
PKA_CPTR	PKA Vector_C Address
PKA_DIVMSW	PKA Most-Significant-Word of Divide Remainder
PKA_DPTR	PKA Vector_D Address
PKA_FUNC	PKA Function
PKA_RAM	Start of PKA RAM space
PKA_RESULTMSW	PKA Most-Significant-Word of Result Vector
PKA_SHIFT	PKA Bit Shift Value

PKA Vector_A Length

During execution of basic PKCP operations, the PKA_ALEN register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_ALEN register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

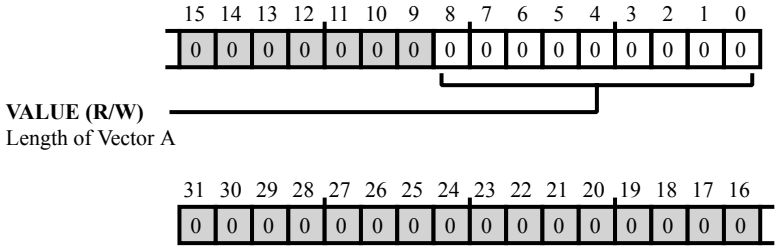


Figure 42-2: PKA_ALEN Register Diagram

Table 42-24: PKA_ALEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Length of Vector A. Length (in 32-bit words) of Vector A.

PKA Vector_A Address

During execution of basic PKCP operations, the PKA_APTR register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_APTR register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

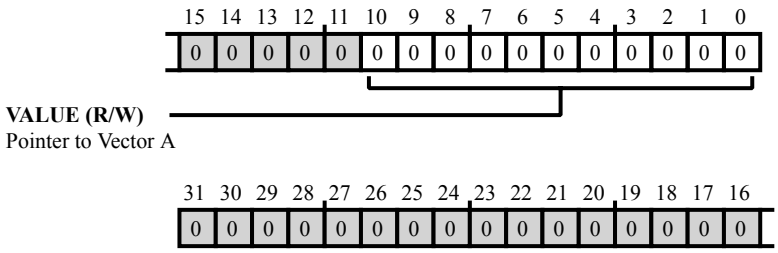


Figure 42-3: PKA_APTR Register Diagram

Table 42-25: PKA_APTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Pointer to Vector A. The PKA_APTR.VALUE bit field is the location of Vector A within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

PKA Vector_B Length

During execution of basic PKCP operations, the PKA_BLEN register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_BLEN register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

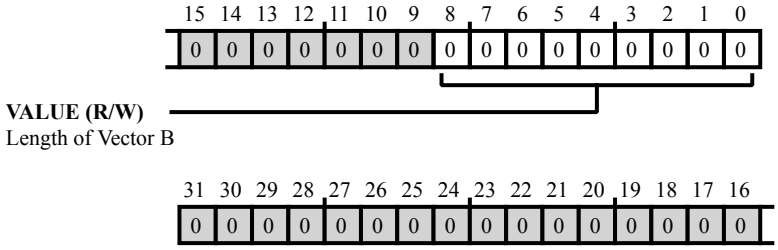


Figure 42-4: PKA_BLEN Register Diagram

Table 42-26: PKA_BLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/W)	VALUE	Length of Vector B. Length (in 32-bit words) of Vector B.

PKA Vector_B Address

During execution of basic PKCP operations, the PKA_BPTR register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_BPTR register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

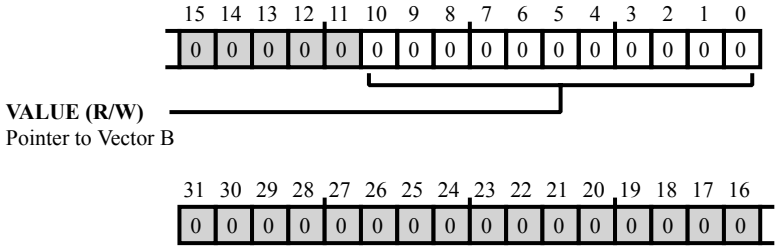


Figure 42-5: PKA_BPTR Register Diagram

Table 42-27: PKA_BPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Pointer to Vector B. The PKA_BPTR.VALUE bit field is the location of Vector B within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

PKA Compare Result

The PKA_COMPARE register provides the result of a basic PKCP Compare operation. It is updated when the PKA_FUNC.RUN bit is reset at the end of that operation. The status after a complex sequencer operation is unknown.

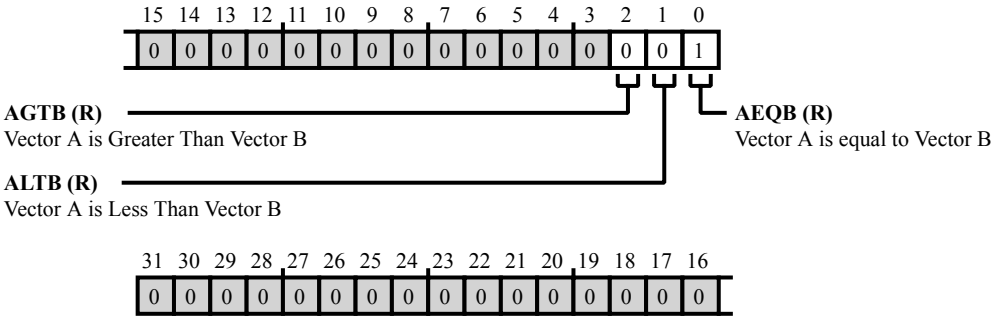


Figure 42-6: PKA_COMPARE Register Diagram

Table 42-28: PKA_COMPARE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2 (R/NW)	AGTB	Vector A is Greater Than Vector B. The PKA_COMPARE.AGTB bit shows the result of the basic compare operation is PKCP Vector_A is greater than Vector_B.
1 (R/NW)	ALTB	Vector A is Less Than Vector B. The PKA_COMPARE.ALTB bit shows the result of the basic compare operation is Vector_A is less than Vector_B.
0 (R/NW)	AEQB	Vector A is equal to Vector B. The PKA_COMPARE.AEQB bit shows the result of the basic compare operation is Vector_A is equal to Vector_B.

PKA Vector_C Address

During execution of basic PKCP operations, the PKA_CPTR register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_CPTR register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

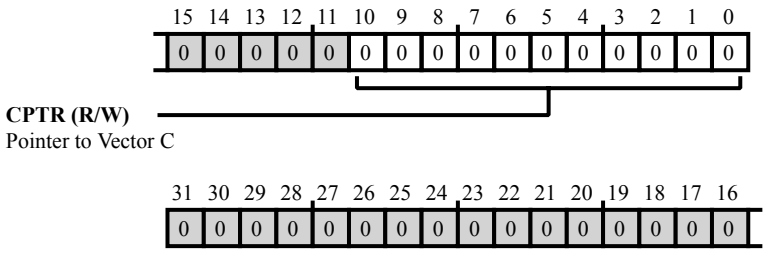


Figure 42-7: PKA_CPTR Register Diagram

Table 42-29: PKA_CPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	CPTR	Pointer to Vector C. The PKA_CPTR.CPTR bit field is the location of Vector C within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

PKA Most-Significant-Word of Divide Remainder

The PKA_DIVMSW register indicates the (32-bit word) address in the PKA RAM where the most significant non-zero 32-bit word of the Remainder result for the basic Divide and Modulo operations is stored. Bits [4:0] are loaded with the bit number of the most significant non-zero bit in the most significant non-zero word when MS one control bit is set. For Divide, Modulo and MS one reporting, this register is updated when the PKA_FUNC.RUN bit is reset at the end of the operation.

For the complex sequencer controlled operations, updating bits [4:0] of this register with the actual result's most significant bit location is done near the end of the operation. Note that the result is only meaningful if no errors were detected and that for ECC operations, the PKA_DIVMSW register provides information for the x-coordinate of the result point only.

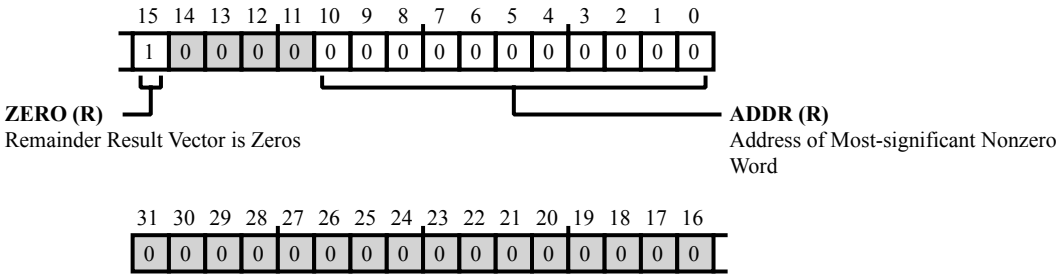


Figure 42-8: PKA_DIVMSW Register Diagram

Table 42-30: PKA_DIVMSW Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/NW)	ZERO	Remainder Result Vector is Zeros. The PKA_DIVMSW.ZERO bit shows the remainder result vector is all zeros, ignore the address returned in bits [10:0].
10:0 (R/NW)	ADDR	Address of Most-significant Nonzero Word. The PKA_DIVMSW.ADDR bit shows the address of the most significant non-zero 32-bit word of the remainder result vector in PKA RAM.

PKA Vector_D Address

During execution of basic PKCP operations, the PKA_DPTR register is double buffered and can be written with a new value for the next operation. When not written, the value remains intact. During the execution of sequencer controlled complex operations, the PKA_DPTR register may not be written and its value is undefined at the conclusion of the operation. The driver software cannot rely on the written value to remain intact.

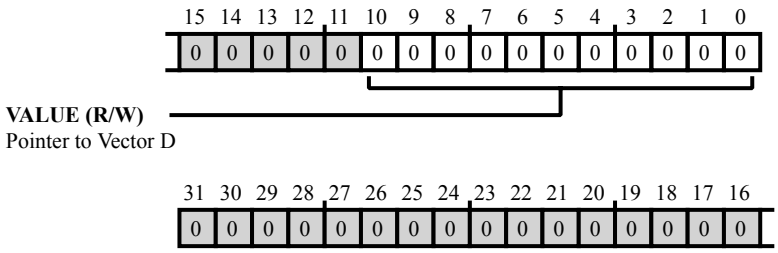


Figure 42-9: PKA_DPTR Register Diagram

Table 42-31: PKA_DPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Pointer to Vector D. The PKA_DPTR.VALUE bit field is the location of Vector D within the PKA RAM. Vectors are identified through the location of their least-significant 32-bit word. Note that bit [0] must be zero to ensure that the vector starts at an 8-byte boundary.

PKA Function

The PKA_FUNC register contains the control bits to start basic PKCP as well as complex sequencer operations. The PKA_FUNC.RUN bit can be used to poll for the completion of the operation. Modifying bits [11:0] is made impossible during the execution of a basic PKCP operation.

During the execution of Sequencer controlled complex operations, this register is modified - the PKA_FUNC.RUN and PKA_FUNC.STALLRSLT bits are set to zero at the conclusion, but other bits are undefined.

Continuously reading this register to poll the PKA_FUNC.RUN bit is NOT allowed when executing complex sequencer operations (the sequencer cannot access the PKCP when this is done).

Leave at least one SCLK cycle between poll operations.

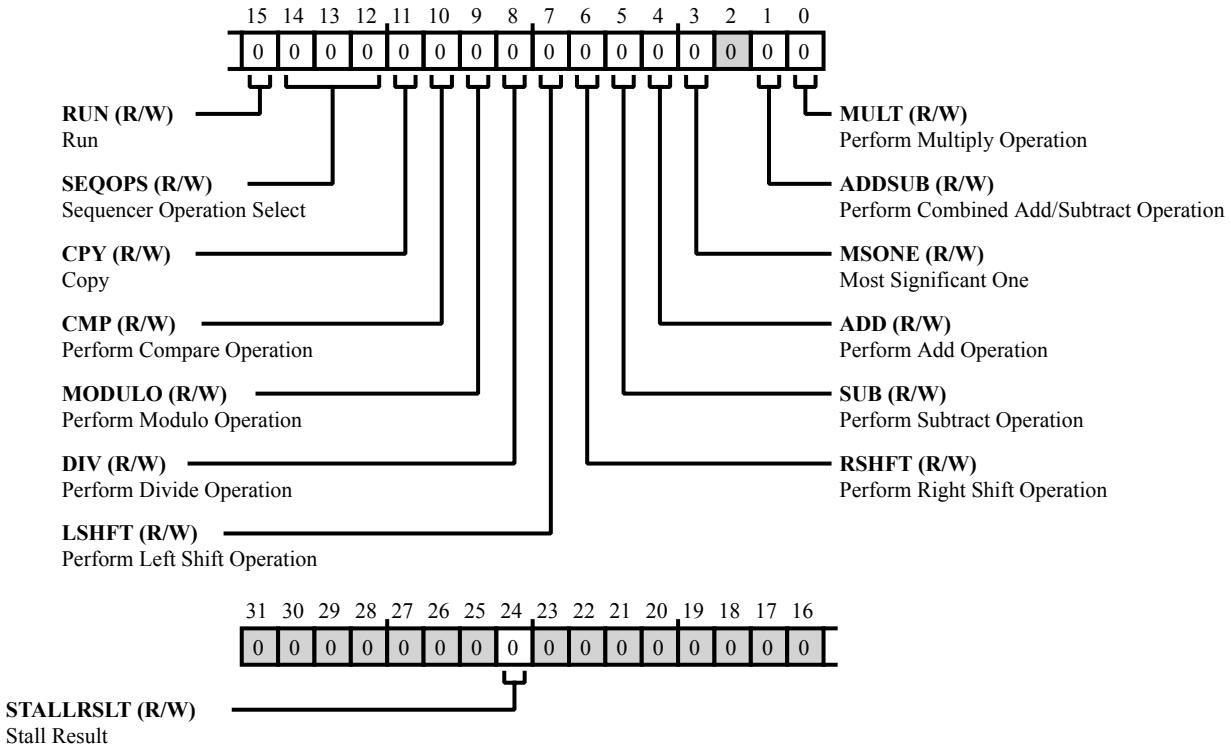


Figure 42-10: PKA_FUNC Register Diagram

Table 42-32: PKA_FUNC Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/W)	STALLRSLT	Stall Result. When the PKA_FUNC.STALLRSLT bit is set, updating the PKA_COMPARE, PKA_RESULTMSW and PKA_DIVMSW registers, as well as resetting the Run bit, is stalled beyond the point that a running operation is actually finished. Use this to allow software enough time to read results from a previous operation when the newly started operation is known to take only a short amount of time. If a result is waiting, the result registers is updated and the Run bit is reset in the clock cycle following writing the Stall Result bit back to 0. The Stall result function may only be used for basic PKCP operations.
15 (R/W)	RUN	Run. Set the PKA_FUNC.RUN bit to instruct the PKA module to begin processing the basic PKCP or complex Sequencer operation. This bit is reset low automatically when the operation is complete. The complement of this bit is output as the pkaint1 interrupt. After a reset, the Run bit is always set to 1b but the first Sequencer firmware instruction sets this bit to 0 immediately after the hardware reset is released. A few clock cycles are needed before the first instruction is executed and the Run bit state has been propagated.
14:12 (R/W)	SEQOPS	Sequencer Operation Select. The PKA_FUNC.SEQOPS bit field select the complex Sequencer operation to perform.
		0 None
		1 ExpMod-CRT
		2 ExpMod-ACT4
		3 ECC-ADD
		4 ExpMod-ACT2
		5 ECC-MUL
		6 ExpMod-variable
		7 ModInv
11 (R/W)	CPY	Copy. Setting the PKA_FUNC.CPY bit performs the copy operation. For more information, see the "PKCP Vector Operations" section.
10 (R/W)	CMP	Perform Compare Operation. Setting the PKA_FUNC.CMP bit performs the compare operation. For more information, see the "PKCP Vector Operations" section.

Table 42-32: PKA_FUNC Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
9 (R/W)	MODULO	Perform Modulo Operation. Setting the PKA_FUNC.MODULO bit performs the modulo operation. For more information, see the "PKCP Vector Operations" section.
8 (R/W)	DIV	Perform Divide Operation. Setting the PKA_FUNC.DIV bit performs the divide operation. For more information, see the "PKCP Vector Operations" section.
7 (R/W)	LSHFT	Perform Left Shift Operation. Setting the PKA_FUNC.LSHFT bit performs the Left shift operation. For more information, see the "PKCP Vector Operations" section.
6 (R/W)	RSHFT	Perform Right Shift Operation. Setting the PKA_FUNC.RSHFT bit performs the right shift operation. For more information, see the "PKCP Vector Operations" section.
5 (R/W)	SUB	Perform Subtract Operation. Setting the PKA_FUNC.SUB bit performs the subtract operation. For more information, see the "PKCP Vector Operations" section.
4 (R/W)	ADD	Perform Add Operation. Setting the PKA_FUNC.ADD bit performs the add operation. For more information, see the "PKCP Vector Operations" section.
3 (R/W)	MSONE	Most Significant One. Setting the PKA_FUNC.MSONE bit loads the location of the Most Significant one bit within the result word indicated in the PKA_RESULTMSW register into bits [4:0] of the PKA_DIVMSW register can only be used with basic PKCP operations, except for Divide, Modulo and Compare.
1 (R/W)	ADDSUB	Perform Combined Add/Subtract Operation. Setting the PKA_FUNC.ADDSUB bit performs the combined Add/Subtract operation. For more information, see the "PKCP Vector Operations" section.
0 (R/W)	MULT	Perform Multiply Operation. Setting the PKA_FUNC.MULT bit performs the multiply operation. For more information, see the "PKCP Vector Operations" section.

Start of PKA RAM space

The PKA_RAM register provides the starting location of the RAM space to hold the input, output and other vectors.

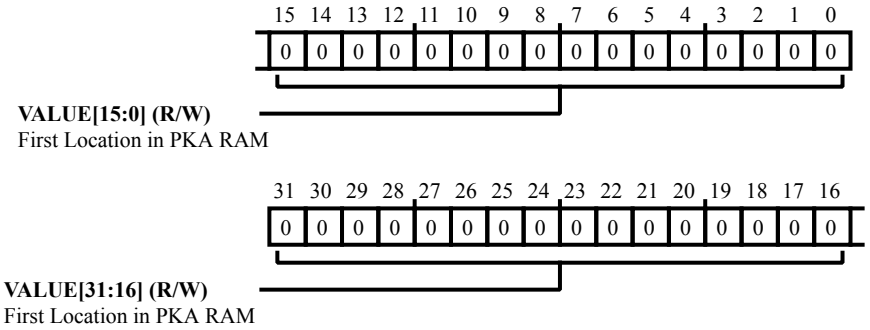


Figure 42-11: PKA_RAM Register Diagram

Table 42-33: PKA_RAM Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	First Location in PKA RAM. The PKA_RAM.VALUE bit field provides the starting location of the RAM space to hold the input, output and other vectors.

PKA Most-Significant-Word of Result Vector

The PKA_RESULTMSW register indicates the (word) address in the PKA RAM where the most significant non-zero 32-bit word of the result is stored and should be ignored for modulo operations. For basic PKCP operations, the PKA_RESULTMSW register is updated when the PKA_FUNC.RUN bit is reset at the end of the operation.

For the complex sequencer controlled operations, updating the final value matching the actual result is done near the end of the operation. Note that the result is only meaningful if no errors are detected and that for ECC operations, the PKA_DIVMSW register provides information for the x-coordinate of the result point only.

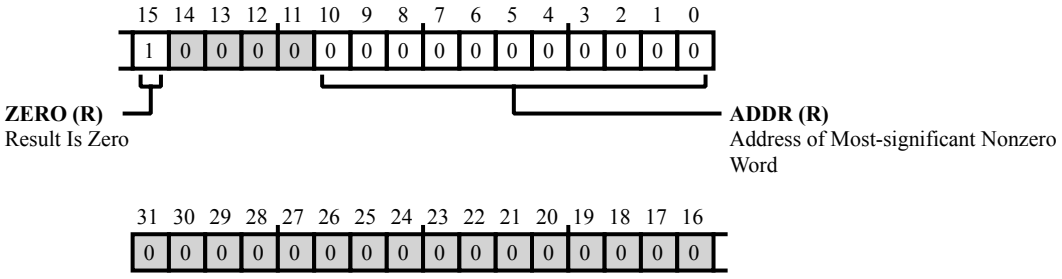


Figure 42-12: PKA_RESULTMSW Register Diagram

Table 42-34: PKA_RESULTMSW Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/NW)	ZERO	Result Is Zero. The PKA_RESULTMSW.ZERO bit indicates the result vector is all zeros, ignore the address returned in bits [10:0].
10:0 (R/NW)	ADDR	Address of Most-significant Nonzero Word. The PKA_RESULTMSW.ADDR bit is the address of the most significant non-zero 32-bit word of the result vector in PKA RAM.

PKA Bit Shift Value

For basic PKCP operations, modifying the contents of the PKA_SHIFT register is made impossible while the operation is being performed. For the ExpMod-variable and ExpMod-CRT operations, the PKA_SHIFT register is used to indicate the number of odd powers to use (directly as a value in the range 1-16). For the ModInv and ECC operations, this register is used to hold a completion code.

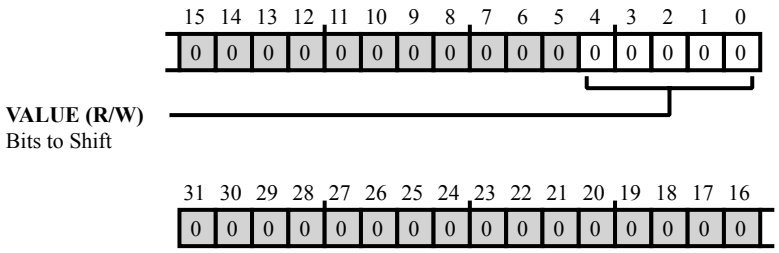


Figure 42-13: PKA_SHIFT Register Diagram

Table 42-35: PKA_SHIFT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
4:0 (R/W)	VALUE	Bits to Shift. The PKA_SHIFT.VALUE bit field is the number of bits to shift the input vector (in the range 0-31) during a Rshift or Lshift operation.

43 Public Key Interrupt Controller (PKIC)

The Public Key Accelerator (PKA) and the True Random Number Generator (TRNG) share a common interrupt controller, the Public Key Interrupt Controller (PKIC). The host processor configures the PKIC to generate interrupts when certain operations are complete, or interrupts are caused by errors.

PKIC Functional Description

The main purpose and function of the PKIC is to capture the interrupts from different sources, either from the PKA or the TRNG and combine them to one interrupt output. The interrupt controller is managed using the following register groups:

- Control for polarity, edge, and level detection and enabling of individual interrupts
- Acknowledgment (to clear edge detected interrupts)
- Status:
  - A raw source status register after edge detection, if edge selected.
  - A status register after masking with the interrupt enable control bits.

ADSP-SC59x PKIC Register List

The Public Key Processor Interrupt Controller (PKIC) provides security-related features. A set of registers governs PKIC operations. For more information on PKIC functionality, see the PKIC register descriptions.

Table 43-1: ADSP-SC59x PKIC Register List

Name	Description
PKIC_ACK	Acknowledge Register
PKIC_EN_CLR	Enable Clear Register
PKIC_EN_CTL	Enable Control Register
PKIC_EN_SET	Enable Set Register
PKIC_EN_STAT	Enabled Status Register
PKIC_POL_CTL	Polarity Control Register

Table 43-1: ADSP-SC59x PKIC Register List (Continued)

Name	Description
PKIC_RAW_STAT	Raw Status Register
PKIC_TYPE_CTL	Type Control Register

ADSP-SC59x PKIC Interrupt List

Table 43-2: ADSP-SC59x PKIC Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
160	PKIC0_IRQ	PKIC0 Interrupt	Level	

PKIC Programming Model

The following sections provide information on how to program the PKIC.

Enabling/Disabling and Status

The PKIC_EN_STAT register provides the mask to which interrupt source is enabled. There are two status registers, PKIC_RAW_STAT and PKIC_EN_STAT. They allow the host processor to read the status of the interrupt source before and after the mask is applied.

Level or Edge

All of the interrupt sources are level or edge events. The PKIC_TYPE_CTL register configures each interrupt to either level or edge. The PKIC_POL_CTL register controls the polarity of the signal.

These interrupts are latched at both status registers in case edge detection is selected. The edge detectors are reset by clearing the interrupts using the PKIC_ACK registers.

PKIC Programming Concepts

The following concepts help with proper programming for the PKIC module.

Interrupt Handling

When an interrupt is triggered, the handler must first examine this module, the PKIC to determine what triggered the interrupt. By reading the PKIC_EN_STAT, the bits that are set are the pending interrupts of the ones that were not masked. After determining the source of the interrupt, the appropriate action must be taken to service the interrupt from the corresponding module, the PKA, or the TRNG. After handling the interrupt in a particular module, the corresponding interrupt must be acknowledged and cleared in the PKIC to allow further interrupts.

While handling an interrupt, any events that would cause another interrupt would happen without triggering another interrupt.

Overlapping Registers

There are two sets of overlapping registers in the PKIC. The PKIC_EN_STAT and PKIC_ACK registers share one address. If read, the register tells which enabled interrupts are pending. If written to (W1C), the interrupt is acknowledged and cleared. The PKIC_RAW_STAT and PKIC_EN_SET registers are another pair that share the address. When read, the register tells which interrupts are pending and if written to will enabled certain interrupts. This register cannot be used to disable any interrupts.

ADSP-SC59x PKIC Register Descriptions

Public Key Processor Interrupt Controller (PKIC) contains the following registers.

Table 43-3: ADSP-SC59x PKIC Register List

Name	Description
PKIC_ACK	Acknowledge Register
PKIC_EN_CLR	Enable Clear Register
PKIC_EN_CTL	Enable Control Register
PKIC_EN_SET	Enable Set Register
PKIC_EN_STAT	Enabled Status Register
PKIC_POL_CTL	Polarity Control Register
PKIC_RAW_STAT	Raw Status Register
PKIC_TYPE_CTL	Type Control Register

Acknowledge Register

The PKIC_ACK register is used to acknowledge the interrupt and clear the corresponding interrupt bit in the status register.

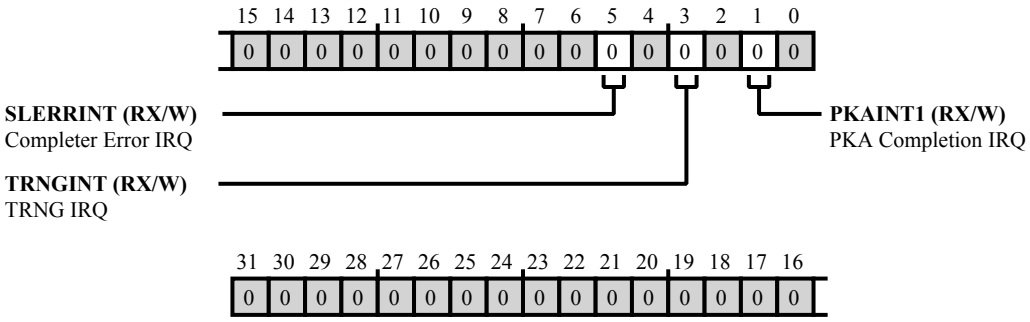


Figure 43-1: PKIC_ACK Register Diagram

Table 43-4: PKIC_ACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (RX/W)	SLERRINT	Completer Error IRQ. The PKIC_ACK.SLERRINT bit is the acknowledge bit for the completer error interrupt. When set =1 the PKIC_ACK.SLERRINT bit acknowledges the interrupt signal and clears the status bit (and is cleared automatically).	
		0	Do not acknowledge interrupt and clear status bit
		1	Acknowledge interrupt and clear status bit
3 (RX/W)	TRNGINT	TRNG IRQ. The PKIC_ACK.TRNGINT bit is the acknowledge bit for the TRNG interrupt. When set =1 the PKIC_ACK.TRNGINT bit acknowledges the interrupt signal and clears the status bit (and is cleared automatically).	
		0	Do not acknowledge interrupt and clear status bit
		1	Acknowledge interrupt and clear status bit
1 (RX/W)	PKAINT1	PKA Completion IRQ. The PKIC_ACK.PKAINT1 bit is the acknowledge bit for the PKA completion interrupt. When set =1 the PKIC_ACK.PKAINT1 bit acknowledges the interrupt signal and clears the status bit (and is cleared automatically).	

Enable Clear Register

The PKIC_EN_CLR register allows the user to disable certain interrupts without enabling others. The disabled interrupts are also reflected in PKIC_EN_CTL register.

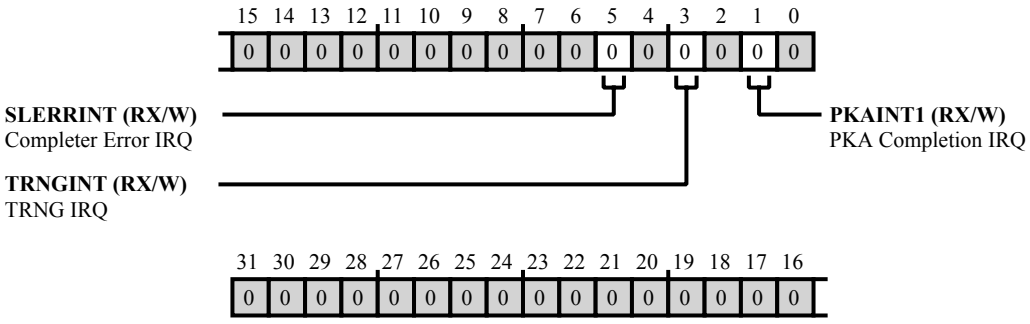


Figure 43-2: PKIC_EN_CLR Register Diagram

Table 43-5: PKIC_EN_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (RX/W)	SLERRINT	Completer Error IRQ. The PKIC_EN_CLR.SLERRINT bit is the individual disable for the completer error interrupt. When set =1 this bit clears/resets the corresponding bit in the PKIC_EN_CTL register to 0 (disables the interrupt). This bit is cleared automatically.	
		0	No action
		1	Clear/reset corresponding CTL bit
3 (RX/W)	TRNGINT	TRNG IRQ. The PKIC_EN_CLR.TRNGINT bit is the individual disable for the TRNG interrupt. When set =1 this bit clears/resets the corresponding bit in the PKIC_EN_CTL register to 0 (disables the interrupt). This bit is cleared automatically. 0= no effect.	
		0	No action
		1	Clear/reset corresponding CTL bit
1 (RX/W)	PKAINT1	PKA Completion IRQ. The PKIC_EN_CLR.PKAINT1 bit is the individual disable for the PKA Completion interrupt. When set =1 this bit clears/resets the corresponding bit in the PKIC_EN_CTL register to 0 (disables the interrupt). This bit is cleared automatically. 0= no effect.	
		0	No action
		1	Clear/reset corresponding CTL bit

Enable Control Register

The PKIC_EN_CTL register provides individual enable bits for the interrupt sources.

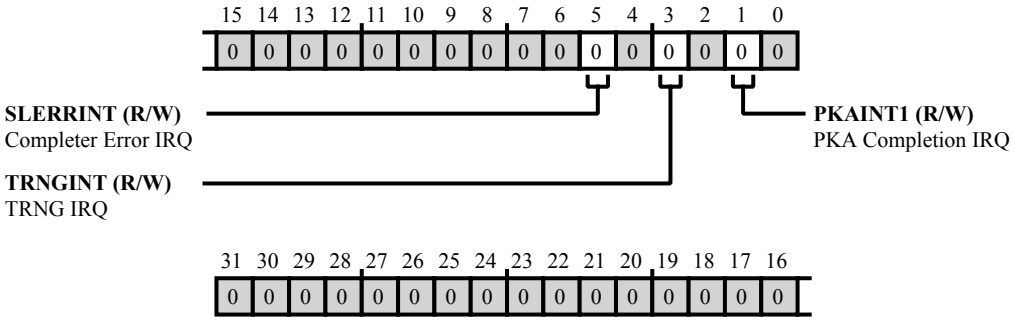


Figure 43-3: PKIC_EN_CTL Register Diagram

Table 43-6: PKIC_EN_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	SLERRINT	Completer Error IRQ. The PKIC_EN_CTL.SLERRINT bit enables control for the completer error interrupt.	
		0	Disable interrupt
		1	Enable interrupt
3 (R/W)	TRNGINT	TRNG IRQ. The PKIC_EN_CTL.TRNGINT bit enables control for the TRNG interrupt.	
		0	Disable interrupt
		1	Enable interrupt
1 (R/W)	PKAINT1	PKA Completion IRQ. The PKIC_EN_CTL.PKAINT1 bit enables control for the PKA completion interrupt.	
		0	Disable interrupt
		1	Enable interrupt

Enable Set Register

The PKIC_EN_SET register allows the user to only set certain interrupt sources without disabling any others. The enabled interrupts are reflected in PKIC_EN_CTL register.

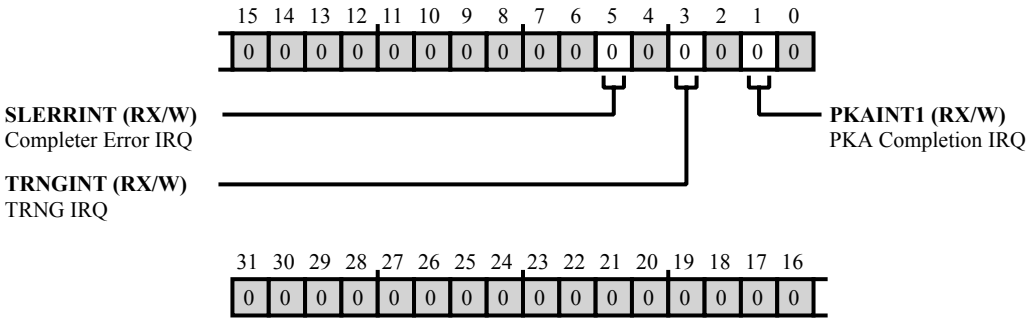


Figure 43-4: PKIC_EN_SET Register Diagram

Table 43-7: PKIC_EN_SET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (RX/W)	SLERRINT	Completer Error IRQ. The PKIC_EN_SET.SLERRINT bit is the individual enable for the completer error interrupt. If =1, sets the corresponding bit in the PKIC_EN_CTL register to 1 (enables interrupt). This bit is cleared automatically. 0=no effect	
		0	No effect
		1	Enable interrupt
3 (RX/W)	TRNGINT	TRNG IRQ. The PKIC_EN_SET.TRNGINT bit is the individual enable for the TRNG interrupt. If =1, sets the corresponding bit in the PKIC_EN_CTL register to 1 (enables interrupt). This bit is cleared automatically.	
		0	No effect
		1	Enable interrupt
1 (RX/W)	PKAINT1	PKA Completion IRQ. The PKIC_EN_SET.PKAINT1 bit is the individual enable for the PKA Completion interrupt. If =1, sets the corresponding bit in the PKIC_EN_CTL register to 1 (enables interrupt). This bit is cleared automatically.	
		0	No effect
		1	Enable interrupt

Enabled Status Register

The PKIC_EN_STAT register is used to tell the status of the interrupts after the gating with the PKIC_EN_CTL register.

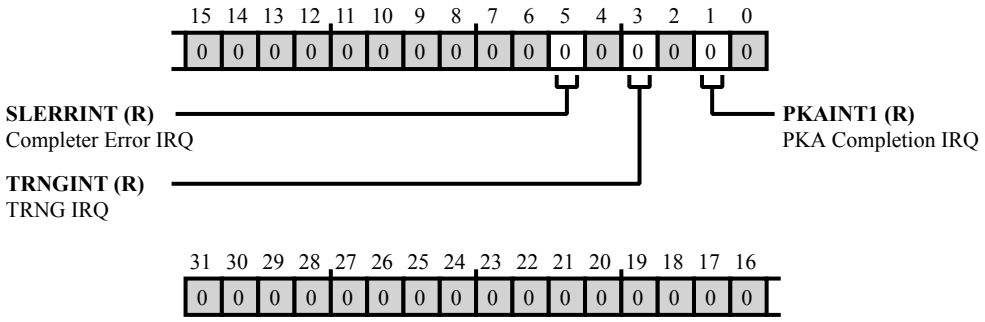


Figure 43-5: PKIC_EN_STAT Register Diagram

Table 43-8: PKIC_EN_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	SLERRINT	Completer Error IRQ. The PKIC_EN_STAT.SLERRINT bit provides the status of the completer error interrupt (after masking from the PKIC_EN_CTL register).	
		0	Interrupt is inactive
		1	Interrupt is pending
3 (R/NW)	TRNGINT	TRNG IRQ. The PKIC_EN_STAT.TRNGINT bit provides the status of the TRNG interrupt (after masking from the PKIC_EN_CTL register).	
		0	Interrupt is inactive
		1	Interrupt is pending
1 (R/NW)	PKAINT1	PKA Completion IRQ. The PKIC_EN_STAT.PKAINT1 bit provides the status of the PKA Completion interrupt (after masking from the PKIC_EN_CTL register).	
		0	Interrupt is inactive
		1	Interrupt is pending

Polarity Control Register

The PKIC_POL_CTL register is used to configure the signal polarity for each individual interrupt. During the initialization phase of the PKA the host processor must set each interrupt in this register to (high level/rising edge or low level/falling edge).

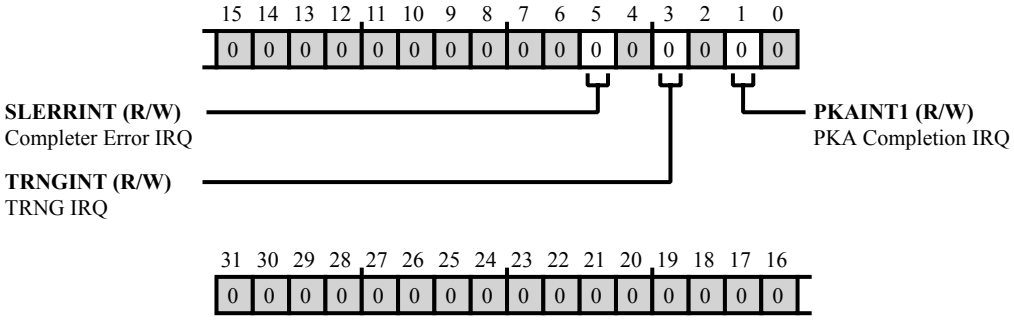


Figure 43-6: PKIC_POL_CTL Register Diagram

Table 43-9: PKIC_POL_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	SLERRINT	Completer Error IRQ. The PKIC_POL_CTL.SLERRINT bit provides polarity control for the completer error interrupt.	
		0	Low level/falling edge
		1	High level/rising edge
3 (R/W)	TRNGINT	TRNG IRQ. The PKIC_POL_CTL.TRNGINT bit provides polarity control for the TRNG interrupt.	
		0	Low level/falling edge
		1	High level/rising edge
1 (R/W)	PKAINT1	PKA Completion IRQ. The PKIC_POL_CTL.PKAINT1 bit provides polarity control for PKA completion interrupt.	
		0	Low level/falling edge
		1	High level/rising edge

Raw Status Register

The PKIC_RAW_STAT register reflects the status of the individual interrupts before masking with the PKIC_EN_CTL register.

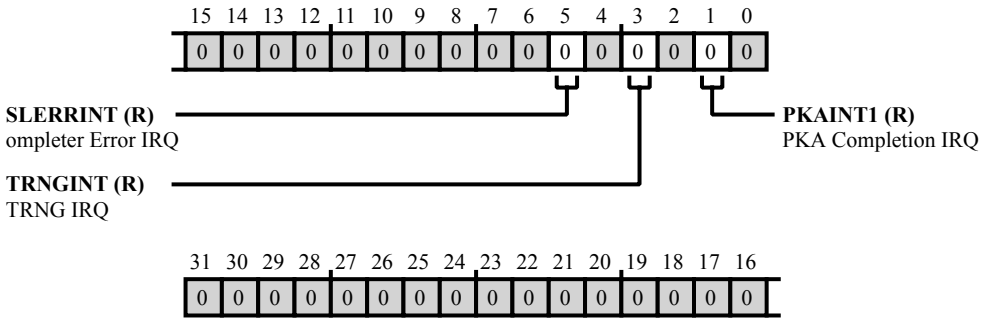


Figure 43-7: PKIC_RAW_STAT Register Diagram

Table 43-10: PKIC_RAW_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/NW)	SLERRINT	ompleter Error IRQ. The PKIC_RAW_STAT.SLERRINT bit provides the raw status of the completer error interrupt.	
		0	Inactive interrupt
		1	Pending interrupt
3 (R/NW)	TRNGINT	TRNG IRQ. The PKIC_RAW_STAT.TRNGINT bit provides the raw status of the TRNG interrupt where 1=pending and 0=inactive.	
		0	Inactive interrupt
		1	Pending interrupt
1 (R/NW)	PKAINT1	PKA Completion IRQ. The PKIC_RAW_STAT.PKAINT1 bit provides raw status of the PKA completion interrupt where 1=pending and 0=inactive.	
		0	Inactive interrupt
		1	Pending interrupt

Type Control Register

The PKIC_TYPE_CTL register is used to configure the signal type for each individual interrupt. During the initialization phase of the PKA the host processor must set each interrupt in this register to level or edge.

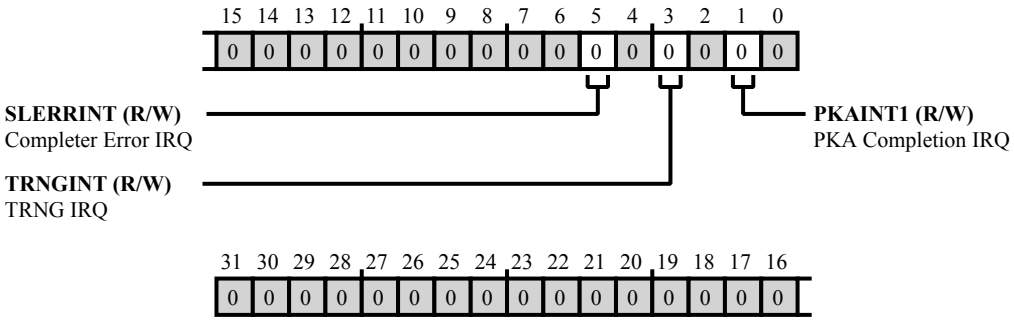


Figure 43-8: PKIC_TYPE_CTL Register Diagram

Table 43-11: PKIC_TYPE_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	SLERRINT	Completer Error IRQ. The PKIC_TYPE_CTL.SLERRINT bit provides signal type control for the completer error interrupt.	
		0	Level
		1	Edge
3 (R/W)	TRNGINT	TRNG IRQ. The PKIC_TYPE_CTL.TRNGINT bit provides signal type control for the TRNG interrupt.	
		0	Level
		1	Edge
1 (R/W)	PKAINT1	PKA Completion IRQ. The PKIC_TYPE_CTL.PKAINT1 bit provides signal type control for the PKA completion interrupt.	
		0	Level
		1	Edge

44 True Random Number Generator (TRNG)

The TRNG engine provides a true, non-deterministic, noise source for generating keys, Initialization Vectors (IVs), and other random number requirements. Other non-cryptographic purposes include statistical sampling, retry timers for communications protocols and noise generation.

TRNG Features

The TRNG features include:

- Hardware-based non-deterministic random number generator
- ANSI X9.31 postprocessing (depending on processor)
- Redundant 'fail-safe' design with self-test circuits
- Reliable shot noise oscillator implementation with auto-tuning
- Debug output to allow monitoring of internal operation
- Alarm count overflow and auto-tuning error interrupts
- Buffer to allow generation of large blocks of random data in the background

TRNG Functional Description

The following sections provide details on the function of the TRNG module.

ADSP-SC59x TRNG Register List

The True Random Number Generator (TRNG) provides random number generation, intended mainly for security-related applications. A set of registers governs TRNG operations. For more information on TRNG functionality, see the TRNG register descriptions.

Table 44-1: ADSP-SC59x TRNG Register List

Name	Description
TRNG_ALMCNT	TRNG Alarm Counter Register

Table 44-1: ADSP-SC59x TRNG Register List (Continued)

Name	Description
TRNG_ALMMSK	TRNG Alarm Mask Register
TRNG_ALMSTP	TRNG Alarm Stop Register
TRNG_BLKCNT	TRNG Block Count Register
TRNG_CFG	TRNG Configuration Register
TRNG_CNT	Counter Register
TRNG_CTL	TRNG Control Register
TRNG_FRODETUNE	TRNG FRO De-tune Register
TRNG_FROEN	TRNG FRO Enable Register
TRNG_INPUT[n]	TRNG Input Registers
TRNG_INTACK	TRNG Interrupt Acknowledge Register
TRNG_KEY[n]	Post-Process Key Registers
TRNG_LFSR_H	TRNG LFSR Access Register
TRNG_LFSR_L	TRNG LFSR Access Register
TRNG_LFSR_M	TRNG LFSR Access Register
TRNG_MONOBITCNT	TRNG Monobit Test Result Register
TRNG_OUTPUT[n]	TRNG Output Registers
TRNG_POKER[n]	TRNG Poker Test Result Registers
TRNG_RUNCNT	TRNG Run Count Registers
TRNG_RUN[n]	TRNG Run Test State and Result Registers
TRNG_STAT	TRNG Status Register
TRNG_TEST	TRNG Test Register
TRNG_V[n]	TRNG Post-Process "V" Value Registers

Random Number Generation

The random numbers that the TRNG generates are produced by sampling Free Running Oscillators (FRO). The (TRNG_CTL.STARTUPCYC) bit field along with TRNG_CFG.MINREFCYC (minimum refill cycles) and TRNG_CFG.MAXREFCYC (maximum refill cycles) bit fields determine the number of samples taken to generate the first random value and subsequent random values.

1. After enabling the TRNG (TRNG_CTL.TRNGEN bit =1), a number of FRO output samples defined by the TRNG_CTL.STARTUPCYC bit field are gathered in the main linear-feedback shift register (LFSR) before taking a snapshot of the LFSR and storing that snapshot in the random data buffer (after optional post-processing).

2. After taking a snapshot of the LFSR, a number of FRO output samples defined by the TRNG_CFG.MINREFCYC bit field, are gathered in the main LFSR. If the random data buffer is full, sample-taking continues until the number of samples (counting from the snapshot) matches the number of samples defined by the TRNG_CFG.MAXREFCYC bit field. At that point, the TRNG switches off the FROs and powers down.
3. If, after the TRNG_CFG.MINREFCYC sampling period, the random data buffer is not completely filled, a new snapshot of the LFSR is taken and stored in the random data buffer (after optional postprocessing). Control branches back to point step 2 above and a new TRNG_CFG.MINREFCYC sample period starts.

Locking Detection and Prevention

Lock detection in functional mode uses the sampled outputs of the individual FROs. A FRO alarm event is declared when a repeating pattern (of up to four sample lengths) is detected continuously for the number of samples defined in the alarm threshold TRNG_ALMCNT.ALMTHRESH bit field. The alarm event is logged by setting the bit that corresponds to the FRO that caused the alarm in the TRNG_ALMMSK register. If that bit was already set, the corresponding bit in the TRNG_ALMSTP register is set. The FRO is switched off to prevent further alarm events from that FRO. If the TRNG_ALMMSK register bit was not yet set, the FRO is restarted automatically in an attempt to break locking.

The shutdown count field in the alarm count TRNG_ALMCNT.SHDNCNT register monitors the number of FROs switched off. (It counts the number of 1 bits in the TRNG_ALMSTP register.) The shutdown threshold field (TRNG_ALMCNT.SHDNTHRESH) can be configured to generate the shutdown overflow interrupt (TRNG_STAT.SHDNOVR). When the shutdown count in the TRNG_ALMCNT.SHDNCNT bit field exceeds the shutdown threshold in the TRNG_ALMCNT.SHDNTHRESH bit field, the shutdown overflow bit (TRNG_STAT.SHDNOVR) is set to 1 (which can be used to generate an interrupt).

Software can use two strategies for the TRNG operation:

- ***Monitored Operation***. Software checks the TRNG_ALMMSK register at regular intervals (on the order of seconds). If a bit is set in that register, then the program must also check the TRNG_ALMSTP register to determine if a FRO was shut down due to multiple alarm events. If no FROs are shut down, the program clears the TRNG_ALMMSK register to remove the incidental alarm events. If one or more FROs are shut down, the host processor can modify the delay selection of those FROs using the TRNG_FRODETUNE register to prevent further locking. For this type of operation, the shutdown threshold is normally set to a low value (two, for example). The shutdown overflow interrupt can then be used to signal abnormal operation conditions or the breakdown of FROs.
- ***Unmonitored Operation.*** Software sets the shutdown threshold to the acceptable number of FROs to be shut down before taking corrective actions. It then uses the shutdown overflow interrupt to initiate corrective actions (clearing the TRNG_ALMMSK and TRNG_ALMSTP registers, toggling bits in the TRNG_FRODETUNE register). The software must monitor the time interval between these interrupts. If they occur too often (for example, within a minute after each other), this frequency indicates abnormal operating conditions or the breakdown of FROs.

Run Testing

Run Test

The TRNG block counts the number of consecutive zeros and ones (runs) in the data stream shifted into the main LFSR. The run length and bit value is then used to increment a specific bucket counter for these values. After 20,000 bits, the bucket counters must be within specified limits for this test to pass. If not, a run fail interrupt (RUNFAIL) is generated.

Table 44-2: Allowable Limits on Runs of 0's and 1's

Run Count	Bit Value	Min (inclusive)	Max (inclusive)
1	0	2267	2733
	1	2267	2733
2	0	1079	1421
	1	1079	1421
3	0	502	748
	1	502	748
4	0	233	402
	1	233	402
5	0	90	223
	1	90	223
6 and up	0	90	233
	1	90	233

Long Run Test

The long run test fails immediately when a run longer than 33 bits is found and a long run fail (`TRNG_STAT.LRUNFAIL`) interrupt is generated.

Noise Source Test

A noise source failure is declared when a run of 48 or more identical bits is found and a noise fail interrupt (`TRNG_STAT.NOISEFAIL`) is generated.

The status and counts are stored in the `TRNG_RUNCNT` and `TRNG_RUN[n]` registers. Unless otherwise indicated, all counters and state bits in these registers are reset when writing a 1 to either the monobit fail acknowledge (`TRNG_INTACK.MBITFAIL`), the run fail acknowledge (`TRNG_INTACK.RUNFAIL`), or the poker fail acknowledge (`TRNG_INTACK.PKRFAIL`) bits.

Monobit Testing

The TRNG block performs the monobit test on blocks of 20,000 bits (in parallel with the run test and poker test). It monitors the number of zeros and ones in the data stream shifted into the main LFSR. At the start of the block,

the counter is initialized to 10,000. Each 1 value increments the counter, each 0 value decrements the counter. After 20,000 bits, the counter value must be within 9310–10690 (inclusive) for this test to pass. If not, a monobit fail interrupt (TRNG_STAT.MBITFAIL) is generated. The AIS-31 standard (test T1, ref 0) specifies this run time testing of the TRNG and the parameters.

NOTE: The actual limits stated here are different than the limits stated in the AIS-31 standard due to the implementation. The circuitry in the TRNG uses an up-down counter while the standard just evaluates the sum of the 20,000 bits.

When the continue poker (TRNG_TEST.CONTPKR) bit is set to 1, the test is not stopped after 20,000 bits. The counter keeps incrementing and decrementing (protected against overflow and underflow). But, no actual limit checking happens. The offset from starting value 10,000 indicates the balance of 0 and 1 bits that were checked since the start of the continuous test. The offset is twice the number of missing or extra 1 bits. (An extra 1 bit adds an increment operation and removes one decrement operation. So, having 10,001 1 bits in the block gives a counter value of 10,002.)

Poker Testing

The poker test is run in parallel with the monobit test and run test. Counters in the TRNG_POKER[n] registers are used to count the occurrences of one specific 4-bit value in the data stream fed into the main LFSR. All of the counters are decremented by one every 64 data bits and reset to their start value every 20,000 bits. All counters start at a value of –1 and are decremented 312 times during the 20,000-bit test run.

Each 8-bit counter holds a two's complement value and does not overflow past the range of –128 to +127. At the end of the 20,000-bits block, the values of the counters that contain a single 1 bit appended at the least significant bit side are individually squared and then added. The poker test fails with a poker fail (PKRFAIL) interrupt when:

- the resulting sum (accumulated in the TRNG_MONOBITCNT register) is outside the range 1288 – 71750 (inclusive), or
- one of the counters tries to increment or decrement outside its limit range

NOTE: The poker test fails when the 4-bit values of the data stream are distributed too evenly (with eight counters having incremented 312 times and the others incremented 313 times). This failure is intentional. The minimum mean deviation from the expected value of 312.5 is 4.5. Failure at counter overflow is not an official part of the poker test as specified in the AIS-31 standard (ref 0). It can be shown that the maximum deviation for one counter's value from the mean value of 312.5 (without the poker test failing) is 129.5. Since this deviation is more than 40% of the mean value, it indicates that something is wrong. Here, the counter overflow failure is combined with the official poker test failure.

When the continue poker (TRNG_TEST.CONTPKR) bit is set to 1, the test does not stop after 20,000 bits. The counters keep incrementing and decrementing (the latter every 64 bits). A PKRFAIL interrupt is generated when one of the counters tries to increment or decrement outside its limit range.

Data for Tests

For the monobit test, run test and poker test circuits self-test, the test data written to the TRNG_INPUT[n]0 register is used for the monobit test and run test bit-by-bit. It executes from bit 31 down to bit 0. For the poker test, the written test data is used in eight blocks of 4 bits, starting from bits 31:28 down to bits 3:0.

When the TRNG_TEST.CONTPKR bit =0 during run test and poker test circuits self-tests, the state of the poker and runs test status registers is frozen after all calculations are made and after inputting 20,000 test bits. This state allows time to read the test results. These calculations take around 20 clock cycles to complete. The status registers are reset to their starting states when the first word for the next test block of test bits is written to the TRNG_INPUT[n]0 register. Then, the contents of this word are processed.

X9.31 Postprocessing

Postprocessing is available on some parts using the TRNG. If available, the postprocessing block is situated after the main LFSRs and before the output buffer that stores the random numbers for consumption. The online test logic for monobit, run, and poker testing is before the postprocessor block. The bits used for testing are from the main LFSR.

The postprocessor is based on the ANSI X9.31 specification using 3-Key 3DES cipher algorithm. It does not provide any more entropy than is what is already achieved from sampling the Free Running Oscillators (FROs). The use of the postprocessor only helps with applications requiring the use of an X9.31 compliant RNG.

The following section is an example of postprocessing.

The variables include:

- DT is a 64-bit date/time vector. This value is the input coming from random bits from the main LFSR.
- I is an intermediate value, a 64-bit value
- R is the final result, a 64-bit value
- V is a 64-bit seed value that is to be kept secret
- K is the 3-key for 3DES, each being 64-bits

The postprocessing uses the steps:

1. The intermediate value I is calculated: I = $3DES_{EDE}$(K, DT) with 3DES.
2. The result R is calculated: R = $3DES_{EDE}$(K, I XOR V).
3. Finally, V is updated: V = $3DES_{EDE}$(K, R XOR I).

TRNG Block Diagram

The ***System Block Diagram of the TRNG*** diagram shows the system block for the TRNG. The system includes:

- Free Running Oscillators (FROs) that are the source of sampled bits
- A post-processing unit (processor dependent) for standards compliance

- Test circuitry to detect non-randomness due to failures in the system

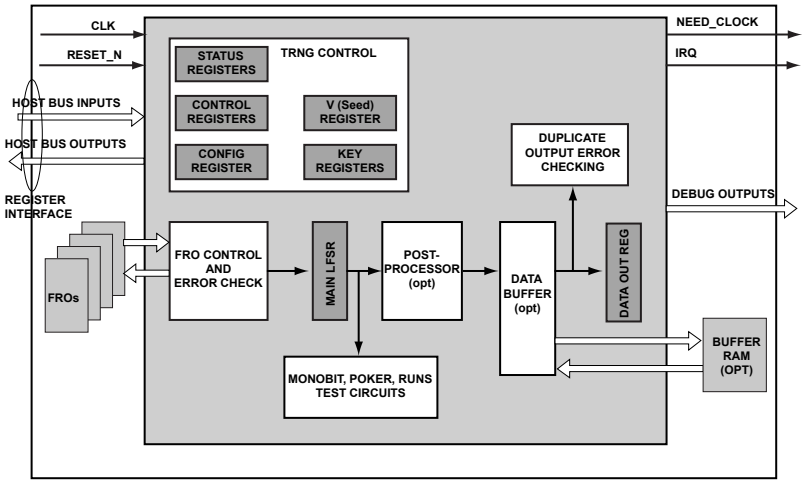


Figure 44-1: System Block Diagram of the TRNG

TRNG Architectural Concepts

The random numbers are accessible to the host processor in four 32-bit registers allowing a single burst read of a 128-bit random number. Acknowledging the data ready (interrupt) state causes the TRNG to move a new value (when available in the data buffer) to the TRNG output register. The TRNG always tries to keep the data buffer completely full. Pulling out data starts the regeneration of a new number by:

- Enabling the FROs
- Capturing their outputs in the LFSR, and
- Cryptographically postprocessing the values from the LFSR

The process produces new random values to replenish the buffer.

The major functional blocks of the TRNG module are:

- The actual TRNG core with control and test circuits, optional post-processor, and optional data buffer control logic
- Free Running Oscillators (FROs) instantiated outside the TRNG core
- A 128-byte buffer RAM instantiated outside the TRNG core

In the TRNG core, the true entropy source uses FROs as the basic building block. The accumulation of timing jitter, caused (for the largest part) by shot noise, creates uncertainty intervals for the output transitions of each FRO. Sampling within an uncertainty interval generates a single bit of entropy, which is ‘accumulated’ in a LFSR. As the uncertainty interval is very narrow compared to the cycle time of a FRO, the mean amount of entropy generated per sample is small (less than 1/100 bit per sample). To increase the entropy generation rate, multiple FROs are used in parallel.

The FROs are asynchronous to one another and asynchronous to the sampling clock to make their behavior truly non-deterministic.

The output signals of the FROs are sampled at regular intervals (in general, at the TRNG core module clock frequency). The samples feed into an error detection circuit that checks for repeating patterns coming out of a FRO. If a repeating pattern persists for a configurable number of samples, the FRO is suspect of having synchronized to (a harmonic of) the sampling interval. This activity drastically reduces the amount of entropy generated by that FRO. The error detection circuit signals this activity as a FRO error event.

Error events can occur during normal operation. The FRO control circuit attempts to restart a FRO that on a first error event. A second error event causes an automatic shutdown of the FRO. Because there are multiple FROs, shutting down a FRO reduces the amount of entropy generated, but it does not immediately jeopardize the TRNG operation. A limit can be configured below which the number of operational FROs does not drop. If this limit is crossed, an interrupt can be generated on the host processor. Software on the host processor can then attempt to prevent frequent locking of a FRO by *de-tuning* it to a slightly different frequency.

An XOR tree combines the sampled outputs and feeds them into an 81-bit LFSR to accumulate entropy and whiten the random bits stream.

NOTE: Here, *whitening* means balancing the number of one and zero bits in the stream.

TRNG Operating Modes

The TRNG has the following operating modes.

Normal Reading Mode

In normal reading mode, random data can only be read out of the output registers of the TRNG module when the ready bit (`TRNG_STAT.RDY`) =1. Acknowledging the data by writing a 1 to the ready acknowledge bit in the `TRNG_INTACK.RDY` register clears the ready bit from the status register and clears the output registers. The registers remain at zero until the next 128-bit data block is available.

Secure Reading Mode

An attacker can try to read the output registers (without acknowledging the data) to obtain a copy of data to be read later by an application. To block this attack, secure reading mode is used. In this mode, enable reading from the output registers (by writing 0x00 to the open read gate bits (`TRNG_INTACK.OPENRDGATE`) before it is possible to access the output registers. Enabling the reading starts a timeout (controlled by the `TRNG_CFG.RDTIMEOUT` bit field). When this timeout expires, reading is disabled, and the offered data is acknowledged so that it is not offered again. The host processor must set this timeout such that there is enough time to read the output registers and perform a normal data acknowledge (which stops the timeout).

Test Mode

In addition to the test circuitry that operates during normal operation, the TRNG has a test mode that allows further diagnosis when errors occur. In test mode, programs have access to the main LFSR through the

TRNG_LFSR_L, TRNG_LFSR_M, and TRNG_LFSR_H registers. Programs can also control the finite state machine sample counter using the TRNG_CNT register.

In test mode, the TRNG can be configured to test individual or a chosen set of FROs. It can also be configured to feed test patterns to the delay chain.

TRNG Data Transfer Modes

The host processor reads four 32-bit registers to access the random numbers. Once the registers are read and the data ready interrupt has been acknowledged, the TRNG moves more data from the internal buffer to the output registers (TRNG_OUTPUT[n]).

TRNG Event Control

There are eight events that the TRNG generates. The events are common error events from the run-time testing of the TRNG. While the TRNG is operating and generating random bits, test circuitry is also running statistical tests. The tests determine if the sources of the random bits have started to fail and are not truly producing random bits. There is also a single event to signal when data is ready in the output registers.

These events are captured in the TRNG_STAT register and can generate a single interrupt in the Public Key Interrupt Controller (PKIC). The individual events can be masked so the interrupt is not triggered. This configuration uses the TRNG_CTL register. The event and associated interrupt can be acknowledged in the TRNG_INTACK register.

The ***TRNG Interrupt Signals*** table lists all events or interrupts that the TRNG can generate.

TRNG Interrupt Signals

The TRNG provides a total of eight interrupts multiplexed into one output.

Table 44-3: TRNG Interrupt Signals

Interrupt	Name	Description
0	RDY	Ready. Random number is ready to be read from registers
1	SHDNOVR	Shutdown Overflow. The number of FRO's automatically shut down due to failures or errors have gone above the threshold specified in TRNG_ALMCNT.SHDNTHRESH.
2	STUCKOUT	Stuck Out. Logic circuitry around the output registers has detected the same output has been provided twice.
3	NOISEFAIL	Noise Fail. Logic circuitry monitoring the data shifted into the main LFSR detected 48 identical bits, which is considered a noise source failure.
4	RUNFAIL	Run Fail. Logic circuitry monitoring the data shifted into the main LFSR detected an out-of-bounds value for at least one of the TRNG_RUN[n] counters after checking 20,000 bits.

Table 44-3: TRNG Interrupt Signals (Continued)

Interrupt	Name	Description
5	LRUNFAIL	Long Run Fail. Logic circuitry monitoring the data shifted into the main LFSR detected 34 identical bits.
6	PKRFAIL	Poker Fail. Logic circuitry monitoring the data shifted into the main LFSR detected an out-of-bounds value in at least one of the 16 TRNG_POKER[n] counters or an out-of-bounds sum of squares values after checking 20,000 bits.
7	MBITFAIL	Monobit Fail. Logic circuitry monitoring the data shifted into the main LFSR detected an out-of-bounds number of 1's after checking 20,000 bits.

ADSP-SC59x TRNG Register Descriptions

True Random Number Generator (TRNG) contains the following registers.

Table 44-4: ADSP-SC59x TRNG Register List

Name	Description
TRNG_ALMCNT	TRNG Alarm Counter Register
TRNG_ALMMSK	TRNG Alarm Mask Register
TRNG_ALMSTP	TRNG Alarm Stop Register
TRNG_BLKCNT	TRNG Block Count Register
TRNG_CFG	TRNG Configuration Register
TRNG_CNT	Counter Register
TRNG_CTL	TRNG Control Register
TRNG_FRODETUNE	TRNG FRO De-tune Register
TRNG_FROEN	TRNG FRO Enable Register
TRNG_INPUT[n]	TRNG Input Registers
TRNG_INTACK	TRNG Interrupt Acknowledge Register
TRNG_KEY[n]	Post-Process Key Registers
TRNG_LFSR_H	TRNG LFSR Access Register
TRNG_LFSR_L	TRNG LFSR Access Register
TRNG_LFSR_M	TRNG LFSR Access Register
TRNG_MONOBITCNT	TRNG Monobit Test Result Register
TRNG_OUTPUT[n]	TRNG Output Registers
TRNG_POKER[n]	TRNG Poker Test Result Registers
TRNG_RUNCNT	TRNG Run Count Registers
TRNG_RUN[n]	TRNG Run Test State and Result Registers

Table 44-4: ADSP-SC59x TRNG Register List (Continued)

Name	Description
TRNG_STAT	TRNG Status Register
TRNG_TEST	TRNG Test Register
TRNG_V[n]	TRNG Post-Process "V" Value Registers

TRNG Alarm Counter Register

The TRNG_ALMCNT register, together with the TRNG_ALMMSK and TRNG_ALMSTP registers, can be used by the host processor to determine if the FRO/sample cycle locking is a problem. Note that incidental alarm events are expected to occur during normal operation. This register also controls the way the monobit test and poker test circuits operate (using the standard 20,000 bit blocks or running continuously).

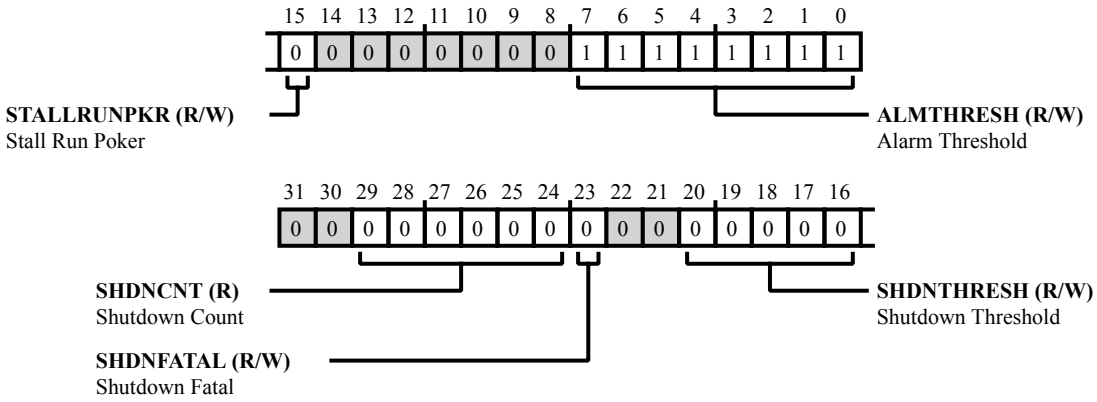


Figure 44-2: TRNG_ALMCNT Register Diagram

Table 44-5: TRNG_ALMCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/NW)	SHDNCNT	Shutdown Count. This read-only field indicates the number of 1 bits in the TRNG_ALMSTP register, the number of FRO's that's been turned off.
23 (R/W)	SHDNFATAL	Shutdown Fatal. When the TRNG_ALMCNT.SHDNFATAL bit field is set, the shutdown overflow (SHDNOVR) interrupt is considered a fatal error requiring taking the complete TRNG engine off-line.
20:16 (R/W)	SHDNTHRESH	Shutdown Threshold. The TRNG_ALMCNT.SHDNTHRESH bit field provides the threshold setting for generating the shutdown overflow (SHDNOVR) interrupt, which is activated when the shutdown count (TRNG_ALMCNT.SHDNCNT) value in this register exceeds the threshold value set here.
15 (R/W)	STALLRUNPKR	Stall Run Poker. When the TRNG_ALMCNT.STALLRUNPKR bit is set, stalls the Monobit Test, Run Test and Poker Test circuits when either the TRNG_STAT.MBITFAIL, TRNG_STAT.RUNFAIL or TRNG_STAT.PKRFAIL bits =1. This allows inspection of the state of the result counters (which would otherwise be reset immediately for the next 20,000 bits block to test).

Table 44-5: TRNG_ALMCNT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	ALMTHRESH	Alarm Threshold. The `TRNG_ALMCNT.ALMTHRESH` bit field sets the alarm detection threshold for the repeating pattern detectors on each FRO. A FRO alarm event is declared when a repeating pattern (of up to four samples length) is detected continuously for the number of samples defined by this fields value. Reset value 255 (decimal) should keep the number of alarm events to a manageable level.

TRNG Alarm Mask Register

A set bit (=1) in the TRNG_ALMMSK register signifies an alarm event and is used by the host processor to determine which of the individual FROs generated the alarm. If a bit in this register is set, the corresponding bit in the TRNG_ALMSTP register is set and the FRO is turned off by clearing the corresponding bit in the TRNG_FROEN register. If a bit is not set, the FRO restarts automatically to try to break sample cycle locking that could have caused the alarm event.

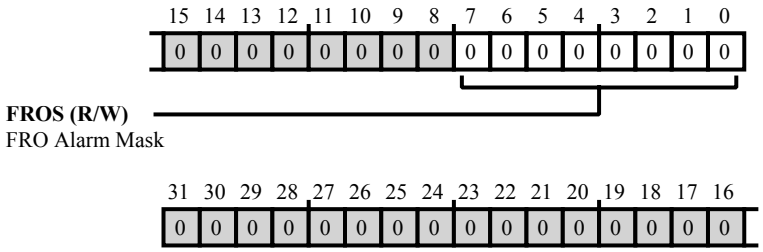


Figure 44-3: TRNG_ALMMSK Register Diagram

Table 44-6: TRNG_ALMMSK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FROS	FRO Alarm Mask. The TRNG_ALMMSK.FROS bit field provides logging for the alarm events of individual FROs. A 1 in bit [n] indicates FRO n experienced an alarm event.

TRNG Alarm Stop Register

The TRNG_ALMSTP register is used by the host processor to determine which of the individual FROs generated more than one alarm event in quick succession. If a FRO generates an alarm event while a previous event is still logged in the TRNG_ALMMSK register, the corresponding bit in this register is set (=1) and the FRO is turned off by clearing (=0) the corresponding bit in the TRNG_FROEN register. The TRNG_ALMCNT.SHDNCNT bit field keeps track of the number of bits that are set in this register.

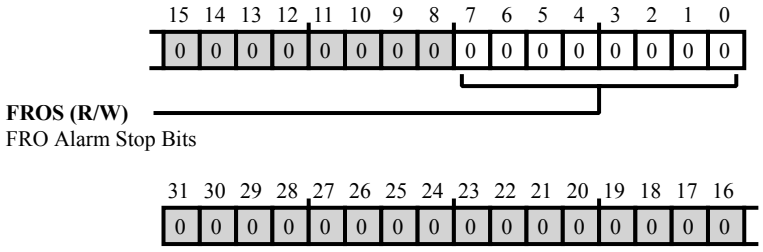


Figure 44-4: TRNG_ALMSTP Register Diagram

Table 44-7: TRNG_ALMSTP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FROS	FRO Alarm Stop Bits. The TRNG_ALMSTP.FROS bit field provides logging for the alarm events of individual FROs. A 1 in bit [n] indicates FRO n experienced more than one alarm event in quick succession and has been turned off. A 1 in this field forces the corresponding bit in the TRNG_FROEN register to 0.

TRNG Block Count Register

The TRNG_BLKCNT register is the counter for the 128-bit blocks generated by the post-processor. These bits are forced to zero when the post-processor is disabled and are cleared to zero when an internal re-seed operation has finished. This register can be used by driver software to determine when to re-seed the post-processor.

The whole 32 bits of this register represent the amount of data (in bytes) generated since a re-seed. The TRNG_BLKCNT register is only present when a post-processor is available.

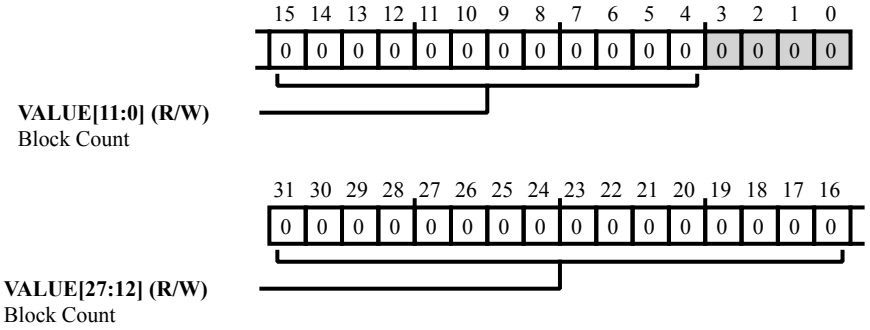


Figure 44-5: TRNG_BLKCNT Register Diagram

Table 44-8: TRNG_BLKCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:4 (R/W)	VALUE	Block Count. The TRNG_BLKCNT.VALUE bit field is the counter for the 128-bit blocks generated by the post-processor. These bits are forced to zero when the post-processor is disabled and are cleared to zero when an internal re-seed operation has finished.

TRNG Configuration Register

The TRNG_CFG register holds the lower and upper limits of the samples taken from the FROs in order to refill the random data buffer. This register also holds the time out value used for secure reading mode.

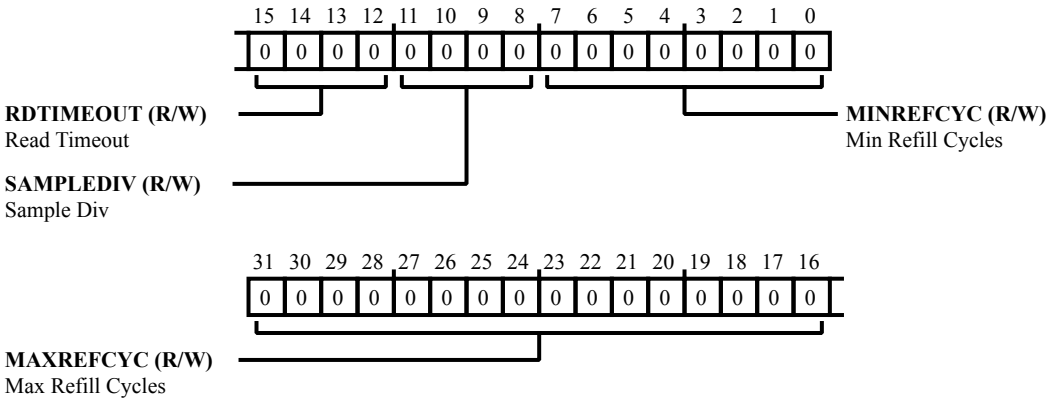


Figure 44-6: TRNG_CFG Register Diagram

Table 44-9: TRNG_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	MAXREFCYC	Max Refill Cycles. The TRNG_CFG.MAXREFCYC bit field determines the maximum number of samples (between 2^8 and 2^{24}) taken to re-generate entropy from the FROs after reading out a 64 bit random number. If the written value of this field is zero, the number of samples is 2^{24}, otherwise the number of samples equals the written value times 2^8. This field can only be modified while the TRNG_CTL.TRNGEN bit =0.
15:12 (R/W)	RDTIMEOUT	Read Timeout. The TRNG_CFG.RDTIMEOUT bit field controls the Secure Reading Mode. When this field is 0, Secure Reading Mode is disabled. Values in the range 115 enable Secure Reading and set a read gate closure timeout of approximately (read_timeout + 1) x 16 clock input cycles. This field can only be modified while the TRNG_CTL.TRNGEN bit =0.
11:8 (R/W)	SAMPLEDIV	Sample Div. The TRNG_CFG.SAMPLEDIV bit field directly controls the number of input cycles between samples taken from the FROs. The default value 0 indicates that samples are taken every cycle, maximum value 15 (decimal) takes one sample every 16 cycles. This field must be set to a value such that the slowest FRO (even under worst-case conditions) has a cycle time less than twice the sample period. The default configuration of the FROs allows this field to remain 0. This field can only be modified while TRNG_CTL.TRNGEN=0.

Table 44-9: TRNG_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MINREFCYC	Min Refill Cycles. The TRNG_CFG.MINREFCYC bit field determines the minimum number of samples (between 2^6 and 2^{24}) taken to re-generate entropy from the FROs after reading out a 64 bit random number. If the value of this field is zero, the number of samples is fixed to the value determined by the maximum refill cycles (TRNG_CFG.MAXREFCYC) field, otherwise the minimum number of samples equals the written value times 64 (which can be up to 2^{14}). The number of samples defined here cannot be higher than the number defined by the TRNG_CFG.MAXREFCYC field (i.e. that field takes precedence). This field can only be modified while the TRNG_CTL.TRNGEN bit =0.

Counter Register

The TRNG_CNT register is used to access the main control Finite State Machine's (FSM) sample counter while the TRNG_CTL.TSTMODE bit =1.

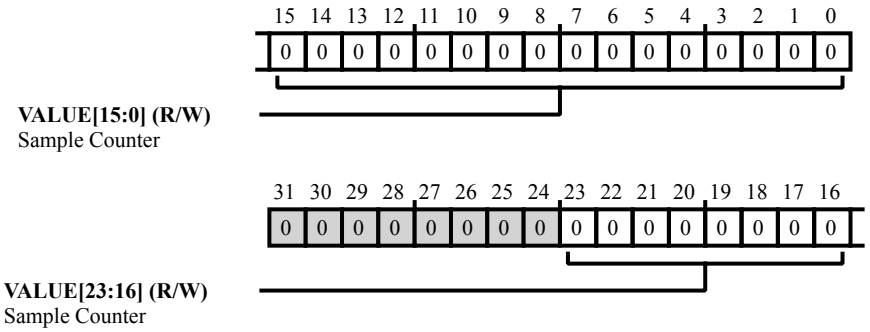


Figure 44-7: TRNG_CNT Register Diagram

Table 44-10: TRNG_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:0 (R/W)	VALUE	Sample Counter. The TRNG_CNT.VALUE bit field is the sample counter used by control finite state machine. This counter can only be accessed when the TRNG_CTL.TSTMODE bit =1.

TRNG Control Register

The TRNG_CTL register must be written to start accumulating entropy before random numbers can be generated. In most cases, the TRNG_CFG register must also be written prior to writing the TRNG_CTL register. To enable the TRNG, set the TRNG_CTL.TRNGEN bit. This register also controls post-processing (if available). Note that when the TRNG_CTL.TRNGEN bit =1, the start up cycles field (TRNG_CTL.STARTUPCYC) and the post-processing enable bit (if available) are locked. Any writes to these fields are ignored.

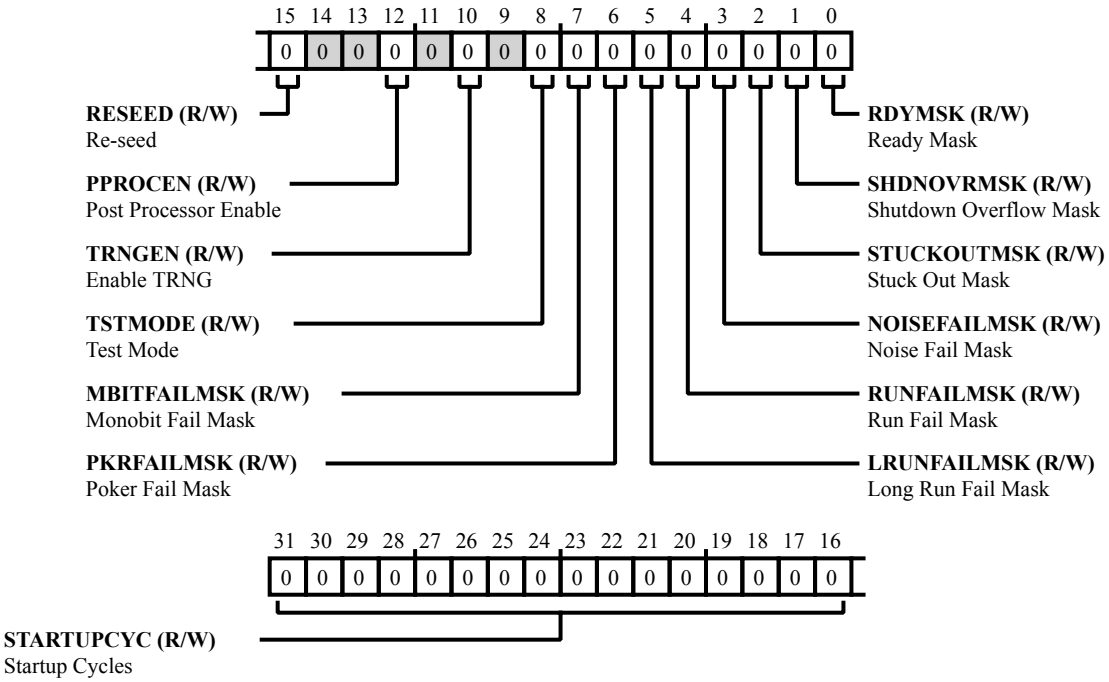


Figure 44-8: TRNG_CTL Register Diagram

Table 44-11: TRNG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	STARTUPCYC	Startup Cycles. The TRNG_CTL.STARTUPCYC bit field determines the number of samples (between 2^8 and 2^{24}) taken to gather entropy from the FROs during startup. If the written value of this field is zero, the number of samples is 2^{24}, otherwise the number of samples equals the written value times 2^8 . This field can only be written when TRNG_CTL.TRNGEN=0 before the write.

Table 44-11: TRNG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	RESEED	Re-seed. The TRNG_CTL.RESEED bit is set-only, writing a 1 starts a re-seed cycle and writing a 0 has no effect. A re-seed cycle entails loading the TRNG_KEY[n] and TRNG_V[n] registers with random values generated internally these values are not visible outside the TRNG core. This bit falls back to 0 automatically after the re-seed operation is complete at that time the TRNG_BLKCNT register is reset to zero and the random data buffer is zeroized so that any new data read from the TRNG will use the new seed values. This bit is only present when post-processing is available and can only be set to 1 when TRNG_CTL.TRNGEN=1 before the write. Note that re-seeding can be done with the post-processor disabled (normally used to seed the post-processor before enabling it). When writing a 1 to this bit, all other bits in this register remain unchanged.
12 (R/W)	PPROCEN	Post Processor Enable. Setting the TRNG_CTL.PPROCEN bit enables the FIPS post-processor. If this bit is reset to 0, the post-processor is forced back into the idle state immediately. This bit is only present when post-processing is available and can only be changed when the TRNG_CTL.TRNGEN bit (enable TRNG) was 0 before the write. To change the TRNG_CTL.PPROCEN bit during operation, first put the TRNG into reset (TRNG_CTL.TRNGEN =0). If the post-processor is enabled, it can be disabled by a subsequent write of 0 to this bit (writing this bit when the TRNG_CTL.TRNGEN bit is still 1 has no effect). The post-processor then stops immediately. To enable it, this bit bit must be written with 1. Changing the enabled/disabled state does not affect the contents of the TRNG_KEY[n] and TRNG_V[n] registers. After changing the state, the TRNG must be started again by setting the TRNG_CTL.TRNGEN bit to 1. Note that it is required to re-gather entropy, so the same number of start-up cycles must be used as when starting the TRNG out of a system reset state.
10 (R/W)	TRNGEN	Enable TRNG. Setting the TRNG_CTL.TRNGEN bit to 1 starts the TRNG, gathering entropy from the FROs for the number of samples determined by the value in the TRNG_CTL.STARTUPCYC (Startup Cycles) field. Resetting this bit to 0 forces all TRNG logic back into the idle state immediately. Resetting this bit to 0 also performs the Un-instantiate operation, clearing all internal post-processor registers.
8 (R/W)	TSTMODE	Test Mode. When the TRNG_CTL.TSTMODE bit is set, access is enabled to the TRNG_CNT and TRNG_LFSR_L, TRNG_LFSR_M and TRNG_LFSR_H registers (the latter are cleared before enabling access) and sets the TRNG_STAT.NEEDCLK bit for testing purposes. This bit must be set to 1 before various test modes in the TRNG_TEST register can be enabled.

Table 44-11: TRNG_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/W)	MBITFAILMSK	Monobit Fail Mask. When the TRNG_CTL.MBITFAILMSK bit is set, this mask allows the TRNG_STAT.MBITFAIL bit to activate the (active HIGH) interrupt output.
6 (R/W)	PKRFAILMSK	Poker Fail Mask. When the TRNG_CTL.PKRFAILMSK bit is set, this mask allows the TRNG_STAT.PKRFAIL bit to activate the (active HIGH) interrupt output.
5 (R/W)	LRUNFAILMSK	Long Run Fail Mask. When the TRNG_CTL.LRUNFAILMSK bit is set, this mask allows the TRNG_STAT.LRUNFAIL bit to activate the (active HIGH) interrupt output.
4 (R/W)	RUNFAILMSK	Run Fail Mask. When the TRNG_CTL.RUNFAILMSK bit is set, this mask allows the TRNG_STAT.RUNFAIL bit to activate the (active HIGH) interrupt output.
3 (R/W)	NOISEFAILMSK	Noise Fail Mask. When the TRNG_CTL.NOISEFAILMSK bit is set, this mask allows the TRNG_STAT.NOISEFAIL bit to activate the (active HIGH) interrupt output.
2 (R/W)	STUCKOUTMSK	Stuck Out Mask. When the TRNG_CTL.STUCKOUTMSK bit is set, this mask allows the TRNG_STAT.STUCKOUT bit to activate the (active HIGH) interrupt output.
1 (R/W)	SHDNOVRMSK	Shutdown Overflow Mask. When the TRNG_CTL.SHDNOVRMSK bit is set, this mask allows the TRNG_STAT.SHDNOVR bit to activate the (active HIGH) interrupt output.
0 (R/W)	RDYMSK	Ready Mask. When the TRNG_CTL.RDYMSK bit is set, this mask allows the TRNG_STAT.RDY bit to activate the (active HIGH) interrupt output.

TRNG FRO De-tune Register

The TRNG_FRODETUNE register is used by the host processor to change the frequencies of individual FROs. This can reduce the number of alarm events generated by a specific FRO.

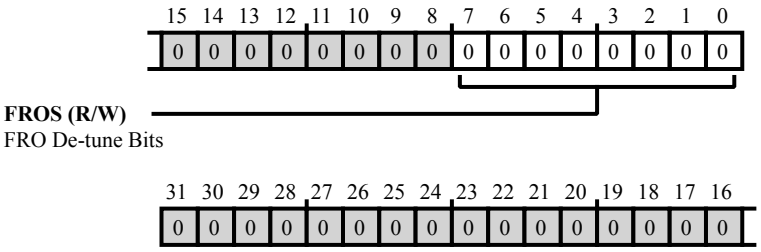


Figure 44-9: TRNG_FRODETUNE Register Diagram

Table 44-12: TRNG_FRODETUNE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FROS	FRO De-tune Bits. The TRNG_FRODETUNE.FROS bits De-tune the FROs. A 1 in bit [n] lets FRO n run approximately 5% faster. The value of one of these bits may only be changed while the corresponding FRO is turned off (by temporarily writing a 0 in the corresponding bit of the TRNG_FROEN register).

TRNG FRO Enable Register

The TRNG_FROEN register can be used by the host processor to enable and disable FROs individually. Only enabled FROs contribute to entropy generation, but require power to do so. Disabled FROs cannot generate alarm events.

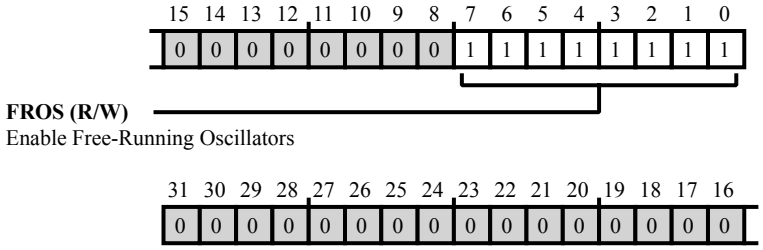


Figure 44-10: TRNG_FROEN Register Diagram

Table 44-13: TRNG_FROEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	FROS	Enable Free-Running Oscillators. The TRNG_FROEN.FROS bits are the enables for the individual FROs. A 1 in bit [n] enables FRO n. The default state is all ones to enable all FROs after power-up. Note that the FROs are not actually started up before the TRNG_CTL.TRNGEN bit is set to 1. These bits are automatically forced to 0 (and cannot be written to 1) when the corresponding bit in the TRNG_ALMSTP register has value 1.

TRNG Input Registers

The TRNG_INPUT[n] registers are used as input for post-processor testing (if post processing is available) and as input for Monobit Test, Run Test and Poker Test functionality tests (TRNG_INPUT0 only). They share their addresses with the corresponding TRNG_OUTPUT[n] registers. The least significant word is contained in the TRNG_INPUT0 register.

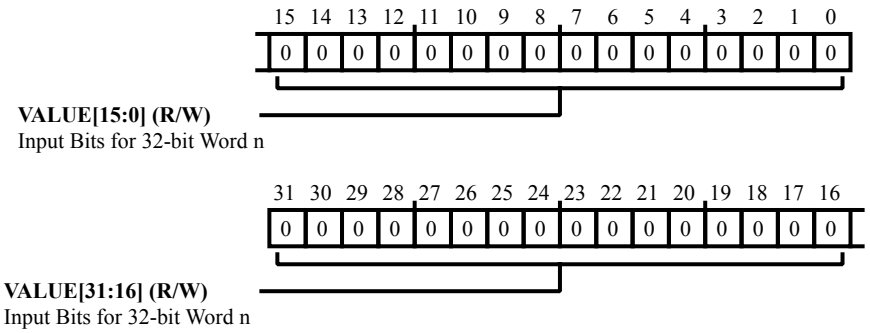


Figure 44-11: TRNG_INPUT[n] Register Diagram

Table 44-14: TRNG_INPUT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Input Bits for 32-bit Word n. The TRNG_INPUT[n].VALUE bit field is used to hold 32-bits of the 64-bit word of test data for 3-DES post-processor (if available). Or, the TRNG_INPUT[n].VALUE bit field is used to hold 32-bit data word for Run Test and Poker Test circuits self test. Can only be written to when the TRNG_STAT.TSTRDY bit =1.

TRNG Interrupt Acknowledge Register

The TRNG_INTACK register is written to acknowledge interrupts indicated in bits [7:0] of the TRNG_STAT register. Writing a 1 to any of the bits [7:2] has side effects in resetting various parts of the TRNG core logic which can also be used even if no interrupts are actually active.

When acknowledging the interrupts, these bits are write '1' to clear the associated bit in TRNG_STAT register. The bit in this register will also automatically be reset to zero.

When secure reading mode is enabled, write bits [7:0] of this register with zeros to enable TRNG data reads from the TRNG_OUTPUT[n] registers. Writing bits [7:0] also starts the (configurable) timeout counter that automatically acknowledges the TRNG data (and disables reads) if the TRNG_INTACK.RDY bit is not written with a 1 within that timeout period.

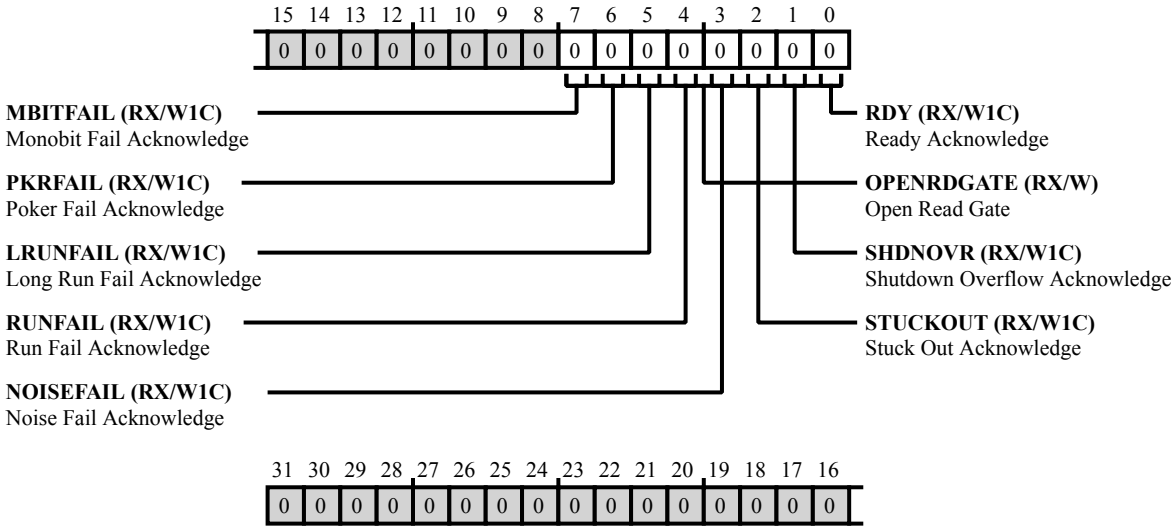


Figure 44-12: TRNG_INTACK Register Diagram

Table 44-15: TRNG_INTACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (RX/W1C)	MBITFAIL	Monobit Fail Acknowledge. Set the TRNG_INTACK.MBITFAIL bit to acknowledge the Monobit Fail Interrupt. This also resets all counter and state bits in the TRNG_RUN[n], TRNG_MONOBITCNT and the TRNG_POKER[n] registers (except for the TRNG_RUNCNT.LENMAX field).
6 (RX/W1C)	PKRFAIL	Poker Fail Acknowledge. Set the TRNG_INTACK.PKRFAIL bit to acknowledge the Poker Fail Interrupt. This also resets all counter and state bits in the TRNG_RUN[n], TRNG_MONOBITCNT and the TRNG_POKER[n] registers (except for the TRNG_RUNCNT.LENMAX field).

Table 44-15: TRNG_INTACK Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
5 (RX/W1C)	LRUNFAIL	Long Run Fail Acknowledge. Set the TRNG_INTACK.LRUNFAIL bit to acknowledge the Long Run Fail Interrupt. Also clears the TRNG_RUNCNT.LENMAX field.
4 (RX/W1C)	RUNFAIL	Run Fail Acknowledge. Set the TRNG_INTACK.RUNFAIL bit to acknowledge the Run Fail Interrupt. Also resets all counter and state bits in the TRNG_RUN[n], TRNG_MONOBITCNT and the TRNG_POKER[n] registers (except for the TRNG_RUNCNT.LENMAX field).
3 (RX/W1C)	NOISEFAIL	Noise Fail Acknowledge. Set the TRNG_INTACK.NOISEFAIL bit to acknowledge the Noise Fail Interrupt. Setting this bit also clears the TRNG_RUNCNT.LENMAX (Run Length Max) field, the random data buffer, the TRNG_OUTPUT[n] registers and the TRNG_STAT.RDY bit.
2 (RX/W1C)	STUCKOUT	Stuck Out Acknowledge. Set the TRNG_INTACK.STUCKOUT bit to acknowledge the Stuck Out Interrupt. Setting this bit also clears the random data buffer, the TRNG_OUTPUT[n] registers and the TRNG_STAT.RDY bit.
1 (RX/W1C)	SHDNOVR	Shutdown Overflow Acknowledge. Set the TRNG_INTACK.SHDNOVR bit to acknowledge the Shutdown Overflow Interrupt.
0 (RX/W1C)	RDY	Ready Acknowledge. The TRNG_INTACK.RDY bit allows a new number (if it is ready in the random data buffer), to directly move into the result register. Once done, the TRNG_STAT.RDY bit is reset, after at most size clock cycles.
7:0 (RX/W)	OPENRDGATE	Open Read Gate. In Secure Reading Mode, the TRNG_INTACK.OPENRDGATE bit writes an all zeros value to bits [7:0] to enable reading of TRNG data from the TRNG_OUTPUT[n] registers. This starts the timeout counter that automatically acknowledges the TRNG data (and disables reading) if the TRNG_INTACK.RDY bit is not written with a 1 within that timeout period.

Post-Process Key Registers

The TRNG_KEY[n] registers are used to load the key used for post-processing (if available). These registers are write-only. Reads return the values of the other registers mapped at the same addresses.

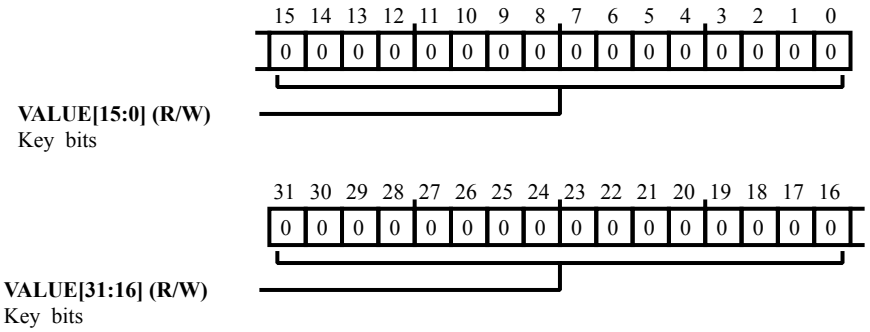


Figure 44-13: TRNG_KEY[n] Register Diagram

Table 44-16: TRNG_KEY[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Key bits. Bits for cipher key used in the post-processor.

TRNG LFSR Access Register

The TRNG_LFSR_H register is used to access bits [80:64] of the main entropy accumulation LFSR while in test mode (TRNG_CTL.TSTMODE =1).

For security reasons, the LFSR contents are zeroed before enabling access.

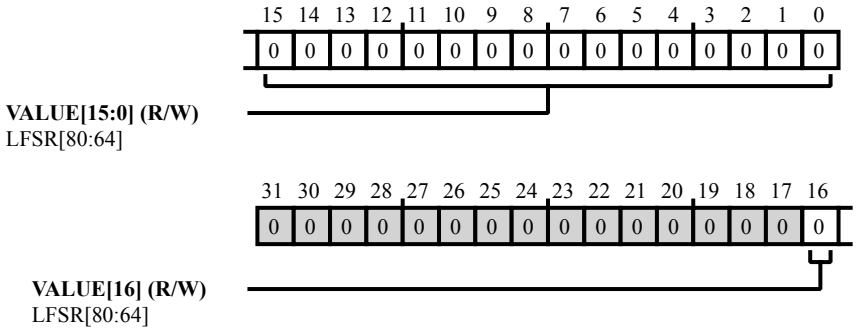


Figure 44-14: TRNG_LFSR_H Register Diagram

Table 44-17: TRNG_LFSR_H Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
16:0 (R/W)	VALUE	LFSR[80:64]. The TRNG_LFSR_H.VALUE bit field contains bits [80:64] of the main entropy accumulation LFSR. This field can only be accessed when the TRNG_CTL.TSTMODE bit =1. Contents are cleared (=0) before access is enabled.

TRNG LFSR Access Register

The TRNG_LFSR_L register is used to access bits [31:0] of the main entropy accumulation LFSR while in test mode (TRNG_CTL.TSTMODE =1).

For security reasons, the LFSR contents are zeroed before enabling access.

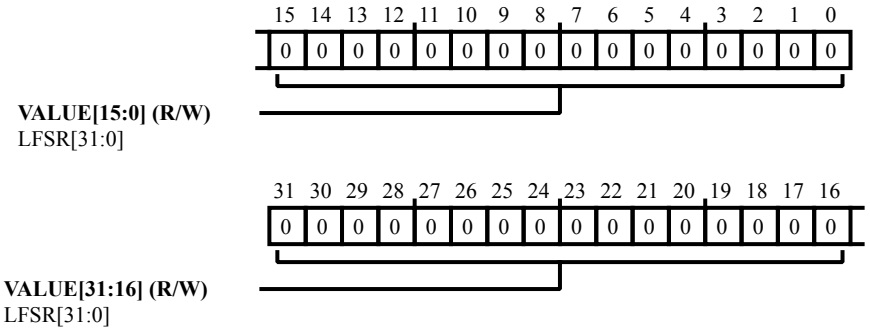


Figure 44-15: TRNG_LFSR_L Register Diagram

Table 44-18: TRNG_LFSR_L Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	LFSR[31:0]. The TRNG_LFSR_L.VALUE bit field contains bits [31:0] of the main entropy accumulation LFSR. This field can only be accessed when the TRNG_CTL.TSTMODE bit =1. Contents are cleared (=0) before access is enabled.

TRNG LFSR Access Register

The TRNG_LFSR_M register is used to access bits [63:32] of the main entropy accumulation LFSR while in test mode (TRNG_CTL.TSTMODE =1).

For security reasons, the LFSR contents are zeroed before enabling access.

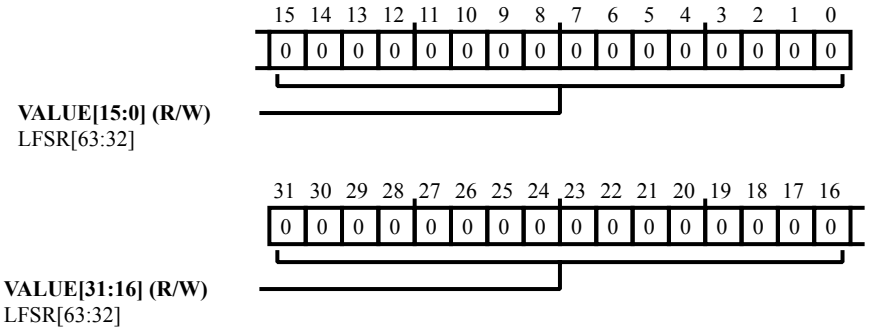


Figure 44-16: TRNG_LFSR_M Register Diagram

Table 44-19: TRNG_LFSR_M Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	LFSR[63:32]. The TRNG_LFSR_M.VALUE bit field contains bits [63:32] of the main entropy accumulation LFSR. This field can only be accessed when the TRNG_CTL.TSTMODE bit =1. Contents are cleared (=0) before access is enabled.

TRNG Monobit Test Result Register

The TRNG_MONOBITCNT register accesses the counter used perform a Monobit Test as specified by the AIS-31 standard (test T1, ref 4). This test is performed on blocks of 20,000 bits (in parallel to the run test and Poker Test).

Note: Immediately after performing the actual Monobit Test at the end of the 20,000 bits block, the counter is used to accumulate the Poker Test results. As a result, the actual Monobit Test count result value can only be read in the TRNG_MONOBITCNT register if the test fails and the stall run Poker (TRNG_ALMCNT.STALLRUNPKR) bit =1.

The monobit test result register is read-only; writing it accesses the registers mapped at the same address. The counter in this register is reset when writing a 1 to either the monobit fail acknowledge (TRNG_INTACK.MBITFAIL), run fail acknowledge (TRNG_INTACK.RUNFAIL) or the poker fail acknowledge (TRNG_INTACK.PKRFAIL) bits.

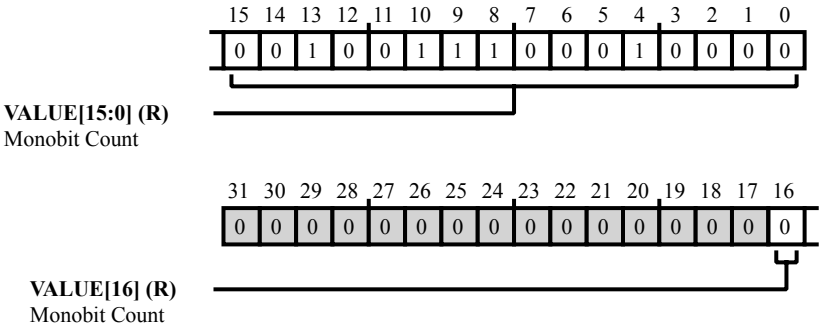


Figure 44-17: TRNG_MONOBITCNT Register Diagram

Table 44-20: TRNG_MONOBITCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
16:0 (R/NW)	VALUE	Monobit Count. The TRNG_MONOBITCNT.VALUE bit field is the up/down counter which monitors 1 and 0 bits. After 20,000 bits, this counter should have a value in the range 9310 through 10690 (inclusive) to pass the Monobit Test. This counter is protected against overflow and underflow.

TRNG Output Registers

The TRNG_OUTPUT[n] registers provide read access to the 128-bit random number output. A subset of these registers are also used as output for post-processor testing (if available). They share their addresses with the TRNG_INPUT0 through TRNG_INPUT3 registers. The least significant word is contained in the TRNG_OUTPUT0 register.

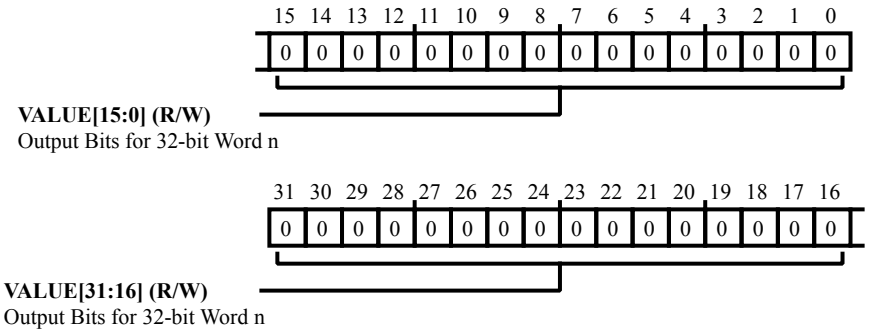


Figure 44-18: TRNG_OUTPUT[n] Register Diagram

Table 44-21: TRNG_OUTPUT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Output Bits for 32-bit Word n. The TRNG_OUTPUT[n].VALUE bit field is used to holds 32 bits of the 128-bit word of random data. Only valid when the TRNG_STAT.RDY bit =1. Alternatively, this register holds the 32-bits of the 42-bit word of result data for 3-DES post-processing testing. Only valid when the TRNG_STAT.TSTRDY bit =1.

TRNG Poker Test Result Registers

The TRNG_POKER[n] registers are used to access the 16 counters used perform a poker test on blocks of 20,000 bits (in parallel to the monobit and run tests).

Poker test result registers are read-only; writing them accesses the registers mapped at these same addresses. All counters in these registers are reset when writing a 1 to either the monobit fail acknowledge (TRNG_INTACK.MBITFAIL), run fail acknowledge (TRNG_INTACK.RUNFAIL) or the poker fail acknowledge (TRNG_INTACK.PKRFAIL) bits.

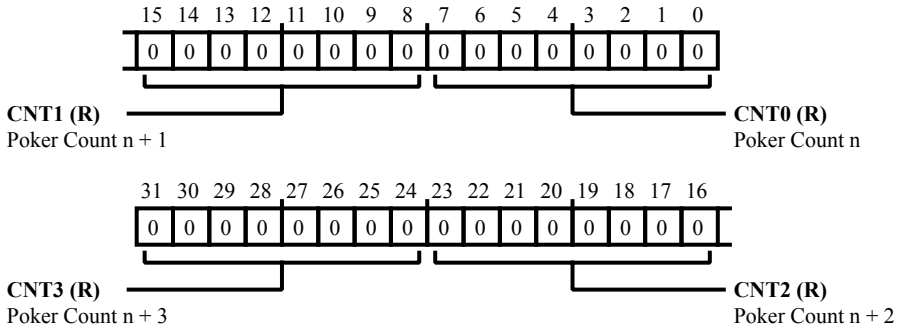


Figure 44-19: TRNG_POKER[n] Register Diagram

Table 44-22: TRNG_POKER[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/NW)	CNT3	Poker Count n + 3. The TRNG_POKER[n].CNT3 bit field provides the counter for 4-bit value 0x3, 0x7, 0xB, 0xF in TRNG_POKER0, TRNG_POKER1, TRNG_POKER2, TRNG_POKER3, respectively.
23:16 (R/NW)	CNT2	Poker Count n + 2. The TRNG_POKER[n].CNT2 bit field provides the counter for 4-bit value 0x2, 0x6, 0xA, 0xE in TRNG_POKER0, TRNG_POKER1, TRNG_POKER2, TRNG_POKER3, respectively.
15:8 (R/NW)	CNT1	Poker Count n + 1. The TRNG_POKER[n].CNT1 bit field provides the counter for 4-bit value 0x1, 0x5, 0x9, 0xD in TRNG_POKER0, TRNG_POKER1, TRNG_POKER2, TRNG_POKER3, respectively.
7:0 (R/NW)	CNT0	Poker Count n. The TRNG_POKER[n].CNT0 bit field provides the counter for 4-bit value 0x0, 0x4, 0x8, 0xC in the TRNG_POKER0, TRNG_POKER1, TRNG_POKER2, TRNG_POKER3, respectively.

TRNG Run Count Registers

The TRNG_RUNCNT registers are used to access the 10 counters that perform a run test and long run test as specified by the AIS-31 standard (tests T3 and T4, ref 4). They are also used to perform the noise source failure test proposed in section E.5 of that same standard.

The TRNG_RUNCNT registers are read-only; writing them accesses the other registers which are mapped at the same addresses. Unless otherwise indicated, all counters and state bits in these registers are reset when writing a 1 to either the Monobit Fail acknowledge (TRNG_INTACK.MBITFAIL), Run Fail acknowledge (TRNG_INTACK.RUNFAIL) or the Poker Fail acknowledge (TRNG_INTACK.PKRFAIL) bits.

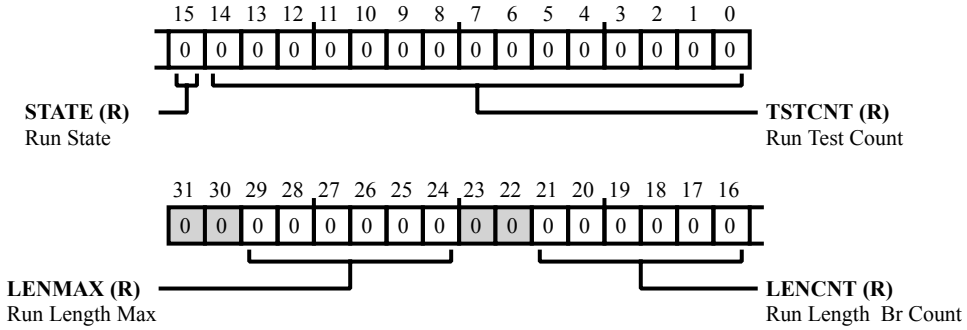


Figure 44-20: TRNG_RUNCNT Register Diagram

Table 44-23: TRNG_RUNCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:24 (R/NW)	LENMAX	Run Length Max. The TRNG_RUNCNT.LENMAX bit field configures the maximum run length count value encountered since start of test. This value is reset back to zero when writing a 1 to either the Noise Fail acknowledge (TRNG_INTACK.NOISEFAIL) or the Long Run Fail acknowledge (TRNG_INTACK.LRUNFAIL) bits.
21:16 (R/NW)	LENCNT	Run Length Br Count. The TRNG_RUNCNT.LENCNT bit field configures the counter for the current run of consecutive 0/1 bits; cannot increment past its maximum value of 63.
15 (R/NW)	STATE	Run State. The TRNG_RUNCNT.STATE bit field provides th state of bits in the current run.
14:0 (R/NW)	TSTCNT	Run Test Count. The TRNG_RUNCNT.TSTCNT bit field configures the block length counter for the run and poker tests - counts up for 20,000 tested bits and then controls testing of the run_X_count_... and poker_count_X counters to contain expected values, after which they - and this counter - are reset for the next block.

TRNG Run Test State and Result Registers

The TRNG_RUN[n] registers holds the counts for the associated run bucket for 1's and 0's.

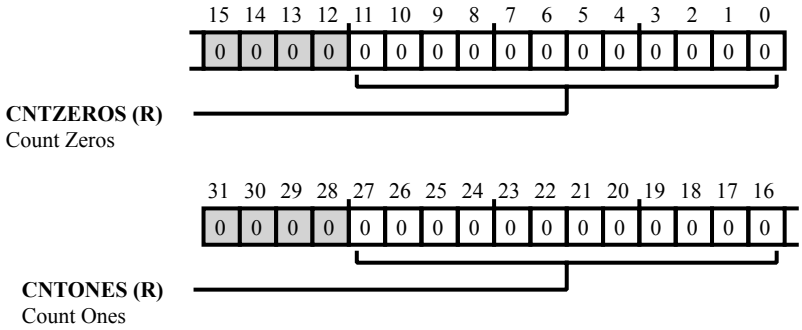


Figure 44-21: TRNG_RUN[n] Register Diagram

Table 44-24: TRNG_RUN[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
27:16 (R/NW)	CNTONES	Count Ones. In TRNG_RUN1, this counter is for single bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 2267 to 2733 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 4095. In TRNG_RUN2, this counter is for two bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 1079 to 1421 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 2047. In TRNG_RUN3, this counter is for three bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 502 to 748 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 1023. In TRNG_RUN4, this counter for four bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 233 to 402 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 511. In TRNG_RUN5, this counter is for five bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 90 to 223 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 255. In TRNG_RUN6, this counter for six and higher bit runs of value one bits. After 20,000 bits, this counter should have a value in the range 90 to 233 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 255.

Table 44-24: TRNG_RUN[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/NW)	CNTZEROS	Count Zeros. In TRNG_RUN1, this counter is for single bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 2267 to 2733 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 4095. In TRNG_RUN2, this counter is for two bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 1079 to 1421 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 2047. In TRNG_RUN3, this counter is for three bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 502 to 748 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 1023. In TRNG_RUN4, this counter is for four bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 233 to 402 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 511. In TRNG_RUN5, this counter is for five bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 90 to 223 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 255. In TRNG_RUN6, this counter is for six and higher bit runs of value zero bits. After 20,000 bits, this counter should have a value in the range 90 to 233 (inclusive) to pass the run test. This counter cannot increment past its maximum value of 255.

TRNG Status Register

The TRNG_STAT register provides status results. This register shares the same address as the Interrupt Acknowledge (TRNG_INTACK) register.

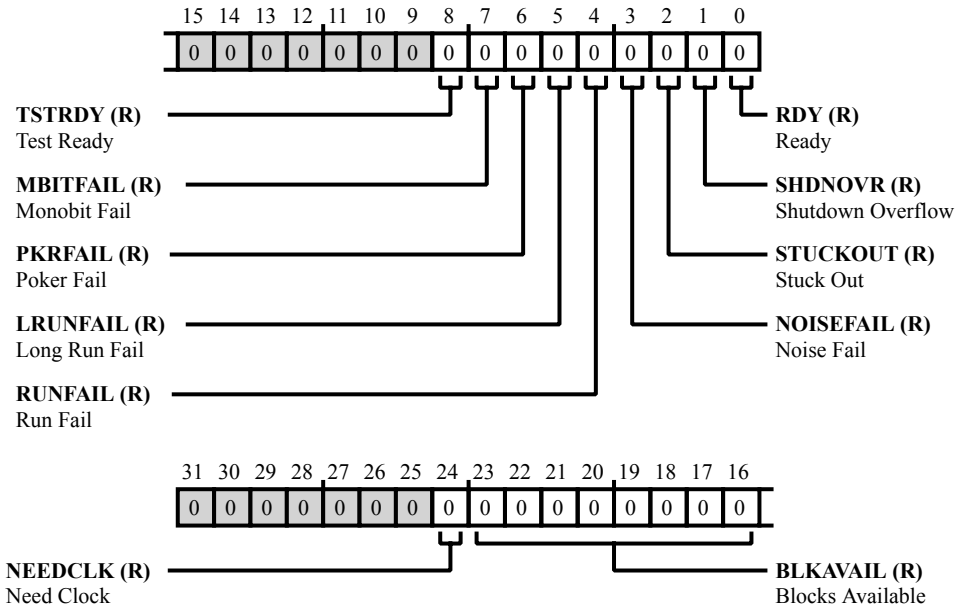


Figure 44-22: TRNG_STAT Register Diagram

Table 44-25: TRNG_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
24 (R/NW)	NEEDCLK	Need Clock. When the TRNG_STAT.NEEDCLK bit is set, it indicates that the TRNG is busy generating entropy or is in one of its test modes the module clock may not be turned off.
23:16 (R/NW)	BLKAVAIL	Blocks Available. This field indicates the number of 128 bits blocks of random data that are available in the random data buffer. If this value is non-zero, the output registers will be re-filled from the random data buffer immediately after acknowledging the TRNG_STAT.RDY by writing a '1' to TRNG_INTACK.RDY.
8 (R/NW)	TSTRDY	Test Ready. When the TRNG_STAT.TSTRDY bit is set, it indicates that data for known-answer tests on the Monobit Test, Run Test, Poker Test and post-processor functions can be written to the TRNG_INPUT[n] registers. When testing the post-processor, result data can be read from those same registers when this bit has become 1 again (after dropping to 0).

Table 44-25: TRNG_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	MBITFAIL	Monobit Fail. When the TRNG_STAT.MBITFAIL bit is set, the Monobit Test logic monitoring data, shifted into the main LFSR, detected an out-of-bounds number of 1s after checking 20,000 bits (test T1 as specified in the AIS-31 standard).
6 (R/NW)	PKRFAIL	Poker Fail. When the TRNG_STAT.PKRFAIL bit is set, the Poker Test logic monitoring data shifted into the main LFSR detected an out-of-bounds value in at least one of the 16 Poker Counters or an out of bounds sum of squares value after checking 20,000 bits (test T2 as specified in the AIS-31 standard).
5 (R/NW)	LRUNFAIL	Long Run Fail. When the TRNG_STAT.LRUNFAIL bit is set, the Run Test logic monitoring data shifted into the main LFSR detected a sequence of 34 identical bits (test T4 as specified in the AIS-31 standard).
4 (R/NW)	RUNFAIL	Run Fail. When the TRNG_STAT.RUNFAIL bit is set, the Run Test logic monitoring data shifted into the main LFSR detected an out-of-bounds value for at least one of the TRNG_RUN[n].CNTZEROS or TRNG_RUN[n].CNTONES counters after checking 20,000 bits (test T3 as specified in the AIS-31 standard).
3 (R/NW)	NOISEFAIL	Noise Fail. When the TRNG_STAT.NOISEFAIL bit is set, the Run Test logic monitoring data shifted into the main LFSR detected a sequence of 48 identical bits, which is considered a noise source failure as proposed in section E.5 of the AIS-31 standard.
2 (R/NW)	STUCKOUT	Stuck Out. When the TRNG_STAT.STUCKOUT bit is set, the logic around the output data registers detected that the TRNG generates the same value twice in a row.
1 (R/NW)	SHDNOVR	Shutdown Overflow. When the TRNG_STAT.SHDNOVR bit is set, the number of FROs shut down after a second error event (the number of 1 bits in the TRNG_ALMSTP register) has exceeded the threshold set by the TRNG_ALMCNT.SHDNTHRESH bit field.
0 (R/NW)	RDY	Ready. When the TRNG_STAT.RDY bit is set, data is available in the TRNG_OUTPUT0 to TRNG_OUTPUT3 registers. If a new number is already available in the random data buffer, that number is directly moved into the result register. In this case the ready status bit is asserted again, after at most six module clock cycles.

TRNG Test Register

The TRNG_TEST register can be used by the host processor to perform a number of tests on the TRNG logic including:

- Register controlled characterization by connecting the tst_fro_clk_out output to a selected FRO clock output
- FRO logic connectivity and error event detection checking by feeding known patterns through the FRO delay line and error event detection circuits
- Direct XOR-ed FRO outputs capture by disabling the main LFSR feedback logic
- Extend the Monobit Test and Poker Test by not resetting the Monobit count and Poker Test X counters after each 20,000 bits block
- Perform known answer tests on the Run Test, Poker Test and post-processor functions.

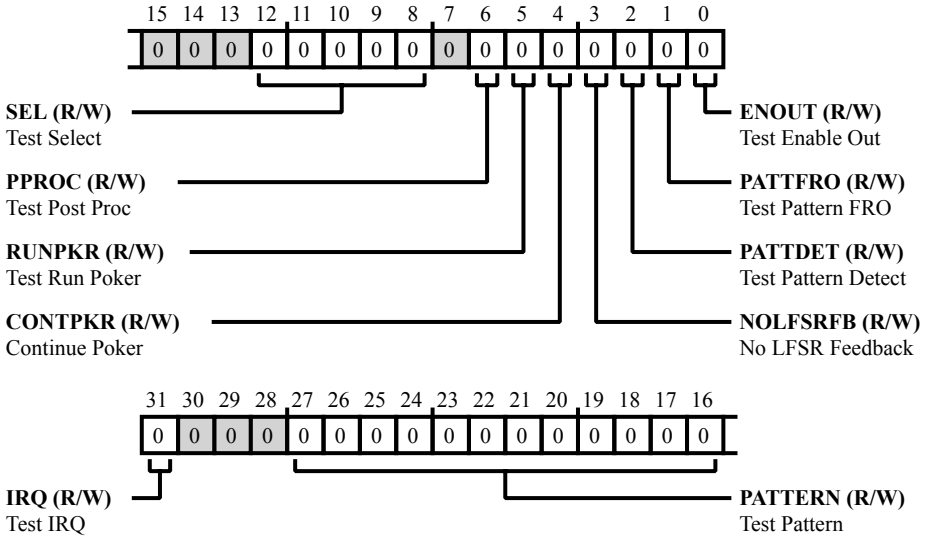


Figure 44-23: TRNG_TEST Register Diagram

Table 44-26: TRNG_TEST Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	IRQ	Test IRQ. When the TRNG_TEST.IRQ bit is set force irq output HIGH for interrupt signal connectivity testing.	
		0	Do not force IRQ high
		1	Force IRQ output high

Table 44-26: TRNG_TEST Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27:16 (R/W)	PATTERN	Test Pattern. The TRNG_TEST.PATTERN bit field sets up a repeating sequence of bits to be fed into the selected FRO delay chain TRNG_TEST.PATTFRO =1 and/or the selected FRO error detection circuit TRNG_TEST.PATTDET =1. This field is rotated right over one bit, once every sample period, when either of these control bits is 1. Therefore, bit [16] is the actual pattern bit fed into the test target.	
12:8 (R/W)	SEL	Test Select. The TRNG_TEST.SEL bit field configures the number of the FRO to be tested, the value should be in the range of 0 to 7.	
6 (R/W)	PPROC	Test Post Proc. When the TRNG_TEST.PPROC bit is set, it provides direct access to the post-processor for known-answer tests (writing input data to the TRNG_INPUT[n] registers). While this bit is set, the TRNG can continue to generate entropy in the main LFSR and any buffered random data is preserved, to be loaded into the output registers as soon as this bit is reset to 0 again. The need clock output is forced active while this bit is set. For X9.31 post-processors, it is advisable to re-seed the post-processor after running known-answer tests as the original key and V values are modified to known values. This bit is only present when post-processing is available and can only be set to 1 when the TRNG_CTL.PPROCEN bit =1 and the test run poker (TRNG_TEST.RUNPKR) bit in this register is 0.	
5 (R/W)	RUNPKR	Test Run Poker. When the TRNG_TEST.RUNPKR bit is set, it provides direct access to the inputs of the Monobit, Run and Poker Test circuits (writing input data in chunks of 32 bits to the TRNG_INPUT0 register). While this bit is 1, the TRNG is not allowed to generate entropy but any buffered random data is preserved, to be loaded into the output registers as soon as this bit is reset to 0 again. The TRNG_STAT.NEEDCLK bit is forced active while this bit is 1. The Monobit, Run and Poker Test circuits are reset to their initial states on any change of this bit.	
4 (R/W)	CONTPKR	Continue Poker. When the TRNG_TEST.CONTPKR bit is set, Monobit Test and Poker Test keep running continuously by not resetting the Monobit count (TRNG_MONOBITCNT) and poker counters (TRNG_POKER[n] register) at the end of each 20,000 bits test block. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Do not continue poker test
		1	Continue poker test

Table 44-26: TRNG_TEST Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/W)	NOLFSRFB	No LFSR Feedback. When the TRNG_TEST.NOLFSRFB bit is set, it removes XNOR feedback from the main LFSR, converting it into a normal shift register for the XOR-ed outputs of the FROs (shifting data in on the LSB side). A 1 also forces the LFSR to sample continuously. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Keep XNOR feedback
		1	Remove XNOR feedback
2 (R/W)	PATTDET	Test Pattern Detect. When the TRNG_TEST.PATTDET bit is set, it repeatedly feeds test pattern (PATTERN) into the error detection circuit of the FRO selected by the test select (TRNG_TEST.SEL) field. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Do not repeat feed test pattern
		1	Repeat feed test pattern
1 (R/W)	PATTFRO	Test Pattern FRO. When the TRNG_TEST.PATTFRO bit is set, it repeatedly feeds test pattern (PATTERN) into the delay chain of the FRO selected by the test select (TRNG_TEST.SEL) field by forcing the corresponding FRO enable (FROEN) output LOW. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Do not repeat feed test pattern
		1	Repeat feed test pattern
0 (R/W)	ENOUT	Test Enable Out. When the TRNG_TEST.ENOUT bit is set, it enables the tst_fro_clk_out output, connecting to the FRO selected by the test select (TRNG_TEST.SEL) field. This bit can only be set to 1 when TRNG_CTL.TSTMODE =1.	
		0	Disable tst_fro_clk_out
		1	Enable tst_fro_clk_out

TRNG Post-Process "V" Value Registers

The TRNG_V[n] registers are used to load the V value used for post-processing (if available). These registers are write-only. Reads return the values of the other registers mapped at the same addresses.

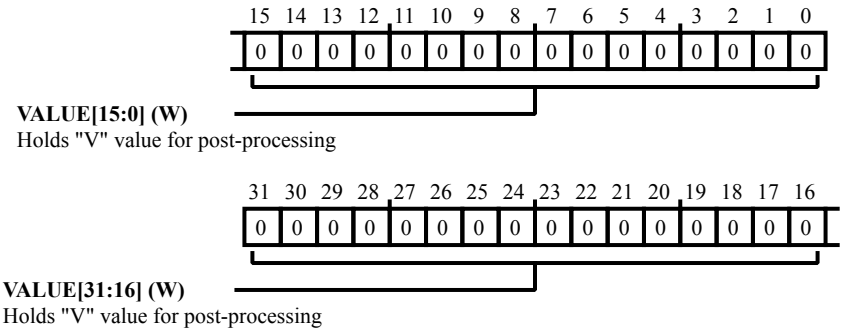


Figure 44-24: TRNG_V[n] Register Diagram

Table 44-27: TRNG_V[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Holds "V" value for post-processing. Bits of the post-processing 'V' value.

45 FIR Accelerator (FIR)

Finite Impulse Response (FIR) filters are frequently used in digital signal processing applications. The FIR accelerator is a dedicated hardware interface used to perform filter processing to reduce the instruction processing load on the core. FIR filters are used in a wide array of applications including multi-rate processing with an interpolator or decimator.

Features

This hardware module can perform FIR filters without core intervention. This allows programs to use the core to implement complex algorithms, effectively adding more bandwidth to the processor.

The FIR supports the following features:

- Fixed-point and 32-bit IEEE floating-point format
- Four SHARC+ core compatible MAC units that operate in parallel
- Rounding modes compatible with SHARC+ core MACs
- Single rate or multi-rate window processing
- Programmable rates with decimation or interpolation mode
- Up to 32 filter channels available in TDM in legacy mode
- Burst transfers on data or coefficient loads

NOTE: The FIR accelerator module has a local memory that the core cannot access during regular operation. Unlike previous SHARC processors, the FIR accelerator modules each have access to the system memory (on-chip or off-chip).

Also, unlike the previous SHARC processors, where only one of the FIR or IIR accelerators can be used at a time, the SHARC+ processor can use both accelerators simultaneously.

Clocking

The FIR accelerator runs at the speed of the core clock frequency (CCLK).

Functional Description

The ***FIR Block Diagram*** shows the 1024-TAP FIR hardware accelerator. The accelerator consists of a 1024 word coefficient memory, a 1024 deep delay line for data, and four MAC units. The accelerator runs at /CCLK frequency.

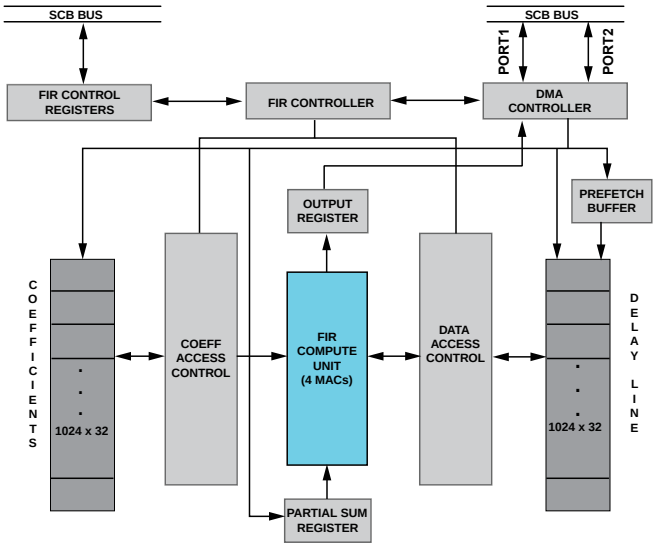


Figure 45-1: FIR Block Diagram

The FIR accelerator has the following logical sub blocks:

- A datapath unit that consists of:
  - A 1024 deep coefficient memory
  - A 1024 deep delay line for the data
  - Four 32-bit floating-point and fixed-point multiplier and adder units
  - One 32-bit prefetch buffer that operates in a pipelined fashion
  - One 32-bit buffer to hold the previous partial sum
  - One 32-bit buffer to hold the output
- Configuration registers for the number of TAPs, number of channels, filter enable, interrupt control, DMA enable, up sample or down sample control, and ratios.
- Core access interface for writing to the DMA and filter configuration registers and reading the status register.
- DMA bus interface for transferring data and coefficients to and from the accelerator.
- DMA configuration registers including chain pointer, input, output, and coefficient registers.

The accelerator block is integrated with the SHARC core. For more information about this, see the SCB Block Diagram.

ADSP-SC59x FIR Register List

The FIR accelerator is a dedicated hardware interface used to perform filter processing to reduce the instruction processing load on the core.

Table 45-1: ADSP-SC59x FIR Register List

Name	Description
FIR_CHNPTR	FIR Chain Pointer Register
FIR_COEFCNT	FIR Coefficient Count Register
FIR_COEFIDX	FIR Coefficient Index Register
FIR_COEFMOD	FIR Coefficient Modifier Register
FIR_CTL1	FIR Global Control Register
FIR_CTL2	FIR Channel Control Register
FIR_DBG_ADDR	Debug Address Register
FIR_DBG_CTL	FIR Debug Control Register
FIR_DBG_RDDAT	FIR Debug Data Read Register
FIR_DBG_WRDAT	FIR Debug Data Write Register
FIR_DMASTAT	FIR DMA Status Register
FIR_INBASE	FIR Input Data Base Register
FIR_INCNT	FIR Input Data Count Register
FIR_INIDX	FIR Input Data Index Register
FIR_INMOD	FIR Input Data Modifier Register
FIR_MACSTAT	FIR MAC Status Register
FIR_OUTBASE	FIR Output Data Base Register
FIR_OUTCNT	FIR Output Data Count Register
FIR_OUTIDX	FIR Output Data Index Register
FIR_OUTMOD	FIR Output Data Modifier Register
FIR_SCTL1	Software Control Register 1
FIR_SCTL2	Software Control Register 2
FIR_SGCTL	Secondary Global Control Register

ADSP-SC59x FIR Interrupt List

Table 45-2: ADSP-SC59x FIR Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
166	FIR0_DMA	FIR0 Core 1 DMA	Edge	
167	FIR0_STAT	FIR0 Core 1 Status	Edge	
176	FIR1_DMA	FIR1 Core 2 DMA	Edge	
177	FIR1_STAT	FIR1 Core 2 Status	Edge	
251	FIR0_BUS_ERR	FIR0 Core 1 FIR Bus Error	Edge	
256	FIR1_BUS_ERR	FIR1 Core 2 FIR Bus Error	Edge	

ADSP-SC59x FIR Trigger List

Table 45-3: ADSP-SC59x FIR Trigger List Generators

Trigger ID	Name	Description	Sensitivity
31	C1_FIR0_DMA	FIR0 Core 1 DMA	Edge
32	C2_FIR0_DMA	FIR1 Core 2 DMA	Edge

Table 45-4: ADSP-SC59x FIR Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
21	C1_FIR0_TRGI	FIR0 Core 1 FIR Wait on Trigger Input	Pulse
22	C2_FIR0_TRGI	FIR1 Core 2 FIR Wait on Trigger Input	Pulse

Compute Block

The MAC unit, shown in the ***FIR MAC Unit*** figure, has four multiply accumulators. The accumulators operate simultaneously on a single filter as described below.

- The MAC unit operates on the data and coefficient fetched from the data and coefficient RAMs
- Each MAC can perform 32-bit floating-point or 32-bit fixed-point MAC operations
- Floating-point format is IEEE-compliant
- Multiply and accumulation operation (addition) are pipelined
- A 32-bit floating-point MAC operation generates 32-bit multiply results
- A 32-bit fixed-point operation generates 80-bit results (64-bit result + 16 guard bits)

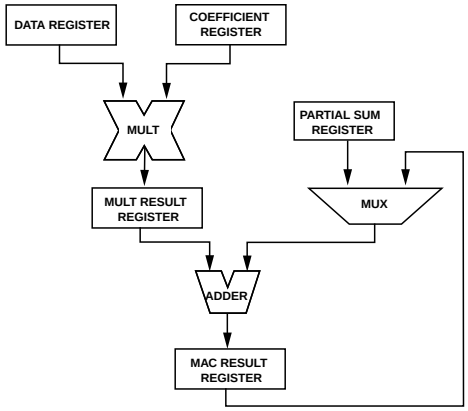


Figure 45-2: FIR MAC Unit

Partial Sum Register

The partial sum register is useful for Floating-point Multi-Iteration mode. For a particular channel, the intermediate MAC result is written to the output buffer of the system memory (on-chip or off-chip). If the same channel is requested again, the partial result register is updated with the intermediate MAC result through DMA from the output buffer of the system memory. The result is added to the current MAC result after each iteration. This process repeats until all iterations complete (the entire soft filter length is processed).

Delay Line Memory

The accelerator has a 1024 TAP delay line to hold the data locally. The DMA controller fetches the data from system memory and loads it into the delay line. Four read accesses can be made to the delay line simultaneously.

Coefficient Memory

The accelerator has a 1024 deep coefficient memory to store the coefficients. The DMA controller loads the coefficients from system memory into coefficient memory. Four coefficients can be fetched from the coefficient memory simultaneously. If the soft filter length is more than 1024, processing happens in multi-iteration mode.

Prefetch Data Buffer

The prefetch data buffer enables pipeline operation. One data sample is prefetched when the compute unit is operating on the delay line corresponding to the current sample. The data prefetched in this buffer is later used to update the delay line for the next sample. This operation happens in parallel again when the compute unit is not accessing the delay line. In other words, it happens when the compute unit is adding the output from the four MACs and the partial sum register.

Table 45-5: Pipeline Operation for Window Size = 1

Cycles	1	2	3	4	5	6
Output DMA			N	N1	N2	N3
Compute		N	N1	N2	N3	

Table 45-5: Pipeline Operation for Window Size = 1 (Continued)

Cycles	1	2	3	4	5	6
Input DMA	N	prefetch N1	prefetch N2	prefetch N3		

Processing Output

The accelerator uses all four MACs simultaneously to calculate one output sample as shown in the ***Multi-Iteration Filtering Flow*** figure and the following procedure.

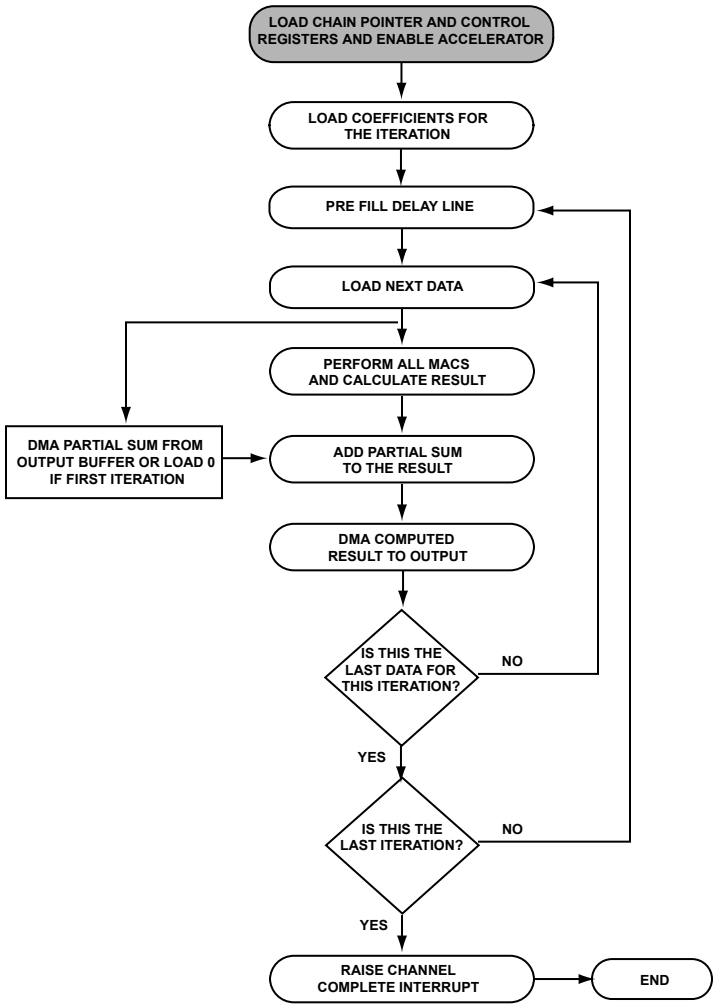


Figure 45-3: Multi-Iteration Filtering Flow

1. The accelerator fetches four input data from the delay line and four corresponding coefficients from the coefficient memory and feeds them to the MAC units for multiply and accumulation operations.
2. The accelerator repeats the procedure with the next four input data and coefficients until all the TAPs complete. For example, this procedure happens N/4 times for an N TAP filter.

3. When all the TAPs are complete, the accelerator adds the four MAC outputs together to the previous partial sum (if any) to calculate the final result.
4. Finally, that output sample is stored back in system memory.

System Memory Storage

The following sections describe the storage format for the accelerator.

CAUTION: Store all data in memory aligned to the word address boundaries. Any other programmed addresses do not flag an error.

Coefficients and Input Buffer Storage

For any N TAP filter with coefficients:

```
C[i] i = 0,1,
...
N - 1
```

store the coefficients in system memory buffer in the order:

```
C[N - 1], C[N - 2]
...
C[1],C[0]
```

and CI should point to `C(N - 1)`.

Single Rate Input Filtering

The total size of the input buffer must at least be equal to `N - 1 + W`. If the input buffer that needs to be processed is:

```
 x[n],x[n+1],x[n+2]
 ...
 x[n+W-1]
```

store it in the memory as

```
x[n-(N-1)], x[n-(N-2)]
...
x[n-1], x[n], x[n+1]
...
x[n+W-1]
```

and the `FIR_INIDX` register value should point to `x[n - (N - 1)]`

Decimation

Assuming M = decimation ratio, the total size of the input buffer should at least be equal to `N - 1 + W x M`. If the input buffer that needs to be processed is:

```
x[n],x[n+1],x[n+2]....x[n+WxM-1],
```

store it in the memory as:

```
x[n-(N-1)], x[n-(N-2)]....x[n-1],
x[n], x[n+1]....x[n+WxM-1]
```

and the FIR_INIDX register value should point to x[n-(N-1)].

Interpolation

Assuming L= interpolation ratio, the total size of the input buffer should be at least equal to:

Ceil ((N-1)/L)+W/L.

If the input buffer that needs to be processed is:

```
x[n], x[n+1], x[n+2]....x[n+W/L-1],
and K = Ceil((N-1)/L)
```

store it in the memory as:

```
x[n-k], x[n-(K-1)], x[n-(K-2)]....x[n-1],
x[n], x[n+1].... x[n+W/L-1]
```

and the FIR_INIDX register value should point to x[n-K].

Operating Modes

The FIR core performs a sum-of-products operation to compute the convolution sum. It supports single-rate, decimation, and interpolation functions.

Single Rate Processing

In a single-rate filter, the output result rate is equal to the input sample rate. The filter output Y(n) is computed according to following equation where N is the number of filter coefficients: c(k) k = 0 to N – 1 are the filter coefficients and x(n) represents the input time-series.

$$Y(n)= \sum_{k=0}^{N-1} c(k) \cdot x(n-k)$$

Figure 45-4: Filter Output Calculation

Single Iteration

Results are computed in a single iteration when the soft filter length is less than or equal to 1024.

Floating-point Multi-Iteration

Results are computed in multiple iterations when the soft filter length is greater than 1024 (for example, 2048 TAPs on a 1024 hard filter length). In this mode, the accelerator implements two iterations of 1024 TAPs.

NOTE: If the soft filter length is not a multiple of the hard filter length, the accelerator iterates until the soft filter length is satisfied.

Example: 550 taps on a 256 tap filter. In this example, the FIR accelerator implements two iterations of 256 taps and one iteration of 38 taps.

NOTE: Multi-iteration mode is not supported in fixed-point format.

Window Processing

In window-based mode, multiple output samples (up to 1024) equal to the window size of that channel are calculated. After these calculations are complete, the accelerator begins processing the next channel. A configurable window size parameter is provided to specify the length of the window.

To select sample-based processing mode, configure the window size to 1. In this mode, one sample from a particular channel is processed through all the taps of that channel and the final output sample is calculated.

Multi-Rate Processing

Multi-rate filters change the sampling rate of a signal—they convert the input samples of a signal to a different set of data that represents the same signal sampled at a different rate.

Decimation

A decimation filter provides a single output result for every M input samples, where M is the decimation ratio. The output rate is 1/M'th of the input rate. The filter implementation exploits the low output sample rate by not starting a computation until a new set of M input samples is available.

In this mode, after low-pass filtering (for anti-aliasing), FIR logic discards the ratio – 1 samples of output data. For performance optimization, FIR logic skips the computation of output samples, which are discarded.

The input buffer size for decimation filters is N – 1 + (W × M) where:

- N is the number of taps
- W is the window size
- M is the decimation ratio

The window size (`FIR_CTL2.WINDOW` bits) must be programmed with the number of output samples.

To start this mode, programs set the `FIR_CTL2.RATIO` and `FIR_CTL2.UPSAMP` bits (along with normal filter setting). Also, the `FIR_CTL2.TAPLEN` bit field value should be greater than or equal to the `FIR_CTL2.RATIO` bit field value for the decimation filter.

Interpolation

An interpolation filter provides L output results for each new input sample, where L is the interpolation ratio. The output rate is L times the input rate.

In this mode, according to the ratio specified in configuration register, FIR logic inserts L – 1 zeros between any two input samples (upsampling). It then performs the interpolation (through the FIR filter).

Both upsampling and downsampling do not support multi-iteration mode. Therefore, the filtering operation only happens on up to 1024 TAPs and the ratio of up and downsampling can only be an integer value.

In an interpolation filter, FIR logic inserts L – 1 zeros between each sample. The program must ensure that these zeros fully shift out of the delay line before moving on to the next channel. This operation puts a restriction on window size in terms of L – *the sample ratio* as showing in the expression:

WINDOWSIZE = *n* × *SAMPLERATIO*, where *n* is the number of input samples.

The input buffer size is smallest integer greater than or equal to (N – 1 + W) L for interpolation filters where:

- N is the number of taps
- W is the window size
- L is the interpolation ratio

To start the mode, programs configure the `FIR_CTL2.RATIO` and `FIR_CTL2.UPSAMP` bits (along with filter settings).

Floating-Point Data Format

The FIR accelerator treats data and coefficients in 32-bit floating-point format as the default functional mode.

Fixed-Point Data Format

In fixed-point mode, the 32-bit input data or coefficient is treated as fixed-point. A 32-bit fixed-point MAC operation generates an 80-bit result. Fixed-point data or coefficients can be unsigned integer, unsigned fractional and signed integer.

NOTE: In fixed-point mode, the entire 80-bit result register is always written back in bursts of 3 × 32 bits. The first word is the LSW, the second word is the MSW, and the third word is a 16-bit overflow. The remaining 16 bits are padded with zeros. Therefore, for fixed-point mode: WINDOWSIZE = WINDOWSIZE × 3.

If the signed fractional format is used, the output must be scaled by 2. The MAC does not right shift to remove the redundant sign bit. A final routine must decimate the output buffer to the desired samples.

Multi-iteration mode is not supported in this format. Therefore, the maximum TAP length is 1024.

Auto Configuration Mode (ACM)

The accelerator can be operated in legacy mode or Auto Configuration Mode (ACM), controlled by the `FIR_CTL1.ACM` bit. The default functional mode is legacy mode. The accelerator mode can only be changed when the accelerator is disabled.

The ACM provides the following additional features:

- Halt and queuing

The core may pause the current Transfer Control Block (TCB) chain being processed by setting the FIR_CTL1.HALT bit. The accelerator acknowledges the core by setting the FIR_DMASTAT.HALT_STAT bit. The core can appropriately take action to submit/insert new TCB's. After the core acts, the accelerator processing can be resumed by clearing the FIR_CTL1.HALT bit. Before halting the accelerator, if the initial TCB chain is processed, the accelerator comes to the idle state. In this case, the accelerator has to be disabled and enabled by toggling the FIR_CTL1.EN bit and the FIR_CTL1.HALT bit must be cleared to resume processing.

- No channel number limitation

  Unlike legacy mode, there is no fixed channel number limitation and, therefore, the accelerator ignores the value programmed in the FIR_CTL1.CH field. The application can queue an unlimited number of channels/TCBs dynamically and the accelerator keeps processing the TCBs until the chain pointer becomes null.

- Selective interrupt

  The core can enable/mask interrupt generation for each channel using the FIR_CTL2.IMASK bit. If the bit is cleared, an interrupt is generated after completion of the channel.

- Selective controller / target trigger

  The core can enable/mask trigger generation by the accelerator after the end of processing of each channel using the FIR_CTL2.TMASK bit. The accelerator can also wait for a trigger after loading the TCB and coefficients and before processing a channel for which the FIR_CTL2.TWAIT bit is set.

In addition to the above features, there are three additional fields as part of the TCB FIR_SCTL1, FIR_SCTL2, and FIR_SGCTL registers. The FIR_SGCTL register can be used to change the FIR_CTL1 parameters such as rounding mode and fixed-point mode for each channel. The FIR_SCTL1 and FIR_SCTL2 registers can be used as general-purpose registers.

Data Transfer

The FIR filter works exclusively through DMA.

Chain Assignment

The structure of a TCB is conceptually the same as a traditional linked list. Each TCB has several data values and a pointer to the next TCB. Further, the chain pointer of a TCB can point to itself to continuously re-run the same DMA. The FIR accelerator reads each word of the TCB and loads it into the corresponding register. The end of the chain (no further TCBs are loaded) is indicated by a TCB with a chain pointer register value of zero.

The FIR accelerator DMA supports circular buffer chained DMA. The FIR accelerator does not support circular buffering for the coefficient buffer.

Table 45-6: TCBs for Chained DMA in Legacy Mode

Address	Register
TCB	FIR_CHNPTR
TCB + 0x1	FIR_COEFCNT
TCB + 0x2	FIR_COEFMOD
TCB + 0x3	FIR_COEFIDX
TCB + 0x4	FIR_OUTBASE
TCB + 0x5	FIR_OUTCNT
TCB + 0x6	FIR_OUTMOD
TCB + 0x7	FIR_OUTIDX
TCB + 0x8	FIR_INBASE
TCB + 0x9	FIR_INCNT
TCB + 0xA	FIR_INMOD
TCB + 0xB	FIR_INIDX
TCB + 0xC	FIR_CTL2

Table 45-7: TCBs for Chained DMA in Auto Configuration Mode

Address	Register
TCB	FIR_CHNPTR
TCB + 0x1	FIR_SCTL1
TCB + 0x2	FIR_SCTL2
TCB + 0x3	FIR_SGCTL
TCB + 0x4	FIR_COEFCNT
TCB + 0x5	FIR_COEFMOD
TCB + 0x6	FIR_COEFIDX
TCB + 0x7	FIR_OUTBASE
TCB + 0x8	FIR_OUTCNT
TCB + 0x9	FIR_OUTMOD
TCB + 0xA	FIR_OUTIDX
TCB + 0xB	FIR_INBASE
TCB + 0xC	FIR_INCNT
TCB + 0xD	FIR_INMOD
TCB + 0xE	FIR_INIDX
TCB + 0xF	FIR_CTL2

The FIR_COEFCNT register is loaded with the values in the FIR_COEFCNT TCB field and is decremented from that value onwards. However, coefficient loading continues until the number of coefficients, equal to the tap length, are read. This condition is true even if the FIR_COEFCNT register reaches zero as in the case of a tap length = 10, and the FIR_COEFCNT bit field in the TCB is initialized to 0. The value in the FIR_COEFCNT register is –10 after all coefficients are loaded.

NOTE: Initialize the FIR_CHNPTR register to TCB + 12 in legacy mode and TCB + 15 in ACM mode.

DMA Access

The FIR accelerator has two DMA channels (accelerator input and output) to connect to the system memory. The DMA controller fetches the data and coefficients from memory and stores the result.

Burst Access Support

Burst access enhances the throughput of the DMA channel and reduces the overall load on the system fabric. Burst support is provided for TCB, data, and coefficient loads on the DMA channel. The FIR module supports burst transfers of size SINGLE, INCR4 INCR8 and INCR16. The DMA access are automatically split into these four types of burst by the FIRA accelerator for optimum performance. For a TCB load, a burst access of length 16 is issued which reduces the TCB load time.

An additional bit (FIR_CTL1.BURSTEN) enables the burst feature. Burst transfers are always of size SINGLE when the modifier is other than 1. In cases of burst transfers around the circular buffer boundary, the design ensures that the burst does not cross the buffer boundary.

Data and Coefficient Load in Parallel

To reduce the overall loading time of data and coefficients, the operation occurs over separate channels. After the TCB load, both data and coefficients are loaded in parallel over two DMA channels (CH0 and CH1). If there is no conflict in the system when accessing the data and coefficients, the loading time is significantly improved over when loaded sequentially using a single channel. This mode is enabled by default and can be disabled by setting the FIR_CTL1.DCP_DIS bit. When disabled, coefficient load and data load occur sequentially over a single channel (CH0).

Prefetch Buffer for DMA Channels

An access latency exists in every burst transfer for the address issued for the first data to arrive. To reduce the latency, both channels of the FIR contain a prefetch buffer. When one burst access is in-progress, the FIR prefetch buffer receives the next burst, which reduces the latency for consecutive accesses. The FIR prefetch buffer is disabled by default. Setting the FIR_CTL1.PFB_EN bit enables the prefetch buffer.

Accelerator TCB

The location of the DMA parameters for the next sequence comes from the chain pointer register that points to the next set of DMA parameters stored in the internal memory of the processor. In chained DMA operations, the accelerator automatically initializes and starts another DMA transfer when the current DMA transfer is complete. Each new set of parameters is stored in a user-initialized memory buffer or TCB for a chosen peripheral.

Chain Pointer DMA

The DMA controller supports circular buffer chain pointer DMA. One TCB must be configured for each channel. The DMA controller contains the following:

- A control register value to configure the filter parameters (such as filter tap length, window size, sample rate conversion settings) for each channel. In ACM mode, additional parameters such as interrupt mask, trigger mask, and trigger wait are also available.
- Software control register values in ACM mode for each channel.
- Secondary control register value to configure rounding mode, fixed-point mode, and two's complement for each channel.
- DMA parameter register values for the input data (delay line).
- DMA parameter register values for coefficient load.
- DMA parameter register values for output data.

Intermediate results in multi-iteration mode are saved in the output buffer.

As shown in the ***Circular Buffer Addressing*** figure, the accelerator loads the TCB into its internal registers and uses these values to fetch coefficients and data and to store results. After processing a window of data for any channel, the accelerator writes back the appropriate values to the `FIR_INIDX` and `FIR_OUTIDX` bit fields of the TCB in memory. Then, data processing can begin from where it left off during the next time slot of that channel.

The write-back value for input buffer is:

- `FIR_INIDX` + W for single rate filtering
- `FIR_INIDX` + W × M for decimation (M = decimation ratio)
- `FIR_INIDX` + W/L for interpolation (L = interpolation ratio)
- The write-back value for output buffer in floating point mode is: `FIR_OUTIDX` + W
- The write-back value for output buffer in fixed-point mode is: `FIR_OUTIDX` + 3 × W

NOTE: The `FIR_CTL2` register is part of the FIR TCB. This configuration allows programming individual FIR channels with different control attributes.

The above index updates are valid only for legacy mode. In ACM mode, the FIR_INIDX and FIR_OUTIDX bit fields of the TCB in memory are updated to 0x00000000 and 0xFFFFFFFF by the accelerator after processing a window of data.

In ACM mode, when the FIR_CTL1.SMQ_LIUPS_EN bit is set, the accelerator updates the FIR_INIDX and FIR_OUTIDX bit fields of the TCB in memory after processing a window of data and according to the circular buffer scheme. When the FIR_CTL1.SMQ_LIUPS_EN bit is cleared, the accelerator updates the FIR_INIDX and FIR_OUTIDX bit fields of the TCB in memory to 0x00000000 and 0xFFFFFFFF after processing a window of data. The FIR_CTL1.SMQ_LIUPS_EN bit is only valid in ACM mode.

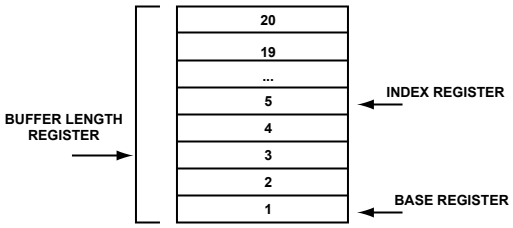


Figure 45-5: Circular Buffer Addressing

Programming Model

The following sections provide general programming information for the FIR accelerator.

Legacy Mode

The ***Wait for Core Intervention ≥ Idle (if auto channel iterate bit = 0)*** figure shows the diagram for multichannel filtering. Multiple channels are processed in a time division multiplexed (TDM) format. After completing all the channels, the accelerator can either repeat the slots or wait for core intervention.

For multichannel filtering, use the following steps.

1. Program the number of channels using the FIR_CTL1.CH bits.
2. Configure the TCBs in system memory with one channel's TCB pointing to the next channel's TCB.
3. Write the first TCB value into the FIR_CHNPTR register and enable the accelerator.
    a. The accelerator fetches the first channel's TCB and, using it as pointer, prefills the delay line and coefficient memory and loads the FIR_CTL2 register to configure the filter parameters corresponding to that channel.
    b. The accelerator then calculates output samples corresponding to one window and stores the data back in internal memory.
    c. At the end of the window the accelerator updates the FIR_INIDX and FIR_OUTIDX registers in the TCB of system memory and moves to the next channel.

d. When all the channels are finished and the auto channel iterate bit (`FIR_CTL1.CAI`) =1, the accelerator processes the first channel again and iterates through the channels. If the `FIR_CTL1.CAI` bit =0, the accelerator waits for core intervention.

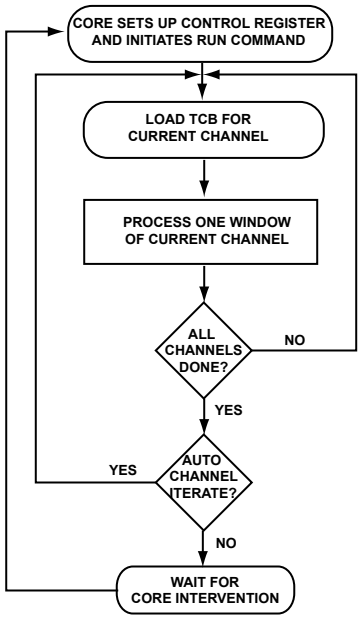


Figure 45-6: Wait for Core Intervention ≥ Idle (if auto channel iterate bit = 0)

NOTE: All the addresses programmed in the TCB correspond to 32-bit address boundaries and should not contain the lower 2 bits (assumed as zeros).

Auto Configuration Mode (ACM)

The figure shows multi-channel filtering in ACM. Multiple channels are processed in a time multiplexed format (TDM).

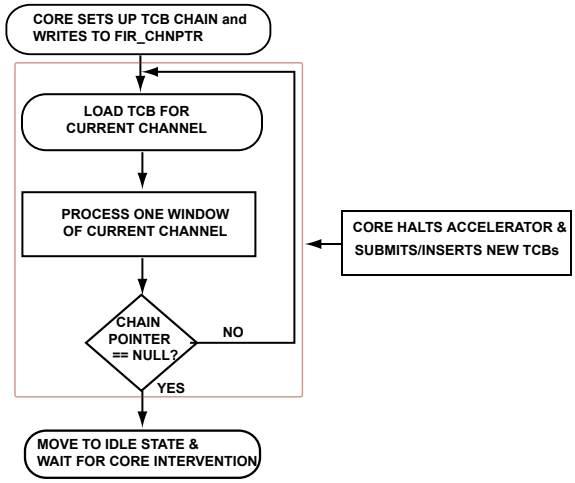


Figure 45-7: Multi-Channel Filtering in Auto Configuration Mode

For multi-channel filtering:

1. Configure the Transfer Control Blocks (TCBs) in system memory with one channel's TCB pointing to the next channel's TCB. There is no limit on the number of channels to be configured.
2. Write the first TCB value to the FIR_CHNPTR register and enable the accelerator.

   The accelerator fetches the first channel's TCB. Using it as pointer, it prefills the delay line and coefficient memory and loads the FIR_CTL2 and FIR_SGCTL registers to configure the filter parameters corresponding to that channel.

   When the FIR_CTL2.TWAIT bit is set, the accelerator waits for a input trigger to start the window processing for the channel.

   The accelerator then calculates output samples corresponding to one window and stores the data back in internal memory. When the FIR_CTL2.TMASK bit is cleared the accelerator sends an output trigger after completion of processing of the channel.

   When the FIR_CTL2.IMASK bit is cleared, the accelerator interrupts the core after completion of processing of a particular channel.

   At the end of the window, the accelerator updates the FIR_INIDX and FIR_OUTIDX registers to 0x00000000 and 0xFFFFFFFF in the TCB of system memory and moves to the next channel.
3. At any instant, as required, the core halts the accelerator. It sets the FIR_CTL1.HALT bit and appropriately takes action to submit or insert new TCBs and clears the FIR_CTL1.HALT bit to resume channel processing.

   If the FIR_CHNPTR register is zero (last channel is being processed or channel processing is complete) after halting the accelerator, the FIR_CHNPTR register is appropriately written before resuming the accelerator channel processing.

   When the accelerator is idle after halting, FIR_CTL1.EN bit is toggled to disable and re-enable the accelerator and the FIR_CTL1.HALT bit is cleared, and the accelerator resumes channel processing.
4. The accelerator continues processing until all the channels are complete. Repeat *Step 3* (if required) to submit/insert new channels.

NOTE: Channel auto iterate (FIR_CTL1.CAI) is not supported in ACM.

Debug Mode

The next sections show the steps required for reading and writing local memory in debug mode.

Write to Local Memory

1. Clear the FIR_CTL1.DMAEN bit.
2. Set the FIR_DBG_CTL.EN, FIR_DBG_CTL.MEM, and FIR_DBG_CTL.HLD bits.
3. Set the FIR_DBG_CTL.ADRINC bit for address auto increment.

4. Write the start address to the FIR_DBG_ADDR register.

   NOTE: If bit 11 in the FIR_DBG_ADDR register is set, coefficient memory is selected.

5. Write data to the FIR_DBG_WRDAT register.

Read from Local Memory

1. Clear the FIR_CTL1.DMAEN bit.
2. Set the FIR_DBG_CTL.EN, FIR_DBG_CTL.MEM, and FIR_DBG_CTL.HLD bits.
3. Set the FIR_DBG_CTL.ADRINC bit for address auto increment.
4. Write the start address to the FIR_DBG_ADDR register.

   NOTE: If bit 11 in the FIR_DBG_ADDR register is set, coefficient memory is selected.

5. Read data from the FIR_DBG_RDDAT register.

Single-Step Mode

Single-step mode can be used for debug purposes. An extra debug register is used in this mode.

1. Enable stop DMA during breakpoint hit in the emulator settings.
2. Clear the FIR_DBG_CTL.HLD bit and enable the FIR_DBG_CTL.EN and FIR_DBG_CTL.RUN bits.
3. Program the FIR module according to the application.
4. In single-step mode, each iteration is updated in the emulator session.

Computing FIR Output, Tap Length Greater than 4096

With little core intervention, the FIR accelerator can also be used to calculate output for a tap length greater than 4096 taps. The section demonstrates the calculation with an example of 8192 taps.

1. Divide the transfer function of an 8192 FIR filter into two 4096 FIR filters:

   $H(Z) = b_0 + b_1Z^{-1} + b_2Z^{-2} + ...b_{4095}Z^{-4095} + b_{4096}Z^{-4096}b_{4097}Z^{-4097} + ...b_{8191}Z^{-8191}$

   $= b_0 + b_1Z^{-1} + b_2Z^{-2} + ...b_{4095}Z^{-4095} + Z^{-4096}(b_{4096} + b_{4097} + ...b_{8191}Z^{-4096})$

2. Divide the filter coefficients of an 8192 tap filter among two 4096 tap FIR filters:

   Filter 1

   Coefficients = b0, b1, b2,……, b4095

   Input data = x[n], x [n – 1],…….x[n – 4095]

   Filter 2

   Coefficients = b4096, b4097,……..,b8191

Input data = x[n – 4096], x[n – 4097],……x[n – 8191]

The accelerator can be used in two-channel mode where:

- channel 1 operates on x[n]…x[n – 4095] input data with the filter coefficients of filter 1 and
- channel 2 operates on x[n – 4096]…x[n – 8191] with the filter coefficients of filter 2.

Once both the channels are processed, add the partial sum output of both the channels to get the final output. Implement this approach (tap length = TAPS = 8192, window size = WINDOW) using the following programming steps.

1. Create a circular input data buffer in system memory (IBUF). The buffer must be large enough to avoid overwriting data before the accelerator processes it. Ideally, the input buffer size for a channel is TAPS + WINDOW – 1.
2. Create a coefficient buffer of size TAPS (8192) (CBUF).
3. Create one output buffer of size WINDOW (OBUF) and another temporary output buffer (OBUF1) to store the partial sum.
4. Create two TCBs in system memory with first TCB chained to the second and second chained to the first in a circular manner.
    a. Point the `FIR_COEFIDX` bit field of the first TCB to the start address of the coefficient buffer (CBUF) and that of the second TCB to 4096 offset from the start of the coefficient buffer (CBUF + 4096).
    b. Point the `FIR_OUTBASE` and `FIR_OUTIDX` bit fields of the first TCB to the start address of OBUF and that of the second TCB to the start address of OBUF1.
    c. Point the `FIR_INIDX` bit field of the first TCB to the start address of IBUF and that of the second TCB to 4096 offset from the start address of IBUF.
    d. Configure the `FIR_CTL2` bit field of both the TCB for tap length = TAP/2 = 4096 and window size = WINDOW.
5. Initialize the `FIR_CHNPTR` register to point to the first TCB.
6. Program the `FIR_CTL1` register to initiate the accelerator processing by setting the `FIR_CTL1.EN` and `FIR_CTL1.DMAEN` bits and the number of channels configured as 2.
7. Wait for the FIR all channels done interrupt (`FIR_DMASTAT.ACDONE`) to occur. Inside the ISR, add the partial sum results using the core from both the output buffers (OBUF and OBUF1) to get the final output. To save memory, replace the contents of the buffer OBUF with the final output result.

Debug Features

The following sections provide information for debugging the FIR accelerator.

Local Memory Access

The contents of FIR delay line and coefficient memories are made observable for debug by setting the FIR_DBG_CTL.EN/FIR_DBG_CTL.MEM and FIR_DBG_CTL.HLD bits. The debug address register (FIR_DBG_ADDR) and two data registers are provided for debug operations. Bit 11 of the FIR_DBG_ADDR register selects coefficient memory when set (=1) and selects delay line memory when cleared (=0).

In the debug mode, the read data register (FIR_DBG_RDDAT) returns the contents of the memory location pointed to by the address register. Data can be written into any memory location using FIR_DBG_WRDAT register writes. If the address auto-increment bit (FIR_DBG_CTL.ADRINC) is set, the address register auto-increments on FIR_DBG_WRDAT writes and FIR_DBG_RDDAT reads. During auto-increment the FIR_DBG_ADDR register cannot cross the data memory or coefficient memory boundary.

Single-Step Mode

Programs can single step through the MAC operations and observe the memory contents after each step. The FIR_DBG_CTL.EN, FIR_DBG_CTL.HLD, and FIR_DBG_CTL.MEM bits control the FIR MAC units.

Emulation Considerations

In FIR debug mode, the DMA operations are not observable.

Interrupts

The ***FIR Interrupt Overview*** table provides the source of interrupt and service instructions for the FIR interrupts.

Table 45-8: FIR Interrupt Overview

Accelerator Mode	Default Programmable Interrupt	Sources	Masking	Service
Legacy Mode	FIR_DMA	Window Complete	N/A	ROC from FIR_DMASTAT + RTI instruction
		All channels complete		
	FIR_STAT	MAC IEEE floating point exceptions	N/A	ROC from FIR_MACSTAT + RTI instruction
		MAC fixed point overflow		
Auto Configuration Mode	FIR_DMA	Window Complete	FIR_CTL2.IMASK	ROC from FIR_DMASTAT + RTI instruction
	FIR_STAT	MAC IEEE floating point exceptions	N/A	ROC from FIR_MACSTAT + RTI instruction
		MAC fixed point overflow	N/A	

Sources

The FIR module drives two interrupt signals: FIR_DMA for the DMA status and FIR_STAT for the MAC status. The FIR module generates interrupts as described in the following sections.

Window Complete

This interrupt is generated at the end of each channel when all the output samples are calculated corresponding to a window and updated index values are written back.

In legacy mode, if the FIR_CTL1.CCINTR bit is set, an interrupt is generated after completion of window processing of each channel. In ACM, the interrupt generation can be selectively masked using the FIR_CTL2.IMASK bit for each channel.

All Channels Complete

This interrupt is generated when all the channels are complete or when one iteration of time slots completes. Note that the interrupt follows the access completion rule, where the interrupt is generated when all data are written back to system memory.

The all channels complete interrupt source is only applicable in legacy mode. If the FIR_CTL1.CCINTR bit is not set in legacy mode, the all channel complete interrupt is generated. In ACM mode, interrupt generation for each channel is controlled using the FIR_CTL2.IMASK bit.

NOTE: The FIR_CTL1.CCINTR bit is valid only in legacy mode. In ACM mode, interrupt generation for each channel is controlled using the FIR_CTL2.IMASK bit.

MAC Status

A MAC status interrupt is generated under the following conditions and is reflected in the FIR_MACSTAT register.

- Multiplier result zero – Set if multiplier result is zero
- Multiplier result infinity – Set if multiplier result is infinity
- Multiply invalid – Set if multiply operation is invalid
- Adder result zero – Set if adder result is zero
- Adder result infinity – Set if adder result is infinity
- Adder invalid – Set if addition is invalid
- Adder overflow – for fixed-point operation

Service

The DMA interrupt status bits are sticky and are cleared when the DMA status register is read. When a MAC status interrupt occurs, programs can find this state by reading the MAC status register (FIR_MACSTAT). The read of the register clears the (sticky) MAC interrupt status bits.

The status interrupt sources are derived from the FIR_MACSTAT register. A status interrupt can occur due to the last set of MAC operations of a processing iteration that correspond to a particular channel. The interrupt is generated continuously and cannot be stopped, even after disabling the accelerator. The interrupt can only be stopped when another processing iteration results in a non-zero or valid multiply or add result.

ADSP-SC59x FIR Register Descriptions

The FIR accelerator (FIR) contains the following registers.

Table 45-9: ADSP-SC59x FIR Register List

Name	Description
FIR_CHNPTR	FIR Chain Pointer Register
FIR_COEFCNT	FIR Coefficient Count Register
FIR_COEFIDX	FIR Coefficient Index Register
FIR_COEFMOD	FIR Coefficient Modifier Register
FIR_CTL1	FIR Global Control Register
FIR_CTL2	FIR Channel Control Register
FIR_DBG_ADDR	Debug Address Register
FIR_DBG_CTL	FIR Debug Control Register
FIR_DBG_RDDAT	FIR Debug Data Read Register
FIR_DBG_WRDAT	FIR Debug Data Write Register
FIR_DMASTAT	FIR DMA Status Register
FIR_INBASE	FIR Input Data Base Register
FIR_INCNT	FIR Input Data Count Register
FIR_INIDX	FIR Input Data Index Register
FIR_INMOD	FIR Input Data Modifier Register
FIR_MACSTAT	FIR MAC Status Register
FIR_OUTBASE	FIR Output Data Base Register
FIR_OUTCNT	FIR Output Data Count Register
FIR_OUTIDX	FIR Output Data Index Register
FIR_OUTMOD	FIR Output Data Modifier Register
FIR_SCTL1	Software Control Register 1
FIR_SCTL2	Software Control Register 2
FIR_SGCTL	Secondary Global Control Register

FIR Chain Pointer Register

The FIR_CHNPTR register contains the chain pointer address.

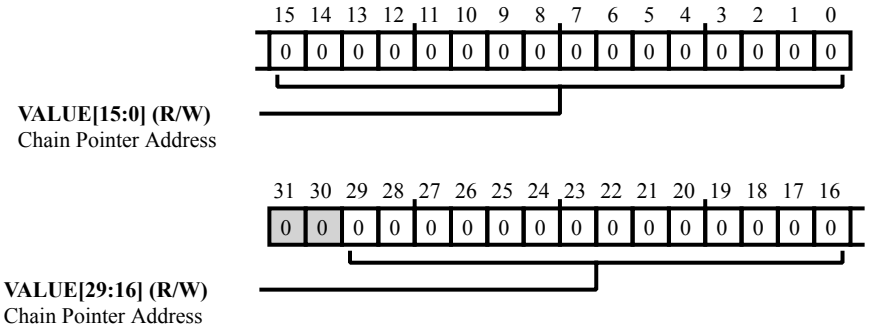


Figure 45-8: FIR_CHNPTR Register Diagram

Table 45-10: FIR_CHNPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Chain Pointer Address. The FIR_CHNPTR.VALUE bit field contains the chain pointer address.

FIR Coefficient Count Register

The FIR_COEFCNT register contains the 16-bit coefficient buffer count.

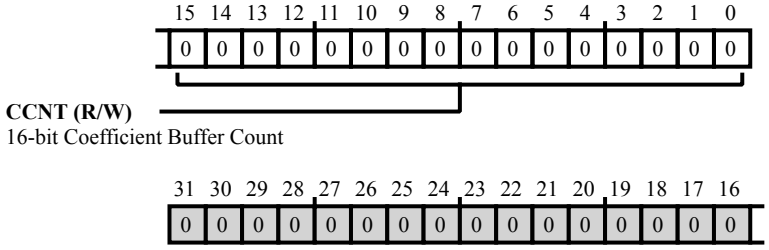


Figure 45-9: FIR_COEFCNT Register Diagram

Table 45-11: FIR_COEFCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	CCNT	16-bit Coefficient Buffer Count. The FIR_COEFCNT.CCNT bit field contains the 16-bit coefficient buffer count.

FIR Coefficient Index Register

The FIR_COEFIDX register contains the coefficient index word address with the lower two bits removed.

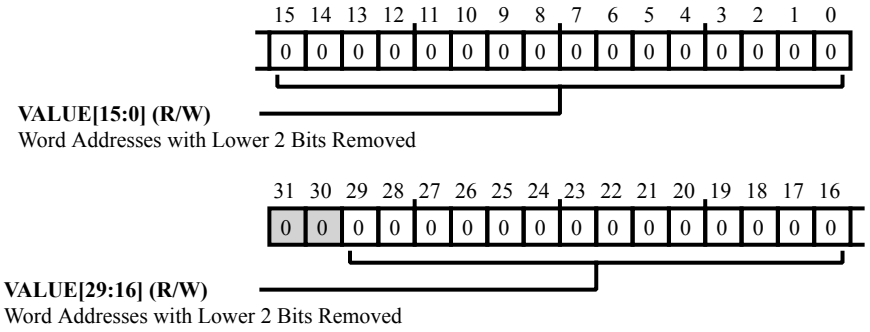


Figure 45-10: FIR_COEFIDX Register Diagram

Table 45-12: FIR_COEFIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Addresses with Lower 2 Bits Removed. The FIR_COEFIDX.VALUE bit field contains the word addresses with the lower 2 bits removed.

FIR Coefficient Modifier Register

The FIR_COEFMOD register contains the 16-bit coefficient index modifier.

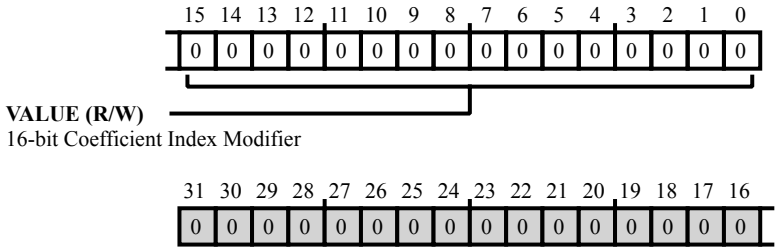


Figure 45-11: FIR_COEFMOD Register Diagram

Table 45-13: FIR_COEFMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Coefficient Index Modifier. The FIR_COEFMOD.VALUE bit field contains the 16-bit coefficient index modifier.

FIR Global Control Register

The FIR_CTL1 register is used to configure the global parameters for the accelerator. These parameters include the number of channels, channel auto iterate, DMA enable, and accelerator enable.

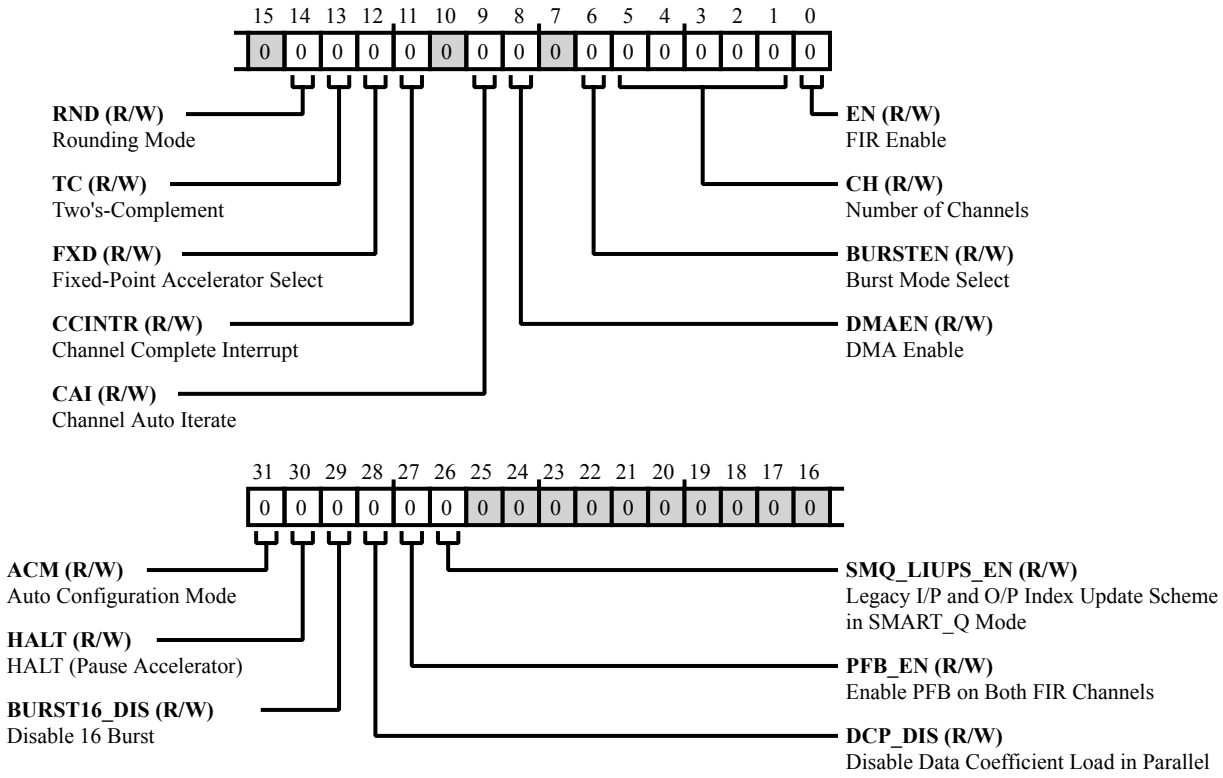


Figure 45-12: FIR_CTL1 Register Diagram

Table 45-14: FIR_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ACM	Auto Configuration Mode. The FIR_CTL1.ACM bit configures the mode for loading the TCB.	
		0	Legacy Mode. Load TCB without the FIR_CTL1 register. Core intervention may be required.
		1	Auto Configuration Mode. Load TCB with the FIR_CTL1 register. Core intervention is not required.

Table 45-14: FIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W)	HALT	HALT (Pause Accelerator). This bit is only valid in Auto Configuration mode (ACM)	
		0	Release. Release accelerator for further processing of data.
		1	Pause. Pause accelerator for core to check status and modify or submit jobs.
29 (R/W)	BURST16_DIS	Disable 16 Burst. The `FIR_CTL1.BURST16_DIS` bit configures the accelerator to disable 16 burst when in burst mode.	
		0	Maximum burst of INCR16 in burst mode for DMA and TCB
		1	Maximum burst of INCR8 in burst mode for DMA and non-burst for TCB
28 (R/W)	DCP_DIS	Disable Data Coefficient Load in Parallel. The `FIR_CTL1.DCP_DIS` bit enables data and coefficient loading to occur in parallel.	
		0	Enable
		1	Disable
27 (R/W)	PFB_EN	Enable PFB on Both FIR Channels. 0- FIR Pre-Fetch Buffer Disabled 1- FIR Pre-Fetch Buffer Enabled	
26 (R/W)	SMQ_LIUPS_EN	Legacy I/P and O/P Index Update Scheme in SMART_Q Mode. The `FIR_CTL1.SMQ_LIUPS_EN` bit configures the scheme the accelerator uses to update the input index (II) and output index (OI). This bit is only valid in ACM.	
		0	Update the II with all '0' and the OI with all 'F', respectively
		1	Update II and OI after circular buffer scheme
14 (R/W)	RND	Rounding Mode. The `FIR_CTL1.RND` bit configures the accelerator to use a rounding mode.	
		0	Round to Nearest
		1	Truncate (round towards zero)

Table 45-14: FIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
13 (R/W)	TC	Two's-Complement. The `FIR_CTL1.TC` bit configures the accelerator to use either unsigned integer or signed integer	
		0	Unsigned Integer
		1	Signed Integer
12 (R/W)	FXD	Fixed-Point Accelerator Select. The `FIR_CTL1.FXD` bit configures the accelerator to use either 32-bit IEEE floating-point or 32-bit fixed-point.	
		0	32-bit IEEE Floating-Point
		1	32-bit Fixed Point
11 (R/W)	CCINTR	Channel Complete Interrupt. The `FIR_CTL1.CCINTR` bit configures the accelerator to generate an interrupt when each or all channels are done. This bit is only valid in Legacy Mode. In Auto Configuration Mode (ACM), interrupt generation for each channel is controlled using the FIR_CTL2.IMASK bit.	
		0	Interrupt is Generated Only When All Channels are Done
		1	Interrupt is Generated After Each Channel is Done
9 (R/W)	CAI	Channel Auto Iterate. The `FIR_CTL1.CAI` bit, if cleared, causes the TDM processing to stop (idle) once all channels are done. If set, processing moves to the first channel and continues TDM processing in a loop when all channels are done. Channel Auto Iterate is not available in Auto Configuration Mode (ACM).	
		0	TDM Processing Stops (IDLE) Once All Channels are Over
		1	Moves to First Channel and Continues TDM Processing in a Loop When All Channels are Over
8 (R/W)	DMAEN	DMA Enable. The `FIR_CTL1.DMAEN` bit enables and disables DMA on the FIR accelerator.	
		0	DMA Disabled
		1	DMA Enabled
6 (R/W)	BURSTEN	Burst Mode Select. When the `FIR_CTL1.BURSTEN` bit is set, burst mode is enabled for coefficient loads and data loads. When cleared, burst mode is disabled. By default, burst mode is disabled.	

Table 45-14: FIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:1 (R/W)	CH	Number of Channels. The `FIR_CTL1.CH` bit field configures the number of channels and is programmable from 0 to 31. This bit field is only valid in Legacy Mode. There is no channel number limitation in Auto Configuration Mode (ACM) as the accelerator keeps processing the TCBs until the chain pointer becomes null.	
		0	Channel 1
		1-30	Channel 2-31
		31	Channel 32
0 (R/W)	EN	FIR Enable. The `FIR_CTL1.EN` bit enables and disables the FIR accelerator.	
		0	Disable FIR
		1	Enable FIR

FIR Channel Control Register

The FIR_CTL2 register is used to configure the channel specific parameters such as filter TAP length, window size, sample rate conversion, up/down sampling and ratio. In Auto Configuration Mode(ACM), this register is also used to configure additional channel specific parameters like interrupts and triggers.

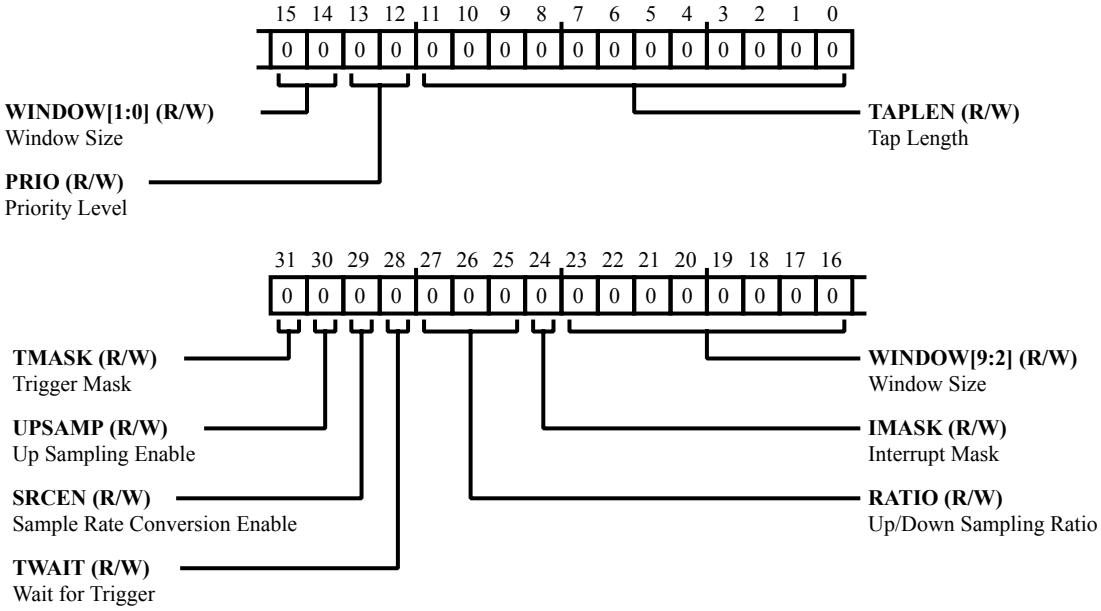


Figure 45-13: FIR_CTL2 Register Diagram

Table 45-15: FIR_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	TMASK	Trigger Mask. The FIR_CTL2.TMASK bit enables trigger generation for the channel. It is only valid in Auto Configuration Mode (ACM).	
		0	Enable
		1	Mask
30 (R/W)	UPSAMP	Up Sampling Enable. The FIR_CTL2.UPSAMP bit enables up sampling.	
		0	Down Sampling
		1	Up Sampling

Table 45-15: FIR_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	SRCEN	Sample Rate Conversion Enable. The `FIR_CTL2.SRCEN` bit field enables sample rate conversion.	
		0	Disabled
		1	Enabled
28 (R/W)	TWAIT	Wait for Trigger. The `FIR_CTL2.TWAIT` bit disables the wait for the trigger. It is only valid in Auto Configuration Mode (ACM).	
		0	Disable wait for trigger for the channel
		1	Enable wait for the external trigger assertion
27:25 (R/W)	RATIO	Up/Down Sampling Ratio. The `FIR_CTL2.RATIO` bit field sets the sampling ratio (`FIR_CTL2.RATIO` + 1).	
24 (R/W)	IMASK	Interrupt Mask. The `FIR_CTL2.IMASK` bit enables interrupt generation for the channel. This bit is only valid in Auto Configuration Mode (ACM).	
		0	Enable
		1	Mask
23:14 (R/W)	WINDOW	Window Size. The `FIR_CTL2.WINDOW` bit field sets the window size which specifies the number of sample/block to process (sample based processing = window size of 0).	
13:12 (R/W)	PRIO	Priority Level. The `FIR_CTL2.PRIO` bit field indicates the priority.	
		0	Level 0 (lowest)
		1	Level 1
		2	Level 2
		3	Level 3 (highest)
11:0 (R/W)	TAPLEN	Tap Length. The `FIR_CTL2.TAPLEN` bit field sets the tap length which is programmable between 0-4095 (Tap Length = `FIR_CTL2.TAPLEN` + 1).	

Debug Address Register

The FIR_DBG_ADDR register holds the debug address.

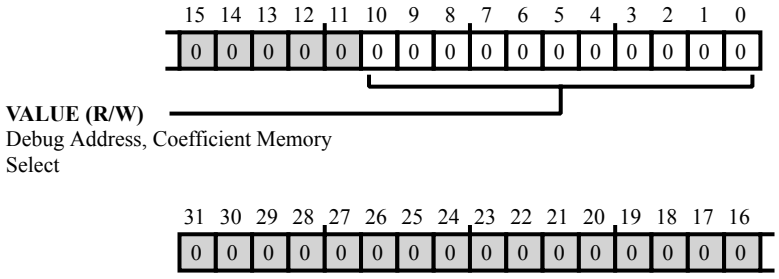


Figure 45-14: FIR_DBG_ADDR Register Diagram

Table 45-16: FIR_DBG_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Debug Address, Coefficient Memory Select. The FIR_DBG_ADDR.VALUE bit field holds the debug address (bits 0-10). Bit 11 configures whether the memory access is to coefficient memory (=0) or to delay line memory (=1).

FIR Debug Control Register

The FIR_DBG_CTL register controls debugging operations such as enabling debug mode running, hold or single stepping.

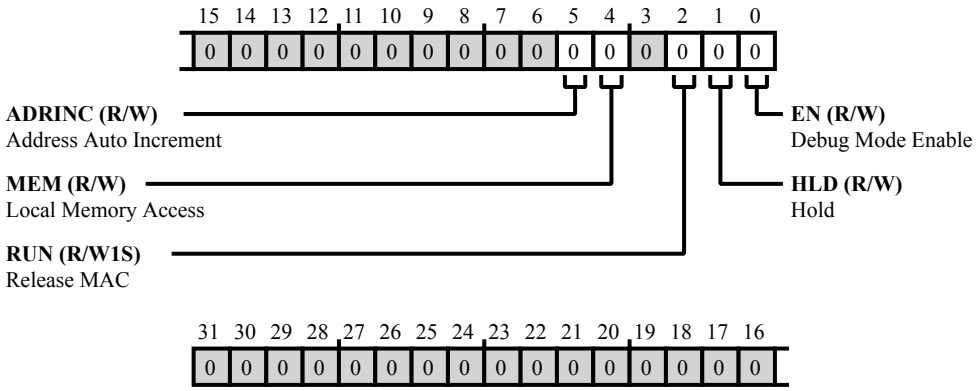


Figure 45-15: FIR_DBG_CTL Register Diagram

Table 45-17: FIR_DBG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W)	ADRINC	Address Auto Increment. The FIR_DBG_CTL.ADRINC bit bit allows the address register to auto-increment on FIR_DBG_WRDAT writes and FIR_DBG_RDDAT reads.	
4 (R/W)	MEM	Local Memory Access. When the FIR_DBG_CTL.MEM bit is set, the data and coefficients memory can be indirectly accessed.	
2 (R/W1S)	RUN	Release MAC. The FIR_DBG_CTL.RUN bit releases the MAC. This bit is self-clearing after one FIR clock cycle.	
1 (R/W)	HLD	Hold. The FIR_DBG_CTL.HLD bit holds or single-steps through the FIR.	
		0	Hold
		1	Single-step
0 (R/W)	EN	Debug Mode Enable. The FIR_DBG_CTL.EN bit enables debug mode. For local memory access, the FIR_CTL1 register can be cleared.	
		0	Disable Debug Mode
		1	Enable Debug Mode

FIR Debug Data Read Register

The FIR_DBG_RDDAT register hold the debug read data.

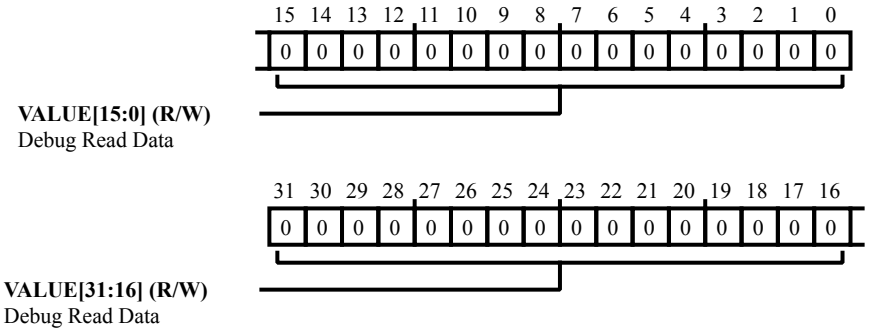


Figure 45-16: FIR_DBG_RDDAT Register Diagram

Table 45-18: FIR_DBG_RDDAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Debug Read Data. The FIR_DBG_RDDAT.VALUE bit field holds the debug read data.

FIR Debug Data Write Register

The FIR_DBG_WRDAT register holds the debug write data.

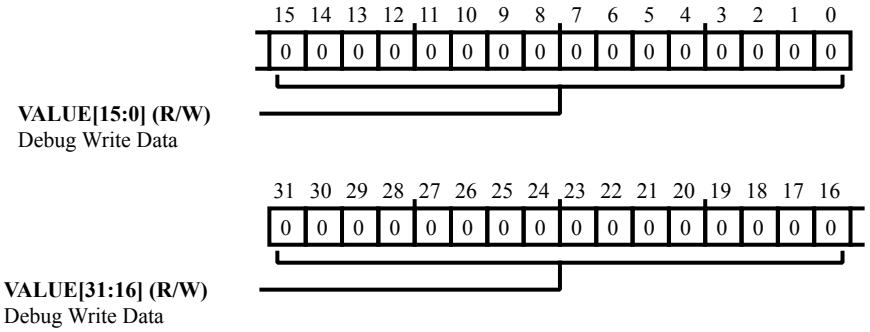


Figure 45-17: FIR_DBG_WRDAT Register Diagram

Table 45-19: FIR_DBG_WRDAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Debug Write Data. The FIR_DBG_WRDAT.VALUE bit field holds the debug write data.

FIR DMA Status Register

The FIR_DMASTAT register provides information about chain pointer loading, coefficient DMA, data preload DMA, processing in progress, window complete, all channels complete.

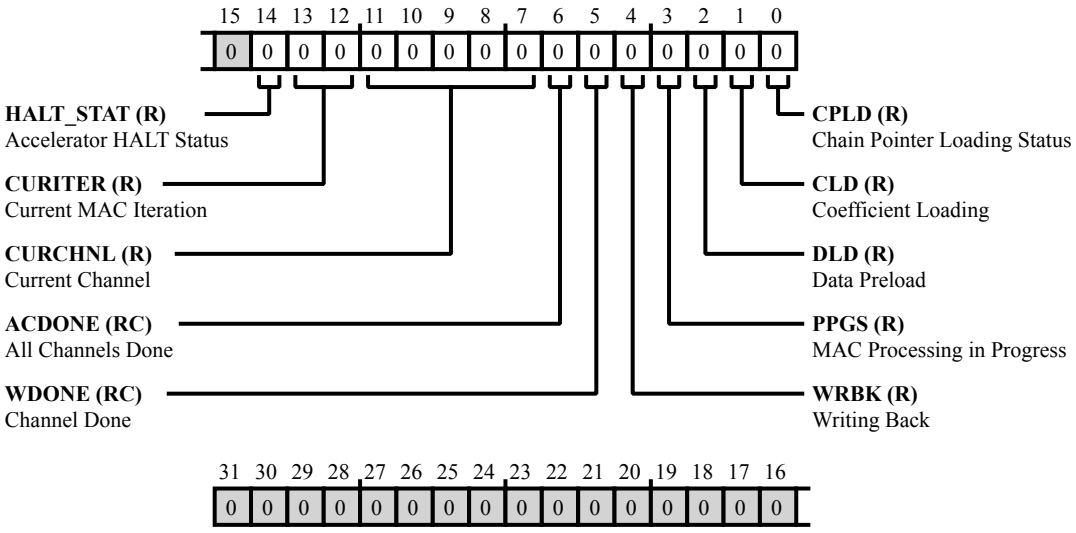


Figure 45-18: FIR_DMASTAT Register Diagram

Table 45-20: FIR_DMASTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/NW)	HALT_STAT	Accelerator HALT Status. Accelerator HALT status. 1 - Acknowledge to core that acceleration is in halt state. 0 - Not in halt state
13:12 (R/NW)	CURITER	Current MAC Iteration. The FIR_DMASTAT.CURITER bit indicates the current MAC iteration in multi-iteration mode. Zero indicates the final iteration.
11:7 (R/NW)	CURCHNL	Current Channel. The FIR_DMASTAT.CURCHNL bit indicates the channel that is being processed in the TDM slot. Zero indicates the last slot.
6 (RC/NW)	ACDONE	All Channels Done. The FIR_DMASTAT.ACDONE bit indicates the accelerator that processing all channels is complete. This is a sticky bit and is cleared on a register read. The FIR_CTL1.CCINTR bit does not affect the FIR_DMASTAT.ACDONE bit.
5 (RC/NW)	WDONE	Channel Done. The FIR_DMASTAT.WDONE bit indicates the accelerator that processing the current channel is complete. This is a sticky bit and is cleared on a register read. The FIR_CTL1.CCINTR bit does not affect the FIR_DMASTAT.WDONE bit.

Table 45-20: FIR_DMASTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
4 (R/NW)	WRBK	Writing Back. The `FIR_DMASTAT.WRBK` bit indicates the accelerator is writing back the updated index registers.
3 (R/NW)	PPGS	MAC Processing in Progress. The `FIR_DMASTAT.PPGS` bit indicates MAC processing in progress.
2 (R/NW)	DLD	Data Preload. The `FIR_DMASTAT.DLD` bit indicates data preloading.
1 (R/NW)	CLD	Coefficient Loading. The `FIR_DMASTAT.CLD` bit indicates coefficient loading.
0 (R/NW)	CPLD	Chain Pointer Loading Status. The `FIR_DMASTAT.CPLD` bit indicates the state machine is in chain pointer load state.

FIR Input Data Base Register

The FIR_INBASE register contains the input word base address with the lower two bits removed.

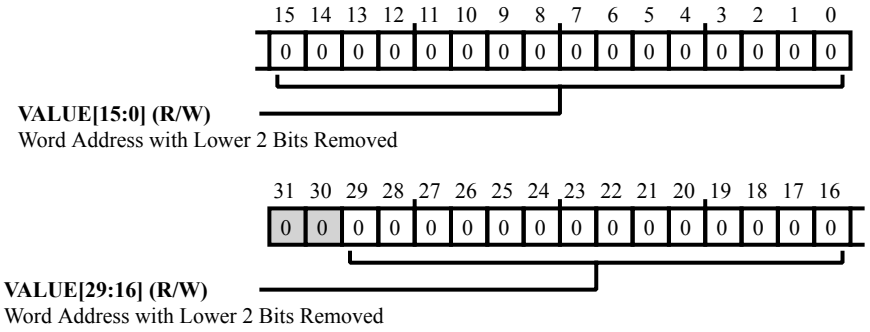


Figure 45-19: FIR_INBASE Register Diagram

Table 45-21: FIR_INBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 Bits Removed. The FIR_INBASE.VALUE bit field contains the word address with the lower 2 bits removed.

FIR Input Data Count Register

The FIR_INCNT register contains the 16-bit input data count.

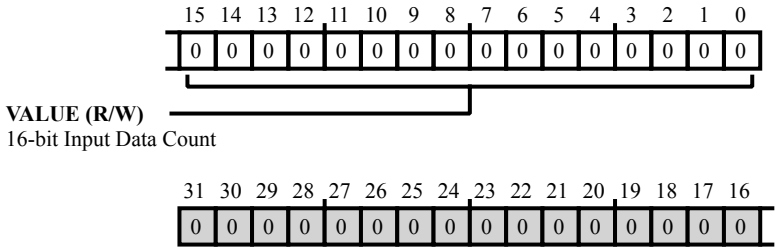


Figure 45-20: FIR_INCNT Register Diagram

Table 45-22: FIR_INCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Count. The FIR_INCNT.VALUE bit field contains the 16-bit input data count.

FIR Input Data Index Register

The FIR_INIDX register contains the input word address with the lower two bits removed.

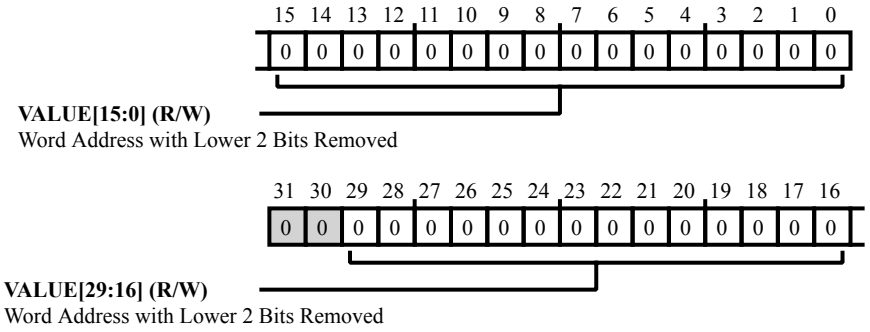


Figure 45-21: FIR_INIDX Register Diagram

Table 45-23: FIR_INIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 Bits Removed. The FIR_INIDX.VALUE bit field contains the input word address with the lower two bits removed.

FIR Input Data Modifier Register

The FIR_INMOD register contains the 16-bit input data buffer modifier.

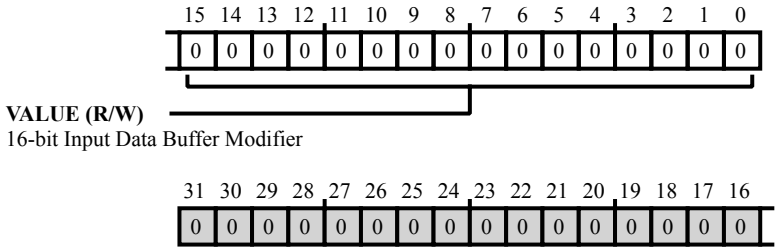


Figure 45-22: FIR_INMOD Register Diagram

Table 45-24: FIR_INMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Buffer Modifier. The FIR_INMOD.VALUE bit field contains the 16-bit input data buffer modifier.

FIR MAC Status Register

The FIR_MACSTAT register provides the status of MAC operations. The status of all four multipliers/adders are available separately for programs to poll. In fixed-point mode only, the ARIx bits are used (all other bits are reserved).

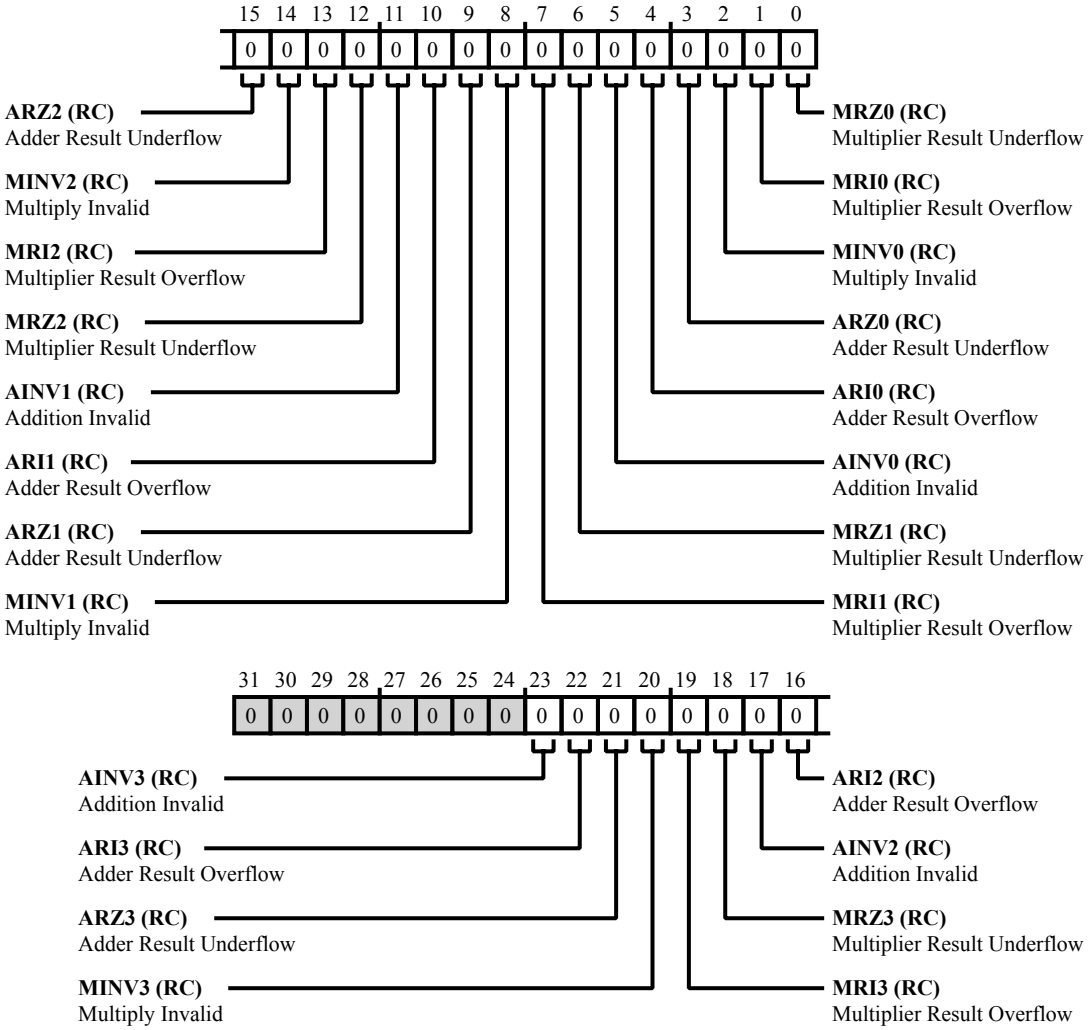


Figure 45-23: FIR_MACSTAT Register Diagram

Table 45-25: FIR_MACSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23 (RC/NW)	AINV3	Addition Invalid. The FIR_MACSTAT.AINV3 bit is set if the adder 3 addition is invalid.

Table 45-25: FIR_MACSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
22 (RC/NW)	ARI3	Adder Result Overflow. The `FIR_MACSTAT.ARI3` bit is set if the adder 3 result is infinity. Indicates overflow in fixed-point mode.
21 (RC/NW)	ARZ3	Adder Result Underflow. The `FIR_MACSTAT.ARZ3` bit is set if the adder 3 result is zero.
20 (RC/NW)	MINV3	Multiply Invalid. The `FIR_MACSTAT.MINV3` bit is set if the multiplier 3 multiply operation is invalid.
19 (RC/NW)	MRI3	Multiplier Result Overflow. The `FIR_MACSTAT.MRI3` bit is set if the multiplier 3 result is infinity.
18 (RC/NW)	MRZ3	Multiplier Result Underflow. The `FIR_MACSTAT.MRZ3` bit is set if the multiplier 3 result is zero.
17 (RC/NW)	AINV2	Addition Invalid. The `FIR_MACSTAT.AINV2` bit is set if the adder 2 addition is invalid.
16 (RC/NW)	ARI2	Adder Result Overflow. The `FIR_MACSTAT.ARI2` bit is set if the adder 2 result is infinity. Indicates overflow in fixed-point mode.
15 (RC/NW)	ARZ2	Adder Result Underflow. The `FIR_MACSTAT.ARZ2` bit is set if the adder 2 result is zero.
14 (RC/NW)	MINV2	Multiply Invalid. The `FIR_MACSTAT.MINV2` bit is set if the multiplier 2 multiply operation is invalid.
13 (RC/NW)	MRI2	Multiplier Result Overflow. The `FIR_MACSTAT.MRI2` bit is set if the multiplier 2 result is infinity.
12 (RC/NW)	MRZ2	Multiplier Result Underflow. The `FIR_MACSTAT.MRZ2` bit is set if the multiplier 2 result is zero.
11 (RC/NW)	AINV1	Addition Invalid. The `FIR_MACSTAT.AINV1` bit is set if the adder 1 addition is invalid.
10 (RC/NW)	ARI1	Adder Result Overflow. The `FIR_MACSTAT.ARI1` bit is set if the adder 1 result is infinity. Indicates overflow in fixed-point mode.
9 (RC/NW)	ARZ1	Adder Result Underflow. The `FIR_MACSTAT.ARZ1` bit is set if the adder 1 result is zero.

Table 45-25: FIR_MACSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
8 (RC/NW)	MINV1	Multiply Invalid. The `FIR_MACSTAT.MINV1` bit is set if the multiplier 1 multiply operation is invalid.
7 (RC/NW)	MRI1	Multiplier Result Overflow. The `FIR_MACSTAT.MRI1` bit is set if the multiplier 1 result is infinity.
6 (RC/NW)	MRZ1	Multiplier Result Underflow. The `FIR_MACSTAT.MRZ1` bit is set if the multiplier 1 result is zero.
5 (RC/NW)	AINV0	Addition Invalid. The `FIR_MACSTAT.AINV0` bit is set if the adder 0 addition is invalid.
4 (RC/NW)	ARI0	Adder Result Overflow. The `FIR_MACSTAT.ARI0` bit is set if the adder 0 result is infinity. Indicates overflow in fixed-point mode.
3 (RC/NW)	ARZ0	Adder Result Underflow. The `FIR_MACSTAT.ARZ0` bit is set if the adder 0 result is zero.
2 (RC/NW)	MINV0	Multiply Invalid. The `FIR_MACSTAT.MINV0` bit is set if the multiplier 0 multiply operation is invalid.
1 (RC/NW)	MRI0	Multiplier Result Overflow. The `FIR_MACSTAT.MRI0` bit is set if the multiplier 0 result is infinity.
0 (RC/NW)	MRZ0	Multiplier Result Underflow. The `FIR_MACSTAT.MRZ0` bit is set if multiplier 0 result is zero.

FIR Output Data Base Register

The FIR_OUTBASE register contains the output word base address with the lower two bits removed

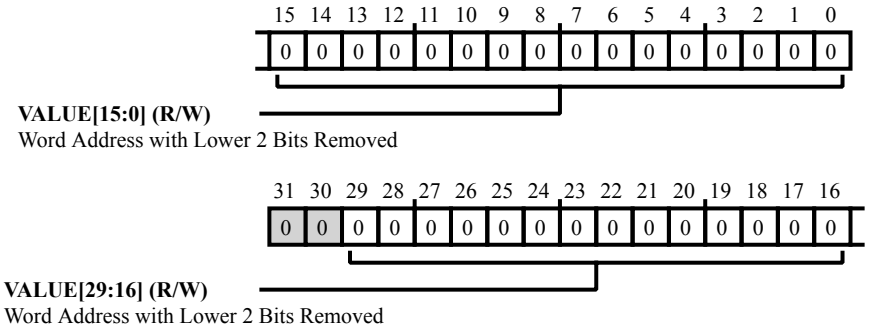


Figure 45-24: FIR_OUTBASE Register Diagram

Table 45-26: FIR_OUTBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 Bits Removed. The FIR_OUTBASE.VALUE bit field contains the word address with the lower 2 bits removed.

FIR Output Data Count Register

The FIR_OUTCNT register contains the 16-bit output buffer count.

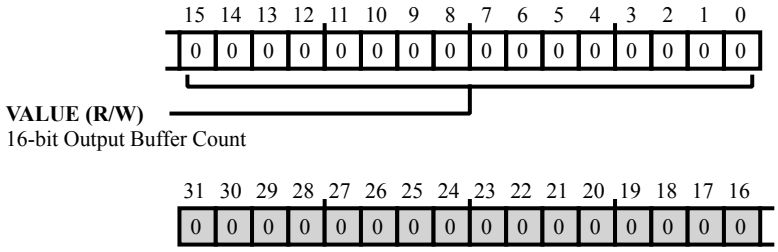


Figure 45-25: FIR_OUTCNT Register Diagram

Table 45-27: FIR_OUTCNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Output Buffer Count. The FIR_OUTCNT.VALUE bit field contains the 16-bit output buffer count.

FIR Output Data Index Register

The FIR_OUTIDX register contains the output word address with the lower two bits removed.

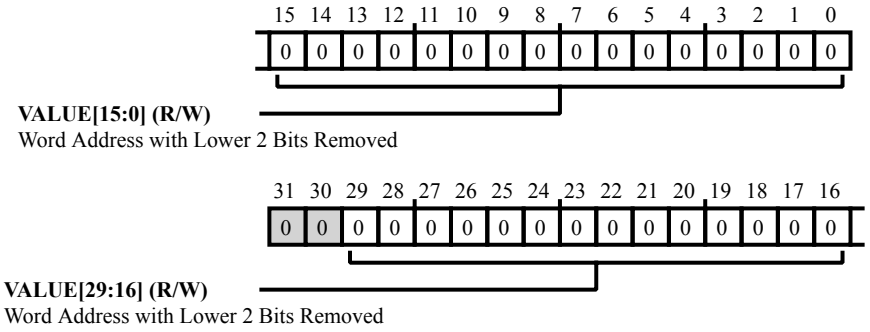


Figure 45-26: FIR_OUTIDX Register Diagram

Table 45-28: FIR_OUTIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Word Address with Lower 2 Bits Removed. The FIR_OUTIDX.VALUE bit field contains the word address with the lower 2 bits removed.

FIR Output Data Modifier Register

The FIR_OUTMOD register contains the 16-bit output data buffer modifier.

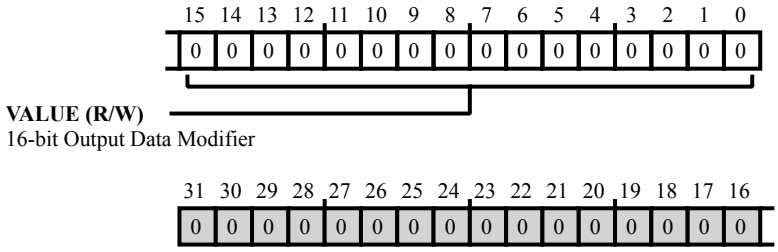


Figure 45-27: FIR_OUTMOD Register Diagram

Table 45-29: FIR_OUTMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Output Data Modifier. The FIR_OUTMOD.VALUE bit field contains the 16-bit output data modifier.

Software Control Register 1

The FIR_SCTL1 register is used to control software.

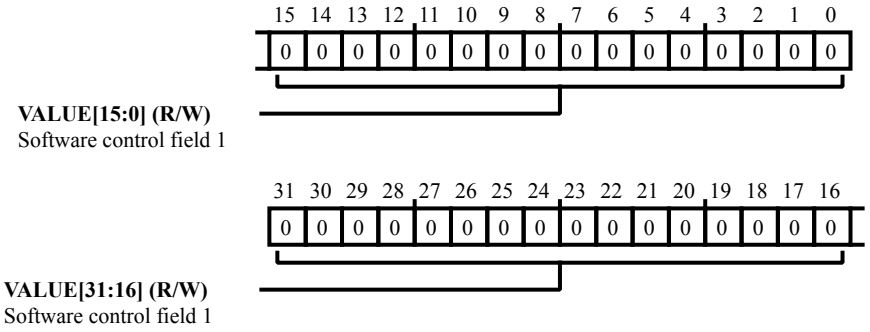


Figure 45-28: FIR_SCTL1 Register Diagram

Table 45-30: FIR_SCTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Software control field 1.

Software Control Register 2

The FIR_SCTL2 register is used to control software.

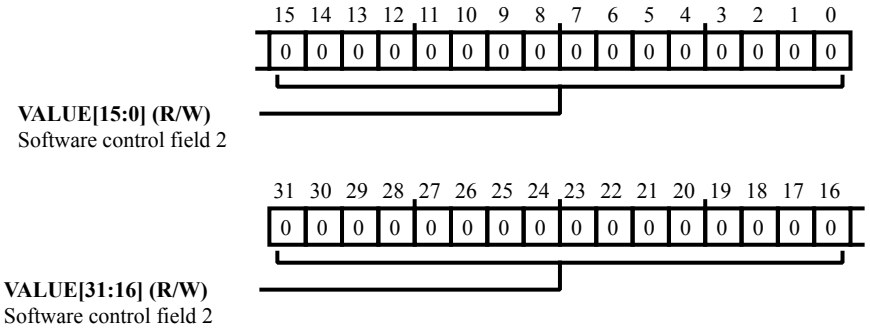


Figure 45-29: FIR_SCTL2 Register Diagram

Table 45-31: FIR_SCTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Software control field 2.

Secondary Global Control Register

The FIRSGCTL register is used to configure the global parameters for the accelerator in ACM mode for loading CTL1 register bits as part of TCB.

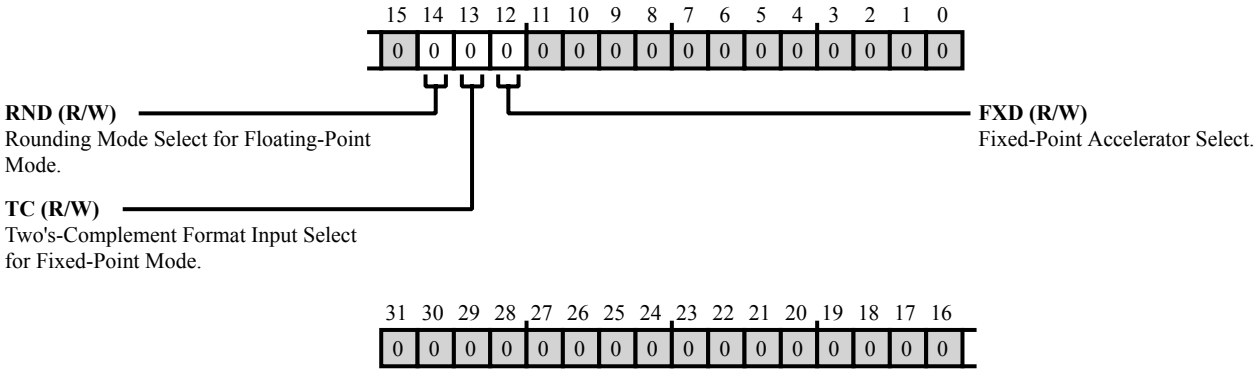


Figure 45-30: FIR_SGCTL Register Diagram

Table 45-32: FIR_SGCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
14 (R/W)	RND	Rounding Mode Select for Floating-Point Mode.. 0- Round to nearest 1 -Truncate (Round towards zero)
13 (R/W)	TC	Two's-Complement Format Input Select for Fixed-Point Mode.. 0 - Unsigned integer 1 - Signed integer
12 (R/W)	FXD	Fixed-Point Accelerator Select.. 0 - 32-bit floating-point compatible to SHARCXI 1 - 32-bit fixed point

46 IIR Accelerator (IIR)

The processor includes an Infinite Impulse Response (IIR) filter accelerator implemented in hardware that reduces the processing load on the core, freeing it up for other tasks.

Features

The accelerator supports a maximum of 32 channels in legacy mode. In Auto Configuration Mode (ACM), there is no limit on number of channels. The accelerator locally stores all the biquad coefficients of 32 channels in legacy mode. Window size can be configured from 1 (sample-based) to 1024.

The IIR has the following features:

- Supports IEEE floating point format 32/40-bit
- Rounding modes compatible with SHARC+ core MACs
- Sample-based or window-based processing
- Up to 64 cascaded biquads per channel
- Up to 32 filter channels available in TDM
- Allows biquad save state storage

NOTE: The IIR accelerator module has local memory that is not accessible by the core during regular operation mode. Unlike in the previous SHARC processors, the IIR accelerator modules each have access to the system memory (on-chip/off-chip).

Unlike in the previous SHARC processors, where only one of the IIR or FIR accelerators can be enabled at a time, the processor can simultaneously use both the IIR and the FIR accelerators.

IIR Definitions

The following definitions are helpful when using the IIR filter.

Infinite Impulse Response (IIR) filter

A recursive filter calculates its output by using current and previous inputs, along with previous outputs. IIR filters are digital filters that have an infinite impulse response.

Automatic Configuration Mode (ACM)

The IIR accelerator loads biquad coefficients for only the current channel before processing the current channel.

Legacy Mode

The IIR accelerator loads biquad coefficients for all channels before processing the first channel.

Task Control Block (TCB)

A method to store task-specific parameters and calculation states in a user-initialized memory buffer and permitting user tasks to directly access the values. The reduction of the amount of time required for context switching can lessen the execution time for user tasks.

Biquad (biquadratic)

A second-order recursive digital linear filter, containing two poles and two zeros. In the Z domain, its transfer function consists of the ratio of two quadratic functions. The biquad form II implements the equation in two parts with difference equations and two less storage delay components.

Multiply and Accumulate (MAC) Unit

The IIR subblock with a pipelined multiplier and accumulator that performs floating-point calculations on data and fetches coefficients. Can perform 32-bit or 40-bit operations.

Window-based Processing Mode

Includes using a window size of one or greater (maximum of 1024) for a channel.

Save Biquad State Mode

The mode where the IIR filter stores the current biquad states in local memory when it must switch to another high-priority accelerator task. After resuming, these states can be reloaded, and IIR processing continued.

Clocking

The IIR accelerator runs at the speed of core clock.

Functional Description

The ***IIR Accelerator Block Diagram*** shows the blocks of the IIR hardware accelerator.

The accelerator has:

- a coefficient memory size of 1440 × 40 bits (288 biquads x 5 coefficients).
- a data memory size of 576 × 40 bits (288 biquads x 2 states).
- one MAC unit with an input data buffer to supply data to the MAC.

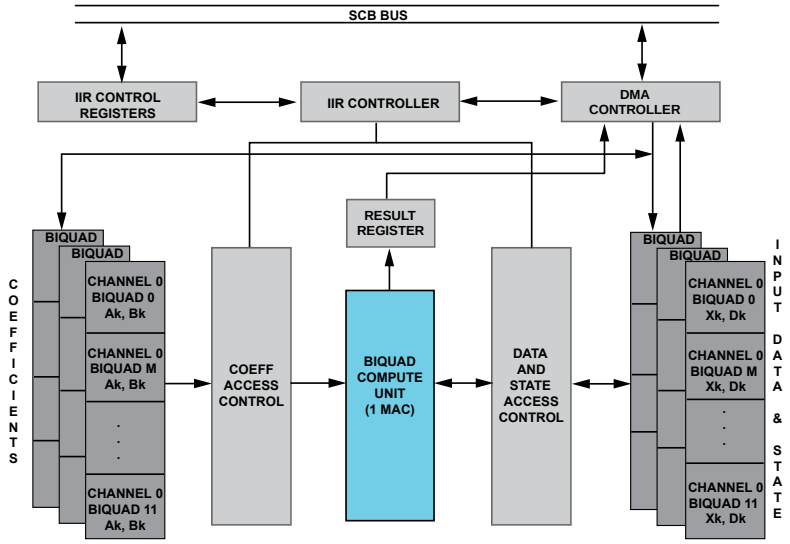


Figure 46-1: IIR Accelerator Block Diagram

A transposed direct form II biquadratic is used to implement the IIR accelerator, which reduces coefficient sensitivity. The ***Transposed Direct Form II Biquad*** figure shows the signal flow graph for the biquad structure.

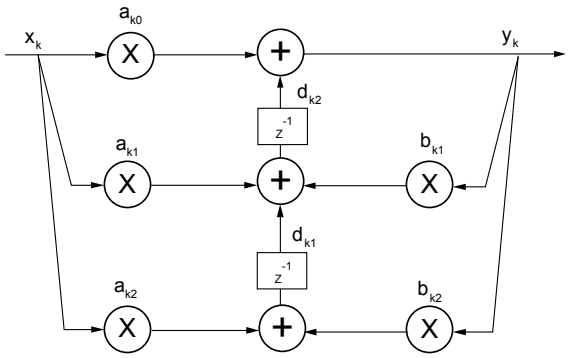


Figure 46-2: Transposed Direct Form II Biquad

The accelerator has the following logical subblocks.

- A data path unit with the following elements:
  - 32/40-bit coefficient memory (Ak, Bk) for storing biquad coefficients

  - 32/40-bit input data (Xk) and state (Dk)
  - One 40/32-bit floating-point multiplier and adder (MAC) unit
  - An input data buffer to efficiently supply data to MAC
  - One 40-bit result register to hold result of the biquad
- Configuration registers for controlling various parameters such as the number of biquads, the number of channels, interrupt control, and DMA control.
- A core access interface for writing the DMA/filter configuration registers and for reading the status registers.
- A DMA bus interface for transferring data to and from the accelerator. This interface also preloads the coefficients (Ak, Bk) and state (Dk) at startup.
- DMA configuration registers for the transfer of input data, output data, and coefficients.

The accelerator block is integrated with the SHARC core. For more information, see Figure 48-5 SHARC Fabric Connectivity.

ADSP-SC59x IIR Register List

The IIR module reduces the processing load on the core. For more information on IIR functionality, see the IIR register descriptions.

Table 46-1: ADSP-SC59x IIR Register List

Name	Description
IIR_CHNPTR	Chain Pointer Register
IIR_COEFIDX	Coefficient Buffer Index Register
IIR_COEFLEN	Coefficient Buffer Length Register
IIR_COEFMOD	Coefficient Index Modifier Register
IIR_CTL1	Global Control Register
IIR_CTL2	Channel Control Register
IIR_DBG_ADDR	IIR Debug Address Register
IIR_DBG_CTL	IIR Debug Control Register
IIR_DBG_RDDAT_HI	IIR Debug Read Data High Register
IIR_DBG_RDDAT_LO	IIR Debug Read Data Low Register
IIR_DBG_WRDAT_HI	IIR Debug Write Data High Register
IIR_DBG_WRDAT_LO	IIR Debug Write Data Low Register
IIR_DMASTAT	DMA Status Register
IIR_INBASE	Input Buffer Base Register

Table 46-1: ADSP-SC59x IIR Register List (Continued)

Name	Description
IIR_INIDX	Input Data Index Register
IIR_INLEN	Input Data Buffer Length Register
IIR_INMOD	Input Data Index Modifier Register
IIR_MACSTAT	MAC Status Register
IIR_OUTBASE	Output Buffer Base Register
IIR_OUTIDX	Output Data Buffer Index Register
IIR_OUTLEN	IIR Output Data Buffer Length Register
IIR_OUTMOD	IIR Output Data Index Modifier Register
IIR_SCTL1	Software Control Register1
IIR_SCTL2	Software Control Register2
IIR_SGCTL	Secondary Global Control Register

ADSP-SC59x IIR Interrupt List

Table 46-2: ADSP-SC59x IIR Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
168	IIR0_DMA	IIR0 Core 1 DMA	Edge	
169	IIR0_STAT	IIR0 Core 1 Status	Edge	
170	IIR1_DMA	IIR1 Core 1 DMA	Edge	
171	IIR1_STAT	IIR1 Core 1 Status	Edge	
172	IIR2_DMA	IIR2 Core 1 DMA	Edge	
173	IIR2_STAT	IIR2 Core 1 Status	Edge	
174	IIR3_DMA	IIR3 Core 1 DMA	Edge	
175	IIR3_STAT	IIR3 Core 1 Status	Edge	
178	IIR4_DMA	IIR4 Core 2 DMA	Edge	
179	IIR4_STAT	IIR4 Core 2 Status	Edge	
180	IIR5_DMA	IIR5 Core 2 DMA	Edge	
181	IIR5_STAT	IIR5 Core 2 Status	Edge	
182	IIR6_DMA	IIR6 Core 2 DMA	Edge	
183	IIR6_STAT	IIR6 Core 2 Status	Edge	
184	IIR7_DMA	IIR7 Core 2 DMA	Edge	

Table 46-2: ADSP-SC59x IIR Interrupt List (Continued)

Interrupt ID	Name	Description	Sensitivity	DMA Channel
185	IIR7_STAT	IIR7 Core 2 Status	Edge	
252	IIR0_BUS_ERR	IIR0 Core 1 IIR Bus Error	Edge	
253	IIR1_BUS_ERR	IIR1 Core 1 IIR Bus Error	Edge	
254	IIR2_BUS_ERR	IIR2 Core 1 IIR Bus Error	Edge	
255	IIR3_BUS_ERR	IIR3 Core 1 IIR Bus Error	Edge	
257	IIR4_BUS_ERR	IIR4 Core 2 IIR Bus Error	Edge	
258	IIR5_BUS_ERR	IIR5 Core 2 IIR Bus Error	Edge	
259	IIR6_BUS_ERR	IIR6 Core 2 IIR Bus Error	Edge	
260	IIR7_BUS_ERR	IIR7 Core 2 IIR Bus Error	Edge	

ADSP-SC59x IIR Trigger List

Table 46-3: ADSP-SC59x IIR Trigger List Generators

Trigger ID	Name	Description	Sensitivity
34	C1_IIR0_DMA	IIR0 Core 1 DMA	Edge
35	C1_IIR1_DMA	IIR1 Core 1 DMA	Edge
36	C1_IIR2_DMA	IIR2 Core 1 DMA	Edge
37	C1_IIR3_DMA	IIR3 Core 1 DMA	Edge
38	C2_IIR0_DMA	IIR4 Core 2 DMA	Edge
39	C2_IIR1_DMA	IIR5 Core 2 DMA	Edge
40	C2_IIR2_DMA	IIR6 Core 2 DMA	Edge
41	C2_IIR3_DMA	IIR7 Core 2 DMA	Edge

Table 46-4: ADSP-SC59x IIR Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
23	C1_IIR0_TRGI	IIR0 Core 1 IIR Wait on Trigger Input	Pulse
24	C1_IIR1_TRGI	IIR1 Core 1 IIR Wait on Trigger Input	Pulse
25	C1_IIR2_TRGI	IIR2 Core 1 IIR Wait on Trigger Input	Pulse
26	C1_IIR3_TRGI	IIR3 Core 1 IIR Wait on Trigger Input	Pulse
27	C2_IIR0_TRGI	IIR4 Core 2 IIR Wait on Trigger Input	Pulse
28	C2_IIR1_TRGI	IIR5 Core 2 IIR Wait on Trigger Input	Pulse

Table 46-4: ADSP-SC59x IIR Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
29	C2_IIR2_TRGI	IIR6 Core 2 IIR Wait on Trigger Input	Pulse
30	C2_IIR3_TRGI	IIR7 Core 2 IIR Wait on Trigger Input	Pulse

Multiply and Accumulate (MAC) Unit

The *IIR MAC Unit* figure shows a pipelined multiplier and accumulator unit that operates on the data and coefficients fetched from the data and coefficient memory. The MAC can perform 32-bit floating-point or 40-bit floating-point MAC operations. The 32-bit floating-point operations generate 32-bit results and the 40-bit floating-point operations generate 40-bit results.

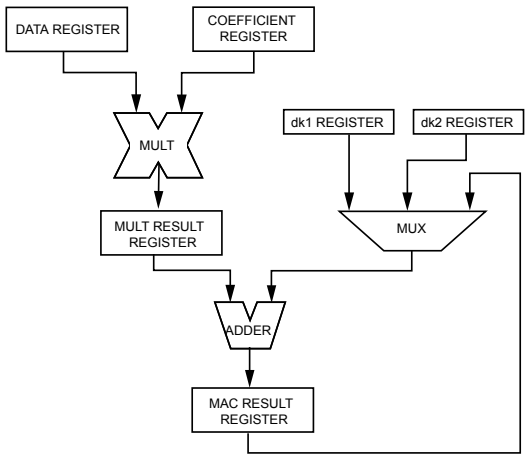


Figure 46-3: IIR MAC Unit

Input Data and Biquad State

The size of the data memory is 576 × 40 bits and locally holds the dk1 and dk2 state of all biquads. The DMA controller fetches the sample data from internal memory and calculates the output plus the dk1 and dk2 values for each biquad and stores them in local data memory.

Coefficient Memory

The size of coefficient memory is 1440 × 40 bits and stores all the coefficients for all biquads. At startup, the DMA loads the coefficients from system memory into local coefficient memory.

Internal Memory Storage

This section describes the required storage model for the IIR accelerator.

Coefficient Memory Storage

Store the coefficients and Dk values for a specific biquad BQD[k] in internal memory in the following order: Ak0, Ak1, Bk1, Ak2, Bk2, Dk2, and Dk1.

NOTE: The naming convention for the filter coefficients used here is different from the one used in MATLAB. The following conversion should be used when using MATLAB generated coefficients:

(Akx = bx and Bkx = -ax).

Store the coefficients for each biquad in the order:

```
b0, b1, -a1, b2, -a2, dk2, dk1
```

For N biquad stages, store the coefficients in the order:

```
b01, b11, -a11, b21, -a21, dk21, dk11,
b02, b12, -a12, b22, -a22, dk22, dk12,
,......
b0N, b1N, -a1N, b2N, -a2N, dk2N, dk1N.
```

where bxN and axN are the coefficients ([b, a]) for the Nth biquad stage.

Operating Modes

The accelerator operates:

- In Window Processing Mode, that also includes sample-based processing mode
- In 40-Bit Floating-Point Mode
- In Save Biquad State Mode
- In Auto Configuration Mode (ACM)

Window Processing Mode

Select the sample-based processing mode by configuring window size to one. All biquads for a particular channel process one sample, and then calculate a final output sample in this mode.

In window-based mode, multiple output samples (maximum of 1024, equal to the window size) of that channel are calculated. After these calculations are complete, the accelerator begins processing the next channel. A register field provides a window size parameter to specify the length of the window.

40-Bit Floating-Point Mode

In 40-bit floating-point mode, the accelerator treats input data and coefficients as a 40-bit floating-point numbers. The 40-bit floating-point MAC operations generate 40-bit results. This mode can be selected by setting the `IIR_CTL1.FORTYBIT` bit.

In ACM mode, 40-bit floating-point mode can be selected by setting the `IIR_SGCTL.FORTYBIT` bit field for each channel.

As the DMA bus width is 32 bits, in 40-bit mode the IIR accelerator performs two packed 32-bit accesses to the memory to:

- fetch one 40-bit input or coefficient data

- store one 40-bit output word

The first 32-bit word provides the lower 32 bits and the 8 LSBs of the second 32-bit word provides rest of the upper 8 bits of the complete 40-bit word. The ***32-Bit to 40-Bit Packing*** figure shows the 32– to 40–bit packing used by the accelerator.

NOTE: Overhead time may be required to pack the input 40-bit data into the format acceptable by the IIR accelerator and for unpacking the output of accelerator to a format acceptable to the rest of the application.

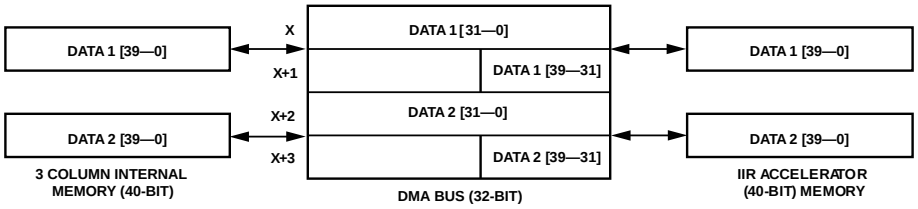


Figure 46-4: 32-Bit to 40-Bit Packing

Save Biquad State Mode

The `IIR_CTL1.SS` bit stores the current biquad states in local memory (writes all the Dk1 and Dk2 states back into the system memory states). This is useful for applications that require fast switching to another high-priority accelerator task—a required IIR to FIR processing transition for example. After resuming, these states can be reloaded and IIR processing can be continued. DMA status is automatically stored after each iteration.

NOTE: In legacy mode, the accelerator loads the biquad coefficients for all the channels before processing the first channel. After processing the last channel, when the `IIR_CTL1.SS` bit is set, the accelerator stores the biquad states of all the channels in local memory (writes all the Dk1 and Dk2 states back into the system memory states). In ACM mode, the accelerator loads the biquad coefficients for only the current channel before starting to process the current channel. When the `IIR_SGCTL.SS` bit is set, the accelerator stores the biquad states of the current channel in local memory before starting calculations for the next channel.

NOTE: During the save state operation, it is not permitted to have write access to any of the IIR accelerator registers that are loaded by chaining. Attempted writes to these registers can result in the blocking of IOP core reads until the save state operation completes.

Auto Configuration Mode (ACM)

The accelerator can be operated in legacy mode or Auto Configuration Mode (ACM). ACM is configured by setting the `IIR_CTL1.ACM` bit. The default functional mode is legacy mode. The accelerator must be disabled before changing the accelerator mode.

The ACM provides the following features:

- Halt and Queuing

  The core can pause processing the current Transfer Control Block (TCB) chain by setting the `IIR_CTL1.HALT` bit. The accelerator acknowledges the core by setting the `IIR_DMASTAT.HALT_STAT`

bit. The core can then submit or insert new TCBs. After the core acts, the accelerator resumes processing by clearing the IIR_CTL1.HALT bit. Before halting the accelerator, when the initial TCB chain is processed, the accelerator goes to the idle state. To do this, disable the accelerator and then enable it by toggling the IIR_CTL1.EN. Clear the IIR_CTL1.HALT bit to resume processing.

- No Channel Number Limitation

  Unlike in legacy mode, there is no fixed channel number limitation, and the accelerator ignores the value programmed in the IIR_CTL1.CH field. The application can queue an unlimited number of channels and TCBs dynamically, plus the accelerator keeps processing the TCBs until the chain pointer is null.

- Selective Interrupt

  The core can enable and mask interrupt generation for each channel by setting the IIR_CTL2.IMASK bit. When the bit is cleared, an interrupt generates after the channel completes.

- Selective Controller/Target Trigger

  The core can enable/mask trigger generation by the accelerator after the end of processing of each channel using the IIR_CTL2.TMASK bit. The accelerator can also wait for a trigger after loading the TCB and coefficients and before processing a channel for which the IIR_CTL2.TWAIT bit is set.

  This feature can also be enabled in legacy mode by setting the IIR_CTL1.L_TIMASK_EN and IIR_CTL1.L_TWAIT_EN bits. In legacy mode, when the IIR_CTL1.L_TIMASK_EN and IIR_CTL1.L_TWAIT_EN bits are set, then the IIR_CTL2.TMASK and IIR_CTL2.TWAIT bits can be used to enable/mask trigger generation and wait for a trigger for each channel respectively.

Additionally, use the IIR_CTL1 register to change parameters such as rounding mode, 40-bit mode, and save state for each channel. The IIR_SCTL1 and IIR_SCTL2 registers can be used as general-purpose registers.

Data Transfers

The IIR filter works exclusively through DMA.

IIR Accelerator TCB

The location of the DMA parameters for the next sequence comes from the chain pointer register. This register points to the next set of DMA parameters stored in the system memory of the processor known as the Task Control Block (TCB). In chained DMA operations, the processor automatically initializes and then begins another DMA transfer once the current DMA transfer completes. Each new set of parameters is stored in a user-initialized memory buffer or TCB for a chosen peripheral.

Chain Assignment

The structure of a TCB is conceptually the same as the structure of a linked list. Each TCB has several data values and a pointer to the next TCB. The chain pointer of a TCB can point to itself to re-run the same DMA continuously. The IIR accelerator reads each word of the TCB and loads it into the corresponding register. A TCB with a chain pointer register value of zero indicates the end of the chain (no further TCBs to load). The IIR accelerator supports

circular buffer chained DMA. The following tables show the required TCBs for chained DMA in Legacy Mode and ACM. TCB refers to the start address of the TCB array.

NOTE: The IIR accelerator DMA has two available TCB loading sequences:

- Loads five parameters for the coefficients (IIR_CTL2, IIR_COEFIDX, IIR_COEFMOD, IIR_COEFLEN, and IIR_CHNPTR).
- Loads 10 parameters for the data (IIR_CTL2, IIR_INIDX, IIR_INMOD, IIR_INLEN, IIR_INBASE, IIR_OUTIDX, IIR_OUTMOD, IIR_OUTLEN, IIR_OUTBASE, and IIR_CHNPTR).

Initialize IIR_CHNPTR to TCB+12 in legacy mode.

Table 46-5: IIR TCBs for Chained DMA in Legacy Mode

Address	Register
TCB	IIR_CHNPTR
TCB + 0x1	IIR_COEFLEN
TCB + 0x2	IIR_COEFMOD
TCB + 0x3	IIR_COEFIDX
TCB + 0x4	IIR_OUTBASE
TCB + 0x5	IIR_OUTLEN
TCB + 0x6	IIR_OUTMOD
TCB + 0x7	IIR_OUTIDX
TCB + 0x8	IIR_INBASE
TCB + 0x9	IIR_INLEN
TCB + 0xA	IIR_INMOD
TCB + 0xB	IIR_INIDX
TCB + 0xC	IIR_CTL2

Table 46-6: IIR TCBs for Chained DMA in ACM

Address	Register
TCB	IIR_CHNPTR
TCB + 0x1	IIR_SCTL1
TCB + 0x2	IIR_SCTL2
TCB + 0x3	IIR_SGCTL
TCB + 0x4	IIR_COEFLEN
TCB + 0x5	IIR_COEFMOD

Table 46-6: IIR TCBs for Chained DMA in ACM (Continued)

Address	Register
TCB + 0x6	IIR_COEFIDX
TCB + 0x7	IIR_OUTBASE
TCB + 0x8	IIR_OUTLEN
TCB + 0x9	IIR_OUTMOD
TCB + 0xA	IIR_OUTIDX
TCB + 0xB	IIR_INBASE
TCB + 0xC	IIR_INLEN
TCB + 0xD	IIR_INMOD
TCB + 0xE	IIR_INIDX
TCB + 0xF	IIR_CTL2

DMA Access

The IIR accelerator has two DMA channels (accelerator input and output) to connect to the system memory. The DMA controller fetches the data and coefficients from memory and stores the result.

Burst Mode Access

Burst mode enhances the throughput of the DMA channel and reduces the overall load on the system. When the IIR_CTL1.BURSTEN bit is set, it loads all the biquads for all the channels with burst enabled. The IIR module supports only INCR8 burst.

Chain Pointer DMA

The DMA controller supports a circular buffer chain pointer DMA. One transfer control block (TCB) must be configured for each channel. The TCB contains:

- A control register value to configure the filter parameters (such as number of biquads, window size) for each channel. For ACM, additional parameters such as an interrupt mask, trigger mask, and trigger wait exist.
- Software control register values in ACM for each channel.
- Secondary control register value in ACM to configure rounding mode, 40-bit mode, and save state for each channel.
- DMA parameter register values for the input data.
- DMA parameter register values for coefficient load.
- DMA parameter register values for output data.

In legacy mode, The chain pointer (IIR_CHNPTR) field of the last channel TCB should point to the first channel TCB. This configuration exists so that when the IIR accelerator is enabled, the module:

1. Loads the coefficients (Ak, Bk) and state variables (Dk) for all the channels into the local coefficient memory of the IIR.
2. Loops back to the first channel again to start fetching the input data for processing.

In ACM, the chain pointer field of the last channel TCB need not point to the first channel TCB.

When the accelerator is enabled, the module:

In ACM, the accelerator keeps processing the TCBs until the chain pointer becomes null.

The accelerator loads the TCB into its internal registers and uses these values to fetch coefficients, fetch data, and store results. After processing a window of data for any channel, the accelerator writes back the `IIR_INIDX` (input index register) and `IIR_OUTIDX` (output index register) values to the TCB in memory. Data processing can then begin from where it left off during the next time slot of that channel.

For 32-bit mode, the write-back values for the index registers are equal to `IIRII` + W and `IIROI` + W.

For 40-bit mode, the write-back values are: `IIR_INIDX` + 2 × W and `IIR_OUTIDX` + 2 × W.

Accelerator input and output channels connect to system memory.

NOTE: The `IIR_CTL2` register is part of the IIR TCB. This configuration allows software to program individual IIR channels having different control attributes.

In ACM, when the `IIR_CTL1.SMQ_LIUPS_EN` bit is set, the accelerator updates the `IIR_INIDX` and `IIR_OUTIDX` bit fields of the TCB in memory after processing a window of data and according to the circular buffer scheme. When the `IIR_CTL1.SMQ_LIUPS_EN` bit is cleared, the accelerator updates the `IIR_INIDX` and `IIR_OUTIDX` bit fields of the TCB in memory to 0x00000000 and 0xFFFFFFFF after processing a window of data. The `IIR_CTL1.SMQ_LIUPS_EN` bit is only valid in ACM.

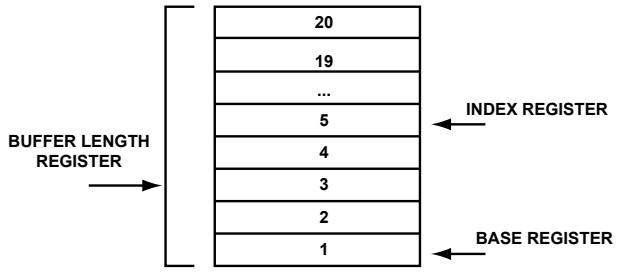


Figure 46-5: Circular Buffer Addressing

Interrupts

The ***IIR Interrupt Overview*** table provides the source of interrupt and service instructions for the IIR interrupts.

Table 46-7: IIR Interrupt Overview

Accelerator Mode	Default Programmable Interrupt	Sources	Masking	Service
Legacy Mode	IIR_DMA	All channels complete	N/A	ROC from IIR_DMASTAT + RTI instruction
		Window complete	IIR_CTL2.IMASK if IIR_CTL1.L_TIMASK_EN is set	
	IIR_STAT	MAC IEEE floating point exceptions	N/A	ROC from IIR_MACSTAT + RTI instruction
		MAC fixed point overflow		
Auto Configuration Mode	IIR_DMA	Window complete	IIR_CTL2.IMASK	ROC from IIR_DMASTAT + RTI instruction
	IIR_STAT	MAC IEEE floating point exceptions	N/A	ROC from IIR_MACSTAT + RTI instruction
		MAC fixed point overflow	N/A	

Sources

The IIR module drives two interrupt signals, IIR_DMA for the DMA status and IIR_STAT for the MAC status. The IIR module generates interrupts as described in the following sections.

Window Complete

This interrupt generates at the end of each channel when all the output samples are calculated for a window and the channel writes back updated index values.

In legacy mode, when the IIR_CTL1.CCINTR bit is set, an interrupt generates after completion of window processing of each channel. In ACM, the interrupt generation can be selectively masked using the IIR_CTL2.IMASK bit for each channel.

Interrupt masking for each channel can be enabled in legacy mode by setting the IIR_CTL1.L_TIMASK_EN bit. In legacy mode, when the IIR_CTL1.L_TIMASK_EN bit is set, interrupt generation for each channel is controlled using the IIR_CTL2.IMASK bit.

All Channels Complete

This interrupt generates when all the channels are complete or when one iteration of time slots completes. The interrupt follows the access completion rule, where the interrupt generates when all data are written back to system memory.

The All Channels Complete interrupt source only applies in legacy mode. When the IIR_CTL1.CCINTR and IIR_CTL1.L_TIMASK_EN bits are not set in Legacy Mode, the all channel complete interrupt generates. In ACM, interrupt generation for each channel is controlled using the IIR_CTL2.IMASK bit.

MAC Status

A MAC status interrupt generates under the following conditions:

- Multiplier result zero–set when multiplier result is zero
- Multiplier result infinity–set when multiplier result is infinity
- Multiply invalid–set when multiply operation is invalid
- Adder result zero–set when adder result is zero
- Adder result infinity–set when adder result is infinity
- Adder invalid–set when addition is invalid

Service

The DMA interrupt status bits are sticky and are cleared when the DMA status register is read. When a MAC status interrupt occurs, programs can find this state (and clear) by reading the MAC status register (`IIR_MACSTAT`). The MAC interrupt status bits are sticky.

The status interrupt sources are derived from the `IIR_MACSTAT` register. A status interrupt can occur due to the last set of MAC operations of a processing iteration that correspond to a particular channel. The interrupt is generated continuously and cannot be stopped, even after disabling the accelerator. The interrupt can only be stopped when another processing iteration results in a non-zero or valid multiply or add result.

Programming Model

The Legacy Mode Programming section provides programming examples for the Legacy and Auto Configuration Mode.

Programming Legacy Mode and ACM

- Legacy Mode Programming
- Auto Configuration Mode Programming

Legacy Mode Programming

The IIR accelerator supports up to 32 channels which are time division multiplexed (TDM). Each channel can have a maximum of 64 biquads. The window size for each channel is configurable using the control registers. The maximum configured window size is 1024 samples.

For programming multi-channel filtering in Legacy Mode:

1. Program the number of channels using the `IIR_CTL1.CH` bits.
2. Configure the TCBs in system memory with one channel TCB pointing to the next channel TCB. The TCB of the last channel should point to the TCB of the first channel.
3. Write the first TCB value into the `IIR_CHNPTR` register and enable the accelerator.

4. When the IIR_CTL1.BURSTEN bit is set, ensure that the coefficient buffer end address plus seven words does not fall outside the valid address range and the buffer is initialized. When the burst feature is enabled, the IIR module issues an INCR8 burst. When the data count % 8== 7, the extra seven words are always read, but ignored by the IIR module.

   NOTE: Data count = ∑(Bn*7) or ∑(Bn*14); where Bn is number of biquads for channel N. Each biquads contains 7 words for 32-bit mode or 14 words in 40-bit mode.

The ***Biquad Processing Program Flow*** figure shows the biquad processing program flow.

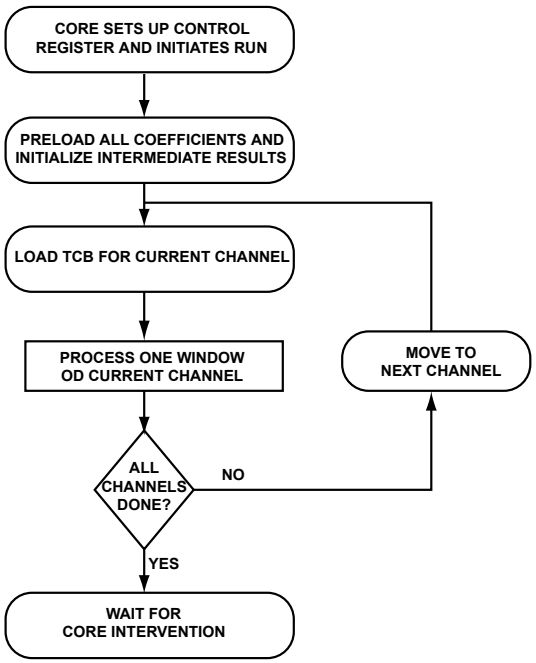


Figure 46-6: Biquad Processing Program Flow

The accelerator uses the following procedure to process biquads.

1. The controller loads all coefficients of all the channels into local storage.
2. Once all the coefficients are loaded, the controller goes to the first biquad of the first channel and calculates the output of the first biquad and updates the intermediate results for that biquad. When the IIR_CTL1.L_TWAIT_EN bit is set and the IIR_CTL2.TWAIT bit is set for the channel, the accelerator waits for the input trigger to start the computation.
3. The accelerator then moves to the next biquad of that channel and repeats the process until all the biquads for that channel are completed and the results are stored to memory.
4. The procedure is repeated for the next sample until processing completes for one window of the corresponding channel.
5. After one channel window is processed, the accelerator moves to the next channel and computes the results.

NOTE: All the addresses must be aligned on 32-bit address boundaries and must not contain the lower two bits (assumed to be zeros).

Auto Configuration Mode Programming

For ACM, there is no limit on the number of channels the accelerator can process. The accelerator keeps processing the TCBs until the chain pointer becomes null. Each channel can have a maximum of 64 biquads. The maximum configured window size is 1024 samples.

The Multi-channel Filtering in Auto Configuration Mode figure shows multi-channel filtering in ACM. Multiple channels are processed in a time division multiplexed (TDM) format.

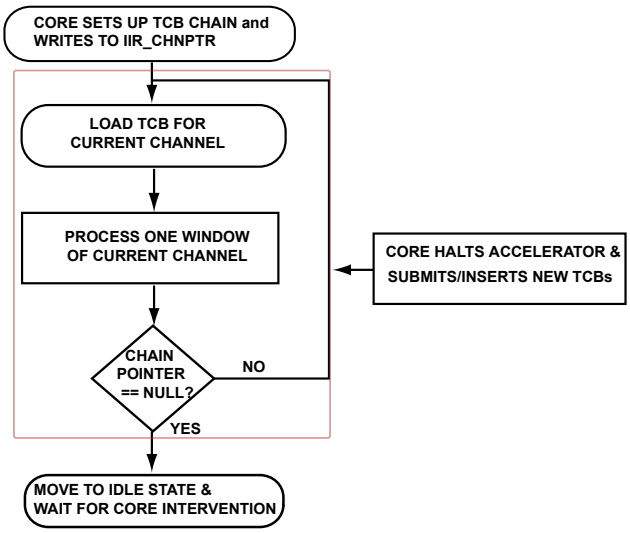


Figure 46-7: Multi-channel Filtering in Auto Configuration Mode

To program and understand the multi-channel filtering in ACM:

1. Configure the TCBs in system memory with the one channel TCB pointing to the next channel TCB. There is no limit on the number of channels to be configured. The TCB of the last channel does not need to point to the TCB of the first channel.
2. Write the first TCB value to the `IIR_CHNPTR` register and enable the accelerator.

   The accelerator fetches the first channel TCB. Using it as pointer, loads the biquad coefficients and biquad states for that channel into the IIR local memory and loads the `IIR_CTL2` and `IIR_SGCTL` registers to configure the filter parameters corresponding to that channel.

   When the `IIR_CTL2.TWAIT` bit is set, the accelerator waits for a input trigger to start the window processing for the channel.

   The accelerator computes the output of the first biquad and updates the intermediate results for that biquad. Then, the accelerator moves to the next biquad of that channel and repeats the process until all the biquads for that channel are completed and the results are stored to memory. The process is repeated with the next sample

until one window of the corresponding channel is processed. When the IIR_CTL1.SS bit is set, the accelerator saves the states of the biquads of the channel into system memory. If the IIR_CTL2.TMASK bit is cleared the accelerator sends an output trigger after completion of processing the channel.

When the IIR_CTL2.IMASK bit is cleared, the accelerator interrupts the core after completion of processing a particular channel.

At the end of the window, the accelerator updates the IIR_INIDX and IIR_OUTIDX registers to 0x00000000 and 0xFFFFFFFF in the TCB of system memory and moves to the next channel.

3. At any instant, the core can halt the accelerator as needed. It sets the IIR_CTL1.HALT bit and appropriately takes action to submit/insert new TCBs and clears the IIR_CTL1.HALT bit to resume channel processing.

   When the IIR_CHNPTR register is zero (last channel is being processed or channel processing is complete) after halting the accelerator, the IIR_CHNPTR register is written before resuming the accelerator channel processing.

   When the accelerator is idle after halt, IIR_CTL1.EN bit is toggled to disable and re-enable the accelerator and the IIR_CTL1.HALT bit is cleared and the accelerator resumes channel processing.

4. The accelerator continues processing until all the channels are complete. Repeat *Step 3* (as required) to submit or insert new channels.

ADSP-SC59x IIR Register Descriptions

The IIR filter accelerator (IIR) contains the following registers.

Table 46-8: ADSP-SC59x IIR Register List

Name	Description
IIR_CHNPTR	Chain Pointer Register
IIR_COEFIDX	Coefficient Buffer Index Register
IIR_COEFLEN	Coefficient Buffer Length Register
IIR_COEFMOD	Coefficient Index Modifier Register
IIR_CTL1	Global Control Register
IIR_CTL2	Channel Control Register
IIR_DBG_ADDR	IIR Debug Address Register
IIR_DBG_CTL	IIR Debug Control Register
IIR_DBG_RDDAT_HI	IIR Debug Read Data High Register
IIR_DBG_RDDAT_LO	IIR Debug Read Data Low Register
IIR_DBG_WRDAT_HI	IIR Debug Write Data High Register
IIR_DBG_WRDAT_LO	IIR Debug Write Data Low Register

Table 46-8: ADSP-SC59x IIR Register List (Continued)

Name	Description
IIR_DMASTAT	DMA Status Register
IIR_INBASE	Input Buffer Base Register
IIR_INIDX	Input Data Index Register
IIR_INLEN	Input Data Buffer Length Register
IIR_INMOD	Input Data Index Modifier Register
IIR_MACSTAT	MAC Status Register
IIR_OUTBASE	Output Buffer Base Register
IIR_OUTIDX	Output Data Buffer Index Register
IIR_OUTLEN	IIR Output Data Buffer Length Register
IIR_OUTMOD	IIR Output Data Index Modifier Register
IIR_SCTL1	Software Control Register1
IIR_SCTL2	Software Control Register2
IIR_SGCTL	Secondary Global Control Register

Chain Pointer Register

The `IIR_CHNPTR` register should be written with word address without the lower 2 bits.

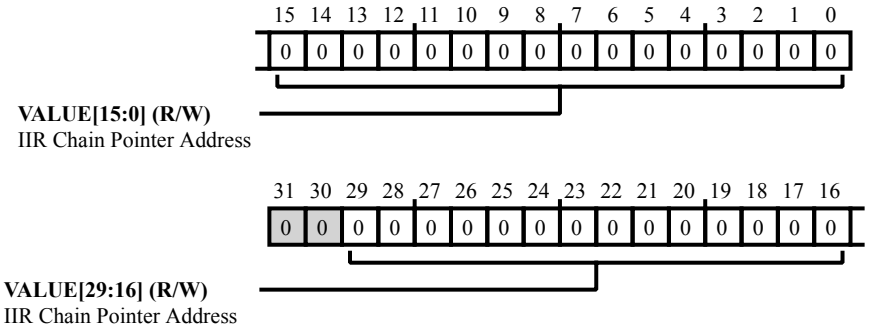


Figure 46-8: IIR_CHNPTR Register Diagram

Table 46-9: IIR_CHNPTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	IIR Chain Pointer Address. The `IIR_CHNPTR.VALUE` bit field contains the chain pointer address.

Coefficient Buffer Index Register

The IIR_COEFIDX register contains the word address with the lower 2 bits removed.

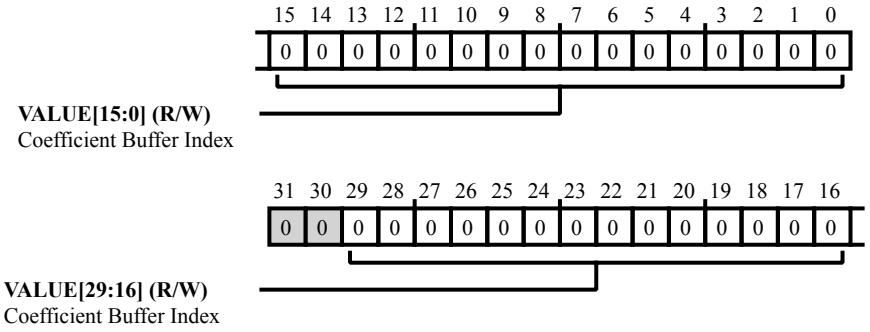


Figure 46-9: IIR_COEFIDX Register Diagram

Table 46-10: IIR_COEFIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Coefficient Buffer Index. The IIR_COEFIDX.VALUE bit field provides the coefficient buffer index.

Coefficient Buffer Length Register

The `IIR_COEFLEN` register provides the coefficient buffer length.

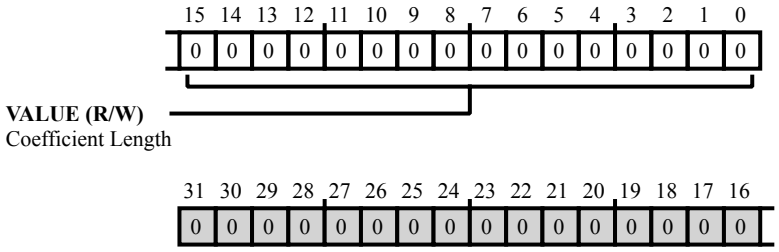


Figure 46-10: IIR_COEFLEN Register Diagram

Table 46-11: IIR_COEFLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Coefficient Length. The `IIR_COEFLEN.VALUE` bit field provides the coefficient buffer length.

Coefficient Index Modifier Register

The IIR_COEFMOD register provides the coefficient index modifier.

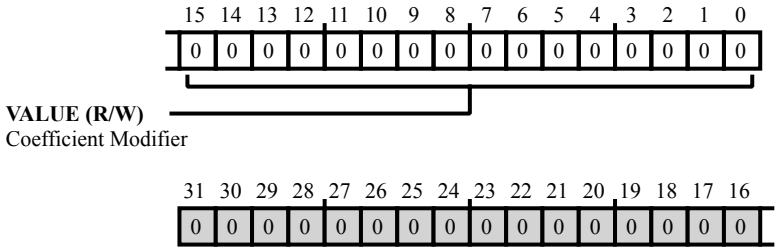


Figure 46-11: IIR_COEFMOD Register Diagram

Table 46-12: IIR_COEFMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Coefficient Modifier. The IIR_COEFMOD.VALUE bit field provides the coefficient modifier.

Global Control Register

The `IIR_CTL1` register is used to configure the global parameters for the accelerator. These parameters include the number of channels, channel auto iterate, DMA enable, and accelerator enable.

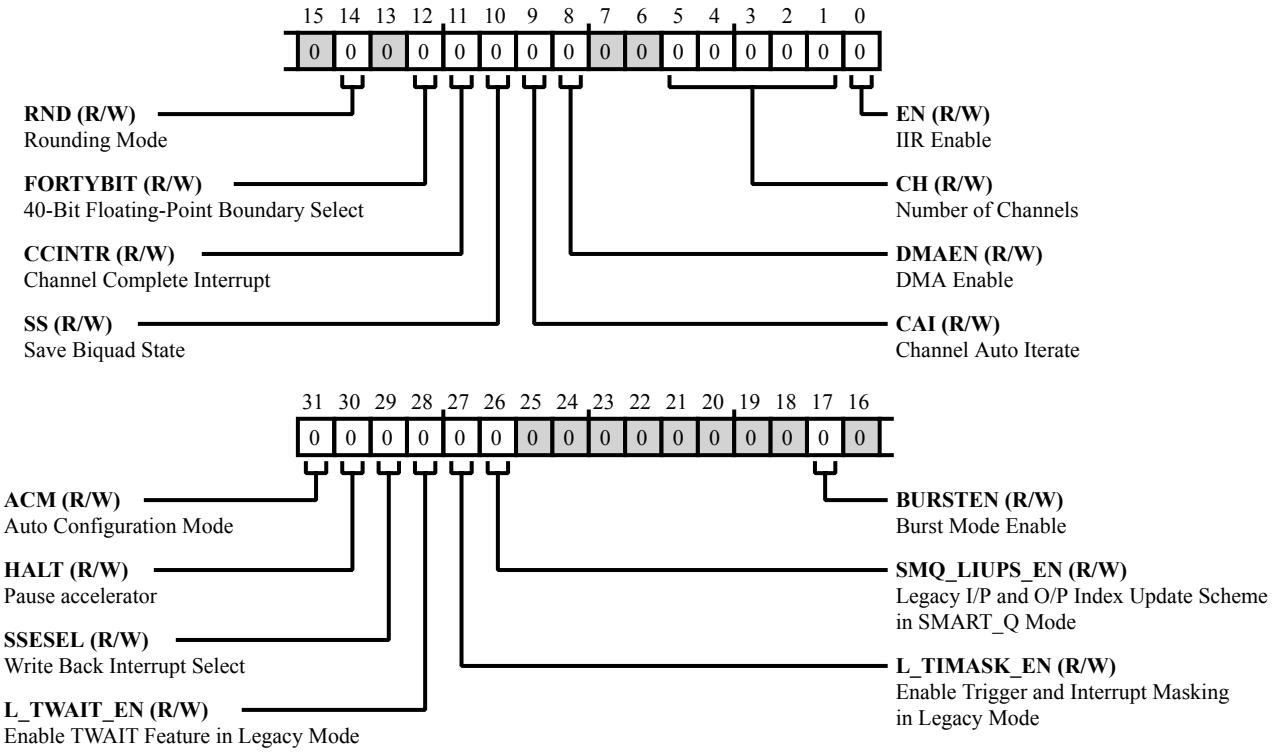


Figure 46-12: IIR_CTL1 Register Diagram

Table 46-13: IIR_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	ACM	Auto Configuration Mode. The `IIR_CTL1.ACM` bit configures the mode for loading the TCB.	
		0	Legacy Mode
		1	Auto Configuration Mode
30 (R/W)	HALT	Pause accelerator. The `IIR_CTL1.HALT` bit determines whether the accelerator pauses so that the core can check the status and modify or submit a job or the accelerator is released for further processing of data. This bit is only valid in Auto Configuration Mode (ACM).	
		0	Release accelerator
		1	Pause accelerator

Table 46-13: IIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	SSESEL	Write Back Interrupt Select. The `IIR_CTL1.SSESEL` bit selects whether to generate an interrupt after the write back or after save state completion.	
		0	Generate Interrupt after save state completion
		1	Generate interrupt after write back completion
28 (R/W)	L_TWAIT_EN	Enable TWAIT Feature in Legacy Mode. The `IIR_CTL1.L_TWAIT_EN` bit indicates whether the TWAIT feature is available in legacy mode. This bit is only valid in legacy mode.	
		0	Not Available
		1	Available
27 (R/W)	L_TIMASK_EN	Enable Trigger and Interrupt Masking in Legacy Mode. The `IIR_CTL1.L_TIMASK_EN` bit enables trigger and interrupt masking in legacy mode. This bit is only valid in legacy mode.	
		0	Disable
		1	Enable
26 (R/W)	SMQ_LIUPS_EN	Legacy I/P and O/P Index Update Scheme in SMART_Q Mode. The `IIR_CTL1.SMQ_LIUPS_EN` bit configures the scheme the accelerator uses to update the input index (II) and output index (OI). This bit is only valid in ACM.	
		0	Update the II with all '0' and the OI with all 'F', respectively
		1	Update II and OI after circular buffer scheme
17 (R/W)	BURSTEN	Burst Mode Enable. The `IIR_CTL1.BURSTEN` bit field is set, burst access is enabled.	
		0	Disable burst mode
		1	Enable burst mode
14 (R/W)	RND	Rounding Mode. The `IIR_CTL1.RND` bit field selects the rounding mode for floating-point format.	
		0	Round to nearest (even)
		1	Truncate (Round towards zero)

Table 46-13: IIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	FORTYBIT	40-Bit Floating-Point Boundary Select. The `IIR_CTL1.FORTYBIT` bit selects between 32-bit IEEE floating-point format or 40-bit IEEE floating-point format.	
		0	32-bit IEEE floating-point
		1	40-bit IEEE floating-point
11 (R/W)	CCINTR	Channel Complete Interrupt. The `IIR_CTL1.CCINTR` bit configures the channel complete interrupt to generate when all channels are done or after each channel is done. This bit is only valid in legacy mode when the IIR_CTL1.L_TIMASK_EN bit is cleared (=0). In Auto Configuration Mode (ACM) or when the IIR_CTL1.L_TIMASK_EN is set (=1), interrupt generation for each channel is controlled using IIR_CTL2.IMASK bit.	
		0	Interrupt is generated only when all channels are done (default)
		1	Interrupt is generated after each channel is done (default)
10 (R/W)	SS	Save Biquad State. The `IIR_CTL1.SS` bit configures the accelerator to store the Dk register settings into the internal memory. This can be used to save the biquad states before switching to another high priority accelerator task. This bit is only valid in legacy mode. In Auto Configuration Mode (ACM), the save state for each channel is controlled using the IIR_SGCTL.SS bit	
9 (R/W)	CAI	Channel Auto Iterate. The `IIR_CTL1.CAI` bit sets whether TDM processing stops (idle) once all channels complete processing or moves to first channel and continues TDM processing in a loop when all channels complete processing. Channel Auto Iterate is not available in Auto Configuration Mode (ACM). The accelerator keeps processing the TCBs until the chain pointer becomes null in ACM.	
		0	TDM processing stops (idle) once all channels complete processing
		1	Moves to first channel and continues TDM processing in a loop when all channels complete processing
8 (R/W)	DMAEN	DMA Enable. The `IIR_CTL1.DMAEN` bit enables DMA on the accelerator.	
		0	Disable
		1	Enable

Table 46-13: IIR_CTL1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5:1 (R/W)	CH	Number of Channels. The `IIR_CTL1.CH` bit field configures the number of channels and is programmable between 0-31 (channels = NCH + 1). This bit field is only valid in legacy mode. In Auto Configuration Mode (ACM), there is no limit on the number of channels. the accelerator keeps processing the TCBs until the chain pointer becomes null.	
0 (R/W)	EN	IIR Enable. The `IIR_CTL1.EN` bit enables or disables the IIR accelerator.	
		0	IIR disabled
		1	IIR enabled

Channel Control Register

The IIR_CTL2 register is used to configure the channel specific parameters. These parameters include the number of biquads and window size. In Auto Configuration Mode (ACM), this register is also used to configure additional channel specific parameters like interrupts and triggers.

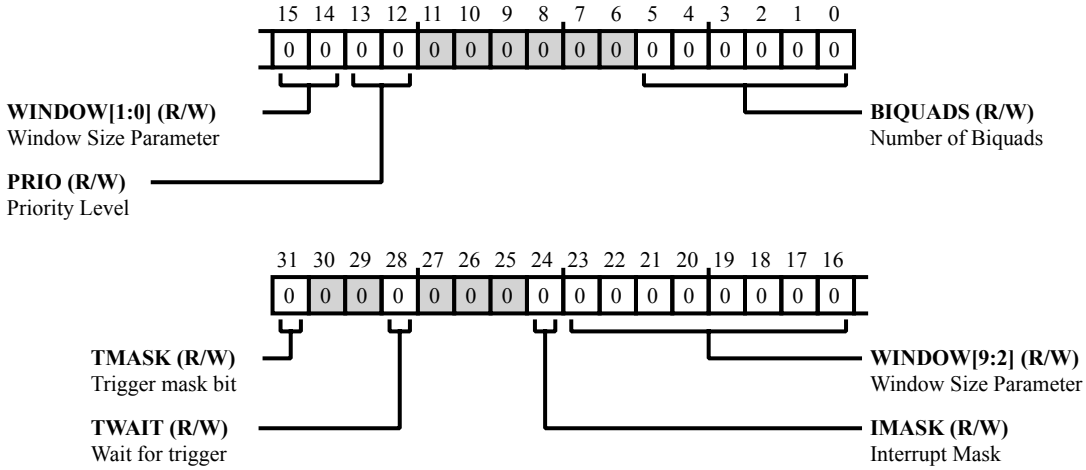


Figure 46-13: IIR_CTL2 Register Diagram

Table 46-14: IIR_CTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	TMASK	Trigger mask bit. The IIR_CTL2.TMASK bit enables trigger generation for the channel. It is valid in Auto Configuration Mode (ACM) or if CTL1.L_TIMASK_EN is set.	
		0	Enable
		1	Mask
28 (R/W)	TWAIT	Wait for trigger. The IIR_CTL2.TWAIT bit disables the wait for the trigger. It is valid in Auto Configuration Mode (ACM) or if CTL1.L_TIMASK_EN is set.	
		0	Disable wait for the trigger for the channel
		1	Enable wait for external trigger input assertion for the channel
24 (R/W)	IMASK	Interrupt Mask. The IIR_CTL2.IMASK bit enables interrupt generation for the channel. This bit is valid in Auto Configuration Mode (ACM) or if CTL1.L_TIMASK_EN is set.	
		0	Enable
		1	Mask

Table 46-14: IIR_CTL2 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
23:14 (R/W)	WINDOW	Window Size Parameter. The `IIR_CTL2.WINDOW` bit field configures the window size which specifies the number of samples/blocks to process (sample based processing = window size of 1). This bit field should be programmed to "actual window size required -1". For example, for sample based processing this bit field should be programmed to 0.	
13:12 (R/W)	PRIO	Priority Level. The `IIR_CTL2.PRIO` bit field indicates the priority.	
		0	Level 0 (lowest)
		1	Level 1
		2	Level 2
		3	Level 3 (highest)
5:0 (R/W)	BIQUADS	Number of Biquads. The `IIR_CTL2.BIQUADS` bit field configures the number of biquads and is programmable between 0-63 (number of Biquads = BIQUADS + 1).	

IIR Debug Address Register

The IIR_DBG_ADDR register holds the debug address. If bit 11 is set, coefficient memory is selected.

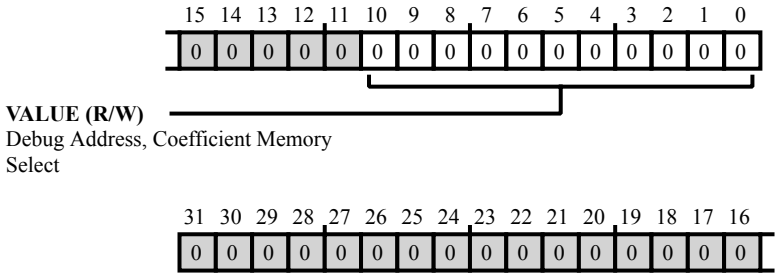


Figure 46-14: IIR_DBG_ADDR Register Diagram

Table 46-15: IIR_DBG_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/W)	VALUE	Debug Address, Coefficient Memory Select. The IIR_DBG_ADDR.VALUE bit field holds the debug address (bits 0-10). Bit 11 configures whether the memory access is to coefficient memory (=0) or to delay line memory (=1).

IIR Debug Control Register

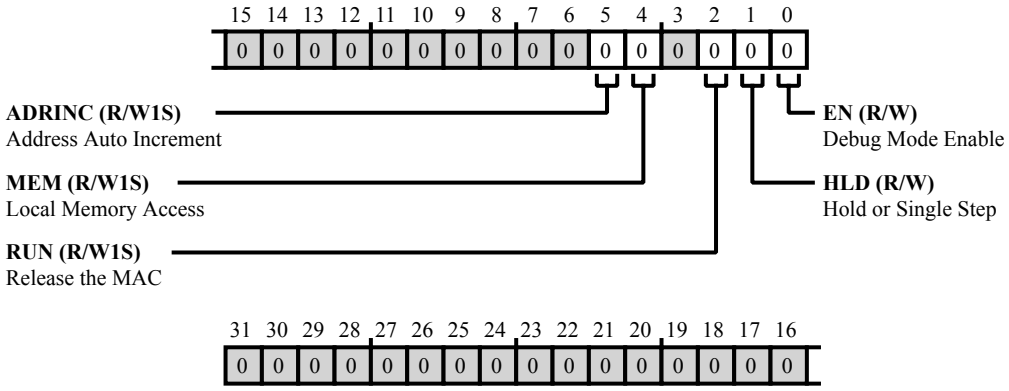


Figure 46-15: IIR_DBG_CTL Register Diagram

Table 46-16: IIR_DBG_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (R/W1S)	ADRINC	Address Auto Increment. The `IIR_DBG_CTL.ADRINC` bit allows the address register to auto increment on `IIR_DBG_WRDAT_HI`/`IIR_DBG_WRDAT_LO` writes and `IIR_DBG_RDDAT_HI`/`IIR_DBG_RDDAT_LO` reads.	
4 (R/W1S)	MEM	Local Memory Access. The `IIR_DBG_CTL.MEM` bit allows the data and coefficients memory to be indirectly accessed.	
2 (R/W1S)	RUN	Release the MAC. The `IIR_DBG_CTL.RUN` bit releases the MAC; it is is self clearing after one IIR clock cycle.	
1 (R/W)	HLD	Hold or Single Step. The `IIR_DBG_CTL.HLD` bit function is based on the `IIR_DBG_CTL.MEM` bit setting. For `IIR_DBG_CTL.MEM` = 0 this bit sets single step. For `IIR_DBG_CTL.MEM` = 1 this bit sets hold data.	
		0	No effect
		1	Single step (IIR_DBGMEM=0) or Hold data (IIR_DBGMEM=1)
0 (R/W)	EN	Debug Mode Enable. The `IIR_DBG_CTL.EN` bit enables debug mode. For local memory access, the `IIR_CTL1` register can be cleared.	
		0	Disable
		1	Enable

IIR Debug Read Data High Register

The `IIR_DBG_RDDAT_HI` register is part of the 40-bit wide debug mode read data register and holds the upper 8 bits.

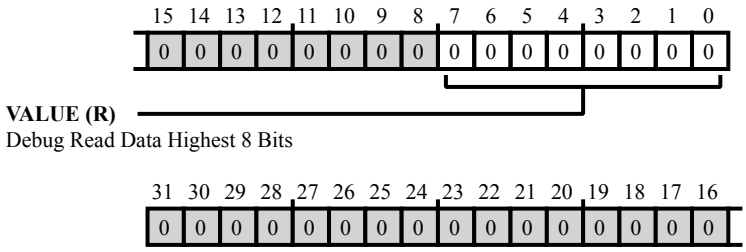


Figure 46-16: IIR_DBG_RDDAT_HI Register Diagram

Table 46-17: IIR_DBG_RDDAT_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	VALUE	Debug Read Data Highest 8 Bits. The `IIR_DBG_RDDAT_HI.VALUE` bit field holds the upper 8-bit read data.

IIR Debug Read Data Low Register

The IIR_DBG_RDDAT_LO register is part of the 40-bit wide debug mode read data register and holds the lower 32 bits.

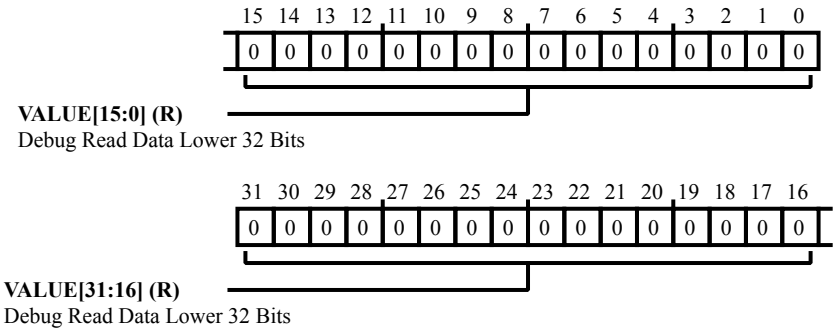


Figure 46-17: IIR_DBG_RDDAT_LO Register Diagram

Table 46-18: IIR_DBG_RDDAT_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	Debug Read Data Lower 32 Bits. The IIR_DBG_RDDAT_LO.VALUE bit field holds the lower 32-bit read data.

IIR Debug Write Data High Register

The `IIR_DBG_WRDAT_HI` register is part of the 40-bit wide debug mode write data register and holds the upper 8 bits.

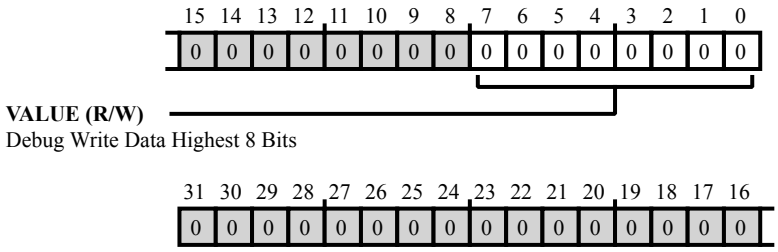


Figure 46-18: IIR_DBG_WRDAT_HI Register Diagram

Table 46-19: IIR_DBG_WRDAT_HI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	VALUE	Debug Write Data Highest 8 Bits. The `IIR_DBG_WRDAT_HI.VALUE` bit field holds the upper 8-bit write data.

IIR Debug Write Data Low Register

The IIR_DBG_WRDAT_LO register is part of the 40-bit wide debug mode write data register and holds the lower 32 bits.

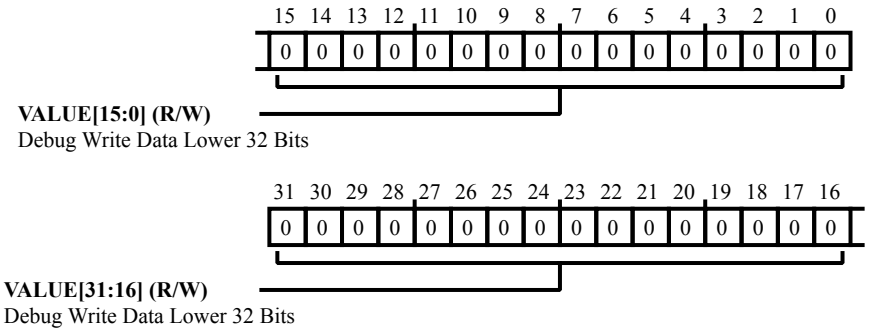


Figure 46-19: IIR_DBG_WRDAT_LO Register Diagram

Table 46-20: IIR_DBG_WRDAT_LO Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Debug Write Data Lower 32 Bits. The IIR_DBG_WRDAT_LO.VALUE bit field holds the lower 32-bit write data.

DMA Status Register

The IIR_DMASTAT registers indicate the status of DMA operations.

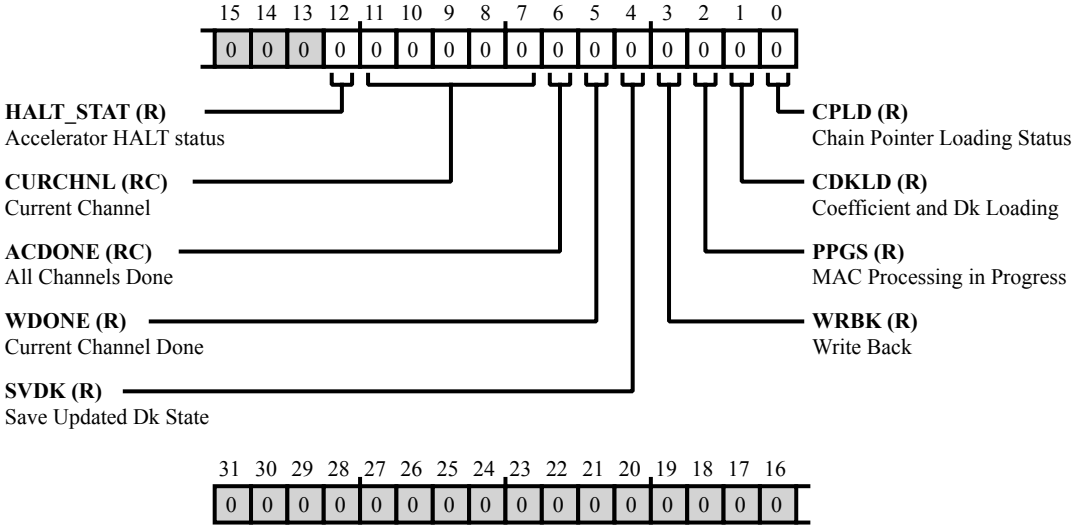


Figure 46-20: IIR_DMASTAT Register Diagram

Table 46-21: IIR_DMASTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
12 (R/NW)	HALT_STAT	Accelerator HALT status. Acknowledge to core that accelerator is in halt state, if set.
11:7 (RC/NW)	CURCHNL	Current Channel. The IIR_DMASTAT.CURCHNL bit field indicates the channel that is being processed in the TDM slot. Zero indicates the last slot.
6 (RC/NW)	ACDONE	All Channels Done. The IIR_DMASTAT.ACDONE bit indicates all channels are done processing. Note that the IIR_CTL1.CCINTR bit does not affect this status bit. This bit is sticky and is cleared on register read.
5 (R/NW)	WDONE	Current Channel Done. The IIR_DMASTAT.WDONE bit indicates the processing of the current channel is complete. Note that the IIR_CTL1.CCINTR bit does not affect this status bit. This bit is sticky and is cleared on register read.
4 (R/NW)	SVDK	Save Updated Dk State. If there is more than one channel (IIR_CTL1.CH>0), the IIR_DMASTAT.SVDK bit toggles between 0 and 1 as it starts and completes the save state operation on one channel at a time.

Table 46-21: IIR_DMASTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (R/NW)	WRBK	Write Back. The `IIR_DMASTAT.WRBK` bit indicates the accelerator is writing back updated index registers.	
2 (R/NW)	PPGS	MAC Processing in Progress. The `IIR_DMASTAT.PPGS` bit indicates MAC processing is in progress.	
1 (R/NW)	CDKLD	Coefficient and Dk Loading. The `IIR_DMASTAT.CDKLD` bit indicates the coefficient and Dk are loading.	
0 (R/NW)	CPLD	Chain Pointer Loading Status. The `IIR_DMASTAT.CPLD` bit indicates the IIR is in the chain pointer load state.	
		0	State machine not in chain pointer load state
		1	State machine in chain pointer load state

Input Buffer Base Register

The IIR_INBASE register contains the word address with the lower 2 bits removed.

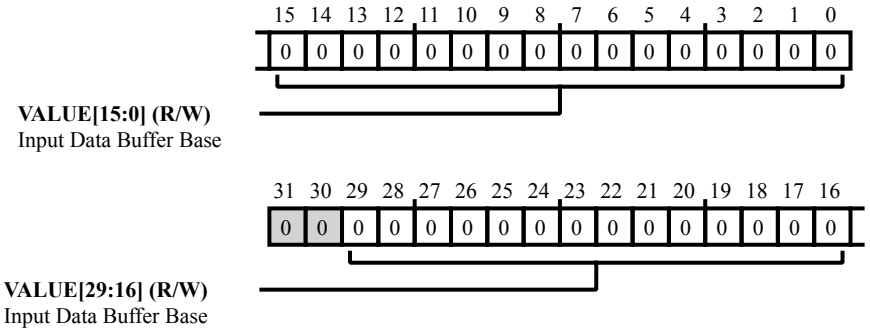


Figure 46-21: IIR_INBASE Register Diagram

Table 46-22: IIR_INBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Input Data Buffer Base. The IIR_INBASE.VALUE bit field value is the input data buffer base address.

Input Data Index Register

The IIR_INIDX register contains a word address with the lower 2 bits removed.

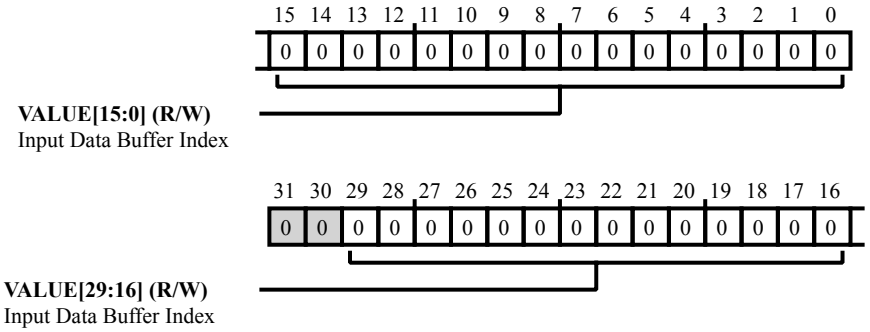


Figure 46-22: IIR_INIDX Register Diagram

Table 46-23: IIR_INIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Input Data Buffer Index. The IIR_INIDX.VALUE bit field value is the input data buffer index.

Input Data Buffer Length Register

The `IIR_INLEN` register provides the input data buffer length.

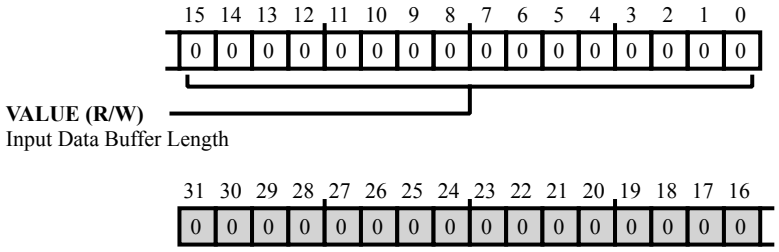


Figure 46-23: IIR_INLEN Register Diagram

Table 46-24: IIR_INLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Input Data Buffer Length. The `IIR_INLEN.VALUE` bit field value is the input data buffer length.

Input Data Index Modifier Register

The IIR_INMOD register provides the 16-bit input data buffer index modifier.

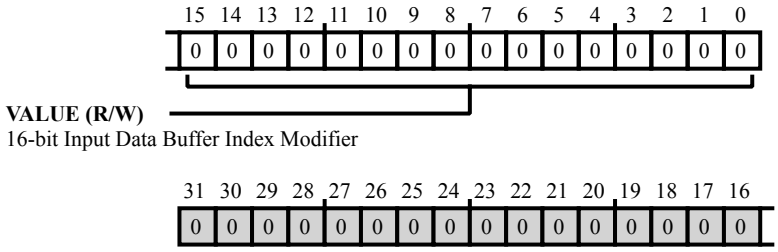


Figure 46-24: IIR_INMOD Register Diagram

Table 46-25: IIR_INMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Buffer Index Modifier. The IIR_INMOD.VALUE bit field value is the 16-bit input data buffer modifier.

MAC Status Register

The IIR_MACSTAT register indicates the status of MAC operations.

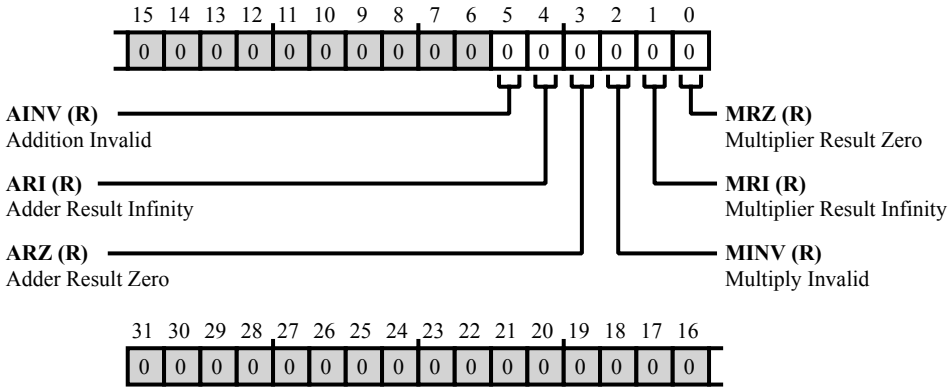


Figure 46-25: IIR_MACSTAT Register Diagram

Table 46-26: IIR_MACSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
5 (R/NW)	AINV	Addition Invalid. The IIR_MACSTAT.AINV bit indicates the addition is invalid.
4 (R/NW)	ARI	Adder Result Infinity. The IIR_MACSTAT.ARI bit indicates the adder result is infinity.
3 (R/NW)	ARZ	Adder Result Zero. The IIR_MACSTAT.ARZ bit indicates the adder result is zero.
2 (R/NW)	MINV	Multiply Invalid. The IIR_MACSTAT.MINV bit indicates the multiply operation is invalid.
1 (R/NW)	MRI	Multiplier Result Infinity. The IIR_MACSTAT.MRI bit indicates the multiplier result is infinity.
0 (R/NW)	MRZ	Multiplier Result Zero. The IIR_MACSTAT.MRZ bit indicates the multiplier result is zero.

Output Buffer Base Register

The IIR_OUTBASE register contains the word address with the lower 2 bits removed.

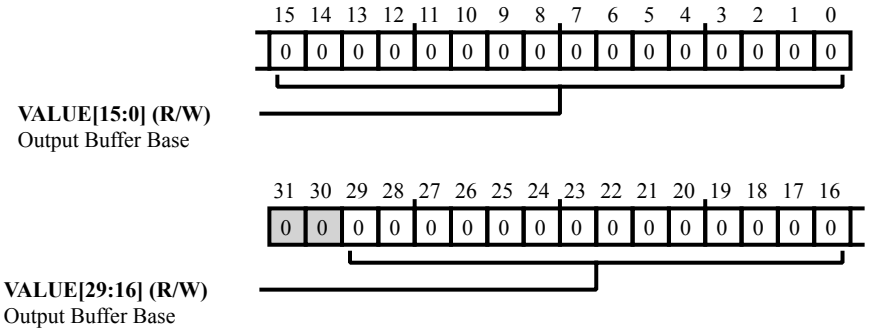


Figure 46-26: IIR_OUTBASE Register Diagram

Table 46-27: IIR_OUTBASE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Output Buffer Base. The IIR_OUTBASE.VALUE bit field provides the output buffer base address.

Output Data Buffer Index Register

The IIR_OUTIDX register is written with word address without the lower 2 bits.

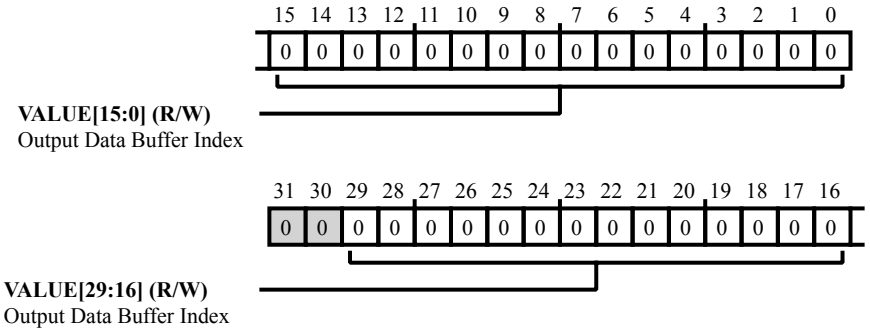


Figure 46-27: IIR_OUTIDX Register Diagram

Table 46-28: IIR_OUTIDX Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
29:0 (R/W)	VALUE	Output Data Buffer Index. The IIR_OUTIDX.VALUE bit field provides the output data buffer index.

IIR Output Data Buffer Length Register

The `IIR_OUTLEN` register provides the output data buffer length.

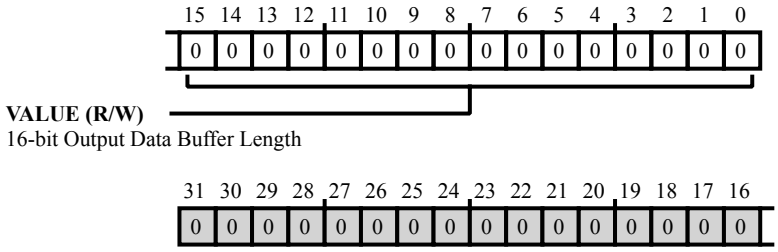


Figure 46-28: IIR_OUTLEN Register Diagram

Table 46-29: IIR_OUTLEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Output Data Buffer Length. The `IIR_OUTLEN.VALUE` bit field provides the output data buffer length.

IIR Output Data Index Modifier Register

The IIR_OUTMOD register provides the output data index modifier.

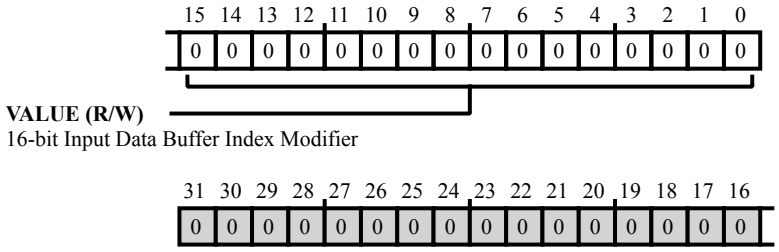


Figure 46-29: IIR_OUTMOD Register Diagram

Table 46-30: IIR_OUTMOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	16-bit Input Data Buffer Index Modifier. The IIR_OUTMOD.VALUE bit field provides the output data buffer index modifier.

Software Control Register1

The IIR_SCTL1 register is used by Smart Queuing APIs in ACM mode to exchange relevant job information amongst APIs.

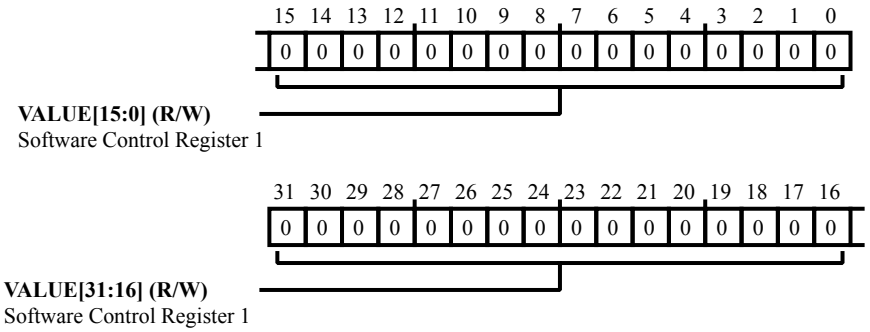


Figure 46-30: IIR_SCTL1 Register Diagram

Table 46-31: IIR_SCTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Software Control Register 1.

Software Control Register2

The IIR_SCTL2 register is used by Smart Queuing APIs in ACM mode to exchange relevant job information amongst APIs.

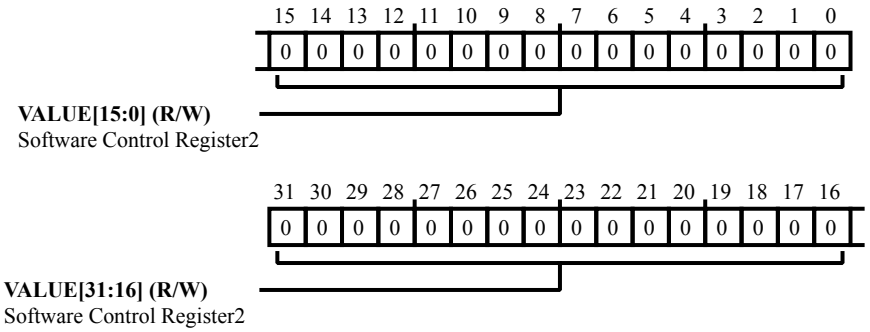


Figure 46-31: IIR_SCTL2 Register Diagram

Table 46-32: IIR_SCTL2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Software Control Register2.

Secondary Global Control Register

The IIR_SGCTL register configures the global parameters for the accelerator in ACM mode for loading CTL1 register as part of TCB.

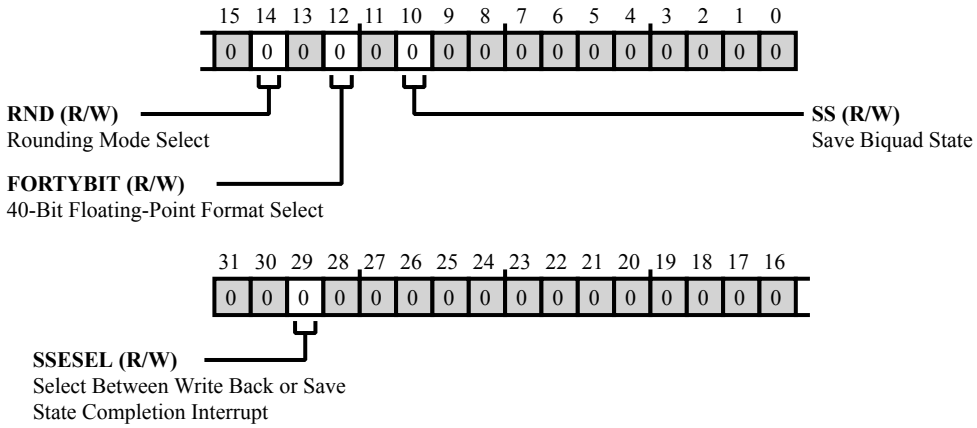


Figure 46-32: IIR_SGCTL Register Diagram

Table 46-33: IIR_SGCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	SSESEL	Select Between Write Back or Save State Completion Interrupt.	
14 (R/W)	RND	Rounding Mode Select. The IIR_SGCTL.RND bit selects the rounding mode for a floating-point mode compatible with SHARC+.	
		0	Round to nearest (even)
		1	Truncate (Round towards zero)
12 (R/W)	FORTYBIT	40-Bit Floating-Point Format Select. The IIR_SGCTL.FORTYBIT bit selects the floating point format.	
		0	32-bit floating point
		1	40-bit floating point
10 (R/W)	SS	Save Biquad State. The IIR_SGCTL.SS bit configures the accelerator to store the Dk register settings into the internal memory. This can be used to save the biquad states before switching to another high priority accelerator task.	

47 Boot ROM and Booting the Processor

Bootstrapping or booting is the series of events that occur when the system applies power to the processor or when the processor enters a hardware reset state. This section gives an in-depth description of these events and how to integrate an application effectively.

On reset, the processor begins fetching instruction from an internal ROM. The boot code contained within the ROM is designed to facilitate loading an application. The boot code can automatically initialize certain peripherals for communication based on a chosen boot mode, then load an application. For more information on what boot modes are available, see the Boot Modes section. The boot code can efficiently load an entire application, code, and data, into appropriate locations after the development tools repackage the application into a boot stream.

A boot stream is an application or data that the boot-loader tool splits into blocks. A 16-byte header in each block provides instruction to the boot code for processing the associated data. The processor can perform several boot functions, depending on the flags set in the header. For more details on what options are available and a description of the stream format, refer to the Boot Loader Stream section.

The boot ROM provides a mechanism through available non-volatile programmable memory (OTP on this processor) to customize different aspects of the boot process. These customizations include overriding default boot-peripheral instance, overriding default peripheral-timing parameters, and disabling boot modes.

NOTE: In this chapter the term boot ROM describes the boot code that cannot be altered. The terms *program* and *application* are used to describe code that is used to customize the boot process.

Several utilities of the boot code are also available to the application. These utilities include features such as copying memory, comparing memory, or loading another boot stream at run time. The APIs may be used to help ensure that application code is more compatible with future products. The boot code also provides the ability to define a custom boot mode. This capability helps when support is not available for a desired boot mode. It allows second stage boot loaders for unsupported boot peripherals to leverage a significant amount of the existing boot ROM functionality.

SRAM Requirements

The boot process reserves 8K bytes of L2 ECC Protected SRAM for dedicated use. This topic describes how the reserved memory region is used during boot.

The boot process requires SRAM resources for stack use and to store various data items that require read/write access during the boot process. 8K bytes of L2 ECC protected SRAM is reserved for this purpose. The ***Boot Process SRAM Requirements*** table describes the various items stored in this memory region.

Table 47-1: Boot Process SRAM Requirements

Address	Size (Bytes)	Item	Description
0x201FE000	4	Reserved	
0x201FE004	4	Pointer to the struct ADI_ROM_BOOT_CONFIG object	Pointer to the boot configuration structure that is located on the stack. This location is used to find the location of the boot structure on the stack for debug.
0x201FE008	8	Reserved	
0x201FE010	1024	Internal Intermediate Buffer 0	The first of two internal buffers used for intermediate storage of boot content when using indirect and page mode accesses and for secure boot operations. Two buffers are used to allow SHA-224 and AES-128 operations to be performed on one buffer while simultaneously loading the other buffer.
0x201FE410	1024	Internal Intermediate Buffer 1	The second of two internal buffers used for intermediate storage of boot content when using indirect and page mode accesses and for secure boot operations. Two buffers are used to allow SHA-224 and AES-128 operations to be performed on one buffer while simultaneously loading the other buffer.
0x201FE810	16	struct ADI_ROM_BOOT_HEADER object	Storage location for all the block headers of the boot stream.
0x201FE820	2912	Storage for Secure Boot related descriptors	Contains a number of buffers for the various descriptors used by the Cryptographic Accelerators and provides storage for the secure header of a secure boot stream.
0x201FF400	3072	Stack for the boot process	The primary booting cores stack. The processor core should locate the stack in this region to preserve security in secure boot operations.

NOTE: To preserve the security of the product, the 8K byte region described here is not a bootable region of memory. If the boot process determines that a block of data in the boot stream is targeted towards this memory region, the boot process terminates and enters either the default error handler or, if applicable, a user-defined error handler. This reserved memory region is free for use after the boot process completes. To preserve security when using the boot API to boot a secure boot stream, the stack used during the execution of the boot API must be located at the default location in this reserved 8K bytes region of memory.

Preboot Operations

Preboot is responsible for configuration of all system resources prior to executing the required boot operation.

The steps performed by the preboot process are described here in the order of execution. Numerous stages of the preboot process are conditional based upon the content of the `RCU_BCODE` register.

NOTE: When a power on reset, hard reset, and software triggered system reset event completes, the processor operates in PLL Bypass mode (default). Partway through the preboot sequence the processor is brought into Full-On mode at the default settings unless the program has provisioned custom CGU settings in the OTP. The Oscillator Watchdog fault, enabled by default when reset completes, is disabled at this stage unless Oscillator Watchdogs settings are also supplied.

Start-up Sequence

This section describes the initial start-up sequence of all cores in the processor.

Upon completion of a power-on reset, hardware reset, or system reset event, only a single core is released from reset. It is responsible for managing the boot process. The following sections describe in detail the sequence of events that occur for each of the cores on the various product derivatives.

Core Reset Sequencing

System and hardware reset events result in the processor state being reset and the boot sequence is executed. Only a single core is initially released from reset and starts execution of the preboot software from the boot ROM. The sequencing between the RCU and the various cores is shown for each of the product derivatives.

For the ADSP-SC598 variant, core 0 is responsible for controlling the boot process. Cores 1 and 2 are held in reset until core 0 releases them from reset during the preboot phase of the boot process.

The initial sequence of events after the completion of the reset sequence are shown below.

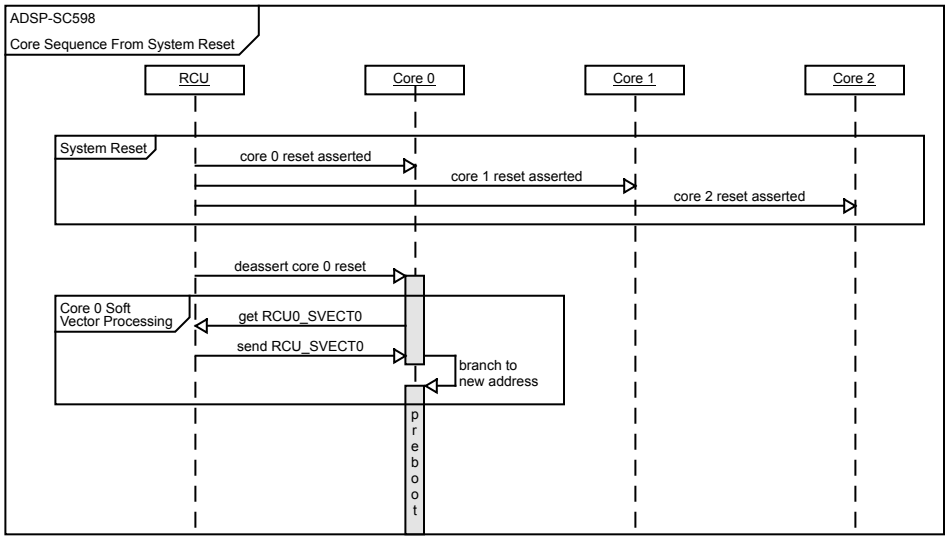


Figure 47-1: Core Reset Sequencing - ADSP-SC598

The ADSP-SC596 variant contains two cores with IDs of 0 and 1. In this device, core 0 is responsible for controlling the boot process; core 1 is held in reset until released by core 0 during the preboot phase of the boot process. The initial sequence of events after the completion of the reset sequence are shown.

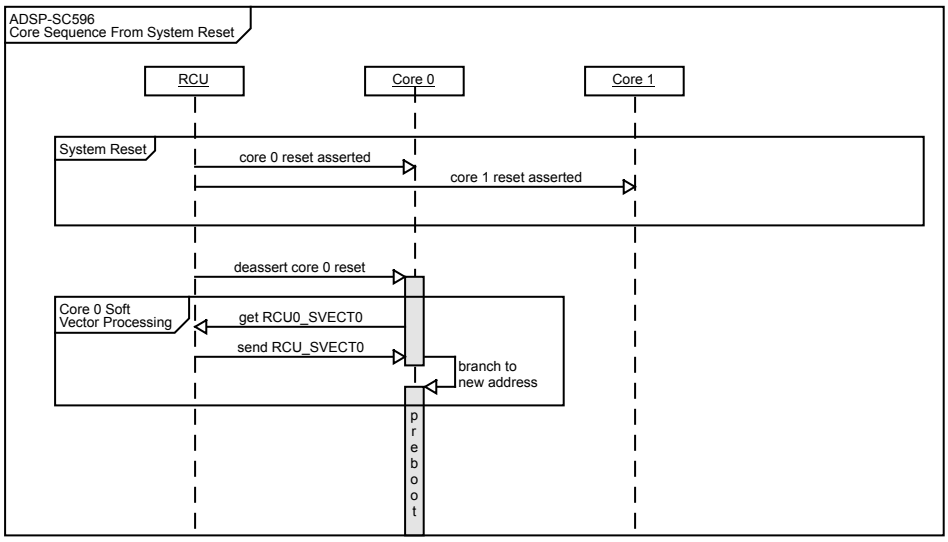


Figure 47-2: Core Reset Sequencing - ADSP-SC596

Core 0 Start-up

Describes the initial operations performed by the core 0 immediately after being released from reset and soft vector processing has completed. This topic is only applicable to the ADSP-SC59x processors.

Upon initial release from reset, the soft vectoring process results in core 0 executing the boot process as long as the `RCU_SVECT0` register contained the default reset value during the soft vector processing.

The core performs the following initial operations:

- Clear `RCU_MSG.C0IDLE`
- Check for OTPC boot completion
- Initialize the register banks X0 to X30 to zero
- Initialize SCTLR_EL3
- Update the vector base address to point to AARCH64 exceptions in VBAR_EL3/EL2/EL1
- Determine if the CPU MIDR is A55. If not jump to forever WFI loop.
- Disable power down bit for Cortex A55 CPU in CORTEX_A55_CPUPWRCTLR_EL1 register.
- Check if the CPU is 0. If not, then jump to forever WFI loop.
- Initialize the translation table control registers.
- Set memory attribute for all the address space as Normal Non-Cacheable. Only MMR and OTP space are configured as Device memory.
- Initialize SCTLR_EL3 without disturbing setting done above:
- Initialize SCR_EL3

- Initialize MDCR_EL3
- Initialize PMCR_EL0
- Enable External Aborts, Error Interrupts, IRQ and FIQ.
- Initialize CPTR_EL3
- Check If Data Independent Timing (DIT) functionality is implemented, always enable DIT in EL3
- Install Faults
- Setup the memory and stack
- Initialize system control register HCR_EL2 to zero
- Initialize SCTLR_EL2/EL1 to 0x30C50838
- Read the `RCU_STAT` register and store locally in register set
- Call the preboot function passing the previously read RCU_STAT contents as an argument

Core 1 Start-up

Describes the initial operations performed by the core 1 immediately after being released from reset and soft vector processing has completed.

Upon initial release from reset, the soft vectoring process results in core 1 executing from the boot ROM as long as the `RCU_SVECT1` register contained the default reset value during the soft vector processing. The operations the core performs varies depending on the product.

ADSP-SC598 Processor - Core 1 is not responsible for the booting of the processor from reset and thus enters a safe IDLE state allowing access to the cores L1 resources by any other core.

The core performs the following initial operations:

- Clear `RCU_MSG.C1IDLE`
- Check for OTPC boot completion.
- Disable Software Return feature in the BTB.
- Disable the BTB.
- Disable cache via `MODE2` register.
- Flush the cache.
- Set `IRPTL` to 0x00000000.
- Set `RCU_MSG.C1IDLE`
- Enter an endless IDLE loop

Table 47-2: Primary/Secondary DAG Configuration for C Run-Time Setup

Primary/Secondary DAG Register	Value	Description
M7, M15	-1	Dedicated registers must always be set to -1
M6, M14	1	Dedicated registers must always be set to 1
M5, M13	0	Dedicated registers must always be set to 0
L0-L5	0	Preserved registers set initially to 0
L6, L7	0	Stack Length Register set initially to 0
L8-L15	0	Preserved registers set initially to 0
B6, B7	0x201FF400	Stack Base Registers
I7	0x201FFFFC	Stack Pointer
L6, L7	0x000002FF	Stack Length Registers
I6	0x201FFFFC	Frame Pointer

Core 2 Start-up

Describes the initial operations performed by core 2 immediately after being released from reset and soft vector processing has completed.

Upon initial release from reset the soft vectoring process results in core 2 executing from the boot ROM as long as the `RCU_SVECT2` register contained the default reset value during the soft vector processing. The operations the core performs varies depending on the product.

In the *ADSP-SC598 Processor*, core 1 is not responsible for the booting of the processor from reset and thus enters a safe IDLE state allowing access to the cores L1 resources by any other core. Core 2 has no entry point into the main boot rom to execute the preboot process or controller boot process from reset of the device.

From reset, core 2 can perform the following operations:

- Clear `RCU_MSG.C2IDLE`
- Check for OTPC boot completion.
- Disable Software Return feature in the BTB.
- Disable the BTB.
- Disable cache via `MODE2` register.
- Flush the cache.
- Set `IRPTL` to 0x00000000.
- Set `RCU_MSG.C2IDLE`
- Enter an endless IDLE loop

Fault Configuration

Describes the initial fault sources that are enabled allowing the processor to signal a fault to the system.

The following faults are enabled via the SEC. Only the faults are enabled, the boot process does not install any SEC interrupts.

Table 47-3: Initial Faults installed during Preboot

SEC Fault ID	SEC Fault Name	Description
0	INTR_SEC0_ERR	SEC Error
3	INTR_WDOG0_EXP	WDOG Expire
4	INTR_WDOG1_EXP	WDOG Expire
6	INTR_OTPC0_ERR	OTPC Expire
10	INTR_L2CTL0_ECC_ERR	L2 ECC Error
76	INTR_SOFT3_INT	Software Driven Interrupt 3. This is raised in the boot ROM error handler should it be entered.
190	INTR_CRC0_ERR	CRC Error
191	INTR_CRC1_ERR	CRC Error
194	INTR_MDMA1_SRC_ERR	MDMA 1 Source DMA Channel Error
195	INTR_MDMA1_DST_ERR	MDMA 1 Destination DMA Channel Error

The SEC is configured for a fault delay of 0x100 via the SEC_FDLY.COUNT and SEC_FSRDLY.COUNT bit fields. This allows for a delay before the fault assertion is a custom error handler is installed and any SEC interrupts are enabled and handled for more advanced second stage boot scenarios.

The SYSFAULT pin is configured via the SEC to support both incoming and outgoing faults by enabling the SEC_FCTL.FIEN and SEC_FCTL.FOEN bits. The boot process captures an incoming fault if that fault is asserted by the external system on handover to the user application after boot.

NOTE: The installation of the fault sources is enabled by default and may be optionally bypassed using the RCU_BCODE.NOFAULTS bit.

NOTE: In the ADSP-SC598 processor, the SYSFAULT pin is dedicated. However the SYSFAULT signal also appears on the PC_07 in pin mux.

L2 Controller Configuration

The L2 controller is configured to ensure all L2 memory banks are ECC enabled and that the L2 memory scrub feature is disabled.

NOTE: The L2 Controller configuration is enabled by default and may be optionally bypassed by setting the RCU_BCODE.NOL2CONFIG bit.

L2 Memory Initialization

The L2 memory subsytem is ECC protected by default. The L2 memory must be initialized to ensure that no read from the L2 memory generates an ECC error. All L2 memory banks are initialized using the L2CTL_INIT bit. The boot process waits for each memory bank to signal initialization complete using the L2CTL_ISTAT bit. The RCU_MSG.L2INIT is set when L2 memory initialization is complete.

NOTE: The L2 memory initialization process is enabled by default and may be optionally bypassed by setting the RCU_BCODE.NOMEMINIT bit.

Idle On Entry

The Idle On Entry feature instructs the debugger to halt the boot code before continuing with any further preboot operations.

When this feature is enabled the processor executes a WFI/IDLE instruction and then continues once an event (such as an emulator exception) is serviced.

NOTE: Idle On Entry processing is disabled by default and is enabled by setting the RCU_BCODE.IDLEONENTRY bit prior to performing a system reset operation.

SPU Configuration

The SPU is configured differently depending upon the detected security state of the device. The first operation clears any existing security violations that may be indicated in the SPU_STAT register.

None of the peripherals are configured to ignore security signals and only the following controllers are configured to generate secure transactions.

Table 47-4: SPU Secure Controllers During Boot

SPU Endpoint ID	Controller Name
147	MDMA0 Source DMA Channel
148	MDMA0 Destination DMA Channel
149	MDMA1 Source DMA Channel
150	MDMA1 Destination DMA channel
145	CRC0
146	CRC1
178	PKTE

SMPU Configuration

The SMPU is used to restrict access to various memory regions in the processor. The configuration applied during boot differs depending on the locked state of the processor.

By default the SMPU instances only allow secure read and write transactions. The ***SMPU Configuration*** table describes the configurations for the different security states.

Speculative reads are also disabled by setting the SMPU_CTL.RSDIS bit.

Table 47-5: SMPU Configuration

SMPU Instance	SMPU Instance	Open SMPU_SECURECTL Value	Locked SMPU_SECURECTL Value
SMC	0	SMPU_SECURECTL.WNSEN | SMPU_SECURECTL.RNSEN	0x00000000 (Default Reset Value)
Core_L2_RAM_Boot_ROM0	2	SMPU_SECURECTL.WNSEN | SMPU_SECURECTL.RNSEN	0x00000000 (Default Reset Value)
DMA_L2_RAM_Boot_ROM0	3	SMPU_SECURECTL.WNSEN | SMPU_SECURECTL.RNSEN	0x00000000 (Default Reset Value)
DMC	9	SMPU_SECURECTL.WNSEN | SMPU_SECURECTL.RNSEN	0x00000000 (Default Reset Value)
SPI	11	SMPU_SECURECTL.WNSEN | SMPU_SECURECTL.RNSEN	0x00000000 (Default Reset Value)
Core_L2_ROM1_Boot_ROM1	4	SMPU_SECURECTL.WNSEN | SMPU_SECURECTL.RNSEN	0x00000000 (Default Reset Value)
Core_L2_ROM2_Boot_ROM2	6	SMPU_SECURECTL.WNSEN | SMPU_SECURECTL.RNSEN	0x00000000 (Default Reset Value)
DMA_L2_ROM1_Boot_ROM1	5	SMPU_SECURECTL.WNSEN | SMPU_SECURECTL.RNSEN	0x00000000 (Default Reset Value)

Secure Debug Key Processing

In the event the processor is locked, the debug tools must submit a secure debug key that is matched with a key on the processor. A 128-bit Secure Debug Key must be provided the application program prior to locking the device.

The secure debug key is read from the OTP memory and then written to the corresponding register in the TAPC. After the key has been written the TAPC_SDBGKEY_CTL.VALID bit is set. Once the debug tools then submit their key a key compare operation is performed.

Debug tools must wait for the boot software to load the key before setting the TAPC_SDBGKEY_CTL.VALID bit before submitting the key for comparison.

The 128-bit Secure Debug Key is loaded as follows from the storage area in OTP.

Table 47-6: Secure Debug Key Load Procedure

Secure Debug Key[127:0]	Register
Secure Debug Key[31:0]	TAPC_SDBGKEY0
Secure Debug Key[63:32]	TAPC_SDBGKEY1
Secure Debug Key[95:94]	TAPC_SDBGKEY2
Secure Debug Key[127:96]	TAPC_SDBGKEY3

NOTE: In the ADSP-SC59x processor family, the boot ROM supports two 128-bit secure debug keys, `secure_emu_key0` and `secure_emu_key1`, where only one of them is used by the boot ROM at a time. Each of the secure debug keys is associated with a bypass key (`emu_key0_disable` or `emu_key1_disable`) which decides the validity of the corresponding key. Any non-zero value written into the 16-bit bypass key invalidates that particular secure debug key.

The validity of the secure debug keys can be determined as shown in the ***Secure Debug Key Validation*** table:

Table 47-7: Secure Debug Key Validation

Bypass Key 0 (emu_key0_disable) Value	Bypass Key 1 (emu_key1_disable) Value	Valid Secure Debug Key
Zero	Zero	secure_emu_key0
Zero	Non-zero	secure_emu_key0
Non-zero	Zero	secure_emu_key1
Non-zero	Non-zero	Both keys are valid

CAUTION: The boot code bypasses the key load operation entirely when a 16-bit non-zero value is programmed on both `emu_key0_disable` and `emu_key1_disable` fields in OTP *or* a key of 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF is programmed in both secure debug key fields provided in theOTP. When debug access is subsequently needed, the program must load the key to the TAPC. When the processor fails to boot (due to corrupted firmware) then there is no debug access. The only way to gain access is to load an authenticated boot image that can then load the required keys prior to attempting to connect with a debugger.

CGU Configuration

This step in the preboot process reconfigures the internal clocks on the processor for improved boot performance.

The boot process can optionally configure the CGU in order to improve boot performance. The settings for the CGU are located within the `struct ADI_ROM_OTP_BOOT_CGU_INFO` structure that has storage allocated in the OTP as part of the struct ADI_ROM_OTP_BOOT_INFO structure.

Typically, CGU configuration is performed using an Init Block in the boot stream. This provides greatest flexibility. In situations where boot time must be kept to a minimum, settings in the OTP can be applied at this stage of preboot as opposed to during the boot process itself. When a processor is locked, the boot process does not support an Init Block in the boot stream. For a locked processor programs must use the OTP to reconfigure the clocks without adopting a multi-stage boot strategy.

On release from reset, the CGU is configured for PLL Bypass mode. To improve boot performance, the boot software reconfigures the CGU so that Full-On mode is entered with the default CGU settings.

If the program doesn't provide settings in OTP to configure the Oscillator Watchdog, then the Oscillator Watchdog fault, enabled by default after reset, is disabled prior to reconfiguring the CGU.

If settings are provided in the OTP to configure the Oscillator Watchdog, the fault is left enabled and, if applicable, the CGU is configured per the provided settings. The remainder of the boot process completes in Full-On mode.

The CGU configuration object (see struct ADI_ROM_OTP_BOOT_CGU_INFO) allows the boot software to configure the CGU for a more efficient boot process.

Table 47-8: ADI_ROM_OTP_BOOT_CGU_INFO Members

Type	Name	Description
uint32_t	ctl_WEN:1 (bitfield)	Enable write to the CGU_CTL register
uint32_t	div_WEN:1 (bitfield)	Enable write to the CGU_DIV register
uint32_t	reserved0:1 (bitfield)	Reserved
uint32_t	div_DSEL:5 (bitfield)	CGU_DIV.DSEL value
uint32_t	div_CSEL:5 (bitfield)	CGU_DIV.CSEL value
uint32_t	div_S0SEL:3 (bitfield)	CGU_DIV.S0SEL value
uint32_t	div_SYSSEL:5 (bitfield)	CGU_DIV.SYSSEL value
uint32_t	div_S1SEL:3 (bitfield)	CGU_DIV.S1SEL value
uint32_t	div_OSEL:7 (bitfield)	CGU_DIV.OSEL value
uint32_t	ctl_DF:1 (bitfield)	CGU_CTL.DF value
uint32_t	ctl_MSEL:7 (bitfield)	CGU_CTL.MSEL value
uint32_t	reserved1:1 (bitfield)	Reserved
uint32_t	ctl_S1SELEXEN:1 (bitfield)	CGU_CTL.S1SELEXEN value
uint32_t	auto_disable:1 (bitfield)	Disable auto-alignment
uint32_t	clkoutsel_CLKOUTSEL:5 (bitfield)	CGU_CLKOUTSEL.CLKOUTSEL value
uint32_t	clkoutsel_WEN:1 (bitfield)	Enable write to the CGU_CLKOUTSEL register
uint32_t	reserved2:8 (bitfield)	Reserved
uint32_t	divex_S1SELEX:8 (bitfield)	CGU_DIVEX.S1SELEX value
uint32_t	oscwctl0_WEN:1 (bitfield)	Enable write to the CGU_OSCWDCTL instance 0 register
uint32_t	oscwctl0_HODF:6 (bitfield)	CGU_OSCWDCTL.HODF value
uint32_t	oscwctl0_HODEN:1 (bitfield)	CGU_OSCWDCTL.HODEN value
uint32_t	oscwctl0_CNGEN:1 (bitfield)	CGU_OSCWDCTL.CNGEN value
uint32_t	oscwctl0_BOUF:5 (bitfield)	CGU_OSCWDCTL.BOUF value
uint32_t	oscwctl0_BOUEN:1 (bitfield)	CGU_OSCWDCTL.BOUEN value
uint32_t	oscwctl0_FAULTEN:1 (bitfield)	CGU_OSCWDCTL.FAULTEN value
uint32_t	oscwctl0_MONDIS:1 (bitfield)	CGU_OSCWDCTL.MONDIS value
uint32_t	oscwctl0_FAULTPINDIS:1 (bitfield)	CGU_OSCWDCTL.FAULTPINDIS value
uint32_t	reserved3:14 (bitfield)	Reserved
uint32_t	oscwctl1_WEN:1 (bitfield)	Enable write to the CGU_OSCWDCTL instance 1 register

Table 47-8: ADI_ROM_OTP_BOOT_CGU_INFO Members (Continued)

Type	Name	Description
uint32_t	oscwctl1_HODF:6 (bitfield)	CGU_OSCWDCTL.HODF value
uint32_t	oscwctl1_HODEN:1 (bitfield)	CGU_OSCWDCTL.HODEN value
uint32_t	oscwctl1_CNGEN:1 (bitfield)	CGU_OSCWDCTL.CNGEN value
uint32_t	oscwctl1_BOUF:5 (bitfield)	CGU_OSCWDCTL.BOUF value
uint32_t	oscwctl1_BOUEN:1 (bitfield)	CGU_OSCWDCTL.BOUEN value
uint32_t	oscwctl1_FAULTEN:1 (bitfield)	CGU_OSCWDCTL.FAULTEN value
uint32_t	oscwctl1_MONDIS:1 (bitfield)	CGU_OSCWDCTL.MONDIS value
uint32_t	oscwctl1_FAULTPINDIS:1 (bitfield)	CGU_OSCWDCTL.FAULTPINDIS value
uint32_t	Reserved4:14 (bitfield)	Reserved

If a CGU_STAT.WDIVERR, CGU_STAT.WDFMSERR, CGU_STAT.LWERR or CGU_STAT.ADDRERR occurs at the entry to or completion of the configuration routine, the default error handler is called and the boot process terminates.

NOTE: Programs can bypass CGU configuration by setting the RCU_BCODE.NOPREBOOT bit when this part of the boot process is reached.

Releasing All Cores from Reset

The control booting core releases all other cores from reset allowing them to then run by default into a safe endless loop. By releasing all other cores from reset, any dedicated core L1 memories become accessible to all cores via the system address space. For one core to load or read data from dedicated L1 memory of another core, the core must be released from the reset state. By default, the other cores in the processor execute a safe endless loop in the boot ROM.

L1 Memory Initialization

The processor initializes all parity and ECC protected memories to perform subsequent read operations without generating an ECC or parity error.

The ***L1 Memory Initialization*** table describes the methods used to initialize the various parity and ECC supported memories on the processor.

Table 47-9: L1 Memory Initialization

Resource To Fill Memory	Memory Type	Address	Count	Fill Value	Flag Set Upon Completion
Core 1	Core 1 L1 Bank 0	0x00048000	0x6000 (LW)	0x00000000	RCU_MSG.C1L1INIT
Core 1	Core 1 L1 Bank 1	0x00058000	0x6000 (LW)	0x00000000	
Core 1	Core 1 L1 Bank 2	0x00060000	0x4000 (LW)	0x00000000	
Core 1	Core 1 L1 Bank 3	0x00070000	0x4000 (LW)	0x00000000	
Core 2	Core 2 L1 Bank 0	0x00048000	0x6000 (LW)	0x00000000	RCU_MSG.C2L1INIT
Core 2	Core 2 L1 Bank 1	0x00058000	0x6000 (LW)	0x00000000	
Core 2	Core 2 L1 Bank 2	0x00060000	0x4000 (LW)	0x00000000	
Core 2	Core 2 L1 Bank 3	0x00070000	0x4000 (LW)	0x00000000	

NOTE: The Memory Initialization process is enabled by default and may be optionally bypassed by setting the RCU_BCODE.NOMEMINIT bit.

Default Entry Point

The first instruction executed for the core in the boot ROM is a read of the RCU_SVECT0 register. The boot ROM then vectors to that location.

The ***Default Entry Point*** table defines the default entry point for each core in the processor

Table 47-10: Default Entry Point

Core ID	Register	Entry Point
0	RCU_SVECT0	0x00000000
1	RCU_SVECT1	0x00500004
2	RCU_SVECT2	0x00500004

The boot code does not set any initial default application entry points by writing the entry point to the RCU_SVECT0 register in the processor. It will hold the default values of the power on reset case. The RCU_SVECT0 register is updated at the end of the boot process. For the secure boot, the register is updated only after the successful authentication of the boot stream.

NO-BOOT Processing

No-Boot mode is executed when selected by the SYS_BMODE[n] pins. The boot mode is intended as a recovery boot mode or for debug purposes. The core simply executes in an endless loop in the boot ROM, terminating further execution of the boot process.

This boot mode is primarily intended for debug sessions when no boot source can be configured. It allows for a debugger to safely connect to the device and take control (assuming the debugger has been granted access rights as defined by the processor security implementation).

NOTE: NO-BOOT processing is usually only entered because of the boot mode pin sampling that results in execution of the No-Boot boot mode. This processing can also be optionally enabled by setting `RCU_BCODE.HALT` field. The `RCU_BCODE.HALT` setting can be especially useful for debug sessions to force the execution of the NO-BOOT mode regardless of the `SYS_BMODE[n]` state allowing a user application to be loadedusing the debug tools without fear of the image being corrupted because of attempting to boot through another source.

SYS_RESOUT Signal

To signal to tan external system that the processor is in a configured state and ready to start the boot process, the boot software de-asserts the `SYS_RESOUT` pinusing the `RCU_CTL.RSTOUTDSRT` bit.

DMC Configuration

To boot to external DDR memories (to support booting to external memory when the processor is locked) the DMC must be configured.

Typically, DMC configuration is done using the Init Block in the boot stream. When the processor is locked the boot process does not support an Init Block in the boot stream. For a locked processor, programs must use the OTP to configure the DMC without adopting a multi-stage boot strategy.

Table 47-68 ADI_ROM_OTP_DMC_CONFIG Members provides details of the OTP region that is used to store the DMC configuration. In the table, the `ADI_ROM_OTP_BOOT_CFG::dmcEn` must be set for the settings to apply.

Table 47-11: ADI_ROM_OTP_DMC_CONFIG Members

Type	Name	Description
uint32_t	ulDDR_DLLCTLCFG:16 (bit field)	Content of DMC CFG[15:0]
uint32_t	ulDDR_DLLCTLCFG:16 (bit field)	Content of DMC DLL CTL[15:0]
uint32_t	ulDDR_EMR2EMR3:16 (bit field)	Contents of DMC_EMR3 [15:0]
uint32_t	ulDDR_EMR2EMR3:16 (bit field)	Contents of DMC_EMR2 [15:0]
uint32_t	ulDDR_CTL	Content of DMC CTL[31:0]
uint32_t	ulDDR_MREMR1:16 (bit field)	Content of DMC EMR1[15:0]
uint32_t	ulDDR_MREMR1:16 (bit field)	Content of DMC MR[15:0]
uint32_t	ulDDR_TR0	Content of DMC_TR0[31:0]
uint32_t	ulDDR_TR1	Content of DMC_TR1[31:0]
uint32_t	ulDDR_TR2	Content of DMC_TR2[31:0]
uint32_t	ulDDR_ZQCTL0	Content of DMC PHY ZQCTL0[31:0]
uint32_t	ulDDR_ZQCTL1	Content of DMC PHY ZQCTL1[31:0]
uint32_t	ulDDR_ZQCTL2	Content of DMC PHY ZQCTL2[31:0]

Table 47-11: ADI_ROM_OTP_DMC_CONFIG Members (Continued)

Type	Name	Description
uint32_t	ulDDRPHY_CACTL	Content of DMC PHY CA_CTL [31:0]
uint32_t	uBypassDelay_LANE0CTL1:6 (bit field)	Content of DMC LANE0_CTL1[15:10]
uint32_t	uBypassDelay_LANE1CTL1:6 (bit field)	Content of DMC LANE1_CTL1[15:10]
uint32_t	uBypassDelay_LANE0CTL0:6 (bit field)	Content of DMC LANE0_CTL0[15:10]
uint32_t	uBypassDelay_LANE1CTL0:6 (bit field)	Content of DMC LANE1_CTL0[15:10]
uint32_t	reserved0:8 (bit field)	Reserved

There is an additional single bit located in OTP, `ADI_ROM_OTP_BOOT_CFG::dmcInv` allowing programs to invalidate the DMC settings stored in OTP.

NOTE: Once `ADI_ROM_OTP_BOOT_CFG::dmcInv` has been set in OTP there is no means to configure the DMC during the preboot phase.

The configuration of the DMC is bypassed if the `RCU_BCODE.NOPREBOOT` bit is set when this part of the boot process is reached.

Bypassing the Boot Process

The boot process can be bypassed allowing a program to start execution from the address stored in the core's soft vector register. This is useful when working in emulation sessions as it provides a mechanism to be able to execute programs directly from an accessible memory that already contains executable code.

To bypass the boot process, set the `RCU_BCODE.NOKERNEL` bit. The processor core vectors to the address stored in the core's corresponding `RCU_SVECT0` / `RCU_SVECT1`/ `RCU_SVECT2` register instead of calling the required boot mode.

This feature is used along with other features such as disabling memory initialization.

Boot Mode Disable

Specific boot modes can be permanently disabled via OTP. If the disabled boot mode is enabled using the `SYS_BMODE[n]` pins, a boot error is generated.

A byte of storage is provided in the OTP to disable up to eight boot modes. The boot mode disable field can be programmed using the `adi_rom_otp_pgm()` routine. The `otp_data::bootModeDisable` member is used in the program operation to disable the various boot modes.

Table 47-12: Boot Mode Disable

`otp_data::bootModeDisable` Bit Position	Corresponding Boot Mode
0	SPI Controller Boot Mode

Table 47-12: Boot Mode Disable (Continued)

otp_data::bootModeDisable Bit Position	Corresponding Boot Mode
1	SPI Target Boot Mode
2	UART Target Boot Mode
3	Linkport Target Boot Mode
4	OSPI Controller Boot Mode
5	eMMC Controller Boot Mode
7–6	Reserved

Boot Command Customization

Boot command customization allows permanent customization of a particular boot mode. For example, it is possible to change the peripheral instance used for boot operation.

Storage is provided in OTP for a command item for each supported boot mode.

Storage is provided in the `struct ADI_ROM_OTP_BOOT_CMD_INFO` member of `struct ADI_ROM_OTP_BOOT_INFO`. Refer to the corresponding boot modes boot command description for details on supported command options.

NOTE: Before programming boot command to the OTP, evaluate the boot command using the adi_rom_Boot() API and ensure that the boot command provides the desired functionality. Once the command is programmed to OTP, it cannot be reverted to original default settings.

Boot Mode Specific SPU Configuration

Prior to performing actual boot process, the processors SPU resources specific to the boot mode selected are configured. This is performed in the preboot phase as opposed to within the boot mode itself when calling the boot API, as it isolates the security functionality of the processor allowing it to be handled specifically by a separate process.

The following additional SPU resources are configured as secure controllers according to the boot mode selected.

Table 47-13: Boot Mode Specific SPU Configuration

Boot Mode	SPU Endpoint ID	Controller Name
SPI Controller Boot (Memory-Mapped Mode) and OSPI Controller Boot	147, 148	MDMA0 Source DMA Channel, MDMA0 Destination DMA Channel
SPI Controller Boot (Peripheral Mode)	100, 98, 96, 94	SPI3 Receive DMA, SPI2 Receive DMA, SPI1 Receive DMA, SPI0 Receive DMA
eMSI Controller Boot	175	EMSI0
SPI Target Boot	100, 98, 96, 94	SPI3 Receive DMA, SPI2 Receive DMA, SPI1 Receive DMA, SPI0 Receive DMA

Table 47-13: Boot Mode Specific SPU Configuration (Continued)

Boot Mode	SPU Endpoint ID	Controller Name
UART Target Boot	79, 81, 83, 85	UART 0 Receive DMA, UART1 Receive DMA, UART2 Receive DMA, UART3 Receive DMA
LINKPORT Target Boot	5, 6	LINKPORT 0 DMA, LINKPORT 1 DMA

NOTE: In the ***Boot Mode Specific SPU Configuration*** table, for a given boot mode, all the SPU resources are not configured. Only a single peripheral instance is enabled according the peripheral instance selected for boot. For example if the boot command for the boot mode indicates boot from UART0 only the UART0 Receive DMA is configured, the other UART Receive DMAs are not configured for secure access.

Executing the Boot Mode

The boot mode is called adi_rom_Boot() routine. The routine fetches and processes the boot stream from the configured boot source.

The table provides default parameters passed to each of the supported boot modes. For details on the API usage, refer to adi_rom_Boot().

Table 47-14: Default Boot ROM API Parameters

Boot Mode	pAddress	flags	blockCount	pHook	Command
No Boot	0x00000000	0x00000000	0x00000000	Points to an empty routine in ROM	0x00000000
SPI Controller Boot	0x60000000	0x00040000	0x00000000	Points to an empty routine in ROM	0x00000207
SPI Target Boot	0x00000000	0x00000000	0x00000000	Points to an empty routine in ROM	0x00000212
UART Target Boot	0x00000000	0x00000000	0x00000000	Points to an empty routine in ROM	0x00000013
LINKPORT Target Boot	0x00000000	0x00000000	0x00000000	Points to an empty routine in ROM	0x00000014
OSPI Controller Boot	0x60000000	0x00000000	0x00000000	Points to an empty routine in ROM	0x00000008
eMMC Controller Boot	0x00000023	0x00000000	0x00000000	Points to an empty routine in ROM	0x00082009
Reserved	0x00000000	0x00000000	0x00000000	Points to an empty routine in ROM	0x00000000

The hook function installed using pHook on this product does not perform any additional configuration.

NOTE: For SPI controller boot and OSPI controller boot modes, start address (pAddress) is set to 0x60000000, This corresponds to the first memory location in the respective flash device (0x0). Similarly, the eMMC

controller boot start address (pAddress) is set to 0x00000023, which corresponds to the default 34th bock of memory location in the eMMC device.

Boot Modes

The boot implementation provides built-in support for booting from various peripherals.

The ***Booting Modes*** table describes the supported boot modes.

In target boot mode, the processor functions as a target to any host device. In these modes, the host device usually controls the processor SYS_HWRST input. Typically, the host applies the reset sequence and waits until the processor is ready to boot, depending on the peripheral in use, and transmits the boot stream data to the processor. Handshake signals are used to signal to the host that the processor is ready to accept more data.

In controller boot mode, the processor controls the peripheral and requests data via the peripheral as and when required.

Individual boot modes can be disabled. For more information about disabling boot modes, see Boot ROM OTP Customizations.

Table 47-15: Booting Modes

SYS_BMODE[2:0]	Boot Source	Description
000	No Boot	The processor does not boot. Rather the boot kernel executes some of the preboot operations then enters an endless `WFI/IDLE` state.
001	SPI Controller Boot	Boot from integrated Flash memory through the SPI2 peripheral configured for memory-mapped mode.
010	SPI Target Boot	Boot through the SPI2 peripheral configured as a target.
011	UART Boot	Boots through UART0 configured as a target receiver.
100	LINKPORT Boot	Boot through LINKPORT0 peripheral configured as a target receiver. This boot mode is only applicable to derivatives supporting the LP0 instance.
101	OSPI Controller Boot	Boot from integrated Flash memory through the OSPI peripheral configured for memory-mapped mode.
110	eMMC Controller Boot	Boot from eMMC card through eMSI peripheral.
111	Reserved	

No-Boot Mode

No-Boot mode is intended for device recovery purposes caused due to incorrect programming of the boot source memory allowing for target connection through an emulator. Emulation tools can also leverage the No-Boot functionality allowing for debug sessions to run the preboot software prior to loading an application while preventing the boot process from continuing and clobbering data loaded by the emulator.

This boot mode results in several preboot operations being performed before placing the core in a safe, endless loop located in the boot ROM.

For a complete list of operations performed when No-Boot is selected, see Preboot Operations. The core terminates at the NO-BOOT Processing stage of the preboot process.

SPI Master Boot Mode

The SPI master boot routine provides support for booting the processor from SPI flash memories. The SPI boot mode uses a device auto-detection feature that is enabled by default. This lets the boot stream itself instruct updates to the SPI configuration and the read command used allowing for more efficient transactions.

NOTE: For default SPI master mode, the peripheral fault, SPI2_ERR with fault ID 132 is enabled.

Boot From External SPI Flash Devices

The SPI boot mode supports booting from 24-bit or 32-bit addressable flash devices. The boot mode uses the MDMA channels by default and configures the SPI flash for memory-mapped functionality. Peripheral DMA mode is also supported when calling the boot mode via `adi_rom_Boot()`.

When auto-device detection is enabled, the SPI memory is initially read using the standard 0x03 SPI read command with a reduced clocking frequency for maximum compatibility. The first nibble of the boot stream is then used to reconfigure the SPI interface and possible the SPI flash. Refer to SPI Device Detection Routine.

NOTE: Support for automatic device detection via the first nibble of the boot stream is not supported when booting secure boot streams. Instead when signing the boot image an attribute can be set in the image header that specifies the configuration to use.

For booting, the SPI memory is connected as shown in the ***SPI Memory Connections*** figure.

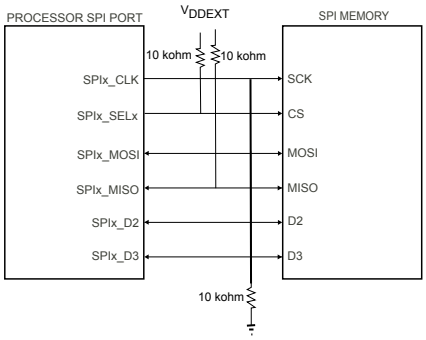


Figure 47-3: SPI Memory Connections

The pull-up resistor on the slave select signal ensures that the memory is deselected when the pin is in a high-impedance mode such as during reset.

Initialization codes are allowed to manipulate the `ADI_ROM_BOOT_CONFIG::dBootCommand` to extend boot mechanism to a second SPI memory connected to another slave select pin. Updating the field that specifies the slave select signal for use, allows the boot process to manage larger boot streams than fit into a single SPI device.

NOTE: If modifying the slave select signal used during the boot process, configure the pin multiplexing to enable the correct functionality for the pin. Once the boot process has proceeded past the configuration function

and the boot process has actually started, the boot kernel will not perform any further pin multiplexing operations.

SPI Device Detection Routine

Since the boot mode supports booting from various SPI memories, the boot kernel automatically detects what type of memory is connected. To determine whether the SPI memory device requires a 24-bit or 32-bit addressing scheme, the boot kernel performs a device detection sequence prior to booting. The `SPI_MISO` signal requires a pull-up resistor. The routine relies on the fact that memories do not drive their data outputs unless the right number of address bytes are received.

Initially, the boot kernel transmits the read command on the `SPI_MOSI` line. Once the command has been sent, the boot kernel proceeds to transmit a single address byte and waits until the receive FIFO indicates that the buffer is no longer empty. The first received byte is discarded. The boot code then proceeds to issue another address byte while simultaneously receiving a byte. The process continues until a non 0xFF or 0x00 byte is received or until the full 4 address bytes is sent without any valid data being returned.

The receiving of a non 0x00 or 0xFF byte tells the boot code whether the memory device requires 24 or 32 address bits. The lower nibble of the received byte is then used to further customize the boot mode. This nibble is referred to as the `BCODE`. The boot code applies settings to the SPI peripheral according to the ***SPI Master Boot BCODE Descriptions***.

If the received value equals 0x00 or 0xFF, it is assumed that the memory device has not driven its data output thus, another zero byte is transmitted and the received data is tested again.

If the value still equals 0xFF, device detection continues. Device detection aborts immediately when a byte different than 0xFF is received. The boot process continues with normal boot operation and it reissues a command to again read from address 0. Two read sequences load the first block header. Separate read sequences load further block headers and block payload fields.

The ***SPI Device Detection Principle*** figure illustrates how individual devices behave.

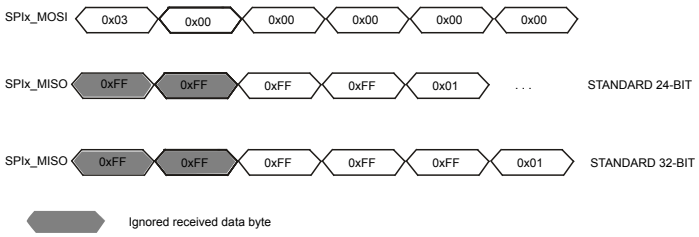


Figure 47-4: SPI Device Detection Principle

Table 47-16: SPI Master BCODE Configuration Lookup Table

Member	BCODE				
	0	1	2	3	4
Transfer Type	Single bit command, address and data	Single bit command, address and data	Single bit command, address and data	Single bit command, address and dual bit data	Single bit command, address and Quad data
ubDummyBytes	0x00	0x00	0x01	0x01	0x01
ubReadCommand	0x03	0x03	0x0B	0x3B	0x6B
ubDataBits	0x00	0x00	0x00	0x01	0x02
ubAddressBytes	0x03	0x03	0x03	0x03	0x03
uwClkLower	0x000F	0x000F	0x0001	0x0001	0x0001
uReserved0	0x0000	0x0000	0x0000	0x0000	0x0000
nTxCtl	0x00000003	0x00000003	0x00040003	0x00040033	0x00040033
nRxCtl	0x00000003	0x00000003	0x00040003	0x00140003	0x00240033
nCmdCtl	0x00000003	0x00000003	0x00000003	0x00000033	0x00000033
pMIOEnFunction	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
nDummy	0x00	0x00	0x00	0x00	0x00

Table 47-17: SPI Master BCODE Configuration Lookup Table

Member	BCODE					
	5	6	7	8	9	A
Transfer Type	Single bit command, Dual bit address and data	Single bit command , Dual bit address and data	Single bit command , Dual bit address and data	Single bit command , Quad bit address and data	Single bit command , Quad bit address and data	Single bit command , Quad bit address and data
ubDummyBytes	0x01	0x02	0x03	0x02	0x03	0x05
ubReadCommand	0xBB	0xBB	0xBB	0xEB	0xEB	0xEB
ubDataBits	0x01	0x01	0x01	0x02	0x02	0x02
ubAddressBytes	0x03	0x03	0x03	0x03	0x03	0x03
uwClkLower	0x0001	0x0001	0x0001	0x0001	0x0001	0x0001
uReserved0	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
nTxCtl	0x00140033	0x00140033	0x00140033	0x00240033	0x00240033	0x00240033
nRxCtl	0x00140033	0x00140033	0x00140033	0x00240033	0x00240033	0x00240033
nCmdCtl	0x00000033	0x00000033	0x00000033	0x00000033	0x00000033	0x00000033
pMIOEnFunction	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Table 47-17: SPI Master BCODE Configuration Lookup Table (Continued)

Member	BCODE					
	5	6	7	8	9	A
nDummy	0x00	0x00	0x00	0x00	0x00	0x00

NOTE: For the above configurations, the addressing scheme can be 3-bytes or 4-bytes depending on the addressing of flash detected in auto-detection. The SPI mode byte issued for all the SPI master peripheral based configurations is 0x00. The mode byte is the first byte transmitted after the address cycles and is used to control the continuous read mode functionality in which the next read operation is not required to issue a command cycle. Continuous read mode is not supported during the boot process.

Supported Quad Mode Enable Methods

The boot ROM does not support enabling quad mode on the SPI flash device. To boot in quad mode, the flash device must be configured outside the boot ROM.

NOTE: It is more beneficial to boot initially in Dual mode and use an initcode to enable Quad mode.

Run-Time API

The following table provides descriptions of the adi_rom_Boot() command parameter.

Table 47-18: SPI Master Boot Command Descriptions

Bits	Name	Setting	Description
31:28	ROM_BCMD_SPIM_SPEED	0 to 1111	SPI clock divider. This value written to the SPI peripherals clock divider register
27	Reserved	Reserved	Reserved
26:22	ROM_BCMD_SPIM_DUMMY	00000	No dummy bytes required after the address
		00001	1 dummy byte required after the address
		....	.....
		11111	31 dummy bytes required after the address
21:20	ROM_BCMD_SPIM_ADDR	00	Flash device requires an 8-bit address
		01	Flash device requires a 16-bit address
		10	Flash device requires a 24-bit address
		11	Flash device requires a 32-bit address
19:16	ROM_BCMD_SPIM_BCODE	0000 to 1111	Boot mode-specific code. Specifies the boot mode-specific code that can further customize and control the boot process.

Table 47-18: SPI Master Boot Command Descriptions (Continued)

Bits	Name	Setting	Description
15	ROM_BCMD_SPIM_CMD	0	Issue the read command over the single bit bus
		1	Issue the read command over the multi-bit bus. Not recommended for use in ADSP-2159x processor.
14:12	ROM_BCMD_SPIM_SSEL	000	Use slave select 1 for SPI chip select
		001	Use slave select 2 for SPI chip select
		010	Use slave select 3 for SPI chip select
		011	Reserved
		100	Reserved
		101	Reserved
		110	Reserved
		111	Reserved
11:8	ROM_BCMD_SPIM_DEVENUM	0 to 16	Boot peripheral enumeration. So, for SPI3 it is set to 3, for SPI2 it is set to 2, for SPI0 it is set to 0. 0x4 to 0xF is reserved.
7	Reserved	Reserved	Reserved
6	ROM_BCMD_SPIM_NOAUTO	0	Automatic device detection disable. Perform automatic device detection and peripheral configuration based on the BCODE value (first nibble) of the boot streams block header
		1	Do not perform automatic device detection
5	ROM_BCMD_SPIM_NOCFG	0	Device configuration enable. Instructs the config routine to perform a pin-muxing and full peripheral configuration
		1	Device configuration disable.
4	ROM_BCMD_SPIM_HOST	0	Master boot mode enable.
		1	Slave boot mode enable.
3:0	ROM_BCMD_SPIM_DEVICE	0000	Reserved
		0001	Reserved
		0010	SPI Boot (Legacy peripheral DMA)
		0011	Reserved
		0100	Reserved
		0101	Reserved
		0110	Reserved
		0111	SPI Memory Mapped Boot

Table 47-18: SPI Master Boot Command Descriptions (Continued)

Bits	Name	Setting	Description
		1000 to 1111	Reserved

NOTE: All bits in the above table that are not defined must be set to zero. Supported features may be limited depending on peripheral instance.

NOTE: SPI master secure boot is only supported in memory mapped mode. SPI2 is the default SPI instance that operates in memory mapped mode. To boot from an SPI instance other than SPI2, ensure that the desired SPI instance supports memory-mapped mode. The ROM_BCMD_SPIM_DEVICE field in the SPI master boot command parameter must be set to 0x7 in OTP or passed as an argument in the ROM API call.

SPI Slave Boot Mode

When using SPI slave mode boot, the processor consumes boot data from an external SPI host device. This mode supports single, dual, and quad-bit modes. The boot kernel always starts in single bit mode and can be changed using the appropriate command. The following figures show the hardware configuration for the modes. As in all slave boot modes, the host device controls the `SYS_HWRST` input of the processor.

NOTE: *Secure Boot Stream Padding*

For slave boot modes, the host must always send data in multiples of 1024 bytes. This requirement is due to the sizing of internal buffers used for DMA.

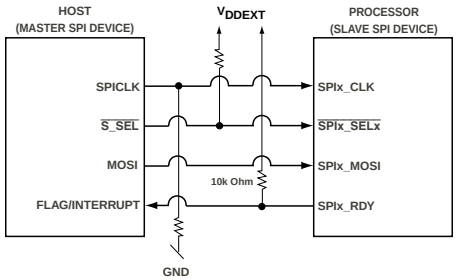


Figure 47-5: Connection between Host (SPI Master) and Processor (SPI Slave)

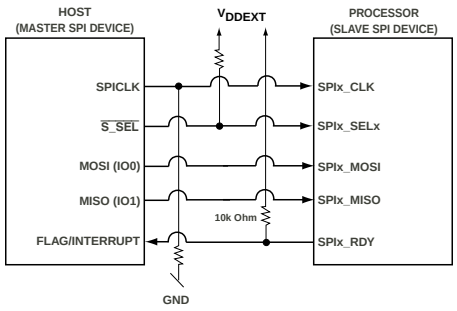


Figure 47-6: Connection between Host (SPI Master) and Processor (SPI Slave) DIOM

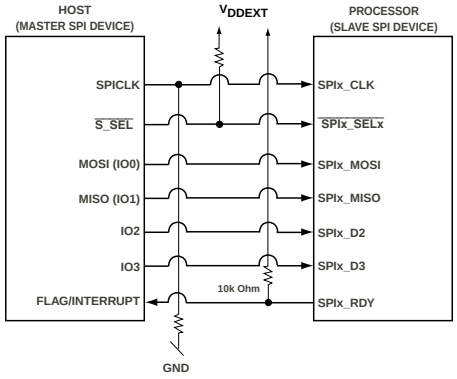


Figure 47-7: Connection between Host (SPI Master) and Processor (SPI Slave) QSPI

The host drives the SPI clock and is responsible for timing. The host must provide an active-low chip select signal that connects to the processor's `SPIx_SS` input signal with each byte transferred or remain low during the entire procedure. 8-bit data is expected and 16-bit mode is not supported.

In SPI slave boot mode, the boot kernel sets the `SPI_CTL.CPHA` bit and clears the `SPI_CTL.CPOL` bit. Therefore the `SPI_MISO` pin is latched on the falling edge of the `SPI_MOSI` pin.

The SPI slave processor detects the correct boot mode from the host SPI device by reading the first byte sent, defined as `SPICMD`. The ***SPICMD Descriptions*** table describes the available codes. These additional bytes must be sent prior to transmitting the data to configure the SPI device.

The ***SPICMD Descriptions*** table describes the following:

- Host starting in single bit mode
- Host starting in a mode other than single bit

Table 47-19: SPICMD Descriptions

SPICMD	Description
If host starts in Single bit Mode	
0x3	Keep single-bit mode
0x7	Switch to dual-bit mode
0xB	Switch to quad-bit mode
If host device starts in DIOM or QSPI	
0xAA,0xBF	Switch to dual-bit mode
0xEE,0xEE,0xFE,0xFF	Switch to quad-bit mode

In SPI slave boot mode, `SPIx_RDY` functionality is critical. The `SPIx_RDY` output is used for back pressure and requires a pulling resistor. The boot code requires the `SPIx_RDY` signal function as active-low. The host is only permitted to transfer data when `SPIx_RDY` is in the active state. This functionality allows the processor to hold off the host while the processor is in reset or executing the pre-boot and processor initialization sequences. The SPI is configured to deassert `SPIx_RDY` when the receive FIFO is filled to 75% or more.

NOTE: For default SPI slave mode, the peripheral fault, SPI2_ERR with Fault ID 132 is enabled.

The ***SPI Program Flow on the Host Side*** figure illustrates the required program flow on the host side.

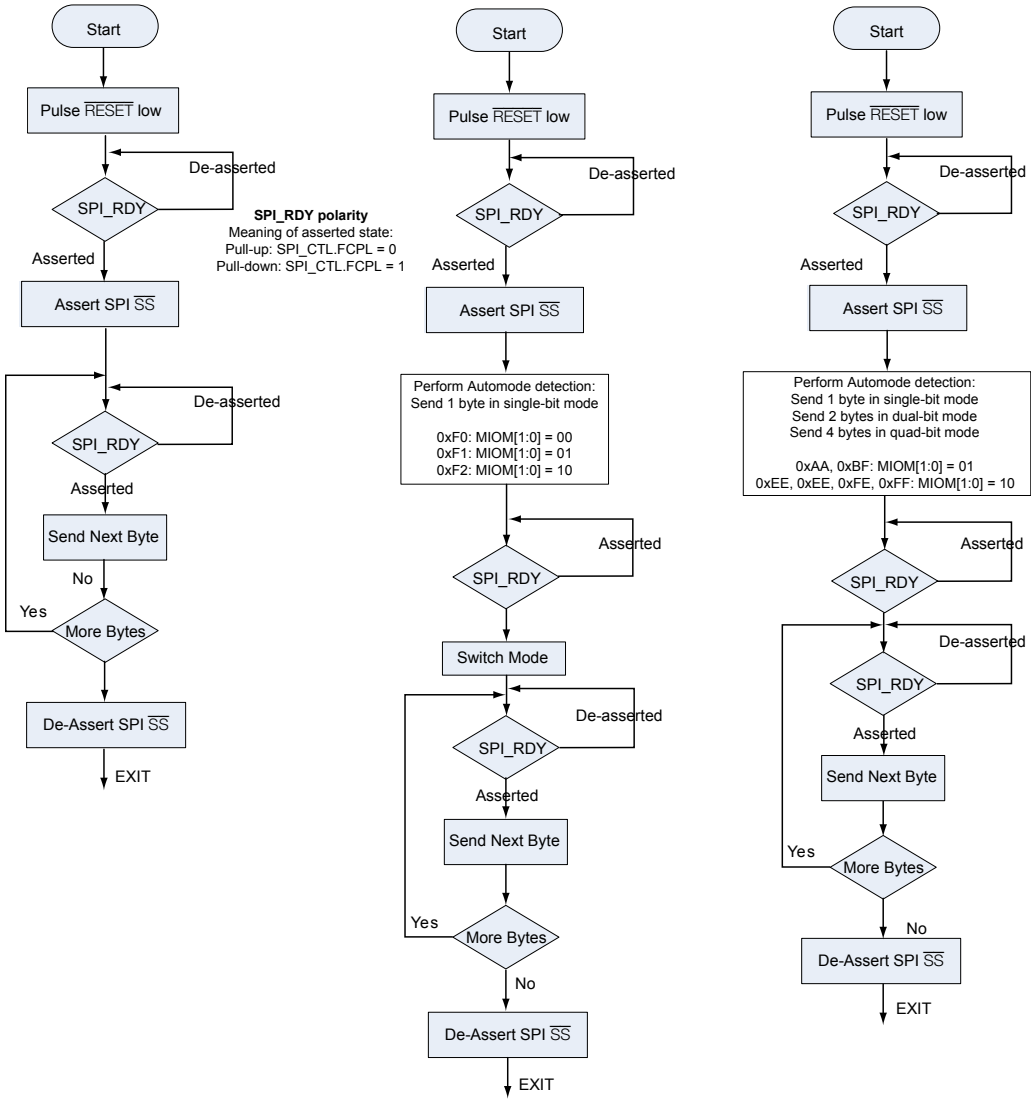


Figure 47-8: SPI Program Flow on the Host Side

Run-time API

The SPI slave boot mode can be called through the Boot Routine API function at run-time. Initiating a boot through the run-time API allows for extra customization such as disabling automatic device configuration or specifying a different SPI device other than the default.

When `ROM_BCMD_NOCFG` flag is specified, it is necessary to program pin multiplexing and other SPI configuration as required, while keeping the `SPI_CTL.EN` bit cleared.

The `ROM_BCMD_NOAUTO` flag can suppress auto mode detection. In that case, the desired configuration must be passed through the `ROM_BCMD_SPIS_BCODE` bit field, even if the `ROM_BCMD_NOCFG` flag is set.

The following table provides descriptions of the `adi_rom_Boot` parameter.

Table 47-20: SPI Slave Boot Command Bit Descriptions

Bit No. (Access)	Bit Name	Description/Enumeration	
19:16	ROM_BCMD_SPIS_BCODE	Boot Mode Specific BCODE. Specifies the boot mode-specific code that can further customize and control the boot process.	
		00xxb	Single bit SPI bus
		01xxb	Dual SPI bus
		10xxb	Quad SPI bus
		11xxb	Reserved
11:8	ROM_BCMD_DEVENUM	Device enumeration. Specifies the SPI device to use.	
		0x0	SPI0
		0x1	SPI1
		0x2	SPI2
		0x3	SPI3
		0x4 - 0xF	Reserved
6	ROM_BCMD_NOAUTO	Automatic device detection disable. When set, this bit disables automatic device detection and uses the setting provided in the other fields of this register to configure the boot mode.	
5	ROM_BCMD_NOCFG	Device configuration disable. When set, this bit disables device configuration. Device configuration includes reconfiguration of the peripherals MMR registers and device pin muxing.	
4	ROM_BCMD_HOST	Host boot mode enable. When set, enables SPI slave boot mode. Otherwise, use the master boot mode.	
3:0	ROM_BCMD_DEVICE	Boot source device. Specifies the device to boot from.	
		0x2	SPI

NOTE: All bits in the table that are not defined must be set to zero. Supported features may be limited depending on peripheral instance.

Link Port Target Boot Mode

This section describes booting from the link port with the processor as a target.

Link port boot is a target boot mode in which the processor receives boot data from an external link port controller through link port 0. The link port is configured for receive mode and all transfers from the link port to memory are

performed under the control of DMA. The maximum supported operating frequency of the link port is 125 MHz for which the controller boot source is responsible for deriving the clock frequency. The link port receiver operates at an asynchronous frequency up to the maximum supported operating frequency.

The link port protocol supports a way to generate link port transmit and receive service requests. The transmit service request is generated on the processor to transmit data when the transmitter is disabled. The receiver drives the `LACKx` signal high to initiate this activity. The receive service request is generated on a receiver when it is disabled. The transmitter drives the `LCLKx` signal high to initiate this activity.

Because the transmitter and receivers can be enabled at different times, external pull-down resistors are required on both the `LCLKx` and `LACKx` signals to eliminate any false service request assertions.

The link port target boot mode initialization phase waits for the receive service request before passing control back to the main kernel. Once this initial receive service request has been detected, the receiving link port is enabled and the boot process completes. The receiving link port is not disabled again until after boot is complete. Once the link port is enabled, the receive DMA channel controls all transfers. The load function for the link port receive boot mode can then point to the peripheral DMA routine of the main kernel in a similar way to the SPI slave boot mode.

NOTE: For default Link Port target mode, the peripheral DMA fault, LP0_DMA_ERR with fault ID 252 is enabled.

Run-time API

The Link Port target boot mode can be called through the boot routine API function at run time. The run-time API allows for more customization. Both device auto-detection and device configuration can be disabled, and a device other than the default LINKPORT0 can be specified.

If `ROM_BCMD_NOCFG` flag is specified, it is the programs responsibility to configure pin multiplexing as required.

The following table provides descriptions of the `adi_rom_Boot` parameter.

Table 47-21: LINKPORT Target Boot `command` Bit Descriptions

Bit No. (Access)	Bit Name	Description/Enumeration	
11:8	ROM_BCMD_ DEVENUM	Device enumeration. Specifies the LINKPORT device to use.	
		0x0	LP0
		0x1	LP1
		0x2 - 0xF	Reserved
6	ROM_BCMD_ NOAUTO	Automatic device detection disable. When set disables automatic device detection and uses the setting provided in the other fields of this register to configure the boot mode.	

Table 47-21: LINKPORT Target Boot `command` Bit Descriptions (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
5	ROM_BCMD_ NOCFG	Device configuration disable. When set, this bit disables device configuration. Device configuration includes reconfiguration of the peripherals MMR registers and device pin muxing.	
4	ROM_BCMD_ HOST	Host boot mode enable. When set, enables SPI slave boot. Otherwise, use the controller boot mode.	
3:0	ROM_BCMD_ DEVICE	Boot source device. Specifies the device to boot from.	
		0x4	LINKPORT

NOTE: All bits in the above table that are not defined must be set to zero. Supported features may be limited depending on peripheral instance.

UART Controller Boot Mode

When using UART target mode boot, the processor receives boot data from a UART host device connected to the UART interface. The device connected to UART0 is initially detected using an autobaud detection sequence. After finishing the UART target boot process, all control and status registers of the used resources are restored.

Further customization, such as disabling autobaud detection, and changing the device, use the boot routine API.

During boot operation, the host device usually relies on the `RTS` output of the UART device. At boot time, the processor does not evaluate RTS signals driven by the host. Since the RTS is in a high impedance state when the processor is in reset, or while executing a pre-boot, an external pull-up resistor to `VDDEXT` is recommended. The ***Connection Between Host and Processor*** figure shows the interconnection required for booting. The figure does not show physical line drivers and level shifters that are typically required to meet the individual UART-compatible standards.

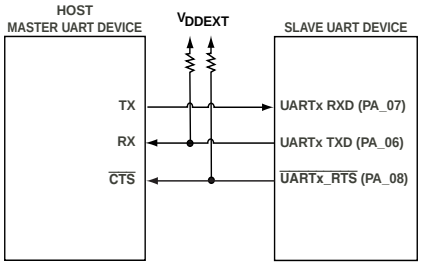


Figure 47-9: Connection Between Host and Processor

When the UART is enabled, the RTS immediately transitions low, encouraging the host to send the first boot stream data as shown in the ***Host Relying on RTS*** figure. For half-duplex UART connections, the host must avoid this action. The host must wait until it has received the 4 bytes from the target processor before sending any data.

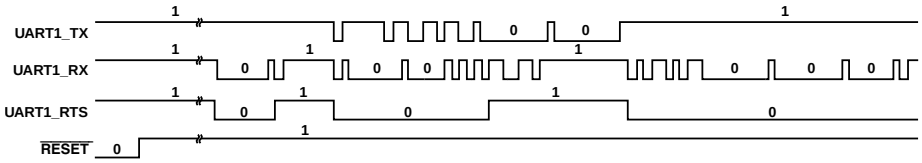


Figure 47-10: Host Relying on RTS

When the boot kernel is processing fill or Initcode blocks, it can require extra processing time and must delay the host from sending more data. This request is signaled using the RTS output.

The ***Host Relying on RTS*** figure shows RTS timing when an extended Initcode routine executes. Since code execution is distracting from the data loading, the host device must be prevented from sending more data. The timing of the RTS depends on the state of the `UART_CTL.RFRT` bit. This bit is cleared during UART target boot mode when RTS is de-asserted, the UART receive FIFO contains 4 or more data words, and another start bit is detected.

NOTE: Secure Boot Stream Padding

For target boot modes, the host must always send data in multiples of 1024 bytes. This requirement is due to the sizing of internal buffers used for DMA.

Autobaud Detection

The kernel supports autobaud detection using the '@' character as data. The host is expected to have its clock set to a rate supported in the UART.

To determine the bit rate when performing autobaud detection, use the following steps:

1. The boot kernel expects an '@' character (0x40, eight bits data, one start bit, one stop bit, no parity bit) on the UART RXD input.
2. The `UART_CLK.EDBO` and the `UART_CLK` register is cleared.
3. The boot kernel acknowledges, and the host then downloads the boot stream. The acknowledgment consists of 4 bytes: 0xBF, `UART_CLK` [15:8], `UART_CLK` [7:0], 0x00.
4. The host is requested to not send further bytes until it has received the complete acknowledge string.
5. Once the 0x00 byte is received, the host can send the entire boot stream.

The host knows the total byte count of the boot stream, but it is not required to know the content of the boot stream.

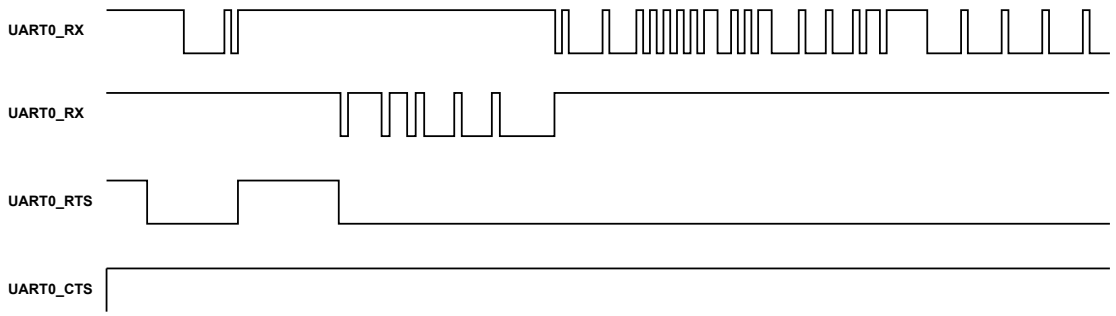


Figure 47-11: UART Autobaud Detection Waveform

The ***UART Autobaud Detection Waveform*** figure provides timing information for UART booting. After the bit rate is known, the UART is enabled and the kernel transmits the 4 acknowledge bytes.

NOTE: For default UART target mode, the peripheral DMA fault, UART2_RXDMA_ERR with fault ID 252 is enabled.

Run-time API

The UART target boot mode can be called through the boot routine API function at run time. The run-time API allows for more customization. Both autobaud detection and device configuration can be disabled, and a device other than the default UART0 can be specified.

If `ROM_BCMD_NOCFG` flag is specified, it is the programs responsibility to configure pin multiplexing as required.

Autobaud detection can be suppressed using the `ROM_BCMD_NOAUTO` flag. In this case, the desired configuration can be passed through the `ROM_BCMD_UART_CLK` bit field. If the `ROM_BCMD_UART_CLK` bit field is zero, `UART_CLK` is evaluated. If a value of 0xFFFF was present, the default error routine of the boot kernel is called and the booting process is aborted. Otherwise, the value in `UART_CLK` remains untouched.

The following table provides descriptions of the `adi_rom_Boot` parameter.

Table 47-22: UART Target Boot `command` Bit Descriptions

Bit No. (Access)	Bit Name	Description/Enumeration	
31:16	ROM_BCMD_ UART_CLK	UART Clock Divider. When set to zero this field is ignored.	
15	ROM_BCMD_ UART_EDBO	UART Clock Divider Mode When set enables EDBO functionality.	
11:8	ROM_BCMD_ DEVENUM	Device enumeration. Specifies the UART device to use.	
		0x0	UART0
		0x1	UART1

Table 47-22: UART Target Boot `command` Bit Descriptions (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0x2	UART2
		0x3	UART3
		0x4 - 0xF	Reserved
6	ROM_BCMD_NOAUTO	Automatic device detection disable. When set disables automatic device detection and uses the setting provided in the other fields of this register to configure the boot mode.	
5	ROM_BCMD_NOCFG	Device configuration disable. When set, this bit disables device configuration. Device configuration includes reconfiguration of the peripherals MMR registers and device pin muxing.	
4	ROM_BCMD_HOST	Host boot mode enable. When set, enables SPI slave boot. Otherwise, use the controller boot mode.	
3:0	ROM_BCMD_DEVICE	Boot source device. Specifies the device to boot from.	
		0x3	UART

NOTE: All bits in the above table that are not defined must be set to zero. Supported features may be limited depending on peripheral instance.

OSPI Controller Boot Mode

OSPI controller boot supports booting from flash devices via the SPI3 controller. Similar to legacy SPI (SPI0/SPI1/SPI2), it supports boot from many different flash devices provided by different flash vendors further increasing the range of devices due to support for DDR modes of operation. This interface enables to reduce boot times by using faster modes of operation supported by the OSPI controller including DDR modes of operation to achieve higher transfer rates. Most flash vendors support at least a basic and standard 0x03 read command allowing for support for a single bit SPI flash interface.

The OSPI boot mode uses a device auto-detection feature that is enabled by default. This lets the boot stream itself instruct updates to the SPI configuration and the read command used allowing for more efficient transactions such as for enabling quad bus widths and DDR modes and faster SPI clocks.

NOTE: The ADSP-SC598 processor family supports OSPI controller boot through PC_XX pins, while SPI controller boot is supported through PA_XX pins. Separate flash devices are required for both the boot modes, as they don't share the common PORT pins.

Boot from External SPI Flash Devices

The OSPI boot mode supports booting from 24-bit or 32-bit addressable flash devices. The boot mode uses the MDMA channels, which works with OSPI controller to get data from flash memory in memory mapped mode.

When auto-device detection is enabled, the SPI memory is initially read using the standard 0x03 SPI read command with a reduced clocking frequency for maximum compatibility. The first nibble of the boot stream is then used to reconfigure the SPI interface.

NOTE: Support for automatic device detection via the first nibble of the boot stream is not supported when booting secure boot streams. Instead when signing the boot image an attribute can be set in the image header that specifies the configuration to use.

For booting, the SPI memory is connected as shown in the ***SPI Memory Connections*** figure.

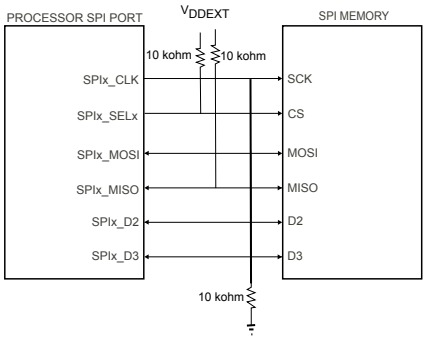


Figure 47-12: SPI Memory Connections

The pull-up resistor on the target select signal ensures that the memory is deselected when the pin is in a high-impedance mode such as during reset.

Initialization codes are allowed to manipulate the `ADI_ROM_BOOT_CONFIG::dBootCommand` to extend boot mechanism to a second SPI memory connected to another target select pin. Updating the field that specifies the target select signal for use, allows the boot process to manage larger boot streams than fit into a single SPI device.

NOTE: If modifying the target select signal used during the boot process, configure the pin multiplexing to enable the correct functionality for the pin. Once the boot process has proceeded past the configuration function and the boot process has started, the boot kernel does not perform any further pin multiplexing operations.

OSPI Device Detection Routine

As boot mode supports booting from various SPI memories, the boot kernel automatically detects what type of memory is connected. To determine whether the SPI memory device requires a 24-bit or 32-bit addressing scheme, the boot kernel performs a device detection sequence prior to booting. The `OSPI_MISO` signal requires a pull-up resistor. The routine relies on the fact that memories do not drive their data outputs unless the right number of address bytes are received.

The SPI flash autodetection routine determines the SPI configuration to be used by data in the boot stream. The autodetection first determines the number of address cycles needed to address the flash device.

This operation is performed by configuring the controller in STIG mode initially with the number of expected address bytes programmed to 3 and a basic read command (0x3). A read command from address 0x00000000 with a 3-byte address is issued. If the flash returns a value other than 0x00 or 0xFF, the flash is in 3-byte address mode.

Otherwise, the same process is repeated with the STIG controller configured with four address bytes. Only three and four byte address cycles are supported.

The lower nibble of the received byte is then used to further customize the boot mode. This nibble is referred to as the BCODE. The boot code applies settings to the OSPI peripheral according to the ***OSPI controller Boot BCODE Descriptions***.

The boot process continues with normal boot operation and it re-issues a command to re-read from address 0. Two read sequences load the first block header. Separate read sequences load further block headers and block payload fields.

The ***SPI Device Detection Principle*** figure illustrates how individual devices behave.

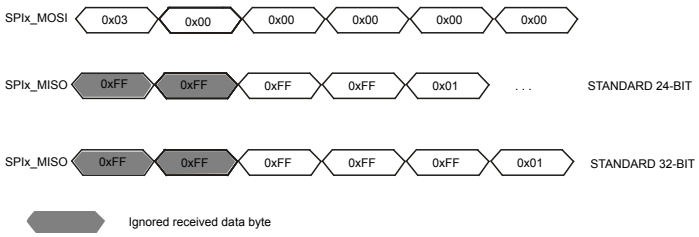


Figure 47-13: SPI Device Detection Principle

Table 47-23: OSPI Controller BCODE Configuration Lookup Table

Member Name	BCODE				
	0	**1**	**2**	**3**	**4**
Transfer Type	Single bit command, address and data	Single bit command, address and data	Single bit command, address and data	Single bit command, dual address and data	Single bit command, dual address and data
ubDummyCycles	0x00	0x00	0x08	0x04	0x08
ubReadCommand	0x03	0x03	0x0B	0xBB	0xBB
ubDataBits	0x00	0x00	0x00	0x01	0x01
ubAddressBytes (This field is retained. Not used by boot kernel)	0x03	0x03	0x03	0x03	0x03
uwClkLower	0x000F	0x000F	0x0003	0x0003	0x0003
uReserved0	0x0000	0x0000	0x0000	0x0000	0x0000
nCfg	0x00000081	0x00000081	0x00000081	0x00000081	0x00000081
nDsr	0x00000002	0x00000002	0x00000002	0x00000002	0x00000002
nDrir	0x00000000	0x00000000	0x00000000	0x00011000	0x00011000
uReserved1	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
nDummy	0x00	0x00	0x00	0x00	0x00

Table 47-24: OSPI Controller BCODE Configuration Lookup Table

Member Name	BCODE				
	5	6	7	8	9
Transfer Type	Single bit command, Quad bit address and data	Single bit command, Quad bit address and data	Single bit command, single bit DDR address and data	Single bit command, single bit DDR address and data	Single bit command, dual DTR address and data
ubDummyCycles	0x06	0x0A	0x06	0x08	0x02
ubReadCommand	0xEB	0xEB	0x0D	0x0D	0xBD
ubDataBits	0x02	0x02	0x00	0x00	0x01
ubAddressBytes (This field is retained. Not used by boot kernel)	0x03	0x03	0x03	0x03	0x03
uwClkLower	0x0003	0x0003	0x0007	0x0007	0x0007
uReserved0	0x0000	0x0000	0x0000	0x0000	0x0000
nCfg	0x00000081	0x00000081	0x00000081	0x00000081	0x00000081
nDsr	0x00000002	0x00000002	0x00000002	0x00000002	0x00000002
nDrir	0x00022000	0x00022000	0x00000400	0x00000400	0x00011400
uReserved1	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
nDummy	0x00	0x00	0x00	0x00	0x00

Table 47-25: OSPI Controller BCODE Configuration Lookup Table

Member Name	BCODE					
	A	B	C	D	E	F
Transfer Type	Single bit command, dual DTR address and data	Single bit command, dual DTR address and data	Single bit command, quad DTR address and data	Single bit command, quad DTR address and data	Single bit command, quad DTR address and data	Reserved
ubDummyCycles	0x04	0x06	0x06	0x07	0x08	0x00
ubReadCommand	0xBD	0xBD	0xED	0xED	0xED	0x0B
ubDataBits	0x01	0x01	0x02	0x02	0x02	0x00
ubAddressBytes (This field is retained. Not used by boot kernel)	0x03	0x03	0x03	0x03	0x03	0x03
uwClkLower	0x0007	0x0007	0x0007	0x0007	0x0007	0x0003
uReserved0	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000

Table 47-25: OSPI Controller BCODE Configuration Lookup Table (Continued)

Member Name	BCODE					
	A	B	C	D	E	F
nCfg	0x00000081	0x00000081	0x00000081	0x00000081	0x00000081	0x00000081
nDsr	0x00000002	0x00000002	0x00000002	0x00000002	0x00000002	0x00000002
nDrir	0x00011400	0x00011400	0x00022400	0x00022400	0x00022400	0x00000000
uReserved1	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
nDummy	0x00	0x00	0x00	0x00	0x00	0x00

Table 47-26: OSPI Controller BCODE Configuration Lookup Table (Applicable to Secure Streams Only)

Member Name	BCODE					
	10	11	12	13	14	15
Transfer Type	Single bit command, dual address and data	Single bit command, quad address and data	Single bit command, single DTR address and data	Single bit command, single DTR address and data	Single bit command, q uad DTR address and data	Single bit command, q uad DTR address and data
ubDummyCycles	0x0C	0x04	0x04	0x05	0x03	0x09
ubReadCommand	0xBB	0xEB	0x0D	0x0D	0xED	0xED
ubDataBits	0x01	0x02	0x00	0x00	0x02	0x02
ubAddressBytes (This field is retained. Not used by boot kernel)	0x03	0x03	0x03	0x03	0x03	0x03
uwClkLower	0x0003	0x0003	0x0007	0x0007	0x0007	0x0003
uReserved0	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
nCfg	0x00000081	0x00000081	0x00000081	0x00000081	0x00000081	0x00000081
nDsr	0x00000002	0x00000002	0x00000002	0x00000002	0x00000002	0x00000002
nDrir	0x00011000	0x00022000	0x00000400	0x00000400	0x00022400	0x00022400
uReserved1	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
nDummy	0x00	0x00	0x00	0x00	0x00	0x00

NOTE: For the above configurations, the addressing scheme can be 3-bytes or 4-bytes depending on the addressing of flash detected in auto-detection. The SPI mode byte issued for all the SPI controller peripheral based configurations is 0x00. The mode byte is the first byte transmitted after the address cycles and is used to control the continuous read mode functionality in which the next read operation is not required to issue a command cycle. Continuous read mode is not supported during the boot process.

Supported Quad Mode Enable Methods

The boot ROM does not support enabling quad mode on the SPI flash device. To boot in quad mode, the flash device must be configured outside the boot ROM.

To enable Quad mode, initially boot in Dual mode and use an initcode.

Run-Time API

The following table provides descriptions of the adi_rom_Boot() command parameter.

Table 47-27: OSPI Controller Boot Command Bit Descriptions

Bits	Name	Setting	Description
31:28	ROM_BCMD_SPIM_SPEED	0-1111b	SPI clock divider used. This value is written to the SPI peripherals clock divider register
27	ROM_BCMD_SPIM_CMDSKIP_EN	0	Issue a read command with every read operation
		1	Do not issue a read command with each read operation only issue the read command on the first read.
26:22	ROM_BCMD_SPIM_DUMMY	00000b	No dummy clock cycles required after the address
		00001b	1 dummy clock cycle required after the address
		....	.....
		11111b	31 dummy clock cycle required after the address
21:20	ROM_BCMD_SPIM_ADDR	00b	Flash device requires an 8-bit address
		01b	Flash device requires a 16-bit address
		10b	Flash device requires a 24-bit address
		11b	Flash device requires a 32-bit address
19:16	ROM_BCMD_SPIM_BCODE	0000b-1111b	Applies an initial configuration as described in the SPI Controller Boot BCODE configurations. Not all options may be available.
15	ROM_BCMD_SPIM_CMD	0	Issue the read command over the single bit bus
		1	Issue the read command over the multi-bit bus. Not recommended for use. Intended for use with products with on chip SPI flash only, and even then it can cause complications.
14:12	ROM_BCMD_SPIM_SSEL	000b	Use target select 1 for SPI chip select
		001b	Use target select 2 for SPI chip select
		010b	Use target select 3 for SPI chip select
		011b	Reserved
		100b	Reserved

Table 47-27: OSPI Controller Boot Command Bit Descriptions (Continued)

Bits	Name	Setting	Description
		101b	Reserved
		110b	Reserved
		111b	Reserved
11:8	ROM_BCMD_SPIM_DEVENUM	0 to 16	Specifies the boot peripheral enumeration. For OSPI Controller boot, configure this field to 0.
7	Reserved	Reserved	Reserved
6	ROM_BCMD_SPIM_NOAUTO	0	Perform automatic device detection and peripheral configuration based on the BCODE value (first nibble) of the boot streams block header
		1	Do not perform automatic device detection
5	ROM_BCMD_SPIM_NOCFG	0	Instructs the config routine to perform a pin-muxing and full peripheral configuration
		1	The config routine does not change the configuration of the peripheral and idoes not enable pin muxing. This can be used when the boot software does not support specific features of a peripheral, users can configure the peripheral prior to calling the boot routine.
4	ROM_BCMD_SPIM_HOST	0	Controller boot mode
		1	Not to be used for OSPI Controller Boot
3:0	ROM_BCMD_SPIM_DEVICE	0000b	Not to be used for OSPI boot
		0001b	Not to be used for OSPI boot
		0010b	Not to be used for OSPI boot
		0011b	Not to be used for OSPI boot
		0100b	Not to be used for OSPI boot
		0101b	Not to be used for OSPI boot
		0110b	Not to be used for OSPI boot
		0111b	Not to be used for OSPI boot
		1000b	OSPI Memory Mapped Boot (Memory mapped mode and MDMA)
		1001b – 1111b	Not to be used for OSPI boot

NOTE: All bits in the above table that are not defined must be set to zero.

To support the above lookup table at varying SPICLK frequency range, boot kernel programs the Read data capture register such that sampling time does not go beyond one full cycle of SPI CLK for STR and

half cycle of SPI CLK for DTR mode of operation. When this timing is not sufficient to support all flash vendors, there is a provision to update the Read data capture register from the Customer OTP space with a worst case value corresponding to a different flash device used by user. This value has to be derived based on the minimum-maximum AC timing propagation delay from OSPI controller to pin.

eMSI Controller Boot Mode

eMSI boot mode allows the processor to boot from the onboard eMMC devices. Boot support for eMMC devices is available, but, SD card boot is not supported. Data is loaded from the eMMC device in page mode; it is read 1024 bytes at a time. This process optimizes boot time by loading the data in two buffers in a ping-pong fashion, filling one buffer while another buffer is being processed by the boot kernel. In eMSI boot, there is support for booting from either from the user area of eMMC device or the boot area of eMMC device. The user can choose between the reliability provided by the boot area memory of eMMC device and the abundance of space to store a large boot stream in the user area.

The boot flow for eMSI boot mode is shown in the figure:

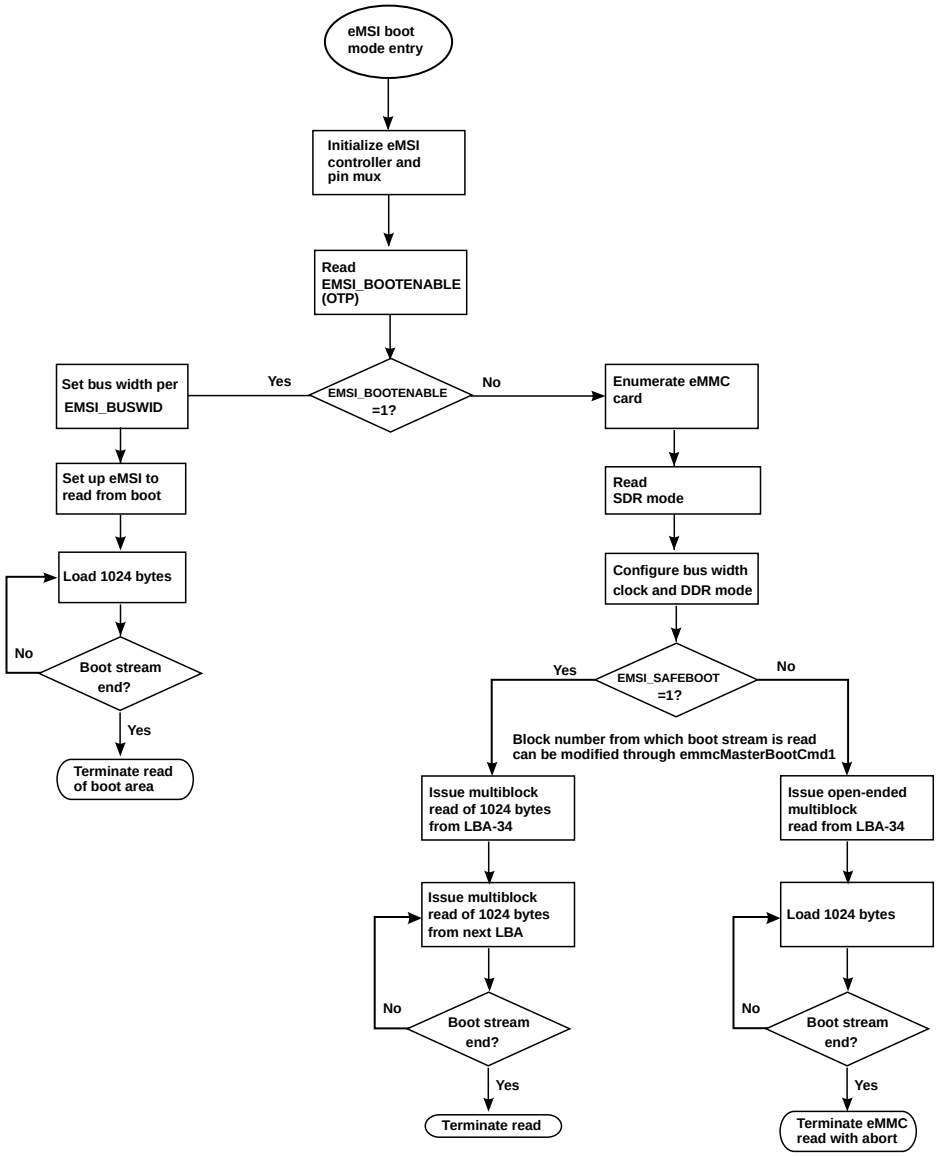


Figure 47-14: eMSI Boot Mode Flow

After the eMSI controller is initialized to read from the eMMC device, the ROM_BCMD_EMSI_BOOTENABLE field in `dBootCommand` determines whether the boot is attempted from the user area or the boot area of eMMC device. (The field is read from the OTP).

When ROM_BCMD_EMSI_BOOTENABLE = 0 (default), boot is attempted from the *user* area of the eMMC device. (An engineer can program the OTP to update the ROM_BCMD_EMSI_BOOTENABLE field).

Booting from the *user* area includes:

- Enumerating the connected eMMC devices –the process allocates RCA to the card.
- Loading boot data–eMSI can be configured to read the data either in SDR or DDR mode based on the ROM_BCMD_EMSI_SDRMODE bit in the OTP. By default, the reads are done in 8-bit DDR mode, unless

this bit is configured to initiate the reads in SDR mode. A bus width of 8-bit is used by default to get the maximum throughput. The bus width can be reconfigured to 1-bit or 4-bit through the ROM_BCMD_EMSI_BUSWID field in the `dBootCommand`. In user area boot (default), the boot stream is read from block 34 (LBA-34). It is expected that the user has programmed the boot stream to start at this address. The user can implement MBR or GUID because the initial 33 blocks of eMMC device are available. However, the boot can be done from any other block including block 0 by programming a second `dBootCommand` (`dBootCommand1`) to indicate the block. If the field is left unprogrammed, the boot will happen from block 34 of the user area. To boot from any other block user must program a value of Block Number +1 in `dBootCommand1`. For example to boot from block 20, program a value of 21; to boot from block 0, program a value of 1.

When ROM_BCMD_EMSI_BOOTENABLE =1 in OTP, the boot kernel expects the boot stream to be stored in the boot partition 1 of the eMMC device.

Booting from the *boot* area includes:

- Configuring the controller for boot operation–in this case, boot kernel expects the eMMC device to have the eMMC device configured for 8-bit SDR access to the boot partition for the best throughput for reading the boot stream (DDR mode is not supported by the eMSI controller during boot operation so DDR mode should not be set by user for eMMC boot operation).
- Reading the boot data–the boot kernel reads 1024 bytes in page mode until the end of boot stream and then terminates the read from boot area.

eMSI Boot Configuration

eMSI boot configuration ensures that the eMSI controller is set up properly and the eMMC device is in a state to read the data out. The configuration process can vary depending on the value programmed in the `dBootCommand` in OTP or passed through the ROM API call.

EMSI_BOOTENABLE = 0. Booting from the user area. Configure as follows:

1. Configure the eMSI clock to less than 400 kHz for enumeration.
2. Perform eMMC device enumeration and set up the device RCA.
3. Set up the eMSI and eMMC device bus width and DDR mode as per the BCMD_EMSI_BUSWID and BCMD_EMSI_SDRMODE values in `dBootCommand`.
4. When BCMD_EMSI_SDRMODE = 1 (SDR mode), the BCMD_EMSI_HISPEEDENABLE bit is used to determine whether to put card in high speed mode or at the default speed.
5. Configure the frequency of the eMSI clock as per the clock settings in BCMD_EMSI_SPEED field in `dBootCommand`.

   NOTE: For power up case DDR mode in 8-bit bus width with clocks to configured to maximum frequency possible as per default CGU clocks.

6. Call the load function to read 512 bytes of data from loader stream.

7. Identify the type of boot stream (secure/non-secure).

EMSI_BOOTENABLE =1. Booting from the boot area. Configure as follows:

1. Configure the frequency of the eMSI clock as per the clock settings in the BCMD_EMSI_SPEED field in `dBootCommand`.
2. Configure the bus mode as per the BCMD_EMSI_BUSWID field in `dbootcommand`. When booting from the boot area, only the SDR mode of operation is supported. Therefore, DDR mode should not be configured.
3. Call the load function to read 512 bytes of data from the loader stream.
4. Identify the type of boot stream (secure/non-secure).

eMSI Data Loading

The eMSI load routine is responsible for reading the boot data stored in the eMMC device. It is called by the boot kernel to request data from the device. Boot ROM employs different load routines based on whether user has requested boot from boot area or user area of the eMMC device. For boot from user area (EMSI_BOOTENABLE= 0), the load routines used depend on state of the BCMD_EMSI_SAFEBOOT bit in `dBootCommand`. The BCMD_EMSI_SAFEBOOT bit is provided as an optional way to load the boot stream using multi-block read commands, each reading 1024 bytes of data. This method of loading is slower. Sending multi-block read commands to read each set of 1024 byte adds some delay in reading the data. Each read command has latency that varies across eMMC devices.

When BCMD_EMSI_SAFEBOOT =1, the load function does the following:

- Sets up the block length on device to 512 bytes.
- Sends CMD23 to predefine the length of transfer as 1024 bytes.
- Sets up the descriptor for ADMA2 to read 1024 bytes.
- Issues a multi-block read every time the load function is called to read 1024 bytes.

When BCMD_EMSI_SAFEBOOT = 0 (default), the eMSI load routine is set up to use the unterminated multi-block read command. Data is read from the eMMC device as long as the end of the boot stream is not detected.

On the first call of the load function:

1. The eMSI is configured for a AMDA2 read.
2. Descriptors are set up.
3. A multi-block read command is issued.

For every subsequent call of the load function, no further multi-block read command is issued; only the ADMA2 descriptor is updated to update the destination buffer (where the data is read). Once the descriptor is ready, the ADMA2 requests the data, and the controller continues to read the data from the device for the multi-block read issued on the first call of the load function. The controller does not have to issue the read command every time the data must be read from the eMMC device. This process reduces the latency incurred by the read command. Once

the end of the loader stream is detected, boot is terminated by sending out the stop command to end the multi-block read from the device.

When EMSI_BOOTENABLE= 1 (booting from a boot area), the boot ROM uses a separate subroutine to handle the loading of data from boot area.

Loading from boot area includes:

- Setting up the eMSI controller to read from the boot area.
- Setting up the ADMA2 descriptor.
- Issuing a CMD0 with the 0xf0f0f0f0 argument.

Once the CMD line from controller side is driven low, after some delay, the eMMC device starts driving the data. Each subsequent load function involves updating the ADMA 2 descriptor to read 1024 bytes. Once the end of the loader stream is detected, boot is terminated by deasserting the CMD line.

Page mode is used to read data from the eMMC device. Page mode optimizes memory reads for block-organized devices by always reading a page, rather than reading data on demand. The page size of the physical source device is defined by the `dPageByteCount` variable.

Run-Time API

The following table provides descriptions of the adi_rom_Boot() command parameter.

Table 47-28: eMSI Controller Boot Command Bit Descriptions

Bits	Name	Setting	Description
29:22	ROM_BCMD_EMSI_SPEED	0x00- 0xFF	eMSI clock divider to be used. This value is written to the eMSI clock divider register.
21:20	Reserved	Reserved	Reserved
19	ROM_BCMD_EMSI_HISPEE-DENABLE	0	Disable high speed mode for eMMC device
		1	Enable high speed mode for eMMC device
18	ROM_BCMD_EMSI_SAFE-BOOT	0	Boot whole loader stream using single multiblock read command
		1	Enable boot using multiple read commands
17	ROM_BCMD_EMSI_CMD0DE-LAY	0	No additional delay after CMD0
		1	Additional delay after CMD0
16	ROM_BCMD_EMSI_BOOTEN-ABLE	0	Boot from user area of eMMC device
		1	Boot from boot area of eMMC device
15	Reserved	Reserved	Reserved

Table 47-28: eMSI Controller Boot Command Bit Descriptions (Continued)

Bits	Name	Setting	Description
14	ROM_BCMD_EMSI_SDRMODE	0	Use DDR mode (default for user area boot
		1	Use SDR mode (default for boot area boot). For boot from the boot area of the eMMC device always set this bit to 1.
13:12	ROM_BCMD_EMSI_BUSWID	00b	1-bit mode
		01b	4-bit mode
		10b	8-bit mode
		11b	Reserved
11:8	ROM_BCMD_EMSI_DEVENUM	0 to 16	Specifies the boot peripheral instance. Configure to 0 when there is only one instance of eMSI.
7	Reserved	Reserved	Reserved
6	ROM_BCMD_EMSI_NOCFG	0	Has no effect on eMSI boot
		1	Has no effect on eMSI boot
5	ROM_BCMD_EMSI_NOCFG	0	Instructs the config routine to perform pin-muxing configuration and full peripheral configuration
		1	The config routine will not change the configuration of the peripheral and it will not enable pin-muxing. This can be used if the boot software does not support specific features of a peripheral, users can configure the peripheral prior to calling the boot routine.
4	ROM_BCMD_EMSI_HOST	0	Controller boot mode
		1	Not to be used for eMSI boot
3:0	ROM_BCMD_EMSI_DEVICE	0000b-1000B	Not to be used for eMSI boot
		1001b	eMSI Controller boot mode
		1010b-111B	Not to be used for eMSI boot

Table 47-29: eMSI Controller Start Address Bit Descriptions

Bits	Name	Setting	Description
31:0	eMMCStartAddress	0x00000000-0xFFFFFFFF	Number of Block+1 in user area from where to start reading the boot stream. If this is configured to zero, by default, a boot from LBA-34 will happen. To boot from LBA-0, configure this field to 1 and so on.

Default Booting Peripheral Pin-Mux

By default SPI2, UART0, LP0 and OSPI0 boot peripherals support booting through a default pin-mux combinations given in the following tables.

Table 47-30: SPI2 Default Pin-Mux

Signals	ADSP-2156x	ADSP-2159x	ADSP-SC59x
SPI2_MOSI	PA_01	PA_01	PA_01
SPI2_MISO	PA_00	PA_00	PA_00
SPI2_D2	PA_02	PA_02	PA_02
SPI2_D3	PA_03	PA_03	PA_03
SPI2_CLK	PA_04	PA_04	PA_04
SPI2_SEL1b	PA_05	PA_05	PA_05
SPI2_RDY	PB_05	PB_05	PB_05

Table 47-31: OSPI0 Default Pin-Mux

Signals	ADSP-2156x	ADSP-2159x	ADSP-SC59x
OSPI0_MOSI	PA_01	PA_01	PC_11
OSPI0_MISO	PA_00	PA_00	PC_12
OSPI0_D2	PA_02	PA_02	PC_10
OSPI0_D3	PA_03	PA_03	PC_09
OSPI0_D4	PA_06	PA_06	PD_00
OSPI0_D5	PA_07	PA_07	PC_15
OSPI0_D6	PA_08	PA_08	PC_14
OSPI0_D7	PA_09	PA_09	PC_13
OSPI0_CLK	PA_04	PA_04	PC_08
OSPI0_SEL1b	PA_05	PA_05	PD_01

Table 47-32: UART0 Default Pin-Mux

Signals	ADSP-2156x	ADSP-2159x	ADSP-SC59x
UART0_CTS	PA_09	PA_09	PA_09
UART0_RTS	PA_08	PA_08	PA_08
UART0_TX	PA_06	PA_06	PA_06
UART0_RX	PA_07	PA_07	PA_07

Table 47-33:

Signals	ADSP-2156x	ADSP-2159x	ADSP-SC59x
LP0_DAT0	PB_07	PB_07	PB_07
LP0_DAT1	PB_08	PB_08	PB_08
LP0_DAT2	PB_09	PB_09	PB_09
LP0_DAT3	PB_10	PB_10	PB_10
LP0_DAT4	PB_11	PB_11	PB_11
LP0_DAT5	PB_12	PB_12	PB_12
LP0_DAT6	PB_13	PB_13	PB_13
LP0_DAT7	PB_14	PB_14	PB_14
LP0_CLK	PB_06	PB_06	PB_06
LP0_ACK	PB_04	PB_04	PB_04

Boot Loader Stream

A loader stream is a set of formatted blocks containing instructions for the boot kernel, as well as the application and data for loading to the chip. This section details the different aspects of the stream, its blocks, some common use cases, and the ROM function.

Each block begins with a block header that contains attributes of the block as well as flags to control its processing by the boot ROM. On power-up or reset, the processor begins executing the on-chip boot ROM. The boot stream is either read from memory or received from a peripheral, depending on the boot mode specified. Each block in the boot stream instructs the boot kernel to perform an action, most commonly to load data to a specified location. For a complete list of actions, refer to the Block Types section.

Some common actions include the following:

- running code that initializes a peripheral
- processing data then loading it to a location

As shown in the ***Project Flow*** figure, a utility is required to process the resulting output from the tool chain to create a valid boot stream. This utility can be in the form of a standalone application or script that parses an application image file, elf output file, or text-based file such as Intel hex. A flash programmer utility can format a boot stream internally.

The `elfloader` utility parses the application data and creates a set of blocks representing the segmented application. When processing an actual image that must be loaded to a single contiguous memory block, this representation can contain as little as a single header block. The output of the standalone utility is stored in a loader file (`.ldr`). The loader file contains the boot stream and becomes available to the hardware by:

- Programming or burning it into non-volatile external memory.
- Alternatively, sending it through a peripheral such as the UART during boot time

A loader stream must always begin with a first block and end with a final block. The loader file contains the boot stream and becomes available to hardware by:

- programming or burning it into non-volatile external memory, or
- sending it through a peripheral during boot time

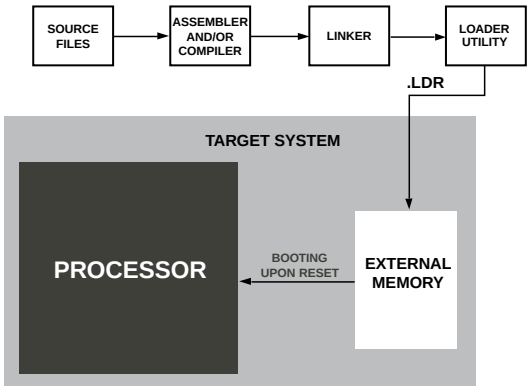


Figure 47-15: Project Flow

The ***Booting Process*** figure shows the parallel or serial boot stream contained in a flash memory device. In host boot scenarios, the non-volatile memory usually connects to the host processor rather than directly to the processor. After reset, the on-chip boot kernel reads and parses the headers, and processes the loader stream block-by-block. Finally, payload data is copied to a destination addresses, either in on-chip L1 and L2 memory, or off-chip to SDRAM or SRAM.

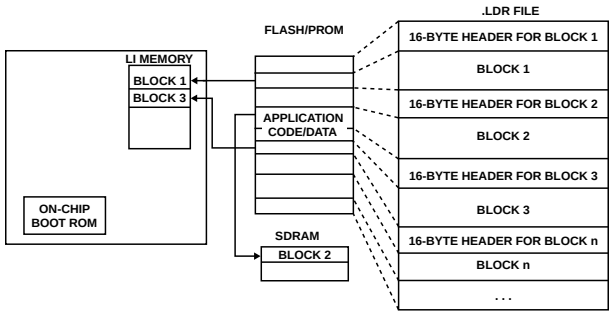


Figure 47-16: Booting Process

In some cases (for example, secure boot or when the `BFLAG_INDIRECT` flag for any block is set), the boot kernel uses another memory block for intermediate data storage. In order to preserve the security of the device processors do not allow these storage regions to be initialized with boot data. The boot stream is loaded to the intermediate storage then processed by the kernel and loaded to the final destination. The final destination cannot be the intermediate storage location otherwise the boot process terminates in an error.

Block Header

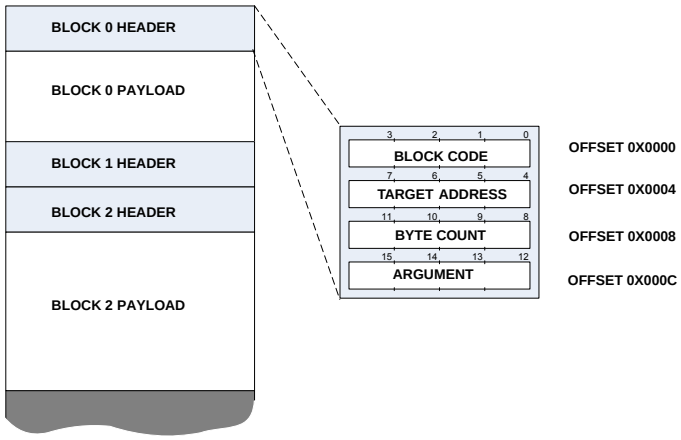


Figure 47-17: Loader Stream Block Structure

A boot stream consists of multiple boot blocks as shown in the figure. A 16-byte block header begins every block. The 16 bytes are functionally grouped into four 32-bit words:

- Block code
- Target address
- Byte count
- Argument field

This section provides a general description describes the fields in general. The use varies depending on the block type and boot mode.

Block Code

Table 47-34: Block Code Flags

Bit	Name	Description
0–3	BCODE	Specific to boot modes (see Boot Modes)
4	BFLAG_SAVE	Intended to allow for a user application to mark blocks for saving the memory of this block to off-chip memory in case of power failure. The on-chip boot kernel does not use this flag.
5	BFLAG_AUX	When set indicates that the byte address space translation for SHARC core boot blocks requires translation to the 48-bit PM address space. When cleared translation to the 32-bit address space is performed.
7–6	Reserved	
8	BFLAG_FILL	Fill the target location with a specified 32-bit value
9	Reserved	
10	BFLAG_CALLBACK	Calls the previously registered callback function

Table 47-34: Block Code Flags (Continued)

Bit	Name	Description
11	BFLAG_INIT	Calls function at target address. If the block contains a payload, the payload is loaded prior to the call.
12	BFLAG_IGNORE	Block payload is ignored
13	BFLAG_INDIRECT	Boots the payload to the intermediate storage location
14	BFLAG_FIRST	Indicates the block that is the beginning of a new application
15	BFLAG_FINAL	Indicates the last block of a loader stream. Booting will complete after processing the block. This flag does not denote the end of an application in a Multi-Application Boot Stream.
16–23	HDRCHK	A simple 8-bit XOR checksum of the other 24 bits in the boot block header.
24–31	HDRSIGN	0xAD, 0xAC or 0xAB. Indicates if the boot block is intended for core 0, core 1 or core 2 respectively.

TARGET_ADDRESS

The TARGET_ADDRESS holds the applicable address for the block, (where the code or data is loaded). However, the interpretation of the field differs depending on what specific flags are set in the block code. Refer to the documentation for each block type for details.

The following attributes must be true:

- The target address must be divisible by 4, as the boot kernel uses 32-bit DMA for certain operations.
- The target address must point to valid on-chip or off-chip memory locations.

BYTE_COUNT

The BYTE_COUNT must be divisible by 4, and can also be zero. This 32-bit field generally holds the size of the block. In some cases, it has a different use (such as when BFLAG_FILL is set). See the Block Types section for information on the variations.

ARGUMENT

The 32-bit field is a user-variable for most block types. The Initcode or callback routine can access this value and use it for optional instructions.

The different block types use the ARGUMENT field in various ways. See the Block Types descriptions for further information.

Block Types

A loader stream is a set of linked blocks and each block is responsible for performing a certain function dependent on the block type. The flags in the block header define a block type. Operations include functions such as loading

data, filling a memory region with data, and instructing the kernel to stop processing. This section describes each block type and its usage within a boot loader stream.

Normal Block

The primary function of a block is to load data into a specified location of memory. A normal block instructs the boot kernel to load the data contained in its payload to the location specified in the TARGET_ADDRESS field. The BYTE_COUNT defines the size of the payload. Once the correct amount of data has been loaded, the kernel moves on to process the next block in the stream.

Table 47-35: Flags

Flag	Required Value	Init
TARGET_ADDRESS	Y	Address where payload is loaded (must be valid)
BYTE_COUNT	Y	Size of block in bytes

First Block

A first block indicates the start of a boot stream and is always needed at the beginning of the boot stream. When a loader stream contains Multi-Application Boot Streams, a first block that occurs within the loader stream indicates the beginning of a new application.

When the kernel processes the first block in a loader stream, the boot kernel uses the TARGET_ADDRESS to determine the application entry location. For more details, refer to Boot Termination and Application Execution .

NOTE: A first block cannot be combined with a fill block.

Table 47-36: Flags

Flag	Required Value	Init
BFLAG_FIRST	Y	1
ARGUMENT	Y	Offset to the next application, or first address following loader stream. Commonly referred to as the NEXTDXE field.
TARGET_ADDRESS	Y	When the block is the first block in a loader stream it defines the start address for the application. If the block is not the first in a loader stream, use the target address as in normal operation.

Final Block

The final block marks the last block in a boot stream. After processing a final block, the boot kernel jumps to the start address of the application. For more information on the definition of the start address, refer to Boot Termination and Application Execution .

There is further customization available to alter the kernel behavior. For example, the kernel can be instructed to return from the boot routine rather than jump to the application using initialization codes or the `adi_rom_Boot()` API.

Before the boot kernel passes program control to the application, it does some housekeeping. Most of the registers in use are set to their default state. However, some register values can differ depending on the boot mode. See Boot Modes for more information.

Table 47-37: Flags

Flag	Required Value	Init
`BFLAG_FINAL`	Y	1

Indirect Block

An indirect block is first loaded to a storage location before being copied to the destination. This block is used in the following situations:

- Some boot modes do not use DMA from the boot peripheral. The core is not always able to access some memory locations directly or efficiently. An intermediate load to a different location improves overall efficiency.
- In some booting situations, the data in the payload must be operated-on or analyzed before it is fully loaded (such as decryption or a checksum calculation). By using an intermediate location, such situations are simplified and are more efficient when loading to off-chip memories.

In some cases, a boot block does not fit into temporary storage memory. Having a larger buffer can improve boot performance. If an entire block cannot fit into the buffer, it is processed in pieces. Initialization code or callback functions can alter the temporary buffer region, including its location and size, by modifying the `ADI_ROM_BOOT_CONFIG::pTempBuffer` and `ADI_ROM_BOOT_CONFIG::dTempByteCount` variables in the struct ADI_ROM_BOOT_CONFIG structure.

Table 47-38: Flags

Flag	Required Value	Init
`BFLAG_INDIRECT`	Y	1
`BFLAG_CALLBACK`	N	Defines a callback function to operate on intermediate data. These 2 flags are often used together.

Ignore Block

It is a block that is ignored by the loader stream (in several cases). These blocks are useful when it is not possible to pass information in another block header. For example, if the first block contains data such as a firmware revision rather than application code, then `BFLAG_IGNORE` can be set with the correct application start address. This step ensures that the loader stream uses the correct start address. Since this block has no other function, identify it as an ignore block. Then, the kernel does not attempt to process any payload.

Ignore blocks clear the following flags, disabling the corresponding blocks from being processed if set along with BFLAG_IGNORE:

- BFLAG_INIT
- BFLAG_CALLBACK
- BFLAG_FINAL
- BFLAG_AUX

NOTE: When BFLAG_IGNORE is set along with BFLAG_FIRST, only the payload associated with the first block is ignored. The application entry point retrieved from the first block is always processed.

Table 47-39: Flags

Flag	Required Value	Init
BFLAG_IGNORE	Y	1
BYTE_COUNT	Y	Size of block to ignore; can be zero

Fill Block

A fill block instructs the boot kernel to perform a 32-bit memory fill of the memory region. Fill blocks help minimize the size of a boot stream when an application contains large arrays of data that need are initialized at startup. Zero fill is the most common fill block type. However, any 32-bit fill value is supported.

Table 47-40: Flags

Flag	Required Value	Init
BFLAG_FILL	Y	1
TARGET_ADDRESS	Y	Address where payload is loaded (must be valid)
BYTE_COUNT	Y	Size of block in bytes (must be multiple of 4)
ARGUMENT	Y	32-bit fill value

Init Block

An initialization block instructs the boot kernel to perform a function call to the target address after the entire block has loaded. The function called is referred to as the *initialization code (Initcode) routine*. Refer to the API reference initcode() for full details.

If the Initcode routine has been previously loaded, the block can declare a zero-size and have no payload.

Initcode routines can be used to speed up and customize booting mechanisms exposed by the boot kernel. Traditionally, an Initcode routine is used to set up the system PLL, bit rates, wait states, and the external memory controllers. Boot time can be significantly reduced when an init block is executed early in the boot process.

Initcode routines are required to follow the C language calling conventions. The expected C prototype is:

```
void initcode(ADI_ROM_BOOT_CONFIG * pBootConfig)
```

NOTE: When programming in assembly, use a return from subroutine instruction for returns.

The structure provided to the Initcode routine by the boot kernel contains information about the block being processed. It includes header information, locations of temporary block data (for indirect blocks), target address, and byte count. See struct ADI_ROM_BOOT_CONFIG for a full list and details on the provided data.

In the simplest case, an Initcode routine consists of only a single block in which the `BFLAG_INIT` flag is set. For larger routines, a sequence of blocks can incrementally load the routine, and only the last block sets the `BFLAG_INIT` flag. In the latter case, the last block has no payload attached, and simply instructs the boot kernel to issue a call to subroutine instruction.

An Initcode routine can be overwritten by a successive block when it is no longer needed. Otherwise, the routine can be called at multiple points during the boot process, and even remain in memory after the application completes booting.

NOTE: The following list provides requirements for Initcode that is written in C or C++.

- Ensure that the Initcode routine does not contain calls to the run-time libraries
- Do not assume that parts of the run-time environment, such as the heap, are fully functional
- Ensure that all run-time components are loaded and initialized before the routine executes

The loader utility and tool set provide mechanisms to aid in implementing initialization codes and organizing them properly within the boot loader stream. A special project type is provided to allow the creation of Initcode routines as separate projects. Options are available to assign particular pieces of the application to be Initcode routines. For details and more information on the utility, see the *Loader and Utilities Manual*.

Table 47-41: Flags

Flag	Required Value	Init
`BFLAG_INIT`	Y	1
`TARGET_ADDRESS`	Y	Location to load payload data. Call to subroutine issued to the same location.
`ARGUMENT`	N	Can be used to supply block specific arguments
`BYTE_COUNT`	Y	Size of payload; can be zero

NOTE: Init blocks cause software not located in the boot ROM during the boot process to execute. In the case of a secure boot scenario, initcode routines are not supported. The secure boot authentication process is performed at the end of the boot process. Execution of any user software prior to the authentication process violates the secure boot requirements.

Callback Block

It instructs the boot kernel to call a pre-registered function when the payload of the block is loaded. The purpose of a callback routine is to apply standard processing to the block payload. The callback routine is registered through an Initcode routine prior to loading a block using the routine. Typically, callback routines contain checksum, decryption, decompression or hash algorithms.

To register a callback, create an Init Block whose Initcode modifies the `ADI_ROM_BOOT_CONFIG::pCallBackFunction` with the address of the callback function to execute. A callback function must be registered prior to processing a callback block.

As callback routines require access to the payload data of the boot blocks, load the block data before processing. Often an Indirect Block is used in combination with a callback block. Indirect blocks in combination with callback blocks allow for post processing of the loaded data before it is then transferred to the final destination.

Callback routines are expected to meet the C language calling conventions. The prototype is as follows:

```
ROM_BOOT_RESULT callback(
      ADI_ROM_BOOT_CONFIG * pBootConfig,
      ADI_ROM_BOOT_BUFFER * pBuffer,
      uint32_t nFlags
      )
```

The `pBootConfig` argument contains a pointer to the struct ADI_ROM_BOOT_CONFIG information and `pBuffer` provides access to the address and size of the block (can vary when using indirect). The `nFlags` parameter is specifically used when the `BFLAG_INDIRECT` flag is also used. The `CBFLAG_DIRECT` flag indicates that the `BFLAG_INDIRECT` bit is not active so that the program only calls the callback routine once per block. When the `CBFLAG_DIRECT` flag is set, the `CBFLAG_FIRST` and `CBFLAG_FINAL` flags are also set.

NOTE: Callback blocks result in execution of software that is not located in the boot ROM during the boot process. In the case of a secure boot scenario callback routines are not supported because the secure boot authentication process is performed at the end of the boot process and execution of any program prior to the authentication process violates the secure boot requirements.

Callback Block in Conjunction with Indirect Block

A block using a callback routine is also loaded indirectly and there are slight behavioral differences. The procedure for loading is:

1. Load data into the temporary buffer defined by `ADI_ROM_BOOT_CONFIG::pTempBuffer`.
2. Issue a call to `ADI_ROM_BOOT_CONFIG::pCallBackFunction`.
3. After the callback routine returns and when the return value is zero, the memory DMA copies data to the destination.

When a block does not fit entirely into the temporary buffer, loading is performed similar to indirect blocks. The software calls the callback function after each chunk is loaded into the temporary storage. The `nFlags` parameter provides information on the specific iteration.

When a block does not fit entirely into the temporary storage area, nFlags tells the callback routine whether it is invoked for the first time (CBFLAG_FIRST) or called the last time (CBFLAG_FINAL) for a specific block.

When the software invokes DMA to copy the data, it relies on the supplied pBuffer data, not the ADI_ROM_BOOT_CONFIG::pTempBuffer and ADI_ROM_BOOT_CONFIG::dTempByteCount members of the boot structure.

The callback routine can control the source of the memory DMA by altering the content of the pBuffer structure. This alteration is necessary when the callback routine performs data manipulation such as decompression.

When the software uses an indirect block, the return value of the callback routine determines whether the DMA transfer occurs. If the value is non-zero, then the transfer does not occur.

Table 47-42: Flags

Flag	Required Value	Init
BFLAG_CALLBACK	Y	1

NOTE: Callback blocks cause software not located in the boot ROM during the boot process to execute. In the case of a secure boot scenario callback routines are not supported because the secure boot authentication process is performed at the end of the boot process and execution of any user software prior to the authentication process violates the secure boot requirements.

Save Block

A save block is intended to mark blocks in a boot stream for saving to off-chip memory. The on-chip boot kernel does not use this flag. A program can process the boot steam to find address of regions of memory for saving to external memory. On a reboot the program may then restore the previously saved contents. This block provides a means of doing a context restore after a reboot.

Table 47-43: Flags

Flag	Required Value	Init
BFLAG_SAVE	Y	1

Single-Block Boot Streams

The simplest boot stream consists of a single block header and one contiguous block of instructions and optionally data. When the appropriate flags are set in the block header, the kernel loads the block to the target address, terminates the boot process, and begins executing from the entry address of the application.

The ***Initial Header for Single-Block Stream*** table shows an example of a single-block boot stream header with settings that can be loaded using any boot mode. The BFLAG_FIRST and BFLAG_FINAL flags are both set at the same time. The desired location and size of the application determines the target address and byte count.

When using single block boot streams on products with multiple cores, the boot stream must be targeted towards the primary booting core that manages the boot process. If core 0 is the primary booting core, the boot stream must contain code that is intended for execution by that core. It is possible to boot a single block boot stream when using the API to load an application to the non-booting core. In this case, the BFLAG_RETURN flag must be set so the boot process returns to user application on the core controlling the boot process.

Table 47-44: Initial Header for Single-Block Stream

Field	Description of Value
BLOCK_CODE	0xAD000000|XORSUM|BFLAG_FINAL|BFLAG_FIRST
TARGET_ADDRESS	Start address of block
BYTE_COUNT	Number of bytes in the block
ARGUMENT	Functions as next-application pointer in multi-application boot streams.

Direct Code Execution

Applications can avoid long boot times and execute code from flash or SDRAM memory with minimal processing by the boot kernel. This feature is called direct code execution.

An initial boot block header is required for the processor to fetch and execute program code from the boot device as early as possible. The block uses safety mechanisms to avoid unpredictable processor behavior when boot memory is not yet programmed with valid data. Safety mechanisms include the header signature and the byte-wise XOR checksum. Rather than blindly executing code, the boot kernel first executes the pre-boot routine for system customization. It then loads the first block header and checks it for consistency. If the block header is corrupt, the boot kernel calls the error handler and does not start code execution.

If the initial block header check is good, the boot kernel interrogates the block flags. If the BFLAG_FINAL flag is set, the boot kernel terminates and executes the sequence as described in the Boot Termination and Application Execution section. If the application requires that the boot kernel customize the starting address in advance, the first block must also have the BFLAG_FIRST flag set. The target address field is then saved as the application start address.

When processing direct code execution blocks, the block instructs the processor executing the boot code to execute from the address specified. It is not possible to have core 0 boot the block and have it instruct core 1 to immediately start execution from the address provided.

For example, when the block header described in the ***Direct Code Execution Header*** table is placed at address 0x20000000, the boot kernel is instructed to issue a JUMP command to address 0x20000020.

Table 47-45: Example Direct Code Execution Header

Field	Value	Comments
BLOCK_CODE	0xAD7BD006	0xAD000000|XORSUM|BFLAG_FINAL|BFLAG_FIRST| BFLAG_IGNORE|BCODE

Table 47-45: Example Direct Code Execution Header (Continued)

Field	Value	Comments
TARGET_ADDRESS	0x20000020	Start address of application code. Provided as an example this would be dependent upon the start address required for a given product.
BYTE_COUNT	0x00000010	Ignores 16 bytes to provide space for control data such as version code and build data. This field is optional and can be zero. The payload is ignored due to the `BFLAG_IGNORE` flag being set.
ARGUMENT	0x00000010	Functions as next-application pointer in multi-application boot streams.

Multi-Application Boot Streams

This section describes loader streams that contain multiple applications.

A boot stream is typically generated from an application file. Loader streams with more than one application are commonly referred to as multi-application booting. A loader utility often accepts multiple application files as input parameters and generates a contiguous boot image. The subsequent applications are appended to the first.

The loader utility must also update the argument field of all `BFLAG_FIRST` blocks. The argument field of a `BFLAG_FINAL` block is called the *next-application* pointer.

The next-application pointer of the first application boot stream is a relative pointer to the start address of the second application boot stream. A multi-application boot image can be seen as a linked list of boot streams. The next-application pointer of the last application boot stream is a relative pointer to the next free address. The ***Multi-Application Boot Stream Example*** figure shows an example.

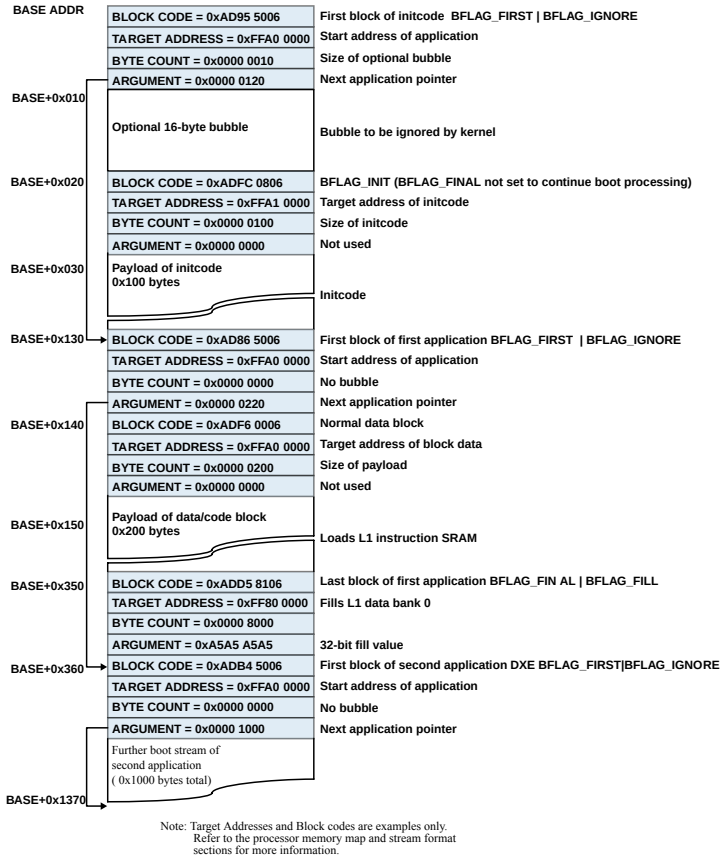


Figure 47-18: Multi-Application Boot Stream Example

The ***Multi-Application Direct Code Execution Example*** figure shows a linked list of initial block headers. The blocks instruct the boot kernel to terminate immediately and to start code execution at the address provided by the target address field of the individual blocks. There is nothing in the boot code that prevents multi-application streams from mixing regular boot streams and direct code execution blocks.

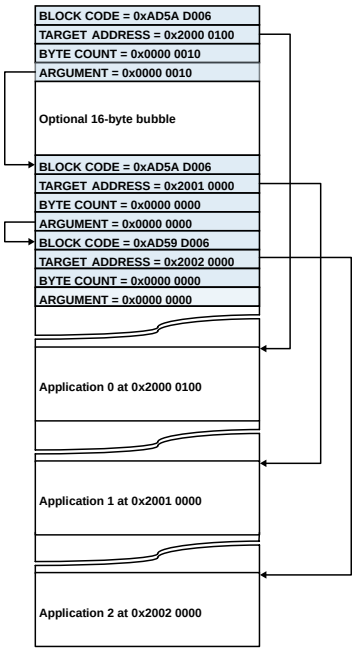


Figure 47-19: Multi-Application Direct Code Execution Example

CRC32 Protection

The boot kernel has mechanisms that allow verification of each block's payload using a 32-bit CRC. The boot ROM contains a function that can be called as an initcode to register the CRC callback and initialize the CRC peripheral with a specified polynomial. CRC32 protected loader streams are supported by the programming tools utlities.

To enable this feature an Block Types must be located in the boot stream with a `TARGET_ADDRESS` that points to the adi_rom_Crc32Init() function in the ROM. The `ARGUMENT` field contains the CRC32 checksum polynomial that is used to initialize the CRC lookup table. Once the CRC initcode function in the ROM has been executed, CRC verification is enabled for all subsequent blocks except:

- Ignore
- First

NOTE: CRC functionality is dependent upon the use of an Initblock and is not supported in secure booting.

Secure Boot

The secure boot process provides a way to integrate security in the processor boot sequence. A chain of trust is established within the system by ensuring the integrity and authenticity of the boot image. Confidentially is also supported.

Secure boot increases protection against malicious, unsecured accesses to critical and confidential resources of the processor. The boot stream application code and data must be digitally signed in order to build up a chain of trust in

the system. This requirement allows the processor to distinguish between authentic and trusted code from non-authentic and untrusted code.

Secure boot also provides confidentiality support. The digitally signed boot image can be optionally encrypted. When loading an encrypted image, the ROM decrypts while loading, then authenticates, before any application code is executed.

Secure boot is an optional feature of the processor that is disabled by default. The feature is enabled using the OTP lock API; secure boot cannot be disabled after it has been enabled. When security is enabled, developers are not dependent on Analog Devices to provision the devices, sign code or provide security certificates. The required tools for signing and encrypting the boot images are provided with the processor development tools.

Integrity and Authenticity Protection

Integrity protection is based on the secure hash SHA-2 224/256-bit algorithm. Authenticity protection is based on the ECDSA algorithm. The ADSP-SC59x Boot ROM uses P-224 or P-256 curves for the implementation of the ECDSA-224/256 algorithm to exercise secure boot.

ECDSA uses public key cryptography consisting of two keys: a private key and a public key. The public key is stored in OTP memory on the processor so that the secure boot process can verify the authenticity of the signed boot image. Only parties in possession of the private key are able to sign the images.

Confidentiality Protection

Confidentiality protection uses the AES algorithm. Two variants are supported: wrapped and unwrapped.

The wrapped variant uses a 128-bit Key Encryption Key (KEK) stored on the processor to decrypt the 128-bit AES decryption key embedded in the secure header. The unwrapped variant stores the AES description key on the processor and utilizes it to decrypt the entire image.

The privacy of the key stored on the device (whether AES or KEK) is paramount to the security of the system. Disclosure of this key compromises security of the entire system.

Anti-Cloning Protection

Anti-cloning protection is based on the confidentiality protection. If each processor in a system uses a unique private key for confidentiality protection, then cloning between these devices can be prevented. The boot image is incompatible with devices using a different private key for decryption.

Anti-Rollback Protection

The secure boot process supports anti-rollback protection using a 32-bit counter in OTP memory. A value of `0x00000000` in the OTP makes anti-rollback disabled by default. If anti-rollback protection is required, then the program can set the rollback ID when signing the boot image. When the boot image is authenticated, the secure boot software updates the counter in OTP (if the rollback ID in the boot image is greater than the value currently stored in the OTP counter).

The rollback ID stored in the secure boot image header is integrity-protected and cannot be altered.

NOTE: To enable anti-rollback protection for secure boot operations, write a non-zero value to the 32-bit counter in OTP memory. When the counter register remains at the default value of zero, anti-rollback protections are not enabled, regardless of the rollback ID located in a secure boot stream.

CAUTION: There are a number of restrictions for the rollback ID in the OTP module. Programs should only use the OTP boot program ROM API to set the counter. Refer to the OTP counters section for information on the implementation strategy.

Terminology

ECDSA

Elliptical Curve Digital Signature Algorithm

BLp

Boot Loader plain text, Plaintext Format

BLx

Boot Loader without key, Keyless Format

BLw

Boot Loader wrapped, Wrapped Format

BLe

Boot Loader encrypted, Encrypted Format

SBLS

Secure Boot Loader Stream

SBH

Secure Boot Header

SBCR

Secure Boot Confidentiality Root

AES

Advanced Encryption Standard

Signing for Secure Boot Images

All boot images must be digitally signed to create secure boot images. The boot image is processed by the security utilities included with the development tools to sign and optionally encrypt the boot image. The security utilities operate with key-pairs consisting of a private and a public key. The private key is used for signing the images, and the public key is used to validate an image being loaded into the processor.

CAUTION: The private key generated from the signing utility, used for signing images, is never required by the processor for successful secure boot. The private key is only ever required by the signing utility and should be made available only within the system responsible for the image signing process.

The image signing utility provides the following functionality:

- Signing and encrypting of images
- Generation of ECDSA key pairs
- Generation of random encryption keys
- Extraction of the public key from an ECDSA key pair
- Setting Secure Boot Image Attributes

For more information on the use of the signing utility, refer to the Loader and Utilities manual.

Secure Boot Image Types

This section provides an overview of the different image types and how to use them.

Plaintext (BLp) Format

The BLp format provides integrity plus authentication protection of the boot image. The boot image is produced using 224-bit or 256-bit Elliptical Curve Digital Signature Algorithm (ECDSA) private key. To authenticate the image, program the corresponding public key into the OTP `public_key` field using the OTP boot program API.

SBH	Boot Loader Stream

Wrapped (BLw) Format

The BLw format provides the highest level of protection: integrity plus authentication, confidentiality, and anti-cloning protection. The image contains an AES wrapped image encryption key (denoted by [K]) within the secure header. The image data is encrypted with the wrapped key, preventing cloning. An extra key is required to unwrap the header; program this key into the OTP `pvt_128key` field using the OTP boot program API.

SBH [K]	Encrypted Boot Loader Stream

Keyless (BLx) Format

The BLx format is similar to the BLw format except that the image does not contain the key at all. This format provides anti-cloning protection only if the secure key is unique per device. Program the decryption key for the data into OTP `pvt_128key` field using the OTP boot program API.

SBH	Encrypted Boot Loader Stream

Secure Boot Image Format

Secure Boot images provide authenticity and integrity protection during the boot process. A secure boot image is comprised of a secure boot header and an optionally encrypted loader stream.

Signed images consist of the following sections to comprise a complete secure boot image:

- Secure Boot Header
- Image Attributes
- Image Section

The Figure 47-20 Secure Boot Image shows that the image attributes are encapsulated within the secure boot header. The image attributes are actually integrity protected along with the image section. The image section contains a standard boot loader stream. Some block types are not allowed as described in Unsupported Boot Stream Blocks.

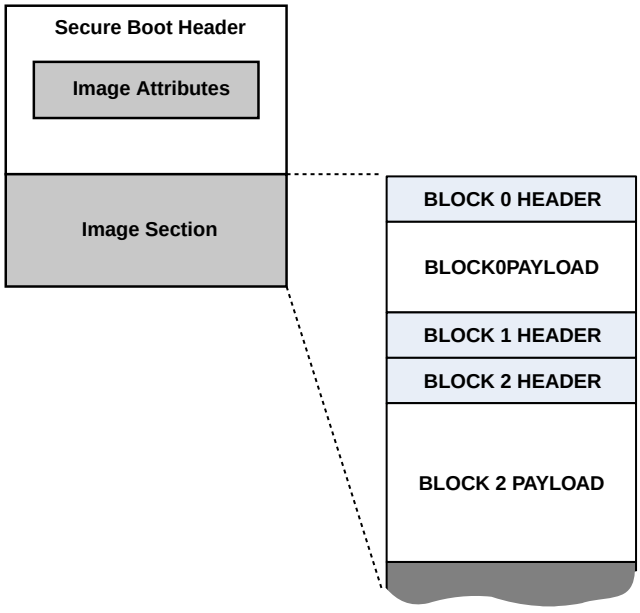


Figure 47-20: Secure Boot Image

Secure Boot Header

Table 47-46: Secure Boot Header

Bytes	Name	Description	Values		
			Keyless Format BLx	Wrapped Format BLw	Plain Text Format BLp
3:0	Type	Format and version of the image. Upper 24 bits are the image format and lower 8 bits are the image version	0x424c7803	0x424c7703	0x424c7003
67:4	Signature	The ECDSA signature of the image	Two 256-bit numbers		
99:68	Hash	Hash of the image	Hash size depends upon whether ECDSA 224 or ECDSA 256 is used. 1. For ECDSA 224 Hash Value: 95:68 bytes 2. For ECDSA 256 Hash Value: 99:68 bytes		
123:100	Key	Confidentiality (only applicable for certain formats)	Reserved	192-bit AES-WRAP data holding a 128-bit AES key	Reserved
139:124	IV	Initialization Vector (only applicable for certain formats)	16-bit IV generated during signing process		Reserved
143:140	Length	The length of the image section in bytes	Maximum supported byte count 0x10000000 bytes		
207:144	Attributes	Image attributes	Support for up to 8 image attributes		
255 :208	Reserved				

Secure Boot Processing Overview

The ***Secure Boot Processing*** figure illustrates how the block is processed using secure features (not all block header type details are shown). For details on the various block types and their function, see Block Types. Some image types are decrypted. Decrypt indicates that the data is decrypted when applicable to the Secure Boot Image Types at that particular stage.

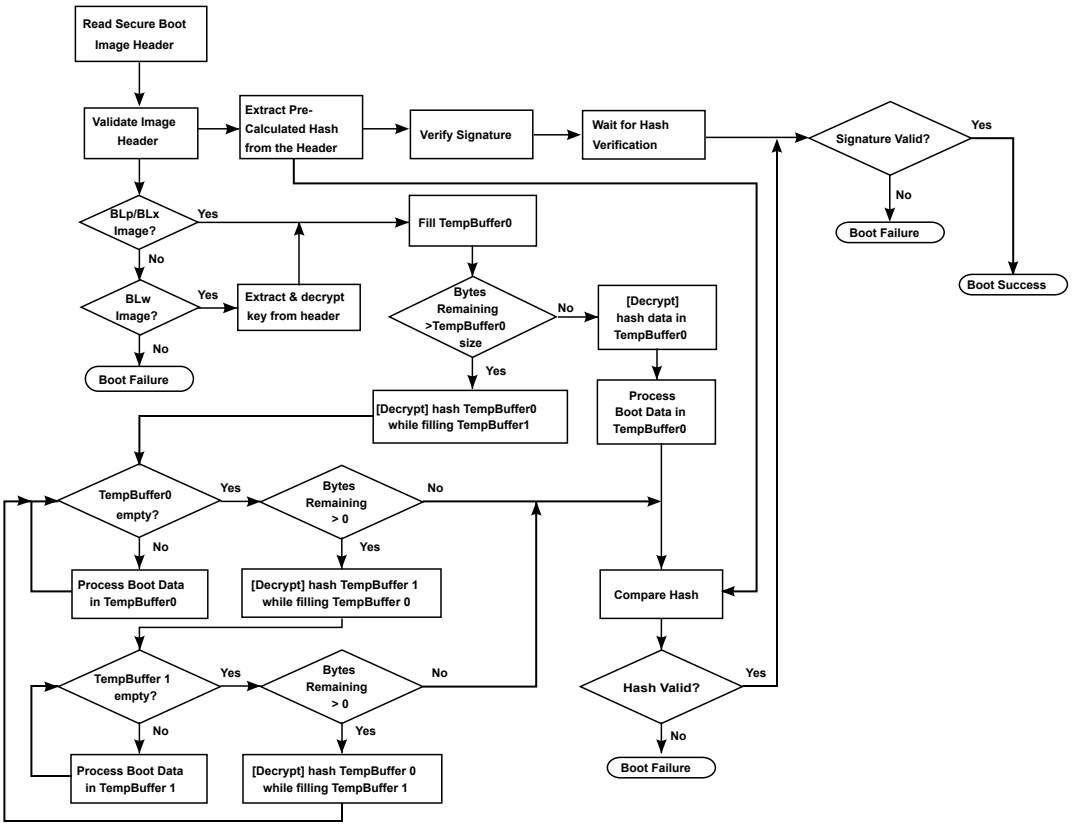


Figure 47-21: Secure Boot Processing

Pre-Hash Authentication

Boot ROM in the ADSP-SC59x family uses the pre-calculated hash (SHA-224 or SHA-256) stored in the secured header to start the authentication process earlier and thereby reduce the boot rom. With pre-calculated hash of the whole boot image available, authentication starts in parallel with rest of the boot process. Since the boot uses the pre-calculate hash for authentication, the last step is to verify that the hash calculated by the boot ROM from the actual image read from the boot source matches the pre-calculated hash. If the hash calculated by boot ROM does not match the pre-calculates hash, the result of ECDSA authentication is ignored and the secure boot fails.

However, if the hash calculated by boot ROM matches the pre-calculated hash, the result of ECDSA authentication is used to decide whether the secure boot happened successfully or not.

Unsupported Boot Stream Blocks

To ensure the security of the processor, the following block types are not supported in a secure boot image. If the boot kernel finds one of these block types, the boot process terminates.

- Init Block: It require a call to the application code prior to the authentication of the boot image. If customizations or optimizations are necessary to improve the load performance, use a second stage loader style implementation. The first application contains only the custom code. Issue a call using the `api_boot ()` routine to boot using the desired device.

- Call Block: It require a call to a user-defined address prior to the authentication of the boot image, and therefore cannot be supported.

NOTE: Secure boot streams use double buffer Page Mode to optimize the boot process. This functionality allows for the performance of decrypt and hash operations on received data while new data is fetched from the boot source. This host in slave boot mode must ensure that more data is sent after the boot stream to ensure that the temp buffer is filled completely. The size of the secure boot stream minus the size of the secure boot header must be a multiple of the size of the temp buffer. The temp buffer default size is 1024 bytes.

Secure Boot Image Attributes

Secure boot image attributes form part of the secure boot header. The attributes provide more information about the content of the secure boot image.

All image attributes are integrity protected using the same algorithm as the image section. When the image authentication process completes and the image is successfully authenticated, the image attributes are known to be trustworthy.

Attributes are specified as type value pairs with both the type and value being a 32-bit value. The boot code supports the following image attributes.

Table 47-47: Secure Boot Image Attributes

ID	Name	Description	Values	
0x00000000	Unused	Unused attribute	0x00000000	
0x00000001	Version	Version of the attribute format	0x00000000	
0x00000002	Rollback ID	Current value of the rollback counter	0x00000000	Rollback disabled
			0x00000001 - 0x0000001F	Current firmware revision. It must be greater than or equal to the value retrieved from OTP.
0x80000001	NoRestore	Controls whether the boot process restored registers back to default values and clears sensitive security related information from the stack and dedicated structures.	0x1	Do not restore registers and clear security information from stack and dedicated structures.
			Other	Restore registers and clear security sensitive information from the stack and dedicated structures.
0x80000002	BCODE	Used by the boot mode drivers that support autodetection to configure the	Values between 0x00000000 and 0x0000000F supported.	

Table 47-47: Secure Boot Image Attributes (Continued)

ID	Name	Description	Values
		device from a range of preconfigured settings	
0x80000003	ECDSA Type	Used by boot ROM to perform Authentication using ECDSA-224 or ECDSA-256 algorithm.	224 : Authentication using ECDSA-224 algorithm 256 : Authentication using ECDSA-256 algorithm

Secure Boot Streams

The secure boot image sections contain the code and data for loading to the various memory regions of the processor. The content is in the form of a boot stream format consisting of block headers that provide a descriptor for the associated block payloads.

The image section within the secure boot stream consists of a standard Boot Loader Stream consisting of block headers and payloads as generated by the supporting tools and utilities.

The secure boot image section can contain application images for just a single core or for multiple cores allowing for a flexible booting strategy.

A single boot image containing program and data for multiple cores permits a full system boot initializing all internal memories without needing a second stage loader approach. If boot images are required to initialize external memories, then a multi-stage loader approach can be required to configure the memory interfaces. The advantage of a single boot image for multiple cores is that authentication and decryption of the boot image requires only one execution. However, it is over a larger boot image.

Multiple single-core boot images permit one core to boot then execute that core's boot image without booting the other cores. This booting strategy can be needed when a single processor must be brought up quickly to deal with some initial tasks before booting the rest of the system. Or, it can be used to initialize extra peripherals to use for external memory interfaces. The core that has previously booted can then control the boot process for the remaining cores.

Single Core Images

Single core boot images are the result of processing a single core's output from the linker and converting it to a compliant boot stream. The resulting boot stream has a single first block at the beginning of the boot stream and a final block at the end of the boot stream.

The Block Header `HDRSIGN` field of the resulting boot stream is used to identify which core the image is intended for. This identification is required so that the boot code can set the correct (`RCU_SVECT0` - `RCU_SVECT2`) register from the target address in the first block once the authentication is successful. When the boot image is loaded and authentication is successful, the boot code jumps to the location stored in the cores corresponding (`RCU_SVECT0` - `RCU_SVECT2`) register.

This boot stream must then be signed using the secure boot utilities resulting in the final secure boot image for a single core.

A second stage loader option is also available giving maximum flexibility both in terms of using ROM functionality, or creating a custom booting strategy. The simplest option is to have the application use `adi_rom_Boot()` to boot the main application image or indeed a third stage loader when one is required.

The boot stream is generated such that all image contents are in the system address space. Core specific L1 memory sections are converted during the boot stream generation process such that they are loaded through the multiprocessor memory space. The core executing the second stage loader can boot an application image intended for another core.

NOTE: The `ROM_BFLAG_RETURN` flag is set when calling the boot routine for the other cores than the core executing the second stage loader then cleared when loading the cores own image. Failure to do so would result in unintentional behavior.

When the core is used to boot a separately generated boot stream for another core, the `ROM_BFLAG_RETURN` flag should be used when calling `adi_rom_Boot()` to instruct the boot kernel to return to the calling application, which in this case is back to the core that was running the second stage loader. Failure to set this flag results in the core vectoring to the location in its own (`RCU_SVECT0` - `RCU_SVECT2`) register which is not the intention.

By default when implementing a scheme such as previously described where one processor is responsible for booting images for another core, the other cores in the system will remain in their existing idle state that they should be placed in prior to the boot commencing. To allow the cores to execute new application, the core that is responsible for booting the other cores must reset the cores via the `RCU_CRCTL` then release them again from reset. If there is a requirement to release the core from the reset state, then this must be done within the application code of the running core.

Releasing other cores from reset to run application software while other processors are running a boot process requires careful system design. The drivers used by the boot kernel for the boot process assume the various peripheral and infrastructure resources such as MDMA channels and peripherals are available for use, and are not being used by another core. If the boot routine is being executed while other cores are running applications, then those applications must ensure that all the required boot resources are freed up and remain free for the boot process to complete on the remaining cores.

Multi-Core Boot Images

Multi-core boot images are generated as a result or processing the multiple linker output files for multiple cores together to create a single compliant boot stream. The resulting boot stream has a first header block at the beginning of each core boot stream and a single final block at the end of the boot stream. The boot stream must only have a single final block to allow the boot kernel to continue processing the entire boot stream. A final block results in the boot kernel terminating triggering the final public key authentication sequence.

The block headers block code `HDRSIGN` field of the resulting boot stream is used to identify the intended core for the image. This identification is required such that the boot code can set the correct (`RCU_SVECT0` - `RCU_SVECT2`) register when the first block is read to set the application start address for that core. When the boot image is loaded and authentication is successful, the core executing the boot sequence jumps to the location stored in the cores corresponding (`RCU_SVECT0` - `RCU_SVECT2`) register. However, since the boot stream has multiple first blocks present for each of the cores, the (`RCU_SVECT0` - `RCU_SVECT2`) register of the other cores is set for

their applications. Upon the booting core running its application, it must release the other cores from reset for them to start running their loaded applications.

When a product supports three or more cores, it is acceptable to create a dual core boot image and then load the other cores later using single core boot images stored elsewhere in the boot source. There are no restrictions on a multi-core boot stream that must contain an application for all the cores in a product.

The resulting multi-core boot stream must then be signed and optionally encrypted resulting in a complaint secure boot stream that can be used to boot a secure device.

Multiple core boot images are advantageous in the fact that applications can be loaded to all cores in the system in a single boot sequence resulting in the requirement to decrypt and authenticate only a single boot image.

NOTE:

If there is a requirement for a multi-core boot stream to load code to memory spaces that require a memory controller to be initialized that is not supported by the boot process a multi-stage booting approach is required that can initialize the peripheral. This boot stream is then authenticated, and executed prior to the loading of the boot stream.

Secure Debug Access

The TAPC controller provides a way to restrict access to secure resources of the processor. Secure access through the debug port is protected through a 128-bit security key that must match a key that has been loaded into OTP for access.

To access a locked processor, the TAPC must allow access to the part. The TAPC only allows access to the part if it is provided with a matching key to the data loaded into its `TAPC_USERKEYn` registers.

With the processor in a locked state, on initial boot the boot ROM reads either the 128-bit `secure_emu_key0` or the 128-bit `secure_emu_key1` from the OTP memory and programs the key into the `TAPC_SDBGKEY0` through the `TAPC_SDBGKEY3` registers before then setting the `TAPC_USERKEY_CTL.USERKEY_VALID` bit. The TAPC is then able to access a matching outside key to allow access. See Secure Debug Key Processing for more information about secure debug key validation.

CAUTION: A 16 bit non-zero value programmed on both `emu_key0_disable` and `emu_key1_disable` fields in OTP or a key of 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF programmed in both the secure debug key fields provided in OTP results in the boot code bypassing the key load operation entirely. When debug access is ever required, the key must be loaded to the TAPC by the program. When the processor fails to boot (due to corrupted firmware) then there is no debug access. The only way to gain access is to load an authenticated boot image that can then load the required keys prior to attempting to connect with a debugger.

The key is set in OTP using the OTP boot program API to program `secure_emu_key0`/ `secure_emu_key1` . The key is read and loaded by the ROM in the following sequence:

1. Bits 31:0 of the key in OTP are stored to `TAPC_SDBGKEY0` bits 31:0

2. Bits 63:32 of the key in OTP are stored to TAPC_SDBGKEY1 bits 31:0
3. Bits 95:64 of the key in OTP are stored to TAPC_SDBGKEY2 bits 31:0
4. Bits 127:96 of the key in OTP are stored to TAPC_SDBGKEY3 bits 31:0

Once the ROM has loaded the user key, a test key can be provided to the TAPC through JTAG. Refer to the Emulator manual for details for providing the key.

A key failure indication can be detected through the TAPC_SDBGKEY_STAT register. The boot code does not check the key status, nor does it enable any associated interrupts to signal key failure. The boot code continues to boot upon a key failure in a secure manner. The key failure status remains intact so that the application loaded can check for a failed challenge on the debug port.

The boot code can be configured to bypass the loading of the key during the boot sequence by setting the value of secure_emu_key0 or secure_emu_key1 in the OTP to all ones (see struct OTP_DATA). In this case, the only way to gain access to the secure resources through the debug port is to load an alternate key using the application. The alternative key must always reside in a secure region of memory. Or, if sent remotely, it should be transmitted over a secure connection.

NOTE: In the ADSP-SC59x processor family, the boot ROM supports two 128-bit secure debug keys, secure_emu_key0 and secure_emu_key1, where only one of them is used by the Boot ROM at a time. Each of the secure debug keys is associated with a bypass key (emu_key0_disable or emu_key1_disable) which decides the validity of the corresponding key. Any non-zero value written into the 16-bit bypass key invalidates that secure debug key.

Failure Analysis

The boot ROM supports failure analysis on locked parts. Calling adi_rom_otp_fa_enable() activates this feature. The user must activate this feature on a locked part before sending it for failure analysis. Without activation, failure analysis on a locked part is impossible due to security reasons. Do not enable this feature on open parts because it results in boot failure.

Anti -rollback NV Counter

Use an anti-rollback counter to prevent loading of older firmware during secure boot. Boot ROM implements the counter using the OTPC module to allow easy reading or writing of the counter, without dealing with the complexities of rewriting OTP memory sections that are ECC protected.

The implementation involves two functional APIs for the counter, which are functionally unique in the way that they set and retrieve data as counters in OTP memory.

It uses a different method to count bits because each bit in OTP memory can only be set =1 once, and the ECC protects each 16-bit unit. This functionality essentially means that each 16-bit unit can only be written to once. Therefore, a counter that can count 0–31 requires 32 × 16 bits of memory.

The counter API receives and returns the value of the counter as a uint8_t binary number. Providing a value less than the current value of the counter or greater than the maximum value results in an error. A greater value will result in updating the counter value equal to the one provided.

To implement this functionality, the driver counts by shifting 1's from the right, treating each block as 1 bit. A three-bit counter is encoded as follows.

Bit 2	Bit 1	Bit 0	Value
0000	0000	0000	0
0000	0000	0001	1
0000	0001	0001	2
0001	0001	0001	3

Boot ROM Errors and Failures

Any errors encountered while processing a secure boot image results in the ROM jumping to the Error Handler with a specific error code. The error can be of any boot specific error or product specific error. The below table shows different types of errors which are supported by the Boot ROM.

Warning: In secured booting, an incorrect key supplied during boot does not cause a boot failure and the processor continues to boot as normal. A program that supplies an incorrect key is unable to gain access to any secure resources of the processor.

Table 47-48: Secure Boot Header

Error Name	Error ID	Description
ROM_BOOT_ERR_GENERAL	0x0	General Boot Error
ROM_BOOT_ERR_UNSUPPORTED	0x1	Unsupported configuration (device/mode selection)
ROM_BOOT_ERR_BMODE_INIT	0x2	Initialization failure for the selected boot mode
ROM_BOOT_ERR_BMODE_CONFIG	0x3	Configuration failure for the selected boot mode
ROM_BOOT_ERR_BMODE_HOOK	0x4	Hook function failure
ROM_BOOT_ERR_BMODE_REG_FAILURE	0x5	Bmode registration failure
ROM_BOOT_ERR_SEC_AUTH_FAIL	0x6	Authentication failure
ROM_BOOT_ERR_SEC_DECRYPT	0x7	Decryption Failure
ROM_BOOT_ERR_ID	0x8	The image being loaded has a lower ID than was previously recorded, indicating image ID is out dated
ROM_BOOT_ERR_CLEANUP	0x9	Cleanup failure
ROM_BOOT_ERR_HDRSGN	0xA	Block header signature failure
ROM_BOOT_ERR_HOOK	0xB	Hook function failure
ROM_BOOT_ERR_HDRCHK	0xC	Header checksum failure

Table 47-48: Secure Boot Header (Continued)

Error Name	Error ID	Description
ROM_BOOT_SECURE_ERR_HDRCHK	0xD	Secure header failure
ROM_BOOT_ERR_NOFINAL	0xE	No final block
ROM_BOOT_ERR_BLKFILL	0xF	Failure to process fill block
ROM_BOOT_ERR_BLKCRC	0x10	Failed crc for a block
ROM_BOOT_ERR_DMA	0x12	DMA transfer failure
ROM_BOOT_ERR_CALLBACK	0x13	Callback failure
ROM_BOOT_ERR_NEXTDXE	0x14	Next dxe failure
ROM_BOOT_ERR_PLLCFG	0x15	PLL configuration failure
ROM_BOOT_ERR_INT	0x16	Internal ROM error
ROM_BOOT_ERR_NOKERNEL	0x17	Application returned to the bootrom
ROM_BOOT_ERR_OTPREAD	0x18	Failure to read OTP
ROM_BOOT_ERR_NODEVICE	0x19	No device selected in the bootCommand
ROM_BOOT_ERR_NOBMODE	0x1A	No boot mode selected
ROM_BOOT_ERR_BMODEDIS	0x1B	Boot mode is disabled
ROM_BOOT_ERR_NOPUBKEY	0x1C	No public key
ROM_BOOT_ERR_NOPVTKEY	0x1D	All private keys have been disabled
ROM_BOOT_ERR_SPICFG	0x1E	Failed SPI configuration
ROM_BOOT_ERR_UARTCFG	0x1F	failed UART configuration
ROM_BOOT_ERR_SECURITY_FAILURE	0x20	Failure to initialize secure features
ROM_BOOT_ERR_ECDSATYPE	0x21	Wrong ECDSA Type detected other than 224 and 256
ROM_BOOT_ERR_OTPLOCKBIT	0x22	Error in OTP area holding the Lock and Guard bits
ROM_BOOT_ERR_INVALID_FAEN	0x23	Error due to Failure Analysis Enable bit being set in open part
ROM_BOOT_ERR_SEC_HASH_COMPARE_FAIL	0x24	Error due to hash compare fail between preHash and Computed Hash
ROM_BOOT_ERR_EMSI_LOAD_FAIL	0x25	EMSI Load Routine Failure
ROM_BOOT_ERR_A55_SYNC_EXCEPTION_SP_EL	0x26	Synchronous Exception at current EL level with SP0
ROM_BOOT_ERR_A55_IRQ_SP_EL	0x27	Normal priority interrupt at current EL level with SP0
ROM_BOOT_ERR_A55_FIQ_SP_EL	0x28	Fast priority interrupt at current EL level with SP0
ROM_BOOT_ERR_A55_SERROR_SP_EL	0x29	SError exception at current EL level with SP0
ROM_BOOT_ERR_A55_SYNC_EXCEPTION_SP_ELX	0x2A	Synchronous Exception at current EL level with SPx

Table 47-48: Secure Boot Header (Continued)

Error Name	Error ID	Description
ROM_BOOT_ERR_A55_IRQ_SP_ELX	0x2B	Normal priority interrupt at current EL level with SPx
ROM_BOOT_ERR_A55_FIQ_SP_ELX	0x2C	Fast priority interrupt at current EL level with SPx
ROM_BOOT_ERR_A55_SERROR_SP_ELX	0x2D	SError exception at current EL level with SPx
ROM_BOOT_ERR_A55_SYNC_EXCEP-TION_AARCH64	0x2E	Synchronous Exception at lower EL level using Aarch64
ROM_BOOT_ERR_A55_IRQ_AARCH64	0x2F	Normal priority interrupt at lower EL level using Aarch64
ROM_BOOT_ERR_A55_FIQ_AARCH64	0x30	Fast priority interrupt at lower EL level using Aarch64
ROM_BOOT_ERR_A55_SERROR_AARCH64	0x31	SError exception at lower EL level using Aarch64
ROM_BOOT_ERR_A55_SYNC_EXCEP-TION_AARCH32	0x32	Synchronous Exception at lower EL level using Aarch32
ROM_BOOT_ERR_A55_IRQ_AARCH32	0x33	Normal priority interrupt at lower EL level using Aarch32
ROM_BOOT_ERR_A55_FIQ_AARCH32	0x34	Fast priority interrupt at lower EL level using Aarch32
ROM_BOOT_ERR_A55_SERROR_AARCH32	0x35	SError exception at lower EL level using Aarch32
ROM_BOOT_ERR_SH_RCU_SVECT_IRQ	0x36	RCU SVECT Exception
ROM_BOOT_ERR_SH_PARI_IRQ	0x37	L1 Parity Error
ROM_BOOT_ERR_SH_ILOPI_IRQ	0x38	Illegal opcode error
ROM_BOOT_ERR_SH_CB7I_IRQ	0x39	Software stack (Circular Buffer 7) Overflow
ROM_BOOT_ERR_SH_IICDI_IRQ	0x3A	Unaligned LW/BW access + unintentional CMMR/ SMMR access
ROM_BOOT_ERR_SH_SOVFI_IRQ	0x3B	Status loop or mode stack overflow; or PC stack full
ROM_BOOT_ERR_SH_ILADI_IRQ	0x3C	Illegal Address Space detected
ROM_BOOT_ERR_SH_IIR2_IRQ	0x3D	IIR 2 error
ROM_BOOT_ERR_SH_IIR3_IRQ	0x3E	IIR 3 error
ROM_BOOT_ERR_SH_TMZHI_IRQ	0x3F	Core Timer error
ROM_BOOT_ERR_SH_BKPI_IRQ	0x40	User Hardware Breakpoint
ROM_BOOT_ERR_SH_FIR_IRQ	0x41	FIR error
ROM_BOOT_ERR_SH_IIR0_IRQ	0x42	IIR 0 error
ROM_BOOT_ERR_SH_SECI_IRQ	0x43	System event controller interrupt
ROM_BOOT_ERR_SH_IIR1_IRQ	0x44	IIR 1 error

Table 47-48: Secure Boot Header (Continued)

Error Name	Error ID	Description
ROM_BOOT_ERR_SH_DAI0SPT1	0x45	
ROM_BOOT_ERR_SH_DAI1SPT0	0x46	
ROM_BOOT_ERR_SH_DAI1SPT1	0x47	
ROM_BOOT_ERR_SH_RINSEQI_IRQ	0x48	Restricted Instruction Sequence
ROM_BOOT_ERR_SH_CB15I_IRQ	0x49	Circular Buffer 15 Overflow
ROM_BOOT_ERR_SH_TMZLI_IRQ	0x4A	Core Timer (Low Priority Option)
ROM_BOOT_ERR_SH_FIXI_IRQ	0x4B	Fixed-point overflow exception
ROM_BOOT_ERR_SH_FLTOI_IRQ	0x4C	Floating-point overflow exception
ROM_BOOT_ERR_SH_FLTUI_IRQ	0x4D	Floating-point underflow exception
ROM_BOOT_ERR_SH_FLTII_IRQ	0x4E	Floating-point invalid exception
ROM_BOOT_ERR_SH_EMULI_IRQ	0x4F	Emulator low priority interrupt
ROM_BOOT_ERR_SH_SFT0I_IRQ	0x50	User software interrupt 0
ROM_BOOT_ERR_SH_SFT1I_IRQ	0x51	User software interrupt 1
ROM_BOOT_ERR_SH_SFT2I_IRQ	0x52	User software interrupt 2
ROM_BOOT_ERR_SH_SFT3I_IRQ	0x53	User software interrupt 3
ROM_BOOT_ERR_OTPBOOT_IRQ	0x54	OTP Boot Exception
ROM_BOOT_ERR_OTP_INVALIDSTATE_IRQ	0x55	OTP Invalidate State Exception

Boot ROM Programming Model

This section describes the programming model for booting the processor. The programming model includes booting functions, API calls, and data structures.

Boot Mode Driver API

The kernel provides a mechanism to provide a customization of supported boot modes or for implementation of completely new boot modes as second stage boot loaders. This allows programs to customize booting while still taking advantage of the rest of the booting framework. A custom boot mode may provide support for a peripheral that is not supported for boot by the ROM, or it could support one of the same peripherals but with a different configuration.

All the same security features can be supported when using a custom boot mode.

A full boot mode, as perceived by the boot implementation, is a collection of following functions:

1. Register–installs the driver functions listed below so they can be accessed by the boot process
2. Initialization–initialize the boot source

3. Configuration–configure the boot source
4. Load–read from the boot source
5. Cleanup–called after booting

The boot kernel is only aware of functions 2 through 5 and has a requirement to support the Load function. It is this function that is responsible for fetching the boot stream from the boot peripheral. The other functions are used prior to executing the kernel or for cleaning up after the kernel has completed processing the boot stream.

To install a custom boot mode:

1. Create a first stage boot application to define a Load function
2. Use the adi_rom_BootKernel() API to call the boot kernel after the boot peripheral and pin muxing has configured. Ensure all the fields of struct ADI_ROM_BOOT_CONFIG are configured accordingly prior to performing the call.

The boot mode can use the `pModeData` member of `ADI_ROM_BOOT_CONFIG` to preserve and access shared data across the different function calls if required.

All functions have the following prototype:

```
void apiFunction(ADI_ROM_BOOT_CONFIG* pBootStruct);
```

Another way to support custom boot mode is:

1. Create a first stage boot application to define all Init, Config, Load and Cleanup routines.
2. Use `adi_rom_Boot()` API with hook function installed to update the Init, Config, Load and Clean up functions after the preregister initialization is complete. This is checked by boot ROM to override the above functions inside the `ROM_HOOK_CALL_CAUSE: ROM_HOOK_REG_COMPLETE`

NOTE: The boot command for the custom boot method is 0xF.

Load Function

The load function is required to read data from the source into the specified destination, according to the parame-
Bootters given through the configuration `struct` parameter. The structure provides all of the required information read from the block header, or specified by the kernel to read the block header. The load function often makes use of the DMA APIs in order to simplify the load function implementation.

As the kernel processes the stream, it calls the load function to request data. Initially, the request is for the header, then the kernel makes requests according to the block flags it parses. The load function must only read from the device, and write where requested.

Relevant fields within the `ADI_ROM_BOOT_CONFIG` object for the load function are (but not limited to): `uwDataWidth`, `pSource`, `dByteCount`, `pDestination`, `loadType`.

Custom load functions must meet the following requirements.

- Protect against `dByteCount` values of zero

- Use multiple DMA units if `dByteCount` is greater than 65536 and the peripheral does not support byte count transfers greater than 65536
- The `pSource` and `pDestination` pointers must be properly updated after loading.

In slave boot modes, the boot kernel uses the address of the `dArgument` field in the `pHeader` block as the destination for the required dummy DMAs when payload data is consumed from `ROM_BFLAG_IGNORE` blocks. The load function must read the `ARGUMENT` word of the block early in the function (if required).

Initialization/Configuration Function

The Initialization and Configuration functions are called in sequence when calling a boot operation using an already supported boot peripheral via the adi_rom_Boot() API. These functions are used to configure the boot peripheral prior to calling the boot kernel. Both functions are called in sequence separated only by a call to a user-defined hook function. This hook function is useful when using built-in boot modes to further customize their function. The initialization and configuration functions are responsible for applying any required settings to any devices in use. For example, pin multiplexing may be required and data or pointers that are used by the load function must be initialized. The specific actions depend on the device and functionality used.

Cleanup Function

The cleanup function is called after the entire boot stream is read, and the kernel has completed its boot mode-specific function. This is only performed when using the adi_rom_Boot() API. Resetting any status registers, or device parameters is done to prepare the environment for the execution of the newly loaded application.

Error Handler

This section describes the default error handler for the ROM including information on how to customize the error handling.

The default error handler eventually puts the core into an idle state. This functionality can be overridden by using an Init Block (see Block Types) to modify the error function point in the struct ADI_ROM_BOOT_CONFIG structure. The error handler has access to the entire boot info structure and receives the instruction address that triggered the error.

When a part is locked, and the boot type has not disabled secure boot, only the default error handler is called.

The expected prototype is:

```
void ErrorFunction(ADI_ROM_BOOT_CONFIG* pBootStruct, void *pFailingAddress);
```

The error handler saves the failing address to the `ADI_ROM_BOOT_CONFIG` structure then raises the `INTR_SOFT3` fault signaling a fault condition to the system before then entering an endless loop in the boot rom.

NOTE: When using the adi_rom_Boot() function to perform a boot action, programs may need to manually configure the `INTR_SOFT3` fault signaling depending on the previous application software executed. Calling the boot process using `adi_rom_Boot()` does not result in a reset of the SEC and Fault installation and configuration as described in Preboot Operations.

Page Mode

Page operations are beneficial for page oriented boot source devices, and to improve boot performance for secure boot operations. Page mode optimizes memory reads for block organized devices by always reading a page, rather than reading data on demand. Two 1024 byte buffers are used in page mode where the contents of one buffer are processed by the boot kernel while DMA is used to load the next data into the second buffer.

Blocking DMA is used to load the active buffer, forcing the process to pause until the DMA is complete. Non-blocking DMA is used to load the not active buffer, allowing the active buffer to process while loading the new data in parallel.

Page mode can be enabled when calling a boot mode using `adi_rom_Boot()`. Refer to the API documentation for the various modes supported by this API. Additionally programs can set the flag using the `dFlags` function (see struct ADI_ROM_BOOT_CONFIG) when using hook functions

NOTE: Do not customize page mode settings from the default installed by the boot process.

Boot Hook Function

The boot software allows installation of callback hooks using the adi_rom_Boot() APIs hook function parameter. Using this feature, it is possible to alter the state of the processor at different stages of the boot process and customize the boot structures to alter the behavior of the boot process.

The hook function must adhere to the following prototype:

```
nt32_t
hookFunction(ADI_ROM_BOOT_CONFIG* pBootconfig,
ROM_HOOK_CALL_CAUSE cause);
```

By modifying settings in the `ADI_ROM_BOOT_CONFIG` structure, many alterations of the boot process can be achieved. Much of the same functionality that is available in an Init Block can be provided through the hook function, with even more flexibility for customization. The hook function is called once after executing the boot modes Init routine then once again after executing the boot modes Config routine. A flag passed to the hook function allows software to determine at which point the call took place to allow for conditional processing to occur at different stages of the setup phase.

The hook function must return a zero value for normal booting to continue. A non-zero return value causes the ROM to omit loading any data and immediately transfer control according to Boot Termination and Application Execution .

When the hook function is called, a parameter is passed that indicates why the hook function was called. See enum ROM_HOOK_CALL_CAUSE for more details.

enum ROM_HOOK_CALL_CAUSE

Enumeration Type Declaration: `ROM_HOOK_CALL_CAUSE`

Passed to a user hook routine to indicate the reason of the call.

An optional hook routine is provided as a callback when calling a boot mode via adi_rom_Boot. This hook routine is called by the boot software first after the execution of the boot modes initialization routine then again after execution of the boot modes configuration routine. This parameter allows the users routine to identify at which point the call was made allowing the user to perform different actions for each call.

Table 47-49: ROM_HOOK_CALL_CAUSE Members

Enumerator	Description
ROM_HOOK_CALL_INIT_COMPLETE	Call was because of the completion of the boot modes initialization function
ROM_HOOK_CALL_CONFIG_COMPLETE	Call was because of the completion of the boot modes configuration function
ROM_HOOK_REG_COMPLETE	Call was because of the completion of the boot modes pre-register initialization

Boot Return Feature

The adi_rom_Boot() API provides a feature to bypass calling the loaded application when boot completes, and to return to the routine that made the call instead. The boot software returns the next address after the last loaded application block in the boot source when this feature is enabled.

To enable this feature, set the ROM_BFLAG_RETURN flag in the adi_rom_Boot() flags argument when calling the API.

Boot Termination and Application Execution

When the boot kernel completes the processing of the boot stream, a sequence of events is required to then pass control to the loaded application.

When the boot process is complete, the core is required to vector to the application start address and start executing the newly loaded application. Typically, the first block of a boot stream, which is marked with the BFLAG_FIRST flag, contains the address of the application. In a multi-core system there may be multiple first blocks in the bootstream indicating the start address of the application for each core. The application entry point for each core is loaded into the cores corresponding RCU_SVECTn register.

Upon boot completion only the core that performed the boot process will vector and start executing the loaded application. This core must then manage the process of resetting then releasing from the reset the other cores in the system in order to make them execute their newly loaded applications.

Execution of the loaded application can be bypassed when calling the boot mode using adi_rom_Boot() and setting the ROM_BFLAG_RETURN flag.

Table 47-50: Application Entry Point Registers

Core ID	Corresponding RCU_SVECTn Register
	RCU_SVECT0

Table 47-50: Application Entry Point Registers (Continued)

Core ID	Corresponding RCU_SVECTn Register
0	
1	RCU_SVECT1
2	RCU_SVECT2

Boot ROM OTP Customizations

The boot ROM provides a mechanism through available non-volatile programmable memory (OTP on this processor) to customize different aspects of the boot process. These customizations include overriding default boot-peripheral instance, overriding default peripheral-timing parameters, and disabling boot modes.

Data in the OTP memory controls all ROM customizations. The struct ADI_ROM_OTP_BOOT_INFO data structure accounts for most of the options.

CGU Initializations

Refer to CGU Configuration

Boot Command Customization

Refer to Boot Command Customization

DMC Configuration

Refer to DMC Configuration

Secure Boot Customization

All the public and private keys can be invalidated using the various key invalidation fields provided in the `ADI_ROM_OTP_BOOT_INFO` structure. This configuration is useful when a new key is required. The boot ROM always uses the lowest valid key enumeration. If key0 is valid, then it is used, if key0 is invalid and key1 is valid, then key1 is used. Refer to Secure Boot for details on the secure boot functionality.

Disabling Boot Modes

Refer to Boot Mode Disable

API Reference

The APIs defined in this section are exposed for widespread use.

adi_rom_Boot()

Provides access to boot an application at run-time through a supported peripheral.

API Details

```
void * adi_rom_Boot(
                     void * pAddress,
                     uint32_t flags,
                     int32_t blockCount,
                     ROM_BOOT_HOOK_FUNC * pHook,
                     uint32_t command
                    )
```

pAddress

Pointer to source address of the boot stream.

flags

Global flags applied to the entire boot process

blockCount

Number of blocks to boot. Zero results in processing until a final block is reached.

pHook

Pointer to user implemented hook function for enabling callbacks during the registering of the boot mode with the boot kernel

command

The boot command defining the boot mode to use, the peripheral instance to boot from as well as some boot mode specific configuration

Returns

The 32-bit address of the next address in the boot source to process

Function Description

This function is used for any second-stage boot for a currently supported boot mode. It provides options to boot from any peripheral enumeration. In SPI Master boot the function supports using any SPI slave select signal.

Boot modes may support an auto-detection mechanism to identify the type of connected device. This function provides options to bypass such auto-detection and use custom configuration options. Options are also provided to bypass peripheral configurations such as pinmux settings or a peripheral configuration if an existing peripheral is more appropriately configured to allow communication with the boot source.

These features are all provided via the command parameter which is specific for each particular boot mode.

The source address of the boot stream is required for master boot modes that require that an address is issued in order to request data from the boot source. Slave boot modes are under full control of the host and use a handshake mechanism to indicate that the processor is ready to receive data. For boot modes such as UART Slave and SPI Slave

this parameter is of little value in regards to the boot process itself. However it can prove useful in debug to see how far through the boot stream the boot process progressed in the event of a boot failure.

NOTE: The processor supports both SPI Memory-Mapped boot as well as Peripheral based SPI Boot. When the boot mode is called to boot from the memory-mapped boot mode via the command argument, the address must coincide with the processors memory-mapped SPI address space as defined by the processors internal memory map. When using the peripheral based boot mode use the absolute address of the boot stream in flash.

Flags passed using the flags argument are *global* flags and the operation is applied throughout the entire boot process. These must not be confused with the boot *block specific* flags which are part of the boot stream and indicate how a particular block in the boot stream is processed. Internally, the boot kernel takes the global flags supplied using this function call and combines them with a boot block's local flags to determine all the operations to perform on a given block. After processing the boot block the local flags are cleared and are ready to populate from the next boot block while the global flags remain.

Table 47-51: Global Flags

Bit Position	Flag Name	Description
18	ROM_BFLAG_HOOK	Calls the application supplied hook function after execution of bootmode init and config routines
19	ROM_BLAG_PAGEMODE	Enables page mode processing where blocks of data are fetched and processed from internal memory
20	ROM_BFLAG_NOFIRS-THEADER	Set if calling the boot mode and the first block header has already been fetched and is present in the block header storage location
21	ROM_BFLAG_HEADER	Not set by the application, set by the boot code each time it is fetched by a block header
22–25	Reserved	
26	ROM_BFLAG_SLAVE	Slave boot mode. Causes different handling of ignore blocks by the kernel
27	ROM_BFLAG_WAKEUP	Enables conditional processing of boot blocks intended for wakeup events but not exclusively
28	ROM_BFLAG_NEXTDXE	Parses the stream using the next DXE pointer
29	ROM_BFLAG_RETURN	Returns the application after calling the API instead of running the new application
30	ROM_BFLAG_NORESET	No Reset
31	ROM_BFLAG_NORESTORE	Do not execute the boot peripherals cleanup routine to restore register contents

The `blockCount` argument specifies the number of blocks to process before terminating the boot process (default = 0x00000000). This value instructs the boot software to continue processing a boot stream until the `ROM_BFLAG_FINAL` flag is set. To load only a specified number of blocks, programs can use `blockCount` argument.

When `blockCount` is used in combination with the `ROM_BFLAG_NEXTDXE` flag then the block count is re purposed as a next application count. The boot kernel navigates the first blocks of multiple boot streams similar to a linked list. When the requested application count is reached, the program returns the pointer to this application in the boot source. This allows programs to use the boot kernel to find a specific application when multiple application boot streams are stored contiguously in the boot source.

The `pHook` argument is a function pointer to a hook routine. When used with the `ROM_BFLAG_HOOK` global flag the boot mode calls the hook routine after calling the boot modes `init` and functions in the boot mode driver so that the boot software can be customized, and the configuration changed.

The `pHook` argument when used with the `ROM_BFLAG_HOOK` global flag can enable new features not supported by the boot software and to install a custom load function or error handler for example.

The command argument describes the boot peripheral to boot from, the peripheral instance and contains additional boot mode specific configuration information and flags specific to the boot mode.

NOTE: When calling a boot mode via this API the program must first ensure that the boot peripheral is configured in the SPU as a secure master. The boot software does not configure the peripheral security via this API so that device security is fully controlled by a dedicated task.

adi_rom_BootKernel()

Calls the boot kernel allowing for implementation of custom boot modes.

API Details

```
void * adi_rom_BootKernel(ADI_ROM_BOOT_CONFIG * pBoot)
```

pBoot

Pointer to the `struct ADI_ROM_BOOT_CONFIG` boot structure containing the complete context of the boot configuration.

Returns

Pointer containing the address of the byte immediately following the end of the boot stream.

Function Description

The boot kernel performs the core processing of the boot stream. The boot kernel calls a load function to load data from the peripheral to the required destination. The boot kernel itself has no concept of what the boot peripheral is or how that peripheral is configured. The kernel calls the registered load function, which must then analyze the boot structure and provide the requested amount of data to the required destination.

The load function called by the kernel is provided via the `struct ADI_ROM_BOOT_REGISTRY` member of `struct ADI_ROM_BOOT_CONFIG`.

The boot kernel fetches a boot stream block header then a payload if one exists. The boot kernel takes care of the size of the data being requested and the destination address.

The load function that is registered with the kernel is required to update the `pSource`. Keeping this control under the load function as opposed to the boot kernel itself allows load functions to better control where the next block of data is fetched in the event the boot stream is fragmented or split into different areas of the boot source.

This function is used to implement a second stage boot loader for a peripheral in which there is no driver support in the boot ROM. The application is responsible for initializing the peripheral and the complete `struct ADI_ROM_BOOT_CONFIG` object before calling this function. The application is responsible for performing a vector to the newly loaded application on return from the function.

NOTE: Programs must ensure that when a new application is loaded it does not clobber the load function and the part of the software responsible for making the core jump to the start of the newly loaded application.

adi_rom_Crc32Init()

The CRC32 Initcode function in the boot rom that is called to enable CRC32 support of boot stream payloads.

API Details

```
ROM_BOOT_RESULT adi_rom_Crc32Init(ADI_ROM_BOOT_CONFIG * pBootConfig)
```

pBootConfig

Pointer to the `struct ADI_ROM_BOOT_CONFIG` object containing the complete boot configuration

Returns

Returns the following results

- `enum ROM_BOOT_RESULT` when `pBootConfig` or `pBootConfig->pHeader` are zero
- `enum ROM_BOOT_RESULT` when the callback is registered and lookup table initialized

Function Description

The boot process supports CRC32 protection of all boot block payloads. In order to enable this feature a global callback must be registered with the boot process using `struct ADI_ROM_BOOT_CONFIG` and the CRC peripherals look up table initialized from the application's polynomial.

An init block header where the `ROM_BFLAG_INIT` flag is set must be included in the boot stream to enable CRC functions. The `ADI_ROM_BOOT_HEADER::pTargetAddress` field must be set to the address of this function and the application polynomial is provided using the block's `struct ADI_ROM_BOOT_HEADER` member.

When the boot kernel processes the init block, it calls this function in the boot ROM, registers the callback with the kernel, and performs the look up table initialization.

CRC functionality is enabled on MDMA channel 1 interfaced to the CRC0 peripheral instance.

adi_rom_Crc32Poly()

Initializes the CRC peripheral for use with the user supplied polynomial.

API Details

```
ROM_BOOT_RESULT adi_rom_Crc32Poly(
                                    uint32_t CrcPoly,
                                    ROM_BOOT_MDMA_REGS const *const pDma
                                   )
```

CrcPoly

None

pDma

None

Function Description

The CRC lookup table must be initialized for the CRC polynomial of choice to prepare the CRC peripheral for use.

adi_rom_GetAddress()

Used to find the location of various look-up tables and data objects used during the boot process.

API Details

```
int32_t adi_rom_GetAddress(ROM_GETADDR_VALUE value)
```

value

The `enum ROM_GETADDR_VALUE` enumeration specifying the object to retrieve the address of in the ROM memory

Returns

The byte address of the object in memory

Function Description

The function returns the address of the object specified by the enumerator provided as an argument to the function. Using this function can make software more code compatible with future products and silicon revisions.

adi_rom_MemCompare()

Verifies that a block of data is filled with an application supplied 32-bit value.

API Details

```
ROM_BOOT_RESULT adi_rom_MemCompare(
                                    ROM_DMA_MDMA_CONFIG * pDmaCfg,
                                    ROM_BOOT_MDMA_REGS const *const pDma
                                   )
```

pDmaCfg

None

pDma

None

Function Description

The CRC peripheral is used in compare mode and a source MDMA channel is used to read data from a buffer and supply each 32-bit value to the CRC. The CRC peripheral checks that the incoming 32-bit value matches the 32-bit value to compare against.

adi_rom_MemCopy()

Performs a Memory-to-Memory DMA (MDMA) operation using a source and destination pair of DMA channels.

API Details

```
ROM_BOOT_RESULT adi_rom_MemCopy(
                                  ROM_DMA_MDMA_CONFIG * pDmaCfg,
                                  ROM_BOOT_MDMA_REGS const *const pDma
                                )
```

pDmaCfg

Pointer to the `struct ROM_DMA_PDMA_CONFIG` object containing the peripheral DMA configuration

pDma

Pointer to the struct ROM_BOOT_MDMA_REGS objects that provides access to the DMA channel MMRs and associated CRC peripheral

Returns

Returns the following results

- enum ROM_BOOT_RESULT for a successful operation or when byte count is 0 as no operation is performed
- `enum ROM_BOOT_RESULT` if a configuration error was detected in the DMA channel prior to configuring the channels for the new operation
- if a configuration error occurred in the source MDMA channel
- if a configuration error occurred in the destination MDMA channel

Function Description

The memory copy routine performs transfers of blocks of data from one memory location to another. The routine takes a basic descriptor providing configuration details of the operation to perform using the `struct`

ROM_DMA_MDMA_CONFIG object passed as the first argument. The second argument is a descriptor that provides access to the DMA channel's MMR registers and the associated CRC peripheral. When called from the higher level adi_rom_MemDma() routine this object is retrieved from the ROM.

NOTE: Applications are expected to make use the adi_rom_MemDma() routine for all MDMA operations as there is little additional optional configuration that is supported by using this routine.

adi_rom_MemCrc()

Performs CRC32 verification of a block of data by reading the contents and comparing them with an expected result.

API Details

```
ROM_BOOT_RESULT adi_rom_MemCrc(
                                  ROM_DMA_MDMA_CONFIG * pDmaCfg,
                                  ROM_BOOT_MDMA_REGS const *const pDma
                                 )
```

pDmaCfg

Pointer to the struct ROM_DMA_PDMA_CONFIG object containing the peripheral DMA configuration

pDma

Pointer to the struct ROM_BOOT_MDMA_REGS objects that provides access to the DMA channel's MMRs and associated CRC peripheral

Function Description

The routine uses an MDMA channel pair source DMA channel and the CRC peripheral to calculate a CRC32 result of a data block using a previously supplied polynomial.

The polynomial is supplied through the adi_rom_Crc32Poly() routine to ensure consistent CRC peripheral configuration for the look up table initialization that uses the polynomial and the

NOTE: Applications are expected to make use the adi_rom_MemDma() routine for all MDMA operations, there is little additional optional configuration that is supported by using this routine.

adi_rom_MemDma()

Provides access to all MDMA operations supported by the boot ROM implementation.

API Details

```
ROM_BOOT_RESULT adi_rom_MemDma(ROM_DMA_MDMA_CONFIG * pDmaCfg)
```

pDmaCfg

Pointer to the struct ROM_DMA_PDMA_CONFIG object containing the MDMA configuration

Returns

Returns the following results using the `enum ROM_BOOT_RESULT` enumeration.

- ROM_BOOT_SUCCESS Success. General success can be used to indicate any general functional success for an operation during the boot process. This must be the return result for a boot mode drivers initialization, configuration, load and cleanup routines when overriding their functionality in second stage boot loaders to use custom functions. . Successful operation or when byte count is 0 as no operation is performed
- ROM_BOOT_MDMA_ID_ERR Illegal MDMA Channel ID. Returned by adi_rom_MemDma() if the MDMA channel ID is not supported. For supported channel IDs, see ROM_DMA_MDMA_ID . A MDMA channel ID is provided that is not supported
- ROM_BOOT_CRC_SUPPORTED_ERR CRC Not Supported Error. Returned by adi_rom_MemDma(), adi_rom_MemFill(), adi_rom_MemCompare() and adi_rom_Crc32Poly() if the supplied DMA configuration specified a MDMA channel that does not support CRC operations. . A CRC operation was attempted on a MDMA channel that does not support CRC
- ROM_BOOT_MDMA_OPERATION_ERR Illegal MDMA operation Specified. Returned by adi_rom_MemDma() if the MDMA operation is not supported. For supported operations, see ROM_DMA_MDMA_OPERATION . The operation is not supported
- ROM_BOOT_DMA_FAILURE DMA Failure. Returned by the DMA routines if an error was detected in the DMA_STAT.IRQERR prior to setting up a new DMA operation with the newly supplied configuration. . A configuration error is detected in the DMA channel prior to configuring the channel for the new operation
- ROM_BOOT_DMA_FAILURE DMA Failure. Returned by the DMA routines if an error was detected in the DMA_STAT.IRQERR prior to setting up a new DMA operation with the newly supplied configuration. . A configuration error was detected in the DMA channel and the operation requested involves only a single DMA channel
- A configuration error occurred in the source MDMA channel
- A configuration error occurred in the destination MDMA channel
- ROM_BOOT_CRC_COUNT_ERR CRC Byte Count was not a multiple 4. The CRC peripheral operates on 32-bit data only and as such all CRC operations must have a byte count that is a multiple of 4. This result is returned by the higher level adi_rom_MemDma() routine and the underlying adi_rom_MemCompare() and adi_rom_MemFill() routines if the byte count is not a multiple of 4 bytes. . A CRC operation is being requested and the byte count is not a multiple of 4
- ROM_BOOT_CRC_FAILURE MDMA CRC32 Failure. Returned by the higher level adi_rom_MemDma() routine and the underlying adi_rom_MemCompare() routine if the CRC32 result of the block of data did not match the expected result. . A CRC32 verification fails

- ROM_BOOT_CRC_FAILURE MDMA CRC32 Failure. Returned by the higher level adi_rom_MemDma() routine and the underlying adi_rom_MemCompare() routine if the CRC32 result of the block of data did not match the expected result. . A 32-bit memory compare fails

Function Description

The supported MDMA operations are listed in `enum ROM_DMA_MDMA_OPERATION`.

NOTE: The MDMA and CRC peripherals support on-the-fly CRC32 calculations during the transfer of data from one location to another—MDMA in the boot ROM software does not. For CRC calculations data is instead read back from its final destination and verified.

This function is the main entry point for all the MDMA functions supported by the boot ROM software. The individual functions that are called for each operation type are also exposed using the public API.

NOTE: Use this function for all operations. The lower level functions allow for some basic reconfiguration of default parameters but such modifications are not required where these basic MDMA operations are required.

The boot ROM has an MDMA configuration data structure that is used to specify the overall MDMA configuration of the processor. It provides details on the DMA channel ID associated with each MDMA channel's source and destination DMA channel. It provides information about CRC support and the CRC peripheral instance that is used for a given MDMA channel. See `struct ROM_BOOT_MDMA` and `struct ROM_BOOT_MDMA_REGS` for full details of the content stored.

The configuration provided as the only argument to the function is provided in `Table 47-72 ROM_DMA_MDMA_CONFIG Members`.

For a basic MDMA transfer from source to destination the program needs to configure:

- The struct ROM_DMA_MDMA_CONFIG type as enum ROM_DMA_MDMA_OPERATION
- The MDMA channel to use with the eId MDMA Channel ID , for example
- Set the address of the source data via pSource Source Pointer
- Set the destination address of the source data via pDestination Destination Pointer
- Set the byte count using the ByteCount Byte Count field
- Set `eDoneDetect DMA Done Detection Method` to

  enum ROM_DMA_DONE_DETECT_METHOD to poll for DMA completion

NOTE: The implementation of the MDMA operations does not support interrupt driven data transfers. The routines are implemented for polling DMA status during the boot process. The boot stream is restricted where byte counts and source and destination address alignment must all be a multiple of 4 bytes. MDMA routines must comply with these restrictions.

To use the CRC32 function of the MDMA routines, the application must first initialize the CRC lookup table from the application supplied polynomial. This operation can be performed by setting `ROM_DMA_MDMA_CONFIG::eOperation` type as `ROM_DMA_MDMA_OPERATION::ROM_DMA_CRC_LUT_INIT`. If an MDMA channel is specified that does not support CRC function, an error result is returned.

For further details of the individual operations supported see the following API references:

- `adi_rom_MemCopy()`
- `adi_rom_MemCrc()`
- `adi_rom_MemFill()`
- `adi_rom_MemCompare()`
- `adi_rom_Crc32Poly()`

adi_rom_MemFill()

Fills a block of memory with a 32-bit user supplied value.

API Details

```
ROM_BOOT_RESULT adi_rom_MemFill(
                                 ROM_DMA_MDMA_CONFIG * pDmaCfg,
                                 ROM_BOOT_MDMA_REGS const *const pDma
                               )
```

pDmaCfg

None

pDma

None

Returns

Returns the following results

- `enum ROM_BOOT_RESULT` for a successful operation or when byte count is 0 as no operation to be performed
- `enum ROM_BOOT_RESULT` if a configuration error was detected in the DMA channel prior to configuring the channel for the new operation
- A configuration error occurred in the source MDMA channel
- A configuration error occurred in the destination MDMA channel

Function Description

The CRC peripheral is configured for fill mode and the destination MDMA channel is configured to fill a block of memory with a fixed 32-bit value.

adi_rom_PeriphDma()

Provides access to any peripherals dedicated DMA channel for receive operations only.

API Details

```
ROM_BOOT_RESULT adi_rom_PeriphDma(ROM_DMA_PDMA_CONFIG * pDmaCfg)
```

pDmaCfg

Pointer to the `struct ROM_DMA_PDMA_CONFIG` object containing the peripheral DMA configuration

Returns

Returns the following results

- `ROM_BOOT_SUCCESS Success. General success can be used to indicate any general functional success for an operation during the boot process. This must be the return result for a boot mode drivers initialization, configuration, load and cleanup routines when overriding their functionality in second stage boot loaders to use custom functions.` for a successful operation or when byte count is 0 as no operation is performed
- `ROM_BOOT_DMA_ACTIVE DMA Channel is Active Returned only by the peripheral DMA routine when an attempt to run another peripheral DMA operation is attempted and the DMA channel is already running. This is not currently implemented for MDMA operations.` if the DMA channel is currently running
- `ROM_BOOT_DMA_FAILURE DMA Failure. Returned by the DMA routines if an error was detected in the DMA_STAT.IRQERR prior to setting up a new DMA operation with the newly supplied configuration.` if a configuration error was detected in the DMA channel after starting the DMA operation

Function Description

The peripheral DMA routine is used by the load routines of boot peripherals that have dedicated DMA channels and do not support MDMA channel pairs. Examples are the SPI when not configured for memory-mapped mode and UART peripherals.

In the boot implementation this routine is called from the peripheral load function to request data from the boot source. The routine supports both polling on DMA completion and non-blocking operation to allow for immediate return after starting the DMA operation and continuing with further processing.

NOTE: The function only supports read operations from the peripheral to memory. Transmit operations from memory to peripheral are not supported

callback()

Implements custom callbacks to previously loaded code during boot.

API Details

```
ROM_BOOT_RESULT callback(
                          ADI_ROM_BOOT_CONFIG * pBootConfig,
                          ADI_ROM_BOOT_BUFFER * pBuffer,
                          uint32_t nFlags
                         )
```

pBootConfig

Pointer to the `struct ADI_ROM_BOOT_CONFIG` object containing the complete context of the boot procedure

pBuffer

Pointer to the `struct ADI_ROM_BOOT_BUFFER` object containing details of the payload associated with the callback

nFlags

The callback flags as set by the boot kernel

Function Description

A single callback function may be registered with the boot kernel using `pCallBackFunction Pointer to the callback function that is called when processing boot blocks with the callback flag set` . This function is then called whenever a block is processed with the `ROM_BFLAG_CALLBACK` flag set. Only a single callback function can be registered for the complete boot process.

Callbacks may be used alongside indirect blocks when post processing of the received boot data is required before sending the data to the final destination. An example of this is compression applied to block payloads. The compressed payload is loaded indirectly to the intermediate buffer where it is decompressed by the callback. The callback can modify the source address and byte count for the final MDMA transfer of the decompressed payload using the `pBuffer` parameter. The callback returns the boot kernel then handles the final transfer of the uncompressed data to its destination.

There are restrictions on the amount of data that can be loaded using indirect blocks depending on the size of the intermediate buffer. For this reason the nFlags parameter is used to indicate the status of the callback when handling larger blocks of indirect data. The table below defines the supported flags:

Bit Position	Flag Name	Description
0	ROM_CBFLAG_DIRECT	When set indicates the call was from the processing of a block header with the ROM_BFLAG_CALLBACK flag set
1	ROM_CBLAG_PAGESTART	Indicates the callback was a result of a fetch of a page of data to the intermediate buffers
2	ROM_CBFLAG_FIRST	Set if the first fetch of payload data
3	ROM_CBFLAG_FINAL	Set if the final fetch of payload data
31:4	Reserved	Reserved

When a callback block header is received by the boot kernel, a call to the callback is performed with the ROM_CBFLAG_DIRECT flag set. If the ROM_BFLAG_INDIRECT flag or the ROM_BFLAG_PAGEMODE flags are set they indicate the use of indirect or page mode the ROM_CBFLAG_FIRST and ROM_CBFLAG_FINAL flags are cleared. If the transfer is a direct transfer straight to the final destination and not via the intermediate buffers, then the ROM_CBFLAG_FIRST and ROM_CBFLAG_FINAL flags are also set.

This allows software to identify a callback call based on the processing of a block header with the ROM_BFLAG_CALLBACK flag set.

Callbacks are also called when processing payloads indirectly or when page mode is enabled.

- The ROM_CBFLAG_DIRECT flag is cleared when the callback is a result of processing the payload data using the intermediate buffers.
- The ROM_CBFLAG_FIRST flag is set if the callback is a result of fetching the first block of data in the payload. This flag is also set if the complete block of data fits in the intermediate buffer.
- If the payload does not fit completely in the intermediate buffers multiple fetches must take place and thus multiple callbacks are generated.
- If no flags are set it indicates a callback on a payload transfer is neither the first nor the last block of data in the payload, so there is still further data in the payload to fetch.
- If only the ROM_CBFLAG_FINAL flag is set then it is the final block in a payload transfer.

The following table provides an overview of the flag states and their meaning for the processing of callbacks.

ROM_CBFLAG_DIRECT	ROM_CBFLAG_PAGESTART	ROM_CBFLAG_FIRST	ROM_CBFLAG_FINAL	Description
1	0	0	0	Callback as a result of processing a block header with indirect or pagemode enabled
1	0	1	1	Callback as a result of processing a block header with indirect and pagemode disabled

ROM_CBFLAG_DIRECT	ROM_CBFLAG_PAGESTART	ROM_CBFLAG_FIRST	ROM_CBFLAG_FINAL	Description
0	1	0	0	Callback as a result of fetching a page of data in pagemode
0	1	0	1	Callback as a result of fetching a page of data in pagemode and the final page in the block
0	0	1	0	Callback as a result of fetching the first part of payload in an indirect payload
0	0	0	0	Callback as a result of fetching an indirect payload, not first or last transfer in payload
0	0	0	1	Callback as a result of fetching the final part of payload in an indirect payload
0	0	1	1	Callback as a result of fetching the complete payload in an indirect payload

initcode()

Implements custom callbacks to previously loaded code during boot.

API Details

```
void initcode(ADI_ROM_BOOT_CONFIG * pBootConfig)
```

pBootConfig

Pointer to the `struct ADI_ROM_BOOT_CONFIG` object containing the complete context of the boot procedure

Function Description

Initcode functions are embedded into the boot stream to execute application code during the boot phase. Initcode functions help to optimally configure the CGU or any external memory interface that requires initialization to boot data to those memories.

A boot stream may have any number of initcodes present. The only requirement is that the code must be loaded prior to processing the `BFLAG_INIT` block.

The initcode routine is passed by the pointer to the complete boot context so that extensive boot customization tasks can be supported.

adi_rom_idle_loop()

Jumps to the IDLE loop inside the boot ROM space.

API Details

```
void adi_rom_idle_loop(void)
```

Function Description

This function is used to jump to the forever idle loop inside the boot ROM space.

adi_rom_CguInit()

Can be used for CGU initialization.

API Details

```
bool adi_rom_CguInit(const ADI_ROM_OTP_BOOT_CGU_INFO * const pSettings)
```

pSettings

Pointer to the struct `struct ADI_ROM_OTP_BOOT_CGU_INFO` object containing complete context of CGU initialization

Function Description

This function is used for CGU initialization.

adi_rom_DmcPhyCalibration()

Used for DMC PHY calibration.

API Details

```
void  adi_rom_DmcPhyCalibration(ADI_ROM_OTP_DMC_CONFIG *pConfig,int csel_dsel_r)
```

pConfig

Pointer to the struct `struct ADI_ROM_OTP_DMC_CONFIG` object containing the complete context of the DMC PHY calibration

csel_dsel_r

Holds the CSEL to DSEL ratio

Function Description

This function is used for DMC PHY calibration.

adi_rom_DmcInit()

Used for DMC initialization.

API Details

```
void adi_rom_DmcInit(ADI_ROM_OTP_DMC_CONFIG *pConfig, uint32_t csel_dsel_r)
```

pConfig

Pointer to the struct `struct ADI_ROM_OTP_DMC_CONFIG` object containing the complete context of the DMC PHY calibration.

csel_dsel_r

Holds the CSEL to DSEL ratio.

Function Description

This function is used for DMC PHY calibration.

adi_rom_ShaInit()

Initializes the PKTE module for the SHA-224/256 operation.

API Details

```
void adi_rom_DmcInit(ADI_ROM_OTP_DMC_CONFIG *pConfig, uint32_t csel_dsel_r)
```

pCrypto

Pointer to Crypto descriptors

IV

Initialization Vector

Ecdsa_Word

ECDSA word count for 224/256 bit operation

Function Description

This API is used for the PKTE initialization for the SHA-224/256 bit operation

adi_rom_Sha()

Computes SHA-224/256 bit digest from the PKTE module.

API Details

```
adi_rom_Sha(CRYPTO_DESCRIPTORS * pCrypto,uint32_t Ecdsa_Word, void *output_p,
void *input_p, int size, bool finalFlag)
```

pCrypto

Pointer to Crypto descriptors

Ecdsa_Word

ECDSA word count for 224/256 bit operation

output_p

Pointer to the output buffer

input_p

Pointer to the input buffer

size

Size of the input buffer. It must be a multiple of 512

finalFlag

Final block or not

Function Description

This API is used to compute the message digest using SHA-224/256 using the PKTE module.

/* points to bootrom_jumptable_aes128_cbc_decrypt */

```
          typedef ROM_BOOT_RESULT (*_bootrom_jumptable_aes128_cbc_decrypt_t)
(CRYPTO_DESCRIPTORS
               * pCrypto, void *output_p, void *input_p, int size);
          inline static ROM_BOOT_RESULT
adi_rom_Aes128CbcDecrypt(CRYPTO_DESCRIPTORS * pCrypto,
               void *output_p, void *input_p, int size){
            _bootrom_jumptable_aes128_cbc_decrypt_t pTmp =
               (_bootrom_jumptable_aes128_cbc_decrypt_t)
(FUNC_ROM_AES128_CBC_DECRYPT);
           return (*pTmp)(pCrypto,output_p,input_p,size);
           )
```

/* points to bootrom_jumptable_aes128_key_unwrap */

```
           typedef ROM_BOOT_RESULT (*_bootrom_jumptable_aes128_key_unwrap_t)
(CRYPTO_DESCRIPTORS
               * pCrypto, void *WrappedIn_p, void *KeyOut_p, uint32_t KeyBits,
void *KWK_p);
           inline static ROM_BOOT_RESULT
adi_rom_Aes128KeyUnwrap(CRYPTO_DESCRIPTORS * pCrypto,
               void *WrappedIn_p, void *KeyOut_p, uint32_t KeyBits, void *KWK_p){
           _bootrom_jumptable_aes128_key_unwrap_t pTmp =
```

```
                (_bootrom_jumptable_aes128_key_unwrap_t)
(FUNC_ROM_AES128_KEY_UNWRAP);
            return (*pTmp)(pCrypto, WrappedIn_p, KeyOut_p, KeyBits, KWK_p);
           }
```

/* points to bootrom_jumptable_aes128_cbc_loadkey */

```
          typedef void * (*_bootrom_jumptable_aes128_cbc_loadkey_t)(sa_t *
pSARecord, uint32_t
                *key_p);
          inline static void * adi_rom_Aes128CbcLoadkey(sa_t * pSARecord,
uint32_t *key_p)
                {
            _bootrom_jumptable_aes128_cbc_loadkey_t pTmp =
                (_bootrom_jumptable_aes128_cbc_loadkey_t)
(FUNC_ROM_AES128_CBC_LOADKEY);
            return (*pTmp)(pSARecord,key_p);
             }
```

/* points to bootrom_jumptable_pka_init */

```
          typedef PKA_Status_t (*_bootrom_jumptable_pka_init_t)( PKA_IOArea_t *
const IOArea_p,
                Device_Handle_t Device,const uint32_t * const Firmware_p,const
uint32_t
                FirmwareWordCount);
          inline static PKA_Status_t adi_rom_PKA_Init( PKA_IOArea_t * const
IOArea_p,
                Device_Handle_t Device,const uint32_t * const Firmware_p,const
uint32_t
                FirmwareWordCount) {
          _bootrom_jumptable_pka_init_t pTmp =
                (_bootrom_jumptable_pka_init_t)(FUNC_ROM_PKA_INIT);
            return (*pTmp)(IOArea_p,Device,Firmware_p,FirmwareWordCount);
            }
```

/* points to bootrom_jumptable_ecdsa_verify_init */

```
            typedef ROM_BOOT_RESULT
                (*_bootrom_jumptable_ecdsa_verify_init_t)(SBHYBRID_EcdsaContext_t
* const Verify_p,
                const SBIF_ECDSA_PublicKey_t * const PublicKey_p,const
SBIF_ECDSA_Signature_t *
                const Signature_p,uint32_t EcdsaWord);
             inline static ROM_BOOT_RESULT
adi_rom_EcdsaVerifyInit(SBHYBRID_EcdsaContext_t * const
                Verify_p, const SBIF_ECDSA_PublicKey_t * const PublicKey_p,const
                SBIF_ECDSA_Signature_t * const Signature_p,uint32_t EcdsaWord) {
          _bootrom_jumptable_ecdsa_verify_init_t pTmp =
```

```
                (_bootrom_jumptable_ecdsa_verify_init_t)
(FUNC_ROM_ECDSA_VERIFY_INIT);
           return (*pTmp)(Verify_p,PublicKey_p,Signature_p,EcdsaWord);
          }
```

/* points to bootrom_jumptable_ecdsa_verify_set_digest */

```
          typedef void *
                (*_bootrom_jumptable_ecdsa_verify_set_digest_t)
(SBHYBRID_EcdsaContext_t * const
                Verify_p, uint8_t * Digest_p);
           inline static void *
adi_rom_EcdsaVerifySetDigest(SBHYBRID_EcdsaContext_t * const
                Verify_p, uint8_t * Digest_p) {
            _bootrom_jumptable_ecdsa_verify_set_digest_t pTmp =
                (_bootrom_jumptable_ecdsa_verify_set_digest_t)
(FUNC_ROM_ECDSA_VERIFY_SET_DIGEST);
          return (*pTmp)(Verify_p,Digest_p);
           }
```

/* points to bootrom_jumptable_ecdsa_verify */

```
          typedef SB_Result_t (*_bootrom_jumptable_ecdsa_verify_t)
(SBHYBRID_EcdsaContext_t *
                const Verify_p, uint32_t EcdsaWord);
          inline static SB_Result_t adi_rom_EcdsaVerify(SBHYBRID_EcdsaContext_t *
const
                Verify_p, uint32_t EcdsaWord) {
          _bootrom_jumptable_ecdsa_verify_t pTmp =
                (_bootrom_jumptable_ecdsa_verify_t)(FUNC_ROM_ECDSA_VERIFY);
           return (*pTmp)(Verify_p, EcdsaWord);
          }
```

Data Structures

The programming model for booting the processor uses the data structures defined in this section.

struct ADI_ROM_BOOT_BUFFER

Structure Type Declaration: `ADI_ROM_BOOT_BUFFER`

Boot Buffer.

A basic buffer type consisting of a pointer to the buffer and its size

Table 47-52: ADI_ROM_BOOT_BUFFER Members

Type	Name	Description
`void *`	`pBuffer`	Pointer to the buffer
`int32_t`	`dByteCount`	Size of the buffer

pBuffer

Pointer to the buffer

dByteCount

Size of the buffer

struct ADI_ROM_BOOT_CONFIG

Structure Type Declaration: `ADI_ROM_BOOT_CONFIG`

The Boot Configuration Object that contains all context for the boot process.

This structure contains the complete context for the boot process. A pointer to this object is passed through many routines and is presented to customizable routines such as initcodes, custom initialization, configuration, load and cleanup routines. The object is passed to error handlers and callbacks to customize and adapt the boot process for specific applications, especially in regards to multi-stage boot loader development.

Table 47-53: ADI_ROM_BOOT_CONFIG Members

Type	Name	Description
`void *`	`pSource`	Source address from where to fetch the next boot data.
`void *`	`pDestination`	Destination address to store the fetched data.
`int32_t`	`dByteCount`	Number of bytes to fetch from the boot source.
`int32_t`	`dFlags`	Control flags related to the boot kernel processing of blocks.
`uint32_t`	`ulBlockCount`	Limit of blocks processed during boot.
`uint32_t`	`ulBlockCurrent`	The number of blocks currently processed by the boot kernel
`void *`	`pNextDxe`	Pointer to the next application in the boot stream or the first free location after the boot stream.
`uint32_t`	`uByteAddress`	The destination address converted to the byte address space.
`uint32_t volatile *`	`pControlRegister`	Pointer to the boot peripherals control register.
`int32_t`	`dControlValue`	Storage for the boot peripheral main control value to enable that peripheral in a required configuration.
`uint32_t volatile *`	`pPeripheralBase`	Pointer to the boot peripherals base MMR address

Table 47-53: ADI_ROM_BOOT_CONFIG Members (Continued)

Type	Name	Description
`uint32_t volatile *`	`pAuxControlRegister`	Pointer to any register that may be used for auxiliary operations such as a timer control register for UART auto-baud detection
`uint32_t volatile *`	`pAuxPeripheralBase`	Pointer to the base address of any peripheral used for auxiliary operations such as the TIMER block
`uint32_t volatile *`	`pSecControlRegister`	Base MMR address of the SEC SSI instance associated with the boot peripheral if required for advanced second stage boot loader development
`ADI_DMA_TypeDef *`	`pDmaBaseRegister`	Base MMR address of the DMA channel associated with the boot peripheral.
`ROM_DMA_DONE_DETECT_METHOD`	`loadType`	Set by the kernel to specify to the boot peripherals load function if it is requesting a blocking or non-blocking DMA
`ROM_DMA_MDMA_CONFIG`	`MdmaCfg`	An MDMA descriptor that is used by the boot kernel for internal MDMA operations.
`uint16_t`	`uwDataWidth`	The maximum data width supported by the boot peripherals DMA channel. Set to 0 for 8-bit, 1 for 16-bit and 2 for 32-bit
`uint16_t`	`uwSrcModifyMult`	The source modify multiplier used to set `DMA_XMOD` for source MDMA operations or peripheral DMA transmit operations
`uint16_t`	`uwDstModifyMult`	The destination modify multiplier used to set `DMA_XMOD` for destination MDMA operations or peripheral DMA receive operations
`uint16_t`	`uwUserShort`	Free to use
`int32_t`	`dUserLong`	Free to use
`int32_t`	`dReserved0`	Reserved for future use
`void *`	`pModeData`	Pointer to the boot mode specific data structure.
`int32_t`	`dBootCommand`	The boot command value supplied during the call to the `adi_rom_Boot()` routine.
`ADI_ROM_BOOT_HEADER*`	`pHeader`	Pointer to the boot header storage location where all boot stream block headers eventually reside for processing by the kernel.
`void *`	`pTempBuffer`	Pointer to the internal intermediate buffer. Used for processing of indirect blocks.
`void`	`dReserved1`	Reserved
`int32_t`	`dTempByteCount`	Size of the internal intermediate buffer in bytes.

Table 47-53: ADI_ROM_BOOT_CONFIG Members (Continued)

Type	Name	Description
void *	pTempSource	Current source address that is processed in the internal intermediate buffer.
int32_t	dPageByteCount	The page size used for page mode processing. On this product page size is fixed to 1024 bytes and this member is not used for the load requests.
ADI_ROM_BOOT_INTER_BUFFERS	bootBuffers	The internal intermediate buffer descriptors required when using indirect and page mode features.
ROM_BOOT_REGISTRY	bootRegistry	The registry object that is used to register a boot peripheral routines with the kernel.
ROM_BOOT_ERROR_FUNC *	pErrorFunction	Pointer to the error handler called in the event of an error.
ROM_BOOT_CALLBACK_FUNC *	pCallBackFunction	Pointer to the callback function that is called when processing boot blocks with the callback flag set.
ROM_BOOT_CALLBACK_FUNC *	pCrcFunction	Pointer to the CRC function that is used to perform CRC validation of the boot stream payload data.
ROM_BOOT_CALLBACK_FUNC *	pForwardFunction	Feature not supported on this product.
ADI_ROM_BOOT_MODES	bootModes	Access to all boot mode specific resources.
void *	pLogBuffer	Pointer to the log buffer. Logging is disabled by default on this product and must be configured from within initcodes or hook routines.
void *	pLogCurrent	The current position within the log buffer. Logging is disabled by default on this product and must be configured from within initcodes or hook routines.
int32_t	dLogByteCount	The size of the log buffer. Logging is disabled by default on this product and must be configured from within initcodes or hook routines.
ADI_ROM_OTP_BOOT_INFO*	pOtpBootInfo	Pointer to the ADI_ROM_OTP_BOOT_INFO boot information block that is read from OTP and contains boot customization options.
ADI_ROM_BOOT_KEY_TYPE	keyType	When set to a specific value allows keys not stored in OTP to perform secure boot evaluation on an open processor.
ADI_ROM_BOOT_TYPE	bootType	A key to indicate if the boot type is secure or non-secure for open parts.
ROM_SB_IMAGE_TYPE	secureBootImageType	The type of secure boot image.
SBIF_ECDSA_Header_t*	pSecureHeader	Pointer to the secure boot stream header that is loaded by the boot peripheral from the boot source during the configuration phase.

Table 47-53: ADI_ROM_BOOT_CONFIG Members (Continued)

Type	Name	Description
ADI_SBIF_ECDSA_PublicKey_t	publicKey	The public key used for secure boot image authentication.
CRYPTO_DESCRIPTORS	cryptoDescriptors	The descriptor items as required for PKTE operations.
SB_StorageArea_t*	pSB_Storage	Storage area reserved for some crypto operations.
int32_t	secureBytesRemaining	The number of bytes remaining to be processed in the secure boot stream.
uint32_t[4]	aesKey	The 128-bit AES decryption key.
uint32_t[6]	aesWrapKey	The key wrapped key from the BLw secure boot image.
uint32_t[4]	IV	The IV as read from the secure boot header, required to initialize the PKTE.
uint8_t *	pHash	Pointer to the output destination of the SHA-224 result that is required for authentication of the secure boot stream.
uint8_t *	esdsaType	Type of ECDSA algorithm to sign the image.
uint32_t	errorReturn	Storage location for the address of the instruction line following a call to the error handler.

pSource

Source address from where to fetch the next boot data.

The source address must be maintained by the boot peripheral's load function. The kernel does not update the source pointer automatically after requesting data. This lets load routines control and change the source address. It is useful when advanced second stage loaders need to change the source address (due to a fragmented boot stream) or to reset the address (if expanding into a second SPI flash device).

During debug it is useful in identifying the block in the boot stream that is currently being processed.

pDestination

Destination address to store the fetched data.

Used by the boot kernel to indicate the destination address for the fetched data. A boot peripheral's load function must transfer the data to this location before returning back to the kernel. The boot kernel updates this field depending on whether a block header or payload is being fetched. In normal operation mode the kernel loads this field with the storage area location of the block header. After processing the block header loads pDestination with the pTargetAddress Destination address of payload contents read from the fetched block header. When using page mode the destination points to the internal buffers and is then updated to transfer data to the final destination.

dByteCount

Number of bytes to fetch from the boot source.

The kernel sets this parameter to indicate to the load function the number of bytes requested by the kernel. The kernel is responsible for adjusting the byte count for page mode based accesses. The peripheral load function must return the required number of bytes to the destination address provided.

dFlags

Control flags related to the boot kernel block processing.

When calling a boot mode using the `adi_rom_Boot()` routine the `flags` supplied to that routine are used to initialize `dFlags`. These become global flags that remain set through the entire boot process. When a block header is received the lower 16 bits of the block header are OR'ed with the global flags. The boot software may clear some flags if it detects they are not compatible with other flags and then writes the resulting flags back to this member. The boot kernel then processes the block payload as instructed by the combination of global and boot block specific flags. Upon completion of the block processing original set of global flags are restored and the process repeated.

ulBlockCount

Limit of blocks processed during boot.

When calling the boot process the `adi_rom_Boot()` routine can accept a block limit for the number of blocks to process before terminating the boot process. If the block count is set to zero then the boot process will continue until a final block reached indicating end of the boot stream. This member holds the program's specified limit for the number of blocks to process and is used by the boot kernel after processing each block to compare it against the `ADI_ROM_BOOT_CONFIG::ulBlockCurrent` value. Boot process terminates when `ADI_ROM_BOOT_CONFIG::ulBlockCurrent` equals `ADI_ROM_BOOT_CONFIG::ulBlockCount`

ulBlockCurrent

The number of blocks currently processed by the boot kernel

pNextDxe

Pointer to the next application in the boot stream or the first free location after the boot stream.

This member is initialized when processing a first block in the boot stream. The `dArgument Argument function varies depending on operation` field of a first block contains the number of bytes left in the boot stream before we reach the end of that boot stream. This lets this pointer point to the next boot stream or to the first empty location after the boot stream. This allows for a feature when using the `adi_rom_Boot()` routine to find the address of an application in a linked list of boot streams or to find the first empty location after the boot stream.

uByteAddress

The destination address converted to the byte address space.

This member is used to store the byte address space equivalent of SHARC L1 memory addresses, allowing any core to load content to any SHARC cores L1 memory. The SHARC core in which the load is targeted is determined by the block header.

pControlRegister

Pointer to the boot peripherals control register.

This can be used by a boot mode peripherals driver in order to gain efficient access to a control MMR register in the boot peripheral.

NOTE: This is not used in this products boot implementation but may be leveraged by developers of second stage boot loaders if required

dControlValue

Storage for the boot peripheral main control value to enable that peripheral in a required configuration.

This can be used by a boot modes peripheral driver in order to store a control value that can be used to enable the peripheral for a required configuration.

pPeripheralBase

Pointer to the boot peripherals base MMR address

pAuxControlRegister

Pointer to any register that may be used for auxiliary operations such as a timer control register for UART autobaud detection

pAuxPeripheralBase

Pointer to the base address of any peripheral used for auxiliary operations such as the TIMER block

pSecControlRegister

Base MMR address of the SEC SSI instance associated with the boot peripheral should they be required for advanced second stage boot loader development

pDmaBaseRegister

Base MMR address of the DMA channel associated with the boot peripheral.

This is used by the boot kernel to gain access to the DMA channels status when using non-blocking DMA operations when page mode or secure boot is required, so it must be set when implementing custom boot loaders in order for the kernel to get access to that peripherals DMA status.

NOTE: When a custom boot peripheral does not support the standard DMA instance, the custom driver is required to set up a DMA instance in SRAM that this location points to and the load function would need to update the status accordingly to indicate when the DMA was running and when the DMA completed.

loadType

Set by the kernel to specify to the boot peripherals load function if it is requesting a blocking or non-blocking DMA

MdmaCfg

An MDMA descriptor that is used by the boot kernel for internal MDMA operations.

The boot kernel may be required to perform internal MDMA operations outside the control of the boot peripheral driver. Such operations include processing of fill blocks CRC callbacks for CRC verification and MDMA operations from the internal intermediate buffers for indirect block and page mode processing.

NOTE: Users must not use this item or reconfigure this item when developing custom boot drivers, it is intended purely for the internal use by the boot kernel

uwDataWidth

The maximum data width supported by the boot peripherals DMA channel. Set to 0 for 8-bit, 1 for 16-bit and 2 for 32-bit

uwSrcModifyMult

The source modify multiplier used to set `DMA_XMOD` for source MDMA operations or peripheral DMA transmit operations

uwDstModifyMult

The destination modify multiplier used to set `DMA_XMOD` for destination MDMA operations or peripheral DMA receive operations

uwUserShort

Free to use by the user

dUserLong

Free to use by the user

pModeData

Pointer to the boot mode specific data structure.

Can be set by a boot peripheral driver to allow for a single point of access to the boot mode specific object containing control and configuration information specific to that single boot mode.

dBootCommand

The boot command value supplied during the call to the `adi_rom_Boot()` routine

pHeader

Pointer to the boot header storage location where all boot stream block headers eventually reside for processing by the kernel

pTempBuffer

Pointer to the internal intermediate buffer. Used for processing of indirect blocks

dTempByteCount

Size of the internal intermediate buffer in bytes

pTempSource

Current source address that is being processed in the internal intermediate buffer

dPageByteCount

The page size used for page mode processing. On this product page size is fixed to 1024 bytes and this member is not used for the load requests

bootBuffers

The internal intermediate buffer descriptors required when using indirect and page mode features

bootRegistry

The registry object that is used to register a boot peripherals routine with the kernel.

When using the `adi_rom_Boot()` function the boot software calls a peripherals initialization, configuration, load, and cleanup routines from the pointers stored in this object. The kernel itself only makes calls to the load function for the peripheral, so when using the `adi_rom_BootKernel()` function the load function that is called by the boot kernel to fetch data from the boot source must be registered using `pLoadFunctionPointer to the boot modes Load function .`

pErrorFunction

Pointer to the error handler called in the event of an error

pCallBackFunction

Pointer to the callback function that is called when processing boot blocks with the callback flag set

pCrcFunction

Pointer to the CRC function that is used to perform CRC validation of the boot stream payload data

pForwardFunction

Feature not supported on this product

bootModes

Access to all boot mode specific resources

pLogBuffer

Pointer to the log buffer. Logging is disabled by default on this product and thus must be configured from within initcodes or hook routines

pLogCurrent

The current position within the log buffer. Logging is disabled by default on this product and thus must be configured from within initcodes or hook routines

dLogByteCount

The size of the log buffer. Logging is disabled by default on this product and thus must be configured from within initcodes or hook routines

pOtpBootInfo

Pointer to the `struct ADI_ROM_OTP_BOOT_INFO` boot information block that gets read from `OTP` and contains boot customization options

keyType

When set to a specific value allows keys not stored in OTP to be used for secure boot evaluation on an open processor.

By default when performing boot on an open part the decryption keys and the public key are fetched from `OTP`. By setting this field to enum ADI_ROM_BOOT_KEY_TYPE users can disable the fetching of the keys from `OTP` and instead provision the keys directly in the `ADI_ROM_BOOT_CONFIG::publicKey` and `ADI_ROM_BOOT_CONFIG::aesKey` members via hook routines when using the `adi_rom_Boot()` function.

bootType

A key to indicate if the boot type is secure or non-secure for open parts

secureBootImageType

The type of secure boot image

pSecureHeader

Pointer to the secure boot stream header that is loaded by the boot peripheral from the boot source during the configuration phase

publicKey

The public key used for secure boot image authentication

cryptoDescriptors

The descriptor items as required for the PKTE operations

pSB_Storage

Storage area reserved for some crypto operations

secureBytesRemaining

The number of bytes remaining to be processed in the secure boot stream

aesKey

The 128-bit AES decryption key

aesWrapKey

The key wrapped key from the `BLw` secure boot image

IV

The `IV` as read from the secure boot header as required to initialize the PKTE

pHash

Pointer to the output destination of the SHA-224 result that is required for authentication of the secure boot stream

ecdsaType

This holds the ecdsa type to be used for authentication. It can be 224-bit or 256-bit.

errorReturn

Storage location for the address of the instruction line following a call to the error handler

struct ADI_ROM_BOOT_HEADER

Structure Type Declaration: `ADI_ROM_BOOT_HEADER`

Boot Block Header.

Boot block headers control the loading process of the boot stream, For full details on the contents of the block header and supported flags see Boot Stream.

Table 47-54: ADI_ROM_BOOT_HEADER Members

Type	Name	Description
`int32_t`	`dBlockCode`	Instructs the boot kernel how to process the block.
`void *`	`pTargetAddress`	Destination Address of Payload
`int32_t`	`dByteCount`	Byte Count of the Payload
`int32_t`	`dArgument`	Argument functionality varies depending on operation

dBlockCode

Instructs the boot kernel how to process the block.

Contains a number of fields for verification of the block header and flags to indicate the type of block. This allows the kernel to process the block correctly.

pTargetAddress

Destination address of payload

dByteCount

Byte count of the payload

dArgument

Argument function varies depending on operation

struct ADI_ROM_BOOT_INTER_BUFFER

Structure Type Declaration: `ADI_ROM_BOOT_INTER_BUFFER`

The buffer object for the internal intermediate buffers used for indirect and page mode operations.

Table 47-55: ADI_ROM_BOOT_INTER_BUFFER Members

Type	Name	Description
`uint8_t *`	`pBuffer`	Pointer to the buffer
`uint32_t`	`size`	Size of the buffer
`uint32_t`	`pageSize`	Page size for block based devices.

pBuffer

Pointer to the buffer

size

Size of the buffer

pageSize

Page size for block based devices.

NOTE: This field is not used in this product. A fixed page size of 1024 bytes is used.

struct ADI_ROM_BOOT_INTER_BUFFERS

Structure Type Declaration: `ADI_ROM_BOOT_INTER_BUFFERS`

The boot kernels internal buffer object used to access the intermediate buffers and obtain buffer status.

Table 47-56: ADI_ROM_BOOT_INTER_BUFFERS Members

Type	Name	Description
`ADI_ROM_BOOT_INTER_BUFFER[2]`	`buffer`	The two buffer descriptors
`ADI_ROM_BOOT_BUFFER_STATE`	`state`	Buffer Status Information
`void *`	`pSource`	Original source address pointer of data loaded to active buffer. Not used on this product

Table 47-56: ADI_ROM_BOOT_INTER_BUFFERS Members (Continued)

Type	Name	Description
ADI_DMA_TypeDef *	pDma	TBD

buffer

The two buffer descriptors

state

Buffer Status Information

pSource

Original source address pointer of data loaded to active buffer. Not used on this product

struct ADI_ROM_BOOT_LINKPORT

Structure Type Declaration: ADI_ROM_BOOT_LINKPORT

The linkport slave boot mode specific structure.

This structure contains all the boot context information that is specific to the linkport slave boot mode.

Table 47-57: ADI_ROM_BOOT_LINKPORT Members

Type	Name	Description
uint32_t	nFlags	Flags related to linkport boot mode
ADI_LP_TypeDef *	pRegisters	Pointer to the LP peripherals base MMR address, not used on this product
ADI_DMA_TypeDef *	pRxDmaRegisters	Pointer to the DMA peripherals base MMR address, for receive operations, not used on this product
ADI_DMA_TypeDef *	pTxDmaRegisters	Pointer to the DMA peripherals base MMR address, for transmit operations, not used on this product

nFlags

Flags related to linkport boot mode

pRegisters

Pointer to the LP peripherals base MMR address, not used on this product

pRxDmaRegisters

Pointer to the DMA peripherals base MMR address, for receive operations, not used on this product

pTxDmaRegisters

Pointer to the DMA peripherals base MMR address, for transmit operations, not used on this product

struct ADI_ROM_BOOT_MODES

Structure Type Declaration: `ADI_ROM_BOOT_MODES`

Holds all boot mode specific configuration items.

A boot mode may have requirements for some dedicated storage. This object is used to collect all storage items for all the boot modes supported by the boot ROM.

Table 47-58: ADI_ROM_BOOT_MODES Members

Type	Name	Description
`ADI_ROM_BOOT_SPI`	`spi`	Access to all SPI boot mode resources
`ADI_ROM_BOOT_UART`	`uart`	Access to all UART boot mode resources
`ADI_ROM_BOOT_LINKPORT`	`linkport`	Access to all LINKPORT boot mode resources
`ADI_ROM_BOOT_OSPI`	`ospi`	Access to all OSPI boot mode resources
`ADI_ROM_BOOT_EMSI`	`pEmsi`	Access to all EMSI boot mode resources
`ADI_ROM_BOOT_CUSTOM`	`custom`	Access to all custom boot mode resources

spi

Access to all SPI boot mode resources

uart

Access to all UART boot mode resources

linkport

Access to all LINKPORT boot mode resources

pEmsi

Access to all EMSI boot mode resources

custom

Access to all custom boot mode resources

Spi3

Access to all OSPI boot mode resources

struct ADI_ROM_BOOT_REGISTRY

Structure Type Declaration: `ADI_ROM_BOOT_REGISTRY`

Boot Mode Registration.

Used to hold pointers for the boot mode initialization, configuration, load and cleanup functions. Can customize the registered content via hook routines or install load functions or cleanup functions from within init codes.

When using `adi_rom_Boot()` the boot process makes a call to the initialization function and the configuration function before calling the kernel. The kernel then runs and makes calls to the load function. The cleanup function is called when the kernel reaches the end of the boot stream.

When using the adi_rom_BootKernel() function only the load function is called during execution of the software in the boot ROM. All the functions here must return a `enum ROM_BOOT_RESULT` result in order for the boot process to continue. All functions expect a single argument that is the pointer to the boot structure object `struct ADI_ROM_BOOT_CONFIG`.

Table 47-59: ADI_ROM_BOOT_REGISTRY Members

Type	Name	Description
`ROM_BOOT_MODE_INIT_FUNC *`	`pInitFunction`	Pointer to the boot modes Initialization function
`ROM_BOOT_MODE_CONFIG_FUNC *`	`pConfigFunction`	Pointer to the boot modes Configuration function
`ROM_BOOT_MODE_LOAD_FUNC *`	`pLoadFunction`	Pointer to the boot modes Load function
`ROM_BOOT_MODE_CLEANUP_FUNC *`	`pCleanUpFunction`	Pointer to the boot modes Cleanup function
`void *`	`pReserved`	Reserved for future use
`int32_t`	`dReserved`	Reserved for future use

pInitFunction

Pointer to the boot modes Initialization function

pConfigFunction

Pointer to the boot modes Configuration function

pLoadFunction

Pointer to the boot modes Load function

pCleanUpFunction

Pointer to the boot modes Cleanup function

struct ADI_ROM_BOOT_SPI

Structure Type Declaration: `ADI_ROM_BOOT_SPI`

The SPI Master Boot Mode Specific Structure.

This structure contains all the boot context information that is specific for SPI master boot mode. During auto-detection information is copied from the required `::ROM_SPI_LUTENTRY` item into this structure and used to configure the SPI peripheral for the mode of operation.

Table 47-60: ADI_ROM_BOOT_SPI Members

Type	Name	Description
uint8_t	ubReadCommand	Read command to read data from the SPI device
uint8_t	ubDummyBytes	Number of dummy bytes to issue after the read command
uint8_t	ubAddressBytes	Number of address bytes required to access the device
uint8_t	ubDataBits	Bus width when reading the data. 0 = single bit, 1 = dual bit, 2 = quad bit
uint16_t	uwClkLower	SPI clock divider value
uint16_t	uReserved0	Reserved
uint32_t	nTxCtl	The value written to the SPI_TXCTL register used for address transmit operations such as address cycles
uint32_t	nRxCtl	The value written to the SPI_RXCTL register that is used for all receive operations
uint32_t	nCmdCtl	The value written to the SPI_TXCTL register that is used for sending the read command to the SPI Flash
ROM_BOOT_SPIM_IO_ENABLE_FUNC *	pMIOEnFunction	Pointer to the function used to enable quad mode on the SPI flash
uint8_t	nDummy	The Dummy Byte value if dummy byte transfers are required and the bus is not three-stated
uint8_t	nFlags	Flags used for additional SPI configuration processing.
uint16_t	uReserved2	Reserved
void *	pXIPAddress	The memory-mapped SPI address to boot from
ADI_SPI_TypeDef *	pRegisters	Pointer to the SPI peripherals base MMR address, not used on this product
ADI_DMA_TypeDef *	pRxDmaRegisters	Pointer to the DMA peripherals base MMR address, for receive operations, not used on this product
ADI_DMA_TypeDef *	pTxDmaRegisters	Pointer to the DMA peripherals base MMR address, for transmit operations, not used on this product

ubReadCommand

Read command to use to read data from the SPI device

ubDummyBytes

Number of dummy bytes to issue after the read command

ubAddressBytes

Number of address bytes required to access the device

ubDataBits

The bus width used when reading the data. 0 for single bit, 1 for dual, 2 for quad

uwClkLower

The SPI clock divider value

nTxCtl

The value written to the `SPI_TXCTL` register that is used for the address transmit operations such as address cycles

nRxCtl

The value written to the `SPI_RXCTL` register that is used for all receive operations

nCmdCtl

The value written to the `SPI_TXCTL` register that is used for sending the read command to the SPI Flash

pMIOEnFunction

Pointer to the function used to enable quad mode on the SPI flash

nDummy

The Dummy Byte value if dummy byte transfers are required and the bus is not tri-stated

nFlags

Flags used for some additional SPI configuration processing. The flags supported are defined as follows:

Bit Position	Name	Description
0	`ROM_SPI_FLAGS_CMDSKIP_EN`	When set the configuration routine enables command skip mode where the SPI does not issue a read command for read operations
1	`ROM_SPI_FLAGS_MULTICMD_EN`	Configuration routine enables sending command cycles over dual or quad bit bus

pXIPAddress

The memory-mapped SPI address to boot from

pRegisters

Pointer to the SPI peripherals base MMR address, not used on this product

pRxDmaRegisters

Pointer to the DMA peripherals base MMR address, for receive operations, not used on this product

pTxDmaRegisters

Pointer to the DMA peripherals base MMR address, for transmit operations, not used on this product

struct ADI_ROM_BOOT_OSPI

Structure Type Declaration: `ADI_ROM_BOOT_OSPI`

The SPI Master Boot Mode Specific Structure.

This structure contains all the boot context information that is specific for OSPI master boot mode. During auto-detection information is copied from the required `::ROM_OSPI_LUTENTRY` item into this structure and used to configure the OSPI peripheral for the mode of operation.

Table 47-61: ADI_ROM_BOOT_OSPI Members

Type	Name	Description
uint8_t	ubReadCommand	Read command to read data from the OSPI device
uint8_t	ubDummyCycles	Number of dummy cycles to issue after the read command
uint8_t	ubAddressBytes	Number of address bytes required to access the device
uint8_t	ubDataBits	Bus width when reading the data
uint16_t	uwClkLower	OSPI clock divider value
uint16_t	uReserved0	Reserved
uint32_t	nCfg	Value written to lower 16 bits of the OSPI_CFG register
uint32_t	nDsr	Value written to OSPI device size register
uint32_t	nDrir	Value written to OSPI device read instruction control register
uint32_t	reserved1	Reserved
uint8_t	nDummy	The dummy byte value to be used
uint8_t	nFlags	Flags used for some additional SPI configuration processing
uint16_t	uReserved2	Reserved
void *	pXIPAddress	The memory-mapped OSPI address to boot
ADI_OSPI_TypeDef *	pRegisters	Pointer to the OSPI peripherals base MMR address, not used on this product
ADI_DMA_TypeDef *	pRxDmaRegisters	Pointer to the DMA peripherals base MMR address, for receive operations, not used on this product
ADI_DMA_TypeDef *	pTxDmaRegisters	Pointer to the DMA peripherals base MMR address, for transmit operations, not used on this product

ubReadCommand

Read command to use to read data from the SPI device

ubDummyCycles

Number of dummy cycles to issue after the read command

ubAddressBytes

Number of address bytes required to access the device

ubDataBits

The bus width used when reading the data. 0 for single bit, 1 for dual, 2 for quad

uwClkLower

The SPI clock divider value

nCfg

The value written to the Lower 16 bits of the OSPI Control register

nDsr

The value written to the OSPI Device Size register

nDrir

The value written to the OSPI Device Read Instruction Control register

nDummy

The Dummy Byte value if dummy byte transfers are required and the bus is not tri-stated

nFlags

Flags used for some additional SPI configuration processing. The flags supported are defined as follows:

- `ROM_SPI_FLAGS_CMDSKIP_EN` (Bit position 0): When set the configuration routine enables command skip mode where the SPI does not issue a read command for read operations
- `ROM_SPI_FLAGS_MULTICMD_EN` (Bit position 1): Configuration routine enables sending command cycles over dual or quad bit bus

pXIPAddress

The memory-mapped SPI address to boot

pRegisters

Pointer to the OSPI peripherals base MMR address (not used on this product)

pRxDmaRegisters

Pointer to the DMA peripherals base MMR address for receive operations (not used on this product)

pTxDmaRegisters

Pointer to the DMA peripherals base MMR address, for transmit operations (not used on this product)

struct ADI_ROM_BOOT_UART

Structure Type Declaration: `ADI_ROM_BOOT_UART`

The UART slave boot mode specific structure.

This structure contains all the boot context information that is specific for the UART slave boot mode.

Table 47-62: ADI_ROM_BOOT_UART Members

Type	Name	Description
`uint32_t`	`nFlags`	Flags related to UART Boot mode
`ADI_UART_TypeDef*`	`pRegisters`	Pointer to the UART peripherals base MMR address, not used on this product
`ADI_DMA_TypeDef*`	`pRxDmaRegisters`	Pointer to the DMA peripherals base MMR address for receive operations, not used on this product
`ADI_DMA_TypeDef*`	`pTxDmaRegisters`	Pointer to the DMA peripherals base MMR address for transmit operations, not used on this product

nFlags

Flags related to UART Boot mode

pRegisters

Pointer to the UART peripherals base MMR address, not used on this product.

pRxDmaRegisters

Pointer to the DMA peripherals base MMR address for receive operations, not used on this product.

pTxDmaRegisters

Pointer to the DMA peripherals base MMR address for transmit operations, not used on this product.

struct ADI_ROM_OTP_BOOT_CFG

Structure Type Declaration: `ADI_ROM_OTP_BOOT_CFG`

The boot configuration object for storing further boot customization objects.

This is a 192-bit structure that is allocated to one contiguous region in the `OTP` memory array. The functionality allows for individual flags to enable or disable specific features of the boot process. Each flag is allocated in a separate 16-bit word so that each flag can be set at different times and the ECC information will not impact the setting of another flag.

Table 47-63: ADI_ROM_OTP_BOOT_CFG Members

Type	Name	Description
`uint32_t`	`lockMonitor:1`	Flag to enable the status checking of ADI private OTP area
`uint32_t`	`reserved0:15`	Reserved
`uint32_t`	`dstCompare: 1`	Flag to enable the data compare from the destination address for data block

Table 47-63: ADI_ROM_OTP_BOOT_CFG Members (Continued)

Type	Name	Description
uint32_t	reserved1:15	Reserved
uint32_t	pubkey0Inv:1	Invalidate public key0
uint32_t	reserved2:15	Reserved
uint32_t	pubkey1Inv:1	Invalidate public key1
uint32_t	reserved3:15	Reserved
uint32_t	privkey0Inv:1	Invalidate private key0
uint32_t	reserved4:15	Reserved
uint32_t	privkey1Inv:1	Invalidate private key1
uint32_t	reserved5:15	Reserved
uint32_t	privkey2Inv:1	Invalidate private key2
uint32_t	reserved6:15	Reserved
uint32_t	privkey3Inv:1	Invalidate private key3
uint32_t	reserved7:15	Reserved
uint32_t	dmcEn:1	Enables configuration of the DMC
uint32_t	reserved8:15	Reserved
uint32_t	dmcInv:1	Invalidates the DMC values
uint32_t	reserved9:15	Reserved
uint32_t	dmcDecDis:1	Disabled Decryption from off-chip memory
uint32_t	reserved10:31	Reserved

lockMonitor

Flag to enable the status checking of ADI private OTP area.

pubkey0Inv

Invalidate Public Key 0, use next public key for secure boot

dstCompare

Flag to enable the data compare from the destination address for data block

pubkey1Inv

Invalidate Public Key 1, Secure boot will no longer be operational as no further public keys

privkey0Inv

Invalidate Decryption Key 0, use next Decryption key for secure boot

privkey1Inv

Invalidate Decryption Key 1, use next Decryption key for secure boot

privkey2Inv

Invalidate Decryption Key 2, use next Decryption key for secure boot

privkey3Inv

Invalidate Decryption Key 3, once invalidated part will no longer be bootable

dmcEn

Enables configuration of the DMC from values in `OTP` stored in the format of the `ADI_ROM_OTP_DMC_CONFIG` object

dmcInv

Invalidates the DMC values in the `OTP` resulting is bypassing of DMC configuration

dmcDecDis

Disabled decryption from off-chip memory

struct ADI_ROM_OTP_BOOT_CGU_INFO

Structure Type Declaration: `ADI_ROM_OTP_BOOT_CGU_INFO`

The CGU configuration object located in `OTP` for configuration of the CGU by the boot software.

This is a 128-bit structure that is allocated to one contiguous region in the `OTP` memory array. Allows the boot software to configure the CGU for a more efficient boot process.

Table 47-64: ADI_ROM_OTP_BOOT_CGU_INFO Members

Type	Name	Description
`uint32_t`	`ctl_WEN:1` (bitfield)	Enable write to the `CGU_CTL` register
`uint32_t`	`div_WEN:1` (bitfield)	Enable write to the `CGU_DIV` register
`uint32_t`	`reserved0:1` (bitfield)	Reserved
`uint32_t`	`div_DSEL:5` (bitfield)	`CGU_DIV.DSEL` value
`uint32_t`	`div_CSEL:5` (bitfield)	`CGU_DIV.CSEL` value
`uint32_t`	`div_S0SEL:3` (bitfield)	`CGU_DIV.S0SEL` value
`uint32_t`	`div_SYSSEL:5` (bitfield)	`CGU_DIV.SYSSEL` value
`uint32_t`	`div_S1SEL:3` (bitfield)	`CGU_DIV.S1SEL` value
`uint32_t`	`div_OSEL:7` (bitfield)	`CGU_DIV.OSEL` value
`uint32_t`	`ctl_DF:1` (bitfield)	`CGU_CTL.DF` value

Table 47-64: ADI_ROM_OTP_BOOT_CGU_INFO Members (Continued)

Type	Name	Description
uint32_t	ctl_MSEL:7 (bitfield)	CGU_CTL.MSEL value
uint32_t	Reserved1:1 (bitfield)	Reserved
uint32_t	ctl_SISELEXEN:1 (bitfield)	CGU_CTL.S1SELEXEN value
uint32_t	auto_disable:1 (bitfield)	Disable auto alignment
uint32_t	clkoutsel_CLKOUTSEL:5 (bitfield)	CGU_CLKOUTSEL.CLKOUTSEL value
uint32_t	clkoutsel_WEN:1 (bitfield)	Enable write to the CGU_CLKOUTSEL register
uint32_t	Reserved2:8 (bitfield)	Reserved
uint32_t	divex_S1SELEX:8 (bitfield)	CGU_DIVEX.S1SELEX value
uint32_t	oscwctl0_WEN:1 (bitfield)	Enable write to the CGU_OSCWDCTL instance 0 register
uint32_t	oscwctl0_HODF:6 (bitfield)	CGU_OSCWDCTL.HODF value
uint32_t	oscwctl0_HODEN:1 (bitfield)	CGU_OSCWDCTL.HODEN value
uint32_t	oscwctl0_CNGEN:1 (bitfield)	CGU_OSCWDCTL.CNGEN value
uint32_t	oscwctl0_BOUF:5 (bitfield)	CGU_OSCWDCTL.BOUF value
uint32_t	oscwctl0_BOUEN:1 (bitfield)	CGU_OSCWDCTL.BOUEN value
uint32_t	oscwctl0_FAULTEN:1 (bitfield)	CGU_OSCWDCTL.FAULTEN value
uint32_t	oscwctl0_MONDIS:1 (bitfield)	CGU_OSCWDCTL.MONDIS value
uint32_t	oscwctl0_FAULTPINDIS:1 (bitfield)	CGU_OSCWDCTL.FAULTPINDIS value
uint32_t	reserved3:14 (bitfield)	Reserved
uint32_t	oscwctl1_WEN:1 (bitfield)	Enable write to the CGU_OSCWDCTL instance 1 register
uint32_t	oscwctl1_HODF:6 (bitfield)	CGU_OSCWDCTL.HODF value
uint32_t	oscwctl1_HODEN:1 (bitfield)	CGU_OSCWDCTL.HODEN value
uint32_t	oscwctl1_CNGEN:1 (bitfield)	CGU_OSCWDCTL.CNGEN value
uint32_t	oscwctl1_BOUF:5 (bitfield)	CGU_OSCWDCTL.BOUF value
uint32_t	oscwctl1_BOUEN:1 (bitfield)	CGU_OSCWDCTL.BOUEN value
uint32_t	oscwctl1_FAULTEN:1 (bitfield)	CGU_OSCWDCTL.FAULTEN value
uint32_t	oscwctl1_MONDIS:1 (bitfield)	CGU_OSCWDCTL.MONDIS value
uint32_t	oscwctl1_FAULTPINDIS:1 (bitfield)	CGU_OSCWDCTL.FAULTPINDIS value
uint32_t	reserved4:14 (bitfield)	Reserved

ctl_WEN

Enable write to the CGU_CTL register

div_WEN

Enable write to the CGU_DIV register

div_DSEL

CGU_DIV.DSEL value

div_CSEL

CGU_DIV.CSEL value

div_S0SEL

CGU_DIV.S0SEL value

div_SYSSEL

CGU_DIV.SYSSEL value

div_S1SEL

CGU_DIV.S1SEL value

div_OSEL

CGU_DIV.OSEL value

ctl_DF

CGU_CTL.DF value

ctl_MSEL

CGU_CTL.MSEL value

auto_disable

disable polling on auto-alignment of clocks, NOT RECOMMENDED!

clkoutsel_CLKOUTSEL

CGU_CLKOUTSEL.CLKOUTSEL value

clkoutsel_WEN

Enable write to the CGU_CLKOUTSEL register

oscwctl0_WEN

Enable write to the CGU_OSCWDCTL instance 0 register

oscwctl0_HODF

CGU_OSCWDCTL.HODF value

oscwctl0_HODEN

CGU_OSCWDCTL.HODEN value

oscwctl0_CNGEN

CGU_OSCWDCTL.CNGEN value

oscwctl0_BOUF

CGU_OSCWDCTL.BOUF value

oscwctl0_BOUEN

CGU_OSCWDCTL.BOUEN value

oscwctl0_FAULTEN

CGU_OSCWDCTL.FAULTEN value

oscwctl0_MONDIS

CGU_OSCWDCTL.MONDIS value

oscwctl0_FAULTPINDIS

CGU_OSCWDCTL.FAULTPINDIS value

oscwctl1_WEN

Enable write to the CGU_OSCWDCTL instance 1 register

oscwctl1_HODF

CGU_OSCWDCTL.HODF value

oscwctl1_HODEN

CGU_OSCWDCTL.HODEN value

oscwctl1_CNGEN

CGU_OSCWDCTL.CNGEN value

oscwctl1_BOUF

CGU_OSCWDCTL.BOUF value

oscwctl1_BOUEN

CGU_OSCWDCTL.BOUEN value

oscwctl1_FAULTEN

CGU_OSCWDCTL.FAULTEN value

oscwctl1_MONDIS

CGU_OSCWDCTL.MONDIS value

oscwctl1_FAULTPINDIS

CGU_OSCWDCTL.FAULTPINDIS value

struct ADI_ROM_OTP_BOOT_CMD_INFO

Structure Type Declaration: ADI_ROM_OTP_BOOT_CMD_INFO

The boot command object for storing a customized boot command for each boot mode within the OTP memory array.

This is a 160-bit structure that is allocated to one contiguous region in the OTP memory array. Allows the boot ROM software to pass a custom boot command to a specific boot mode which changes the default boot behavior on startup. Can be used to change the default UART instance used for a UART boot operation (for example).

Table 47-65: ADI_ROM_OTP_BOOT_CMD_INFO Members

Type	Name	Description
uint32_t	spiMasterBootCmd	SPI Master Boot Mode
uint32_t	spiSlaveBootCmd	SPI Slave Boot Mode
uint32_t	lpBootCmd	Link Port Target Boot Mode
uint32_t	uartBootCmd	UART Controller Boot Mode
uint32_t	ospiMasterBootCmd	OSPI Controller Boot Mode
uint32_t	emmcMasterBootCmd0	eMSI Controller Boot Mode
uint32_t	emmcMasterBootCmd1	eMSI Controller Boot Mode
uint32_t	lpddrBootCmd	Linkport Slave DDR Boot Mode

spiMasterBootCmd

SPI Master Boot Mode

spiSlaveBootCmd

SPI Slave Boot Mode

lpBootCmd

Link Port Target Boot Mode

uartBootCmd

UART Controller Boot Mode

ospiMasterBootCmd

OSPI Controller Boot Mode

emmcMasterBootCmd0

EMSI Master Boot Mode

emmcMasterBootCmd1

EMSI Master Boot Mode

lpddrBootCmd

Linkport Slave DDR Boot Mode

struct ADI_ROM_BOOT_EMSI

Structure Type Declaration: `ADI_ROM_BOOT_EMSI`

The EMSI boot mode specific structure.

This structure contains all the boot context information that is specific to the EMSI master boot mode.

Table 47-66: ROM_BOOT_EMSI Members

Type	Name	Description
`bool`	`bFirstLoadCall`	Indicates first call to EMSI load function
`bool`	`bFinalLoadCall`	Indicates final call to EMSI load function
`uint32_t`	`Reserved0`	reserved
`uint32_t[2]`	`nRead_Adma2_Des0`	Descriptor 0 which handles the Data load to buffer
`uint32_t[2]`	`nRead_Adma2_Des1`	Descriptor 1 which together with Descriptor 2 creates pause in data read by continuous linking
`uint32_t[2]`	`nRead_Adma2_Des2`	Descriptor 2 which together with Descriptor 1 creates pause in data read by continuous linking

bFirstLoadCall

Indicates first call to EMSI load function

bFinalLoadCall

Indicates first call to EMSI load function

nRead_Adma2_Des0

Descriptor 0 which handles the Data load to buffer

nRead_Adma2_Des1

Descriptor 1 which together with Descriptor 2 creates pause in data read by continuous linking

nRead_Adma2_Des2

Descriptor 2 which together with Descriptor 1 creates pause in data read by continuous linking

struct ADI_ROM_OTP_BOOT_INFO

Structure Type Declaration: `ADI_ROM_OTP_BOOT_INFO`

The 704-bit boot info object located in OTP for boot customization.

This is a 704-bit structure that is allocated to one contiguous region in the OTP memory array. The content in OTP is stored in the format of this structure so boot can read the contents directly into this object.

Table 47-67: ADI_ROM_OTP_BOOT_INFO Members

Type	Name	Description
`struct ADI_ROM_OTP_BOOT_CGU_INFO`	`cgu`	CGU Configuration information
`uint32_t`	`flashStartAddress`	Flash Start address for the SPI/OSPI master boot mode
`struct ADI_ROM_OTP_BOOT_CMD_INFO`	`bcmd`	Boot Command customization for each boot mode to change default boot peripheral instance and configuration
`struct ADI_ROM_OTP_BOOT_CMD_INFO`	`rdcr`	Additional OSPI configuration used for OSPI boot mode
`uint16_t`	`ospi_rdcr_reserved0`	Reserved
`struct ADI_ROM_OTP_BOOT_CFG`	`bcfg`	Additional boot configuration flags for key invalidation and to enable DMC configuration
`uint32_t`	`val14`	Reserved
`uint16_t`	`otpReadTiming`	Reserved, Must always be zero
`uint16_t`	`reserved1`	Reserved

cgu

CGU Configuration information

flashStartAddress

Flash start address for the SPI/OSPI master boot mode

bcmd

Boot Command customization for each boot mode to change the default boot peripheral instance and configuration

rdcr

Additional OSPI configuration info that is used for OSPI boot mode

bcfg

Additional boot configuration flags for key invalidation and for enabling DMC configuration

otpReadTiming

Reserved, Must always be zero

struct ADI_ROM_OTP_DMC_CONFIG

Structure Type Declaration: `ADI_ROM_OTP_DMC_CONFIG`

The 384-bit configuration object located in OTP for configuration of the DMC during preboot.

Makes use of the memories connected to the DMC peripheral during boot without the use of init codes in the boot stream. The settings can be applied to this object in the OTP memory. During preboot the boot software reads the object from OTP and configures the peripheral accordingly.

NOTE: When the device is open programs should avoid using OTP and instead use initcodes to initialize DMC because the initcode method is highly customizable. To lock the device to enable secure boot (to boot to memories interfaced to the DMC) the configuration must be provisioned to OTP because initcodes are not supported in secure boot.

Table 47-68: ADI_ROM_OTP_DMC_CONFIG Members

Type	Name	Description
`uint32_t`	`ulDDR_DLLCTLCFG`	Content of DDR DLLCTL and DMC_CFG register
`uint32_t`	`ulDDR_MR2MR3`	Content of the DDR MR2 and MR3 register
`uint32_t`	`ulDDR_CTL`	Content of the DDR Control register
`uint32_t`	`ulDDR_MRMR1`	Content of the DDR MR and MR1 register
`uint32_t`	`ulDDR_TR0`	Content of the DDR Timing register
`uint32_t`	`ulDDR_TR1`	Content of the DDR Timing register
`uint32_t`	`ulDDR_TR2`	Content of the DDR Timing register
`uint32_t`	`ulDDR_ZQCTL0`	Content of ZQCTL0 register
`uint32_t`	`ulDDR_ZQCTL1`	Content of ZQCTL1 register
`uint32_t`	`ulDDR_ZQCTL2`	Content of ZQCTL2 register
`uint32_t`	`ulDDRPHY_CACTL`	Content of DDRPHY_DDR_CA_CTL register
`uint32_t`	`uBypassDelay_LANE0CTL1:6`	Content of DDR_LANE0_CTL1[15:10] register
`uint32_t`	`uBypassDelay_LANE1CTL1:6`	Content of DDR_LANE1_CTL1[15:10] register
`uint32_t`	`uBypassDelay_LANE0CTL0:6`	Content of DDR_LANE0_CTL0[15:10] register
`uint32_t`	`uBypassDelay_LANE1CTL0:6`	Content of DDR_LANE1_CTL0[15:10] register
`uint32_t`	`reserved0:8`	Reserved

struct ROM_BOOT_DMA_INSTANCE

Structure Type Declaration: `ROM_BOOT_DMA_INSTANCE`

DMA Channel Instance.

Specifies the base MMR address of the DMA channel as well as trigger and interrupt IDs

Table 47-69: ROM_BOOT_DMA_INSTANCE Members

Type	Name	Description
`ADI_DMA_TypeDef *`	`pReg`	Pointer to the base address of the DMA channel MMR registers
`DMA_CHANn_TypeDef`	`eDmaChannelId`	The actual DMA channel ID in the system
`uint8_t`	`TriggerId`	The trigger ID associated with the DMA channel
`uint8_t`	`InterruptId`	The interrupt ID associated with the DMA channel

pReg

Pointer to the base address of the DMA channel MMR registers

eDmaChannelId

The actual DMA channel ID in the system

TriggerId

The trigger ID associated with the DMA channel

InterruptId

The interrupt ID associated with the DMA channel

struct ROM_BOOT_MDMA

Structure Type Declaration: `ROM_BOOT_MDMA`

MDMA Channels.

Provides access to all the MDMA channels and CRC peripherals supported by the processor

Table 47-70: ROM_BOOT_MDMA Members

Type	Name	Description
`ROM_BOOT_MDMA_REGS[PARAM_SYS0_NUM_MDMA_STREAMS]`	`Stream`	Array of MDMA channel configurations supported by the processor

Stream

Array of MDMA channel configurations supported by the processor

struct ROM_BOOT_MDMA_REGS

Structure Type Declaration: `ROM_BOOT_MDMA_REGS`

MDMA Channel Registers.

Contains the Source and Destination MDMA channel instances for access to the MMRs and interrupt and trigger information. Information is also provided on the CRC support of the MDMA channel and access is provided to the corresponding CRC peripheral.

Table 47-71: ROM_BOOT_MDMA_REGS Members

Type	Name	Description
`struct ROM_BOOT_DMA_INSTANCE`	`Src`	The source DMA Channel in the MDMA pair
`ROM_BOOT_DMA_INSTANCE`	`Dst`	The destination DMA Channel in the MDMA pair
`ADI_CRC_TypeDef *`	`pCrc`	The base MMR address of the associated CRC peripheral if one exists
`enum ROM_BOOT_MDMA_CRC_SUPPORT`	`eCrcSupport`	Indicates if the MDMA channel supports CRC or not

Src

The source DMA Channel in the MDMA pair

Dst

The destination DMA Channel in the MDMA pair

pCrc

The base MMR address of the associated CRC peripheral if one exists

eCrcSupport

Indicates if the MDMA channel supports CRC or not

struct ROM_DMA_MDMA_CONFIG

Structure Type Declaration: `ROM_DMA_MDMA_CONFIG`

MDMA Configuration Object.

The configurable structure for controlling the MDMA operation supplied to the `adi_rom_MemDma()` routine.

Table 47-72: ROM_DMA_MDMA_CONFIG Members

Type	Name	Description
enum ROM_DMA_MDMA_OPERATION	`eOperation`	Type of operation to perform
`enum ROM_DMA_MDMA_ID`	`eId`	MDMA Channel ID
`void *`	`pSource`	Source Pointer

Table 47-72: ROM_DMA_MDMA_CONFIG Members (Continued)

Type	Name	Description
void *	pDestination	Destination Pointer
uint32_t	ByteCount	Byte Count
enum ROM_DMA_DONE_DETECT_METHOD	eDoneDetect	DMA Done Detection Method
uint32_t	CrcCtl	CRC_CTL value when CRC operations are required
uint32_t	FillVal	Fill value for memory fill operations
uint32_t	CrcPoly	CRC Polynomial for CRC operations
uint32_t	CrcCompare	Value used for CRC compare operations or for a CRC32 result compare

eOperation

Type of operation to perform

eId

MDMA Channel ID

pSource

Source Pointer

pDestination

Destination Pointer

ByteCount

Byte Count

eDoneDetect

DMA Done Detection Method

CrcCtl

CRC_CTL value when CRC operations are required

FillVal

Fill value for memory fill operations

CrcPoly

CRC Polynomial for CRC operations

CrcCompare

Value used for CRC compare operations or for a CRC32 result compare

struct ROM_DMA_PDMA_CONFIG

Structure Type Declaration: ROM_DMA_PDMA_CONFIG

PDMA Configuration Object.

The user configurable structure for controlling the PDMA operation via the adi_rom_PeriphDma() function.

Table 47-73: ROM_DMA_PDMA_CONFIG Members

Type	Name	Description
enum ROM_DMA_PDMA_OPERATION	eOperation	Type of operation to perform
ADI_DMA_TypeDef volatile *	pRegs	Pointer to the base address of the DMA channel MMR registers
uint16_t	dataWidth	The maximum supported data width of the DMA channel. Used to configure the DMA_CFG.PSIZE PSIZE field in DMA_CFG
uint16_t	dstModifyMult	The modify multiplier, usually set to 1
void *	pSource	Source Pointer used for transmit operations
void *	pDestination	Destination Pointer used for receive operations
uint32_t	byteCount	Number of bytes to transfer
enum ROM_DMA_DONE_DETECT_METHOD	eDoneDetect	DMA Done Detection method used for the transfer
enum ROM_DMA_DONE_DETECT_METHOD	loadType	Defines whether the load function DMA waits for the load to complete or returns to the kernel

eOperation

Type of operation to perform

pRegs

Pointer to the base address of the DMA channel MMR registers

dataWidth

The maximum supported data width of the DMA channel. Used to configure the DMA_CFG.PSIZE PSIZE field in DMA_CFG

dstModifyMult

The modify multiplier, usually set to 1

pSource

Source Pointer used for transmit operations

pDestination

Destination Pointer used for receive operations

byteCount

Number of bytes to transfer

eDoneDetect

DMA Done Detection method used for the transfer

loadType

Defines whether the load function DMA waits for the load to complete or returns to the kernel

struct OTP_DATA

Structure Type Declaration: `otp_data`

Any pointers that are `NULL` will result in the object not being written. Any data values of 0 will not be written.

Table 47-74: otp_data Members

Type	Name	Description
`void (*)`	`reserved`	
`uint32_t(*) [ROM_OTP_SZ_huk]`	`huk`	Pointer to 256-bit Hardware Unique Key
`uint32_t(*) [ROM_OTP_SZ_rkek]`	`rkek`	Pointer to 128-bit Root Key Encryption Key
`uint 32_t(*) [ROM_OTP_SZ_dek]`	`dek`	Pointer to 128-bit Local Encryption Key
`uint32_t(*) [ROM_OTP_SZ_oem_public _key]`	`oem_public_key`	Pointer to 512-bit OEM Public Key
`uint32_t(*) [ROM_OTP_SZ_pvt_ 128key0]`	`pvt_128key0`	Pointer to 128-bit AES Key
`uint32_t(*) [ROM_OTP_SZ_pvt_ 128key1]`	`pvt_128key1`	Pointer to 128-bit AES Key
`uint32_t(*) [ROM_OTP_SZ_pvt_ 128key2]`	`pvt_128key2`	Pointer to 128-bit AES Key

Table 47-74: otp_data Members (Continued)

Type	Name	Description
uint32_t(*) [ROM_OTP_SZ_pvt_128key3]	pvt_128key3	Pointer to 128-bit AES Key
uint32_t(*) [ROM_OTP_SZ_ek]	ek	Pointer to 256-bit endorsement key
uint32_t(*) [ROM_OTP_SZ_secure_emu_key0]	secure_emu_key0	Pointer to 128-bit Secure Debug Key
uint32_t(*) [ROM_OTP_SZ_secure_emu_key1]	secure_emu_key1	Pointer to 128-bit Secure Debug Key
uint32_t	emu_key0_disable :16 (bitfield)	Secure Debug key disable
uint32_t	emu_key1_disable :16 (bitfield)	Secure Debug key disable
uint32_t(*) [ROM_OTP_SZ_public_key0]	public_key0	Pointer to 512-bit public key used for boot stream authentication
uint32_t(*) [ROM_OTP_SZ_public_key1]	public_key1	Pointer to 512-bit public key used for boot stream authentication
uint32_t(*) [ROM_OTP_SZ_boot_info]	boot_info	Pointer to 608-bit boot customization structure, see also struct ADI_ROM_OTP_BOOT_INFO
uint8_t	antiroll_nv_cntr	Anti-rollback counter to prevent loading of older firmware during secure boot.
uint32_t(*) [ROM_OTP_SZ_gp1]	gp1	Pointer to 512-bit General purpose user space
uint32_t	bootModeDisable: 8 (bitfield)	Boot mode disable for permanently disabling specific boot modes
uint32_t(*) [ROM_OTP_SZ_preboot_ddr_cfg]	preboot_ddr_cfg	Pointer to 384-bit DMC configuration. See also struct ADI_ROM_OTP_BOOT_INFO
uint32_t(*) [ROM_OTP_SZ_stageID]	stageID	Pointer to 64-bit staging ID

huk

Pointer to 256-bit Hardware Unique Key

rkek

Pointer to 128-bit Root key Encryption Key

dek

Pointer to 128-bit Local Encryption Key

oem_public_key

Pointer to 512-bit OEM Public Key

pvt_128key0

Pointer to 128-bit AES Key

pvt_128key1

Pointer to 128-bit AES Key

pvt_128key2

Pointer to 128-bit AES Key

pvt_128key3

Pointer to 128-bit AES Key

ek

Pointer to 256-bit endorsement key

secure_emu_key0

Pointer to 128-bit Secure Debug Key

secure_emu_key1

Pointer to 128-bit Secure Debug Key

emu_key0_disable

Secure debug key disable

emu_key1_disable

Secure debug key disable

public_key0

Pointer to 512-bit public key used for boot stream authentication

public_key1

Pointer to 512-bit public key used for boot stream authentication

boot_info

Pointer to 608-bit boot customization structure, see also `struct ADI_ROM_OTP_BOOT_INFO`

antiroll_nv_cntr

Anti-rollback counter to prevent the loading of older firmware during a secure boot.

The counter supports values of 0 through 31. The counter feature is disabled when the counter is initially set to zero.

gp1

Pointer to 512-bit General purpose user space

bootModeDisable

Boot mode disable for permanently disabling specific boot modes

preboot_ddr_cfg

Pointer to 384-bit DMC Configuration. See also `struct ADI_ROM_OTP_DMC_CONFIG`

stageID

Pointer to 64-bit staging ID

Enumerations

The programming model for booting the processor uses the enumerations defined in this section.

enum ADI_ROM_BOOT_KEY_TYPE

Enumeration Type Declaration: `ADI_ROM_BOOT_KEY_TYPE`

Indicates when custom security keys are used for evaluation of secure boot.

By default the boot process fetches all security keys from OTP for use during secure boot. Custom security lets the program set their custom keys in the `struct ADI_ROM_BOOT_CONFIG` item (not from OTP). Secure boot is evaluated using the `adi_rom_Boot()` function without provisioning keys in OTP.

Table 47-75: ADI_ROM_BOOT_KEY_TYPE Members

Enumerator	Description
`ADI_ROM_CUSTOM_SECURITY`	Enable use of custom security keys for authentication and decryption

ADI_ROM_CUSTOM_SECURITY

Enable use of custom security keys for authentication and decryption

enum ADI_ROM_BOOT_TYPE

Enumeration Type Declaration: `ADI_ROM_BOOT_TYPE`

Indicate to the boot kernel (in an open processor) when a secure or non-secure boot is required.

The boot kernel defaults to a secure boot unless the boot structure is configured as a Non-Secure Boot.

Table 47-76: ADI_ROM_BOOT_TYPE Members

Enumerator	Description
ADI_ROM_SECURE_BOOT_DIS	Non-Secure Boot
ADI_ROM_SECURE_BOOT	Secure Boot
ADI_ROM_SECURE_BOOT_HASH_ONLY	Boot kernel supports for hash compare only if this boot type selected using the ROM API. ECDSA verification is skipped.

ADI_ROM_SECURE_BOOT_DIS

Non-Secure Boot

ADI_ROM_SECURE_BOOT

Secure Boot

ADI_ROM_SECURE_BOOT_HASH_ONLY

Boot kernel supports for hash compare only if this boot type selected using ROM API. ECDSA verification is skipped.

enum ROM_BOOT_MDMA_CRC_SUPPORT

Enumeration Type Declaration: ROM_BOOT_MDMA_CRC_SUPPORT

MDMA Channel CRC support.

Specifies whether the MDMA channel supports CRC operations or not

Table 47-77: ROM_BOOT_MDMA_CRC_SUPPORT Members

Enumerator	Description
ROM_BOOT_DMA_CRC_SUPPORTED	MDMA Channel supports CRC
ROM_BOOT_DMA_CRC_NOT_SUPPORTED	MDMA Channel does not support CRC

ROM_BOOT_DMA_CRC_SUPPORTED

MDMA Channel supports CRC

ROM_BOOT_DMA_CRC_NOT_SUPPORTED

MDMA Channel does not support CRC

enum ROM_BOOT_RESULT

Enumeration Type Declaration: `ROM_BOOT_RESULT`

General Boot ROM Return Types.

Used throughout the boot software to indicate various success and failure events that can occur during boot.

Table 47-78: ROM_BOOT_RESULT Members

Enumerator	Description
ROM_BOOT_FAILURE	Failure
ROM_BOOT_SUCCESS	Success
ROM_BOOT_HDR_CHKSUM_ERR	Boot Stream Block Header Checksum Error
ROM_BOOT_HDR_SIGN_ERR	Boot Stream Block Header Sign Failure
ROM_BOOT_HDR_DEST_ERR	Boot Stream Block Payload Destination Error
RESERVED0	Reserved
RESERVED1	Reserved
RESERVED2	Reserved
RESERVED3	Reserved
ROM_BOOT_CGU_WRITE_ERR	CGU Write Error
ROM_BOOT_DMA_SUCCESS	DMA operation was successful
ROM_BOOT_DMA_FAILURE	DMA Failure.
ROM_BOOT_DMA_ACTIVE	DMA Channel is Active
ROM_BOOT_DMA_CONFIG_ERR	DMA configuration error
ROM_BOOT_MDMA_ID_ERR	Illegal MDMA Channel ID
ROM_BOOT_MDMA_OPERATION_ERR	Illegal MDMA operation Specified
ROM_BOOT_MDMA_CONFIG_ERR	Memory DMA configuration error
ROM_BOOT_MDMA_SRC_ERR	MDMA Source Channel Configuration Error
ROM_BOOT_MDMA_DST_ERR	MDMA Destination Channel Configuration Error
ROM_BOOT_MDMA_DONE_DETECT_ERR	Memory DMA Completed with errors
ROM_BOOT_MDMA_SUCCESS	Memory DMA Completed successfully
ROM_BOOT_PDMA_CONFIG_ERR	Peripheral DMA configuration invalid
ROM_BOOT_PDMA_SUCCESS	Peripheral DMA completed successfully
ROM_BOOT_CRC_FAILURE	MDMA CRC32 Failure
ROM_BOOT_CRC_CONFIG_ERR	MDMA CRC32 CONFIG error
ROM_BOOT_CRC_COUNT_ERR	CRC Byte Count was not a multiple 4

Table 47-78: ROM_BOOT_RESULT Members (Continued)

Enumerator	Description
ROM_BOOT_CRC_SUPPORTED_ERR	CRC Not Supported Error.
ROM_BOOT_CRC_INITCODE_ERR	CRC32 Enable Failure During Boot
ROM_BOOT_CRC_CALLBACK_ERR	Error in Execution of the CRC Callback
ROM_BOOT_CRC_SUCCESS	MDMA CRC Success

ROM_BOOT_FAILURE

Failure.

General failure can be used to indicate any general failure throughout the boot process

ROM_BOOT_SUCCESS

Success.

General success can be used to indicate any general functional success for an operation during the boot process.

NOTE: This must be the return result for a boot mode drivers initialization, configuration, load and cleanup routines when overriding their functionality in second stage boot loaders to use custom functions.

ROM_BOOT_HDR_CHKSUM_ERR

Boot Stream Block Header Checksum Error.

Indicates that the 8-bit XOR checksum of the 16-byte block header failed to generate the expected result.

ROM_BOOT_HDR_SIGN_ERR

Boot Stream Block Header Sign Failure.

The 0xAD block required as byte 4 of the boot block header was not found.

ROM_BOOT_HDR_DEST_ERR

Boot Stream Block Payload Destination Error.

The target address field of the block header indicates that the payload for the block is destined towards an address that is not supported. Indicates an attempt to load data to the reserved 8 KB non-bootable region of memory reserved by the boot process.

ROM_BOOT_CGU_WRITE_ERR

CGU Write Error

Returned by the CGU configuration routine if a CGU error is set during its initialization from settings provisioned in the OTP.

ROM_BOOT_DMA_SUCCESS

DMA operation was successful

ROM_BOOT_DMA_FAILURE

DMA Failure.

Returned by the DMA routines if an error was detected in the `DMA_STAT.IRQERR` prior to setting up a new DMA operation with the newly supplied configuration.

ROM_BOOT_DMA_ACTIVE

DMA Channel is Active

Returned only by the peripheral DMA routine when an attempt to run another peripheral DMA operation is attempted and the DMA channel is already running.

NOTE: This is not currently implemented for MDMA operations.

ROM_BOOT_DMA_CONFIG_ERR

DMA configuration error

ROM_BOOT_MDMA_ID_ERR

Illegal MDMA Channel ID.

Returned by `adi_rom_MemDma()` if the MDMA channel ID is not supported. For supported channel IDs, see enum ROM_DMA_MDMA_ID

ROM_BOOT_MDMA_OPERATION_ERR

Illegal MDMA operation Specified.

Returned by `adi_rom_MemDma()` if the MDMA operation is not supported. For supported operations, see `enum ROM_DMA_MDMA_OPERATION`

ROM_BOOT_MDMA_CONFIG_ERR

Memory DMA configuration error

ROM_BOOT_MDMA_SRC_ERR

MDMA Source Channel Configuration Error.

Set by the MDMA routines if after configuring the MDMA source channel to start a DMA operation, an error is generated in the source channels `DMA_STAT.IRQERR`

ROM_BOOT_MDMA_DST_ERR

MDMA Destination Channel Configuration Error.

Set by the MDMA routines when an error is generated in the destination channels `DMA_STAT.IRQERR` bit (after configuring the MDMA source channel to start a DMA operation)

ROM_BOOT_MDMA_DONE_DETECT_ERR

Memory DMA Completed with errors

ROM_BOOT_MDMA_SUCCESS

Memory DMA Completed successfully

ROM_BOOT_PDMA_CONFIG_ERR

Peripheral DMA configuration invalid

ROM_BOOT_PDMA_SUCCESS

Peripheral DMA completed successfully

ROM_BOOT_CRC_FAILURE

MDMA CRC32 Failure.

Returned by the higher level `adi_rom_MemDma()` routine and the underlying `adi_rom_MemCompare()` routine if the CRC32 result of the block of data did not match the expected result.

ROM_BOOT_CRC_CONFIG_ERR

MDMA CRC32 CONFIG error

ROM_BOOT_CRC_COUNT_ERR

CRC Byte Count was not a multiple 4.

The CRC peripheral operates on 32-bit data only and as such all CRC operations must have a byte count that is a multiple of 4. This result is returned by the higher level `adi_rom_MemDma()` routine and the underlying `adi_rom_MemCompare()` and `adi_rom_MemFill()` routines if the byte count is not a multiple of 4 bytes.

ROM_BOOT_CRC_SUPPORTED_ERR

CRC Not Supported Error.

Returned by `adi_rom_MemDma()`, `adi_rom_MemFill()`, `adi_rom_MemCompare()` and `adi_rom_Crc32Poly()` if the supplied DMA configuration specified a MDMA channel that does not support CRC operations.

ROM_BOOT_CRC_INITCODE_ERR

CRC32 Enable Failure During Boot.

Returned by adi_rom_Crc32Init() if the boot process cannot enable the CRC32 functionality due to a `NULL struct ADI_ROM_BOOT_CONFIG` pointer or `NULL struct ADI_ROM_BOOT_HEADER` pointer located in `pHeader Pointer to the boot header storage location where all boot stream block headers eventually reside for processing by the kernel`

ROM_BOOT_CRC_CALLBACK_ERR

Error in Execution of the CRC Callback.

Returned by the default CRC callback function located in the boot ROM if any of the following conditions are met:

- The `struct ADI_ROM_BOOT_CONFIG` pointer passed to the callback is a `NULL` pointer
- The `struct ADI_ROM_BOOT_BUFFER` pointer to the buffer to run CRC validation is a `NULL` pointer
- The `ROM_CBFLAG_DIRECT` flag is not set in the supplied flags parameter indicating it was a direct callback
- The `struct ADI_ROM_BOOT_HEADER` pointer located in `pHeader Pointer to the boot header storage location where all boot stream block headers eventually reside for processing by the kernel` is a `NULL` pointer

ROM_BOOT_CRC_SUCCESS

MDMA CRC Success

enum ROM_CORE_ID

Enumeration Type Declaration: `ROM_CORE_ID`

Core ID.

An enumeration for referencing a particular core

Table 47-79: ROM_CORE_ID Members

Enumerator	Description
`ROM_CORE_ID0`	Core 0
`ROM_CORE_ID1`	Core 1
`ROM_CORE_ID2`	Core 2
`ROM_CORE_NUM_CORES`	Number of Cores

ROM_CORE_ID0

Core 0

ROM_CORE_ID1

Core 1

ROM_CORE_ID2

Core 2

ROM_CORE_NUM_CORES

Number of Cores

enum ROM_DMA_DONE_DETECT_METHOD

Enumeration Type Declaration: `ROM_DMA_DONE_DETECT_METHOD`

DMA Done Detection Method.

Specifies the method used to detect the completion of the requested DMA operation.

When a program requests a non-blocking DMA operation, separate software is required to check the status of the DMA channel. The boot ROM does not provide an API for this operation.

NOTE: Trigger mode is not supported on this product

Table 47-80: ROM_DMA_DONE_DETECT_METHOD Members

Enumerator	Description
`ROM_DMA_DONE_NON_BLOCKING`	Return without waiting for the DMA to complete
`ROM_DMA_DONE_POLL_IRQDONE`	Poll on the IRQDONE bit in the DMA Status register
`ROM_DMA_DONE_WAKEUP_TRIGGER`	Configure a trigger to wake up the core when complete

ROM_DMA_DONE_NON_BLOCKING

Return without waiting for the DMA to complete

ROM_DMA_DONE_POLL_IRQDONE

Poll on the IRQDONE bit in the DMA Status register

ROM_DMA_DONE_WAKEUP_TRIGGER

Configure a trigger to wake up the core on when done

enum ROM_DMA_MDMA_ID

Enumeration Type Declaration: `ROM_DMA_MDMA_ID`

MDMA Channel ID.

The ID of the Memory DMA channel. Used in the `struct ROM_DMA_MDMA_CONFIG` configuration to specify the Memory DMA channel to use for operations accessible using the `adi_rom_MemDma()` routine

Table 47-81: ROM_DMA_MDMA_ID Members

Enumerator	Description
`ROM_DMA_MDMA0`	Memory DMA Stream 0

Table 47-81: ROM_DMA_MDMA_ID Members (Continued)

Enumerator	Description
ROM_DMA_MDMA1	Memory DMA Stream 1
ROM_DMA_MDMA2	Memory DMA Stream 2
ROM_DMA_MDMA3	Memory DMA Stream 3
ROM_DMA_MEMDMA_END_COUNT	Number of Memory DMA Streams

ROM_DMA_MDMA0

Memory DMA Stream 0

ROM_DMA_MDMA1

Memory DMA Stream 1

ROM_DMA_MDMA2

Memory DMA Stream 2

ROM_DMA_MDMA3

Memory DMA Stream 3

ROM_DMA_MEMDMA_END_COUNT

Number of Memory DMA Streams

enum ROM_DMA_MDMA_OPERATION

Enumeration Type Declaration: ROM_DMA_MDMA_OPERATION

MDMA Operation.

The operation determines if only an MDMA is required, or whether a CRC operation must be used in conjunction with the MDMA.

Table 47-82: ROM_DMA_MDMA_OPERATION Members

Enumerator	Description
ROM_DMA_MEM_COPY	Standard MDMA transfer from a source to a destination
ROM_DMA_MEM_CRC	Performs a CRC32 MDMA read operation and compares the result with an expected result
ROM_DMA_MEM_FILL	Uses the CRC peripheral to perform a fill operation with a 32-bit value
ROM_DMA_MEM_COMPARE	Uses the CRC peripheral to compare data with a constant 32-bit value
ROM_DMA_CRC_LUT_INIT	Initializes the CRC LUT from the supplied CRC Polynomial

ROM_DMA_MEM_COPY

Standard MDMA transfer from a source to a destination

ROM_DMA_MEM_CRC

Performs a CRC32 MDMA read operation and compares the result with an expected result

ROM_DMA_MEM_FILL

Uses the CRC peripheral to perform a fill operation with a 32-bit value

ROM_DMA_MEM_COMPARE

Uses the CRC peripheral to compare data with a constant 32-bit value

ROM_DMA_CRC_LUT_INIT

Initializes the CRC LUT from the supplied CRC Polynomial

enum ROM_DMA_PDMA_OPERATION

Enumeration Type Declaration: `ROM_DMA_PDMA_OPERATION`

Table 47-83: ROM_DMA_PDMA_OPERATION Members

Enumerator	Description
`ROM_DMA_PERI_TX`	Peripheral Transmit Operation
`ROM_DMA_PERI_RX`	Peripheral Receive Operation

ROM_DMA_PERI_TX

Peripheral Transmit Operation

ROM_DMA_PERI_RX

Peripheral Receive Operation

enum ROM_GETADDR_VALUE

Enumeration Type Declaration: `ROM_GETADDR_VALUE`

Parameter for `adi_rom_GetAddress()` function to retrieve the address of a data object stored in the boot ROM.

Table 47-84: ROM_GETADDR_VALUE Members

Enumerator	Description
`ROM_GETADDR_CONSTANTS`	Retrieve the address of the `ROM_CONSTANTS_TYPE` object
`ROM_GETADDR_BMODE`	Retrieve the address of the lookup table sorting the default `adi_rom_boot()` parameters for each boot mode

Table 47-84: ROM_GETADDR_VALUE Members (Continued)

Enumerator	Description
ROM_GETADDR_MDMAREGS	Retrieve the address of the struct ROM_BOOT_MDMA_REGS object
ROM_GETADDR_SPILUT	Retrieve the address of the lookup table in the ROM describing the various SPI master boot BCODE configurations
ROM_GETADDR_ECDSA_DOMAIN	Retrieve the address of the domain parameters used for ECDSA

ROM_GETADDR_CONSTANTS

Retrieve the address of the ROM_CONSTANTS_TYPE object

ROM_GETADDR_BMODE

Retrieve the address of the lookup table storing the default adi_rom_boot() parameters for each boot mode

ROM_GETADDR_MDMAREGS

Retrieve the address of the struct ROM_BOOT_MDMA_REGS object

ROM_GETADDR_SPILUT

Retrieve the address of the lookup table in the rom describing the various SPI master boot BCODE configurations

ROM_GETADDR_ECDSA_DOMAIN

Retrieve the address of the domain parameters used for ECDSA

enum ROM_HOOK_CALL_CAUSE

Enumeration Type Declaration: ROM_HOOK_CALL_CAUSE

Passed to a user hook routine to indicate the reason of the call.

An optional hook routine is provided as a callback when calling a boot mode via adi_rom_Boot. This hook routine is called by the boot software first after the execution of the boot modes initialization routine then again after execution of the boot modes configuration routine. This parameter allows the users routine to identify at which point the call was made allowing the user to perform different actions for each call.

Table 47-85: ROM_HOOK_CALL_CAUSE Members

Enumerator	Description
ROM_HOOK_CALL_INIT_COMPLETE	Call was because of the completion of the boot modes initialization function
ROM_HOOK_CALL_CONFIG_COMPLETE	Call was because of the completion of the boot modes configuration function

Table 47-85: ROM_HOOK_CALL_CAUSE Members (Continued)

Enumerator	Description
ROM_HOOK_REG_COMPLETE	Call was because of the completion of the boot modes pre-register initialization

enum ROM_SB_IMAGE_TYPE

Enumeration Type Declaration: ROM_SB_IMAGE_TYPE

Secure Boot Image Types.

The secure boot header contains a type field for the secure boot image type, this enumeration provides a complete list of all image types.

NOTE: The secure boot process does not necessarily support all image types defined.

Table 47-86: ROM_SB_IMAGE_TYPE Members

Enumerator	Description
ROM_SB_IMAGE_UNKNOWN	Unknown Secure Boot image type, used by software to initialize the type before detection of boot image type takes place
ROM_SB_IMAGE_BLP	Plain text BLp secure boot image supports authentication only with no decryption
ROM_SB_IMAGE_BLW	Keywrapped BLw secure boot image supports authentication and decryption, boot stream decryption key wrapped in the secure header
ROM_SB_IMAGE_BLE	Not supported by any Secure boot products. Secure boot image with key stored in plain text form in the secure header
ROM_SB_IMAGE_BLX	BLx Secure boot image supports authentication and decryption, boot stream decryption key wrapped located in OTP
ROM_SB_IMAGE_UNSUPPORTED	Used by software to indicate any other unsupported image type

ROM_SB_IMAGE_UNKNOWN

Unknown Secure Boot image type, used by software to initialize the type before detection of boot image type takes place

ROM_SB_IMAGE_BLP

Plain text BLp secure boot image supports authentication only with no decryption

ROM_SB_IMAGE_BLW

Keywrapped BLw secure boot image supports authentication and decryption, boot stream decryption key wrapped in the secure header

ROM_SB_IMAGE_BLE

Not supported by any Secure boot products. Secure boot image with key stored in plain text form in the secure header

ROM_SB_IMAGE_BLX

`BLx` Secure boot image supports authentication and decryption, boot stream decryption key wrapped located in `OTP`

ROM_SB_IMAGE_UNSUPPORTED

Used by software to indicate any other unsupported image type

enum ROM_SPI_PROTOCOL

Enumeration Type Declaration: `ROM_SPI_PROTOCOL`

The SPI Protocol to use.

SPI Flash devices can support multiple protocols to send commands to the SPI flash. Commands are usually sent over the single-bit bus; however a number of newer devices also support sending commands over the dual- or quad-bit bus.

WARNING: In system reset type events where the processor attempts to reboot and the flash may not have been reset, using the DUALIO or QUADIO protocols for the command cycles risks the boot process being unable to communicate with the SPI flash. Do not enable these features on SPI Flash devices if they are also the primary boot source used for booting from hardware reset and system reset events.

Table 47-87: ROM_SPI_PROTOCOL Members

Enumerator	Description
`ROM_SPI_EXT_PROTOCOL`	Extended protocol where the command cycle is sent on the single bit bus
`ROM_SPI_DUALIO_PROTOCOL`	DualIO protocol where the command cycle is sent on the dual bit bus
`ROM_SPI_QUADIO_PROTOCOL`	QuadIO protocol where the command cycle is sent on the quad bit bus

ROM_SPI_EXT_PROTOCOL

Extended protocol where the command cycle is sent on the single bit bus

ROM_SPI_DUALIO_PROTOCOL

DualIO protocol where the command cycle is sent on the dual bit bus

ROM_SPI_QUADIO_PROTOCOL

QuadIO protocol where the command cycle is sent on the quad bit bus

48 System Crossbars (SCB)

A modern system on a chip (SoCs) contains multi-cores, memory controllers, security blocks, and other high speed peripherals. As system integration increases, SoCs need to provide bus connectivity that allows better throughput to reduce performance bottlenecks. While traditional point-to-point connection buses have performed well in smaller systems, there is a need to use advanced switching based bus architectures for efficient handling of data transfer between multiplicity of data sources and sinks in the system. Additionally, mixing various traffic types in the same SoC (for example control, communication over peripherals and computing) while sharing the same bus resources, create different requirements from the Quality of Service (QoS) perspective.

The system crossbars (SCB) are the fundamental building blocks of a switch-fabric style for (on-chip) system bus interconnection. The SCBs connect system bus requesters to system bus completers, providing concurrent data transfer between multiple bus requesters and multiple bus completers. The SCB architecture addresses the challenges described above. The SCB provides sustainable throughput for simultaneous transactions in the system with configurable quality of service for each type of transaction (traffic) as required. A hierarchical model, built from multiple SCBs, provides a power and area efficient system interconnect, which satisfies the performance and flexibility requirements of a specific system.

SCB Features

The SCBs provide the following features:

- Efficient, pipelined bus transfer protocol for sustained throughput
- Full-duplex bus operation for flexibility and reduced latency
- Concurrent bus transfer support to allow multiple bus requesters to access bus completers simultaneously
- Simple priority-based QoS based arbitration model
- Programmable quality of service

SCB Functional Description

The following sections provide a functional description of the SCB.

- SCB Architectural Concepts

SCB Architectural Concepts

This section describes the components of the SCB and the modules connected to it. The basic elements in the SCB are SCB requesters, completers, requester interfaces, and completer interfaces.

Requesters

The system bus requesters include the peripheral Direct Memory Access (DMA) channels. These include the Serial Port (SPORT) DMA, SPI DMAs, Memory-to-Memory DMA channels (MDMA), the L1 code fill block, and the processor cores.

Completers

Completers are SCB connections that are responding to transfer requests. Completers include MMR registers, memory units, and various peripherals depending upon individual configurations. Each system completer has its own latencies and response times.

SCB Block Diagram

The SCB architectural model is shown in the ***SCB Overview*** figure. This figure shows a high level representation of a basic SCB connecting n completers to x requesters. A variable number of requesters connect to a variable number of completers in each SCB. In this example, all SIs connect to all MIs as indicated by the lines connecting them.

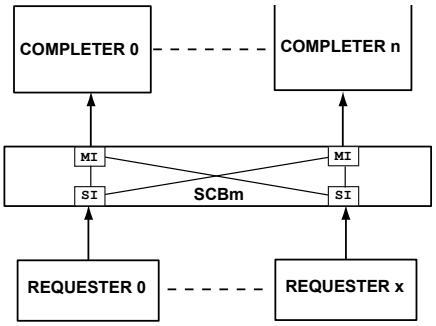


Figure 48-1: SCB Overview

Hierarchy Block Diagram

A system interconnect built from multiple SCBs in a hierarchical model is illustrated in the ***SCB Hierarchy Overview*** figure. The system requester node level SCBs requester connects multiple SIs to a single MI, which in turn connects to an SI of the system completer level node SCB.

As discussed above, all the requesters in the system are distributed across different SCBs. A given SCB at system requester node level connects directly to the system requesters. These SCBs connect to SCB0 through its SIs forming a hierarchal structure. While a requester must access any completer, its first access goes through the SCB it is connected to, and then through SCB0, to reach its intended completer. This simplifies the connecting hardware in the basic SCB block by limiting the requesters. Care must be taken when sharing requesters to allow adequate throughput for their individual data transfer requirements.

In this example, all SIs are connected to all MIs.

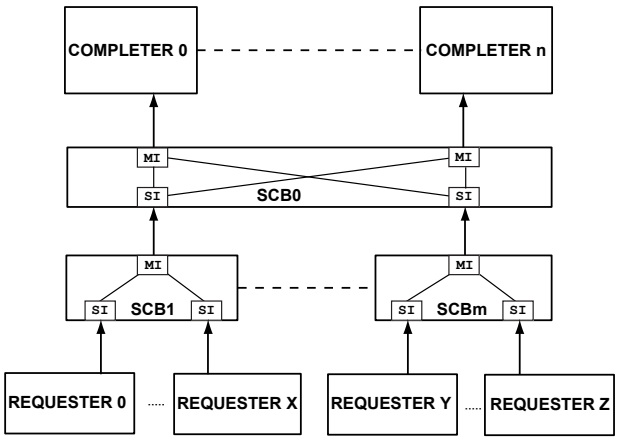


Figure 48-2: SCB Hierarchy Overview

NOTE:

SCB Block Diagram

The ***SCB Block Diagram*** shows the functional blocks of the SCB module.

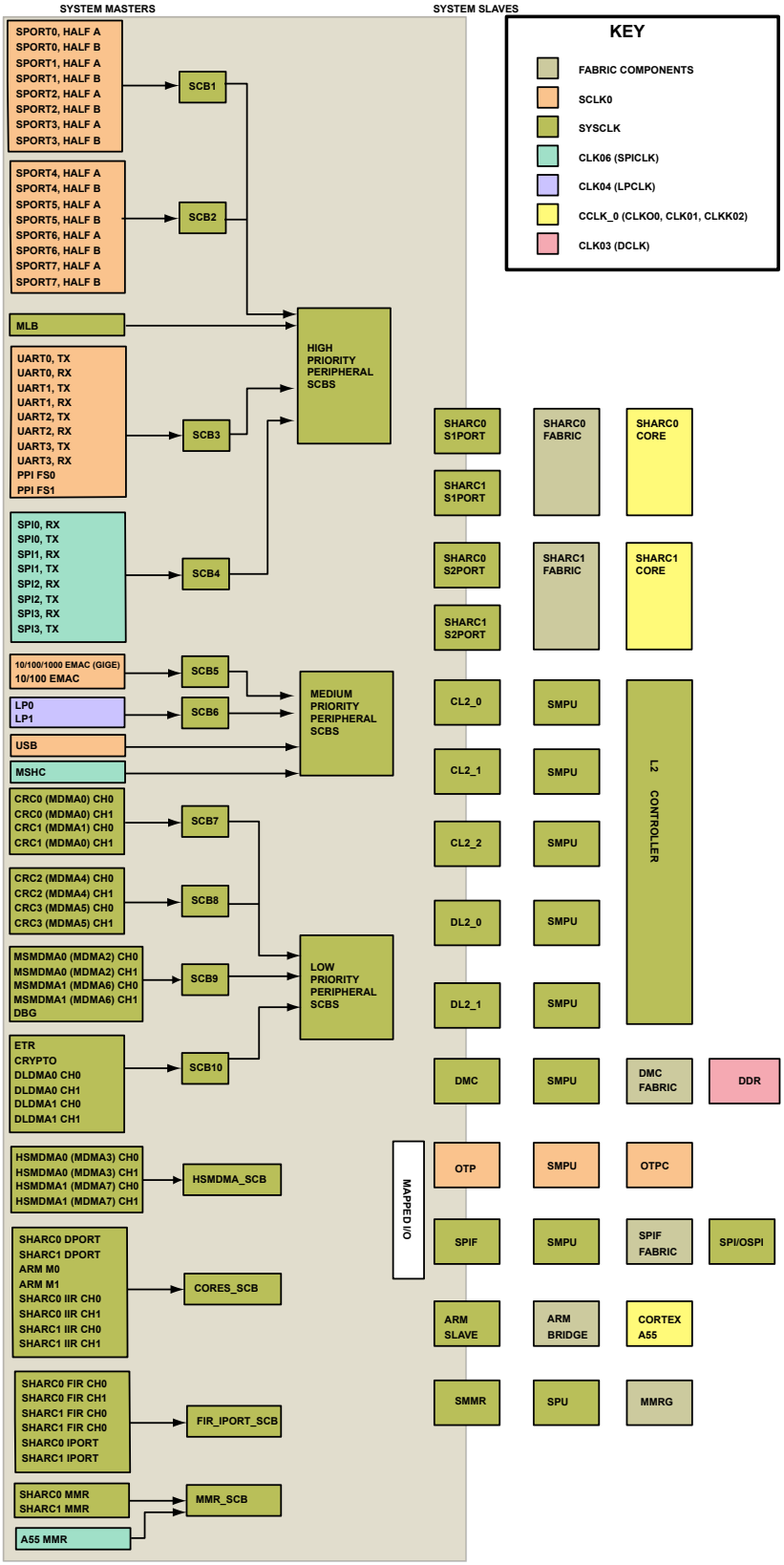


Figure 48-3: SCB Block Diagram

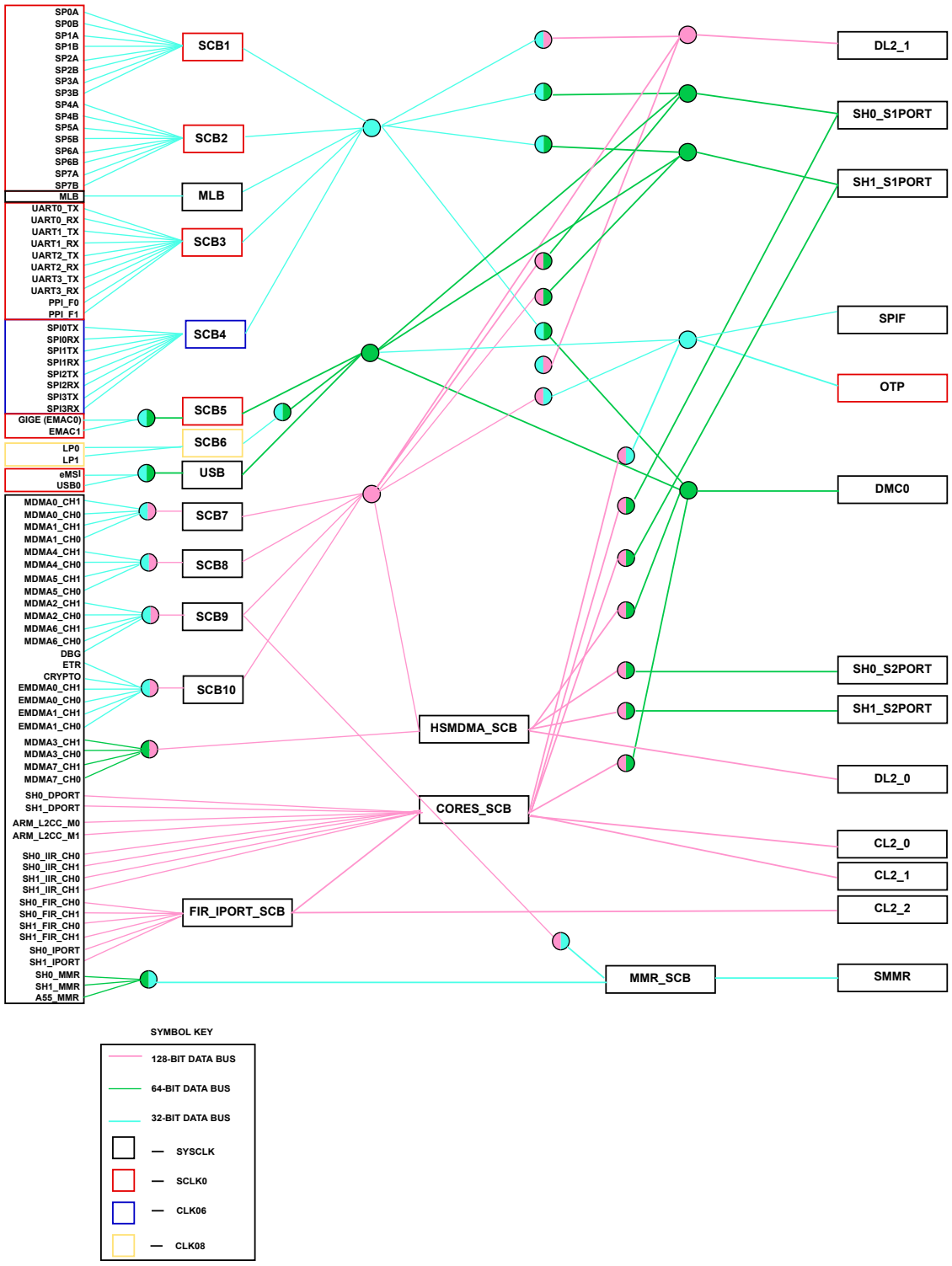


Figure 48-4: SCB Interconnections

The following are important points related to the system fabric on the ADSP-SC59x processors.

- The hierarchy of SCBs manages the system bus interconnections, multiplexing, and arbitration among the peripherals on the processor.

- The SCBs connections support DMA channels for some peripherals and support dedicated connections for others. The connections also support memory-mapped register access for internal memory (L1 and L2) and for external memory.

  The completer interface of the crossbar where requesters such as DMA connect to performs two functions. The first function is arbitration. SCBx handles arbitration. The second function is clock conversion. The programmable QoS registers can be viewed as being associated with the SCBx.

- Most of the peripherals and their SCBs are in the *SCLK0* domain. MDMA0-7, EMDMA0/1, Crypto, FIR/IIR accelerators, MLB, DBG, ETR, SHARC+ and Arm cores and their SCBs are in the *SYSCLK* domain. The link port is in the CLK04 domain. SPI is in the CLK06 domain.

- Each peripheral has a latency for access across the SCB. The latency varies with the nature of the peripheral. Also, the number of active peripherals (especially for cases where multiple peripherals are active on a shared SCB) affects SCB performance. Refer to the L2 System Memory chapter for details on the L2 memory organization.

  MDMA3, MDMA4, MDMA6 and MDMA7 channels are unidirectional. For example, MSMDMA0 CH0 is read-only and MSMDMA0 CH1 is write-only. Access to the MMR space of SCB0 is allowed only in secure mode. The MMR space of SCB0 has registers for programming the QoS of various requesters and controlling the clock domain crossing.

- Access to the GPV (Global Programmers View) space is allowed only in secure mode. This space has registers for programming the QoS and CDC relationships and the remapping of various requesters.

- The S2 completer port of SHARCs is intended primarily for MDMA between the L1 of the 2 SHARCs. Hence, only the HSMDMA0 (MDMA3) and HSMDMA1 (MDMA7) is given access to S2 ports while all other requesters are given access to S1 port. This arrangement also means that S1 and S2 port share the same address space.

- SPI2/OSPI0 is low bandwidth completer – SCLK0 with a 32-bit interface. Only MDMA0/1/2/4/5/6 is used to perform MDMA to these completer nodes. HSMDMA0/1 (MDMA4 and 7) are intended for L1, L2 and DDR (these completers have the same bandwidth capability as HSMDMA0/1).

- SPIF is a read-only flash. Peripherals are not able to read out of the SPI flash directly. Only MDMAs, crypto, cores, and debug tools are given access to it.

- LP to SPI2/OSPI0 access is allowed.

- The SPIF completer in the system fabric has remapping options in the SPIF fabric for SPI2 and OSPI0 in the same address space. The address map configuration is programmable using remap registers.

IMPORTANT: The following points are important changes from ADSP-2156x processors:

- All peripherals (including debugger) access L2 memory through the new DL2_1 subordinate port on the ADSP-SC59x processor. On the ADSP-2156x processor, these accesses were on the DL2_0 port of L2.

- To reduce the probability of port conflict in a multicore system, the cores are connected on the CL2_0 and CL2_1 ports of L2. CL2_0 has access only to the first 1 MB of L2. CL2_1 has access to second 1 MB of L2.
- Booting of the Arm core occurs through the CL2_0 port; booting of the SHARC cores occurs through CL2_1 (to avoid any conflict during multi-core boot). The respective BOOTROM addresses have been routed accordingly.

The following acronyms are used in the ***SCB Interconnections*** figure.

SCB0-10

Indicate SCB interfaces, connecting the system bus requesters and completers

SCLK0, SYSCLK, CLK03, CLK04, CLK06

Indicate clock domains in which the specific SCBs operate. For more information on clock domains, see the Clock Generation Unit (CGU) chapter and the product data sheet.

CDC

Indicates the clock domain crossing

CL2_0, CL2_1, CL2_2

Indicates a 128-bit L2 completer

DL2_0, DL2_1

Indicates a 128-bit L2 completer

L1C1_S1

Indicates the S1 completer of the SHARC1 processor

L1C1_S2

Indicates the S2 completer of the SHARC1 processor

SHARC1_IPORT

Indicates the SHARC1 processor instruction requester port

SHARC1_MMR

Indicates the SHARC1 MMR interface

A5_L2CC_M0/1

Indicates the Cortex A55 requesters (0 and 1)

Arm Completer

Indicates the Cortex A55 completer

SHARC Fabric with FIR/IIR Accelerator

The accelerator block on each SHARC core of ADSP-SC59x processor has one FIR and four IIR accelerators. All instances of the FIR and IIR accelerators operate at the core clock (CCLK) frequency. Accelerator requester ports can directly access the respective SHARC L1 memory with reduced latency. The access does not go through the system fabric. The SHARC core can also directly access the MMR registers of its respective accelerator. The accesses between one SHARC core and the accelerators that belong to the other SHARC core still go through the main system fabric.

The ***SHARC Fabric Connectivity*** figure shows a block diagram of the SHARC fabric with FIR/IIR accelerators integrated closely with the SHARC core.

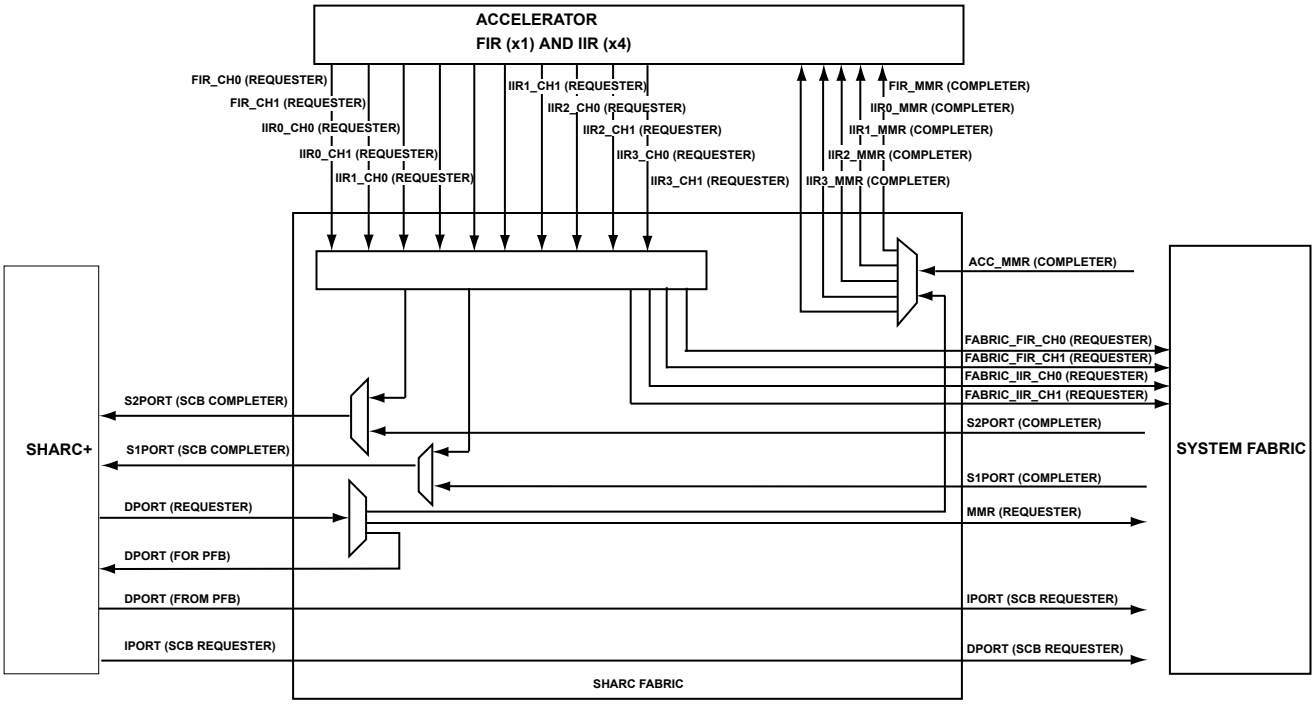


Figure 48-5: SHARC Fabric Connectivity

Note the following:

- The SHARC+ core can directly access the MMR space using its accelerators.

- The MMR port of the SHARC+ core is separate from the data port (DPORT). It is a separate requester port for the main system fabric.
- PFB (prefetch buffer) — after the MMR split, the DPORT comes into the SHARC+ core for the PFB connection and returns to the SHARC+ fabric again for the clock domain crossing.
- Accelerators share the same security level as the respective core. The SHARC+ core associated with the accelerator can always access the MMR space of the accelerator. Accesses from the other requesters such as other SHARC+ core/Arm Core/MSMDMA/DBG are allowed provided that the security requirements of the requester are met.

To ensure complete bandwidth utilization and optimal performance of the L2 ports, the controllers are allocated as shown in ***L2 Port Requester Allocation*** table.

Table 48-1: L2Port Requester Allocation

Requesters	CL2_0	CL2_1	CL2_2	DL2_0	DL2_1
SHARC0 (Core 1)	✓	✓			
SHARC1 (Core 2)	✓	✓			
A55 (Core 0)	✓	✓			
SHARC0 IPORT			✓		
SHARC1 IPORT			✓		
SHARC0_FIR_CH0			✓		
SHARC0_FIR_CH1			✓		
SHARC0_IIR_CH0	✓	✓			
SHARC0_IIR_CH1	✓	✓			
SHARC1_FIR_CH0			✓		
SHARC1_FIR_CH1			✓		
SHARC1_IIR_CH1	✓	✓			
SHARC1_IIR_CH1	✓	✓			
HSMDMA0				✓	
HSMDMA0				✓	
Peripherals					✓

Note the following:

- To reduce the probability of port conflict in a multicore system, the cores are connected on the CL2_0 and CL2_1 ports of L2. CL2_0 has access only to the first 1 MB of L2. CL2_1 has access to second 1 MB of L2
- To avoid conflict during multi-core boot, the booting of the Arm core happens through the CL2_0 port while the booting of the SHARC cores happens through the CL2_1 port. The respective bootROM addresses are routed accordingly.

The addressable range for each of the ports is shown in the ***L2 Port Address Allocation*** table.

Table 48-2: L2 Port Address Allocation

	Memory Address Range			
SMPU Instance	Data		Instruction	
	Start Address	End Address	Start Address	End Address
CL2_0	0x2000_0000	0x200F_FFFF	0x2020_0000	0x2020_FFFF
DL2_0	0x2000_0000	0x2022_FFFF		
CL2_1	0x2010_0000	0x201F_FFFF	0x2021_0000	0x2022_FFFF
CL2_2	0x2000_0000	0x2022_FFFF		
DL2_1	0x2000_0000	0x2022_FFFF		

To support exclusive accesses to the 5 port L2 memory on the processor, the configuration of SMPUs as shown in the ***SMPU Access Scheme for L2 Ports*** figure is used.

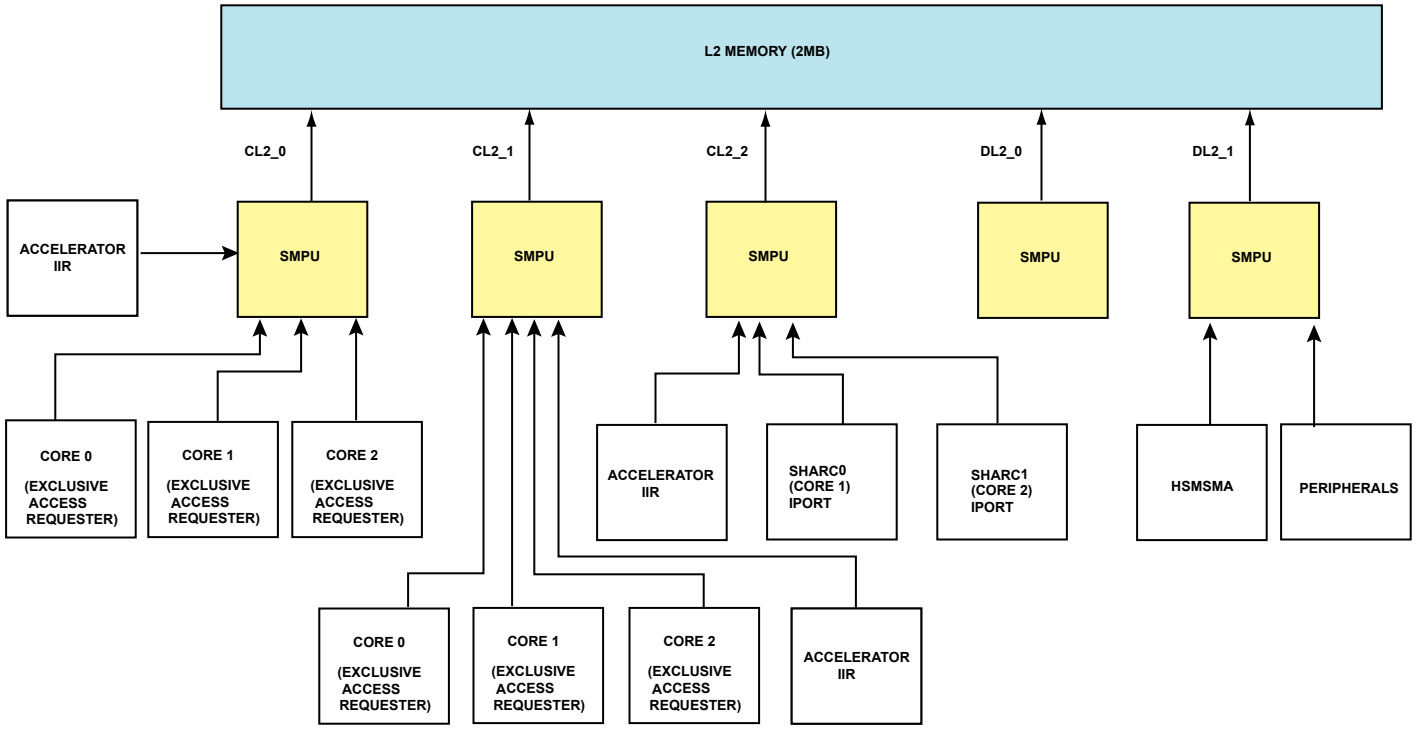


Figure 48-6: SMPU Access Scheme for L2 Ports

Table 48-3: SCB Controlled DMA Channel Peripherals

	Requesters	DDE DMA Channels	Non DDE DMA Channels
SCB1	SPORT0, HALF A	DMA0	
	SPORT0, HALF B	DMA1	

Table 48-3: SCB Controlled DMA Channel Peripherals (Continued)

	Requesters	DDE DMA Channels	Non DDE DMA Channels
	SPORT1, HALF A	DMA2	
	SPORT1, HALF B	DMA3	
	SPORT2, HALF A	DMA4	
	SPORT2, HALF B	DMA5	
	SPORT3, HALF A	DMA6	
	SPORT3, HALF B	DMA7	
SCB2	SPORT4, HALF A	DMA10	
	SPORT4, HALF B	DMA11	
	SPORT5, HALF A	DMA12	
	SPORT5, HALF B	DMA13	
	SPORT6, HALF A	DMA14	
	SPORT6, HALF B	DMA15	
	SPORT7, HALF A	DMA16	
	SPORT7, HALF B	DMA17	
SCB3	UART0, TX	DMA20	
	UART0, RX	DMA21	
	UART1, TX	DMA34	
	UART1, RX	DMA35	
	UART2, TX	DMA37	
	UART2, RX	DMA38	
	UART3, TX	DMA53	
	UART3, RX	DMA54	
	PPI F0	DMA28	
	PPI F1	DMA29	
SCB4	SPI0, TX	DMA22	
	SPI0, RX	DMA23	
	SPI1, TX	DMA24	
	SPI1, RX	DMA25	
	SPI2, TX	DMA26	
	SPI2, RX	DMA27	
	SPI3, TX	DMA55	

Table 48-3: SCB Controlled DMA Channel Peripherals (Continued)

	Requesters	DDE DMA Channels	Non DDE DMA Channels
	SPI3, RX	DMA56	
SCB5	10/100/1000 EMAC (GIGE)	N/A	6 channels (3 receive and 3 transmit)
	10/100 EMAC	N/A	2 channels (1 receive and 1 transmit)
SCB6	LP0	DMA30	
	LP1	DMA36	
SCB7	CRC0/MDMA0	DMA8, DMA9	
	CRC1/MDMA1	DMA18, DMA19	
SCB8	CRC2/MDMA4	DMA45, DMA46	
	CRC3/MDMA5	DMA47, DMA48	
SCB9	MDMA2	DMA39, DMA40	
	MDMA6	DMA49, DMA50	
SCB10	ETR	N/A	N/A
	CRYPTO	N/A	2 channels (1 read, 1 write)
	EMDMA0	N/A	2 channels (1 read, 1 write)
	EMDMA1	N/A	2 channels (1 read, 1 write)
HSMDMA_SCB	MDMA3	DMA43, DMA44	
	MDMA7	DMA51, DMA52	
CORES_SCB	SHARC0 DPORT	N/A	N/A
	SHARC1 DPORT	N/A	N/A
	ARM_L2CC_M0	N/A	N/A
	ARM_L2CC_M1	N/A	N/A
	SHARC0 IIR CH0, CH1	N/A	N/A
	SHARC1 IIR CH0, CH1	N/A	N/A
FIR_IPORT_SCB	SHARC0 FIR CH0, CH1	N/A	N/A
	SHARC1 FIR CH0, CH1	N/A	N/A
	SHARC0 IPORT	N/A	N/A
	SHARC1 IPORT	N/A	N/A

Table 48-3: SCB Controlled DMA Channel Peripherals (Continued)

	Requesters	DDE DMA Channels	Non DDE DMA Channels
MMR_SCB	SHARC0 MMR	N/A	N/A
	SHARC1 MMR	N/A	N/A
	A55 MMR	N/A	N/A
	USB0	N/A	16 channels (8 IN and 8 OUT)
	eMSI	N/A	1 Tx/Rx DMA channel

There are two types of peripherals that use DMA. The first have dedicated DMA channels controlled by the Dedicated DMA Engine (DDE) and have the same operating modes (see DMA Operating Modes) and use the same programming model (DMA Channel Programming Model). The second type is not controlled by the DDE module. These peripherals have their own operating modes and programming models (see the peripheral chapter for this information). The peripheral types are shown in the ***SCB-Controlled DMA Channel Peripherals*** table.

System Crossbars

The System Crossbars (SCB) are the fundamental building blocks of the system bus interconnect. The SCB (often referred to as the system interconnect fabric), is a collection of inter-connection units connecting system requesters to completer memory spaces. The SCB connects one or more requester devices to one or more memory-mapped completer devices. Each connected requester can be a core that originates an SCB transaction, or a requester interface of an upstream SCB cascaded interconnect. Each connected completer can be the final completer of an SCB transaction or a completer interface of a downstream cascaded SCB interconnect (forming a hierarchy of SCBs).

Each SCB that has multiple requesters and completers share the total bandwidth of the SCB. (In a M:N configuration where M requesters are connected to N completers through the SCBx.)

The SCB provides separate channels for reads and writes. Read and write accesses through a given SCB do not share bandwidth. Only SCB0, which is the major SCB in the SCB hierarchy, has the multiple paths between multiple requester and completer interfaces.

All other SCBs in the chip connect to SCB0 through different completer interfaces. Other primary requesters (DMAs, cores, and so on) in the system are distributed across these small SCBs. For a given SCB, all the requester and completers share the total bandwidth of the SCB. (Only SCB0 is the exception). Since different DMA channels are scattered across different SCBs (SCB1, SCB2 SCB3, and so on), they do not conflict for the bandwidth when they are in different SCBs and accessing different completers. SCB0 allows for concurrent data transfer between multiple bus requesters and multiple bus completers, providing flexibility, and full-duplex operation.

If system accesses are carefully architected, SCB has a potential of providing sufficient sustained bandwidth in the end system.

Since the SCBs support burst transfers, it is important to configure the requesting requester appropriately to make best use of available SCB bandwidth. For a DMA requester, choosing the appropriate `DMA_CFG.MSIZE` value, is important from both a functional and a performance perspective. The value in the `DMA_CFG.PSIZE` bit field determines the width of the peripheral bus in use. It can be configured to 1-byte, 2-bytes, or 4-bytes. The

DMA_CFG.MSIZE value determines the actual size of the SCB bus in use. It also determines the minimum number of bytes which are transferred from or to memory corresponding to a single DMA request or grant. The transfer can be 1-, 2-, 4-, 8-, 16-, or 32-bytes. If the DMA_CFG.MSIZE value is greater than the SCB bus width, the SCB performs burst transfers according to the width defined in DMA_CFG.MSIZE. When DMA_CFG.MSIZE is less than the SCB bus width, bursting is not supported and partial bus use results.

Each of the SCB unit in the fabric consists of N completer interfaces (MSTn). Each of these interfaces has controls for read quality of service, write quality of service, and functional mode. A subset of these matrices includes controls for IB (Interface Block) sync mode, and bus functional mode. For more details on IB, see the clock domain synchronization section.

SCB Bus Requester IDs

The SCB ***Bus Requester IDs*** table indicates which requesters are connected to each of the completer ports of SCB0. The tables also indicate the precise value of the ID as seen by the completer. These values are useful for SWU programming.

NOTE: For an overall diagram of all SCB interconnections, see the SCB Block Diagram.

Table 48-4: Bus Requester IDs

Requester	Hex ID Values	Binary Values
DMA0 (SPORT0, HALF A)	0x0000, 0x0100	13'b0000x00000000
DMA1 (SPORT0, HALF B)	0x0010, 0x0110	13'b0000x00010000
DMA2 (SPORT1, HALF A)	0x0020, 0x0120	13'b0000x00100000
DMA3 (SPORT1, HALF B)	0x0030, 0x0130	13'b0000x00110000
DMA4 (SPORT2, HALF A)	0x0040, 0x0140	13'b0000x01000000
DMA5 (SPORT2, HALF B)	0x0050, 0x0150	13'b0000x01010000
DMA6 (SPORT3, HALF A)	0x0060, 0x0160	13'b0000x01100000
DMA7 (SPORT3, HALF B)	0x0070, 0x0170	13'b0000x01110000
DMA8 (Enh BW MDMA0 CRC, CH0)	0x0031, 0x0131	13'b0000x00110001
DMA9 (Enh BW MDMA0 CRC, CH1)	0x0021, 0x0121	13'b0000x00100001
MLB	0x0042	13'b0000001000010
DMA20 (UART0, TX)	0x0003, 0x0103	13'b0000x00000011
DMA21 (UART0, RX)	0x0043, 0x0143	13'b0000x01000011
DMA37 (UART2, TX)	0x0033, 0x0133	13'b0000x00110011
DMA22 (SPI0, TX)	0x0004, 0x0104	13'b0000x00000100
DMA23 (SPI0, RX)	0x0014, 0x0114	13'b0000x00010100
DMA24 (SPI1, TX)	0x0024, 0x0124	13'b0000x00100100
DMA25 (SPI1, RX)	0x0034, 0x0134	13'b0000x00110100

Table 48-4: Bus Requester IDs (Continued)

Requester	Hex ID Values	Binary Values
DMA26 (SPI2, TX)	0x0054, 0x0154	13'b0000x01010100
DMA27 (SPI2, RX)	0x0044, 0x0144	13'b0000x01000100
DMA30 (LP0)	0x0005, 0x005	13'b0000x00000101
DMA34 (UART1, TX)	0x0013, 0x0113	13'b0000x00010011
DMA35 (UART1, RX)	0x0023, 0x0123	13'b0000x00100011
DMA36 (LP1)	0x0015, 0x0115	13'b0000x00010101
CRYPTO	0x0056, 0x0156	13'b0000001010110
SH0_FIR_CH0	0x0X07 (X=don't care)	13'b0xxxx00000111
SH0_FIR_CH1	0x0X17 (X=don't care)	13'b0xxxx00010111
EMDMA0 (CH0)	0x0006	13'b0000000000110
EMDMA0 (CH1)	0x0016	13'b0000000010110
EMDMA1 (CH0)	0x0026	13'b0000000100110
EMDMA1 (CH1)	0x0036	13'b0000000110110
DMA39 (Enh BW MDMA2, CH0)	0x0008, 0x0108	13'b0000x00001000
DMA40 (Enh BW MDMA2, CH1)	0x0018, 0x0118	13'b0000x00011000
DBG	0x0048	13'b0000001001000
ETR	0x0046	13'b0000001000110
DMA18 (Enh BW MDMA1 CRC1, CH0)	0x0011, 0x0111	13'b0000x00010001
DMA19 (Enh BW MDMA1 CRC1, CH1)	0x0001, 0x0101	13'b0000x00000001
DMA38 (UART2, RX)	0x0053, 0x0153	13'b0000x01010011
SH0 (DPORT)	0x0X09(X=don't care)	13'b0xxxx00001001
SH0 (IPORT)	0x0047	13'b0000001000111
DMA43 (High Speed MDMA3, CH0)	0x000A, 0x010A	13'b0000x00001010
DMA44 (High Speed BW MDMA3, CH1)	0x001A, 0x011A	13'b0000x00011010
DMA10 (SPORT4, HALFA)	0x000B, 0x010B	13'b0000x00001011
DMA11 (SPORT4, HALFB)	0x001B, 0x011B	13'b0000x00011011
DMA12 (SPORT5, HALFA)	0x002B, 0x012B	13'b0000x00101011
DMA13 (SPORT5, HALFB)	0x003B, 0x013B	13'b0000x00111011
DMA14 (SPORT6, HALFA)	0x004B, 0x014B	13'b0000x01001011
DMA15 (SPORT6, HALFB)	0x005B, 0x015B	13'b0000x01011011
DMA16 (SPORT7, HALFA)	0x006B, 0x016B	13'b0000x01101011

Table 48-4: Bus Requester IDs (Continued)

Requester	Hex ID Values	Binary Values
DMA17 (SPORT7, HALFB)	0x007B, 0x017B	13'b0000x01111011
SH0_MMR	0x0X0C (X=don't care)	13'b0xxxx00001100
DMA53 (UART3, TX)	0x0063, 0x0163	13'b0000x01100011
DMA54 (UART3, RX)	0x0073, 0x0173	13'b0000x01110011
DMA55 (SPI3, TX)	0x0064, 0x0164	13'b0000x01100100
DMA56 (SPI3, RX)	0x0074, 0x0174	13'b0000x01110100
DMA45 (Enh BW MDMA4 CRC, CH0)	0x001D, 0x011D	13'b0000x00011101
DMA46 (Enh BW MDMA4 CRC, CH1)	0x000D, 0x010D	13'b0000x00001101
DMA47 (Enh BW MDMA5 CRC, CH0)	0x003D, 0x013D	13'b0000x00111101
DMA48 (Enh BW MDMA5 CRC, CH1)	0x002D, 0x012D	13'b0000x00101101
DMA49 (Enh BW MDMA6 CRC1, CH0)	0x0038, 0x0138	13'b0000x00111000
DMA50 (Enh BW MDMA6 CRC1, CH1)	0x0028, 0x0128	13'b0000x00101000
DMA51 (High Speed MDMA3, CH0)	0x002A, 0x012A	13'b0000x00101010
DMA52 (High Speed BW MDMA3, CH1)	0x003A, 0x013A	13'b0000x00111010
DMA28 (PPI F0)	0x0083, 0x0183	13'b0000x10000011
DMA29 (PPI F1)	0x0093, 0x0193	13'b0000x10010011
USB0	0x002E	13'b0000000101110
SH1 (IPORT)	0x0057	13'b0000001010111
SH1 (DPORT)	0x0X19(X=don't care)	13'b0xxxx00011001
SH1_MMR	0x0X2C(X=don't care)	13'b0xxxx00101100
ARM_L2CC_M0	0x0X29(X=don't care)	13'bxxxxx00101001
ARM_L2CC_M1	0x0X39(X=don't care)	13'bxxxxx00111001
GIGE	0x0X0F(X=don't care)	13'b0xxxx00001111
EMAC	0x0X1F(X=don't care)	13'b0xxxx00011111
SH1_FIR_CH0	0x0X27(X=don't care)	13'b0xxxx00100111
SH1_FIR_CH1	0x0X37(X=don't care)	13'b0xxxx00110111
SH0_IIR_CH0	0x0X49(X=don't care)	13'b0xxxx01001001
SH0_IIR_CH1	0x0X59(X=don't care)	13'b0xxxx01011001
SH1_IIR_CH0	0x0X69(X=don't care)	13'b0xxxx01101001
SH1_IIR_CH1	0x0X79(X=don't care)	13'b0xxxx01111001

SCB Programming Model

The following sections provide information for the correct programming of the SCB. The SCB in the ADSP-SC59x processors is based on the ARM CoreLink NIC-400 Network Interconnect. This chapter should be used in conjunction with the ARM CoreLink NIC-400 Network Interconnect Technical Reference Manual. The ARM manual provides more information for programming in the SCB interface.

Programming SCB Arbitration

Each completer interface has a QoS value (priority) associated with both read and write channels. These values are 4 bits present in the `SCB0_MSTx_RQOS` and `SCB0_MSTx_WQOS` registers. At the entry point to the infrastructure, all transactions are allocated this programmable local QoS value. The arbitration of the transaction throughout the infrastructure uses this QoS. At any arbitration node, a fixed priority exists for transactions with a different QoS. The highest value has the highest priority.

If there are coincident transactions at an arbitration node with the same QoS that require arbitration, then the network uses a Least Recently Granted (LRG) algorithm. At each switch, the requester with the highest QoS acquires access and that switch output takes the QoS value of the winner for that transaction. At the next switch completer interface, the requester uses the QoS value of the winner. QoS can have values from 0 (lowest priority) to 15 (highest priority).

For example in the following figure, ***SCB Arbitration***:

1. At SCB1, requesters (1, 2, 3) have RQOS values of (6, 4, 2)
2. At SCB2, requesters (4, 5, 6) have RQOS values of (12, 13, 1)

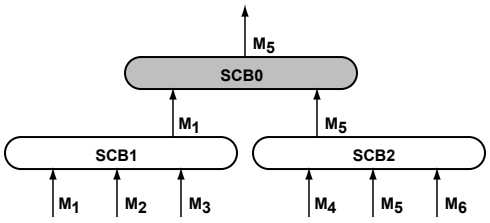


Figure 48-7: SCB Arbitration

In this case, requester 1 wins at SCB1, and requester 5 wins at SCB2. However, in a perfect competition at SCB0, requesters 4 and 5 had the highest overall RQOS values. requesters 4 and 5 would have fought for arbitration directly at SCB0. However, because of the mini*SCBs, requester 1, at a much lower RQOS value, is able to win against requester 4 and make it all the way to SCB0.

Programming Clock Domain Crossing Registers

The ***Clock Domain Crossing Options*** table shows the various clock domain crossings that are available on the ADSP-SC59x processor. The clocking relationships are defined with respect to SYSCLK. The following clock domains are programmable: CLK02:SYSCLK, CLKO3: SYSCLK, CLKO4: SYSCLK and CLKO8: SYSCLK CDC. The registers to configure the CDC mode are present within the GPV space of the respective fabric. For example: To program

the CLKO3: SYSCLK relationship, program the GPV registers within the DMC_CDC fabric and the MMRG fabric.

Table 48-5: Clock Domain Crossing Options

		Clocking Relationships with respect to SYSCLK					
Functional Clocks	**Possible Completers**	**System Fabric**	**MMRG Fabric**	**MC Fabric**	**SHARC Fabric**	**Arm Fabric**	**SPIF Fabric**
SYSCLK	System Fabric and Infrastructure Modules	1:1	1:1	*1	*	*	*
SCLK0	SCLK0 Clock Doamin	Synchronous (m:1)	Synchronous (m:1)	*	*	*	*
CLKO0	SHARC0 and its Accelerators	*	*	*	Synchronous (m:1)	*	*
CLKO1	SHARC1 and its Accelerators	*	*	*	Synchronous (m:1)	*	*
CLKO2	ARM	*	*	*	*	Programmable	*
CLKO3	DMC	*	Programmable	Programmable	*	*	*
CLKO4	CAN	*	Programmable	*	*	*	*
CLKO6	SPI	Synchronous (m:n) (except SPIF completer)	Synchronous (m:n)	*	*	*	Synchronous (m:n)
CLKO8	LP	Programmable	Programmable	*	*	*	*

*1 * Either the interface is not present or CDC is not required

NOTE: Consider the following points in the *Clock Domain Crossing Options* table:

- For DMC fabric, the sync mode register should be programmed for m:n or async (default). Only for sysclk:CLKO3 = 500:500MHz, sync mode can be 1:1.
- For system MMRG fabric – CLKO8/LP sync mode register should be programmed for m:n or async (default).

Table 48-6: Sync Mode Bit Field Description

Sync Mode	Description
0	sync 1:1
1	sync n:1
2	sync 1:n
3	sync m:n

Table 48-6: Sync Mode Bit Field Description (Continued)

Sync Mode	Description
4	async

To change the clock domain crossing mode, follow the actions described in the ***Changing Clock Domain Crossing Modes*** table.

Table 48-7: Changing Clock Domain Crossing Modes

Original Mode	Required Mode	Action
ASYNC	Any other mode	Change the clocks, then change the MMR register
Any mode	ASYNC	Change the MMR register, then change clocks to ASYNC.
m:n	1:1	Change the clocks, then change the register.
1:1	m:n	Change the register, then change the clocks.

SCB Programming Concepts

The SCB arbitration model among requester or completer SCBs of the processor is fixed (not programmable). But, each completer does have a quality of service (QoS) programmable feature that affects arbitration.

The arbitration of transactions in SCB is based on the QoS value or the priority of the transaction. All requesters with the same priority form a priority group. Arbitration is granted to the highest priority group from which a member is trying to win access, and within that group, to the highest requester at that time. When a requester wins arbitration, it is relegated to the bottom of its group to ensure that it cannot prevent other requesters in its group from accessing the completer.

If you configure all requester priorities to different levels, the arbiter implements a fixed priority scheme. This scheme occurs because each requester is in a group of its own, and therefore, requesters maintain their ordering.

The LRG and fixed priority modes concurrently exist when the requester priority value registers are programmed with a combination of identical and unique values.

NOTE: The SCB arbitration hierarchy is fixed (for example, SCB1 requester to SCB1 completer). However, multiple requester inputs to the same completer permit QoS programming.

The *LRG Arbitration Example* figure shows three groups with different QoS values. Requesters in the same group share a QoS value. The arbitration occurs using an LRG scheme.

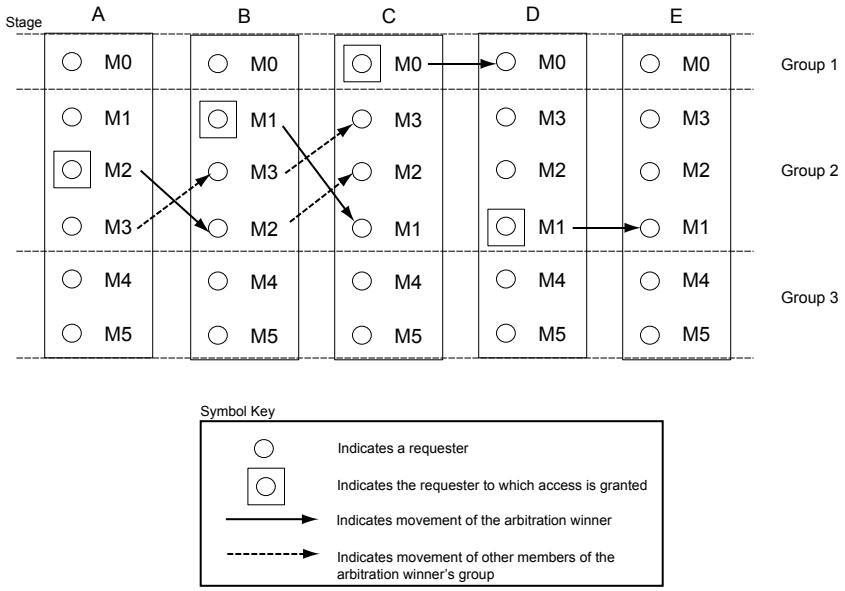


Figure 48-8: LRG Arbitration Example

The QoS value assigned to a transaction at entry point is carried forward by the transaction as it passes through all arbitration stages in the SCB. QoS for all requesters is configured as programmable in the system fabric interconnect.

The priority of the requesters fits into three groups:

- Group A - Peripherals with an external interface, without flow control and with real-time processing requirements
- Group B - Cores, peripherals with flow control, and offload engines
- Group C - MDMAs

Group A: Peripherals with external interface, without flow control and with real-time processing requirements

These requesters are assigned the highest priority. They are latency-critical – a large increase in latency could potentially result in data corruption and catastrophic failure. This high priority SCB group includes requesters connected to SCB1, SCB2, SCB3, and SCB4. Requesters in this group are assigned the QOS reset value of 12.

Group B: Cores, peripherals with flow control, and offload engines

The core is latency sensitive. Once a core requests data, it typically waits for the data without performing anything else in parallel. An increase in the latency has a direct impact on the performance of the core. This group is assigned medium priority and a QOS reset value of seven. Group B requesters are connected to SCB5, SCB6, CORES_SCB, FIR_IPORT_SCB, and MMR_SCB.

Group C: MDMAs

These requesters are assigned the lowest priority. MDMAs are data intensive and do not have any external interface. Requesters in this group are assigned the QOS reset value of 1. The Group C requesters are connected to SCB7, SCB8, SCB9, and SCB10.

Table 48-8: QoS Register Table

Requester ID	Requester	read_qos Reset Value	write_qos Reset Value
1	SPORT0_A_DMA	12	12
2	SPORT0_B_DMA	12	12
3	SPORT1_A_DMA	12	12
4	SPORT1_B_DMA	12	12
5	SPORT2_A_DMA	12	12
6	SPORT2_B_DMA	12	12
7	SPORT3_A_DMA	12	12
8	SPORT3_B_DMA	12	12
9	SPORT4_A_DMA	12	12
10	SPORT4_B_DMA	12	12
11	SPORT5_A_DMA	12	12
12	SPORT5_B_DMA	12	12
13	SPORT6_A_DMA	12	12
14	SPORT6_B_DMA	12	12
15	SPORT7_A_DMA	12	12
16	SPORT7_B_DMA	12	12
	UART0_TX	12	12
	UART0_RX	12	12
	UART1_TX	12	12
	UART1_RX	12	12
	UART2_TX	12	12
	UART2_RX	12	12
	UART3_TX	12	12
	UART3_RX	12	12
	PPI_F0	12	12
	PPI_F1	12	12
	SPI0RX	12	12

Table 48-8: QoS Register Table (Continued)

Requester ID	Requester	read_qos Reset Value	write_qos Reset Value
	SPI0TX	12	12
19	SPI1TX	12	12
20	SPI1RX	12	12
21	SPI2TX	12	12
22	SPI2RX	12	12
23	MLB	12	12
	GIGE	7	7
	EMAC	7	7
	LP0	7	7
	LP1	7	7
	USB	7	7
	SDIO0	7	7
	SH0_DPORT	7	7
	SH0_IPORT	7	7
	SH1_IPORT	7	7
	ARM_L2CC_M0	7	7
	ARM_L2CC_M1	7	7
	SH0_IIR_CH0	7	7
	SH0_IIR_CH1	7	7
	SH1_IIR_CH0	7	7
	SH1_IIR_CH1	7	7
	SH0_FIR_CH0	7	7
	SH0_FIR_CH1	7	7
	SH0_MMR	7	7
	SH1_MMR	7	7
	PL310_MMR	7	7
	CRC0_CH0	1	1
	CRC0_CH1	1	1
	CRC1_CH0	1	1
	CRC1_CH1	1	1
42	CRC2_CH0	1	1

Table 48-8: QoS Register Table (Continued)

Requester ID	Requester	read_qos Reset Value	write_qos Reset Value
43	CRC2_CH1	1	1
	CRC3_CH0	1	1
	CRC3_CH1	1	1
	MDMA2_CH0	1	1
	MDMA2_CH1	1	1
	MDMA3_CH0	1	1
	MDMA3_CH1	1	1
	MDMA6_CH0	1	1
	MDMA6_CH1	1	1
	MDMA7_CH0	1	1
	MDMA7_CH1	1	1
	EMDMA0_CH0	1	1
	EMDMA0_CH1	1	1
	EMDMA1_CH0	1	1
	EMDMA1_CH1	1	1
	CRYPTO	1	1
	ETR	1	1
	DBG	1	1

QoS Programming

Adhere to the following guidelines if software is modifying the reset value of QoS:

- The highest QoS of any requester in a low priority SCB group must be less than the lowest QoS of any requester in a medium priority SCB group
- The highest QoS of any requester in medium priority SCB group must be less than the lowest QoS of any requester in high priority SCB group

Reset values of QoS have been spaced out to allow software to lower or higher the priority of a few requesters without having to modify the priority all the requesters. For example, the QoS of the LP can be reduced from 12 to 11 without violating the guidelines.

SCB6 Programming

Single core A55 integration supports in-order transactions in SCB6. To support this feature, program `SCB6_A55_M0_IB_FN_MOD` = 0x3 (2'b11).

ADSP-SC59x SCB0 Register Descriptions

(SCB0) contains the following registers.

Table 48-9: ADSP-SC59x SCB0 Register List

Name	Description
SCB0_CRC0_CH0_IB_READ_QOS	CRC0 Channel 0 Read Quality of Service Register
SCB0_CRC0_CH0_IB_WRITE_QOS	CRC0 Channel 0 Write Quality of Service Register
SCB0_CRC0_CH1_IB_READ_QOS	CRC0 Channel 1 Read Quality of Service Register
SCB0_CRC0_CH1_IB_WRITE_QOS	CRC0 Channel 1 Write Quality of Service Register
SCB0_CRC1_CH0_IB_READ_QOS	CRC1 Channel 0 Read Quality of Service Register
SCB0_CRC1_CH0_IB_WRITE_QOS	CRC1 Channel 0 Write Quality of Service Register
SCB0_CRC1_CH1_IB_READ_QOS	CRC1 Channel 1 Read Quality of Service Register
SCB0_CRC1_CH1_IB_WRITE_QOS	CRC1 Channel 1 Write Quality of Service Register
SCB0_CRC2_CH0_IB_READ_QOS	CRC2 Channel 0 Read Quality of Service Register
SCB0_CRC2_CH0_IB_WRITE_QOS	CRC2 Channel 0 Write Quality of Service Register
SCB0_CRC2_CH1_IB_READ_QOS	CRC2 Channel 1 Read Quality of Service Register
SCB0_CRC2_CH1_IB_WRITE_QOS	CRC2 Channel 1 Write Quality of Service Register
SCB0_CRC3_CH0_IB_READ_QOS	CRC3 Channel 0 Read Quality of Service Register
SCB0_CRC3_CH0_IB_WRITE_QOS	CRC3 Channel 0 Write Quality of Service Register
SCB0_CRC3_CH1_IB_READ_QOS	CRC3 Channel 1 Read Quality of Service Register
SCB0_CRC3_CH1_IB_WRITE_QOS	CRC3 Channel 1 Write Quality of Service Register
SCB0_CRYPTO_IB_READ_QOS	CRYPTO Read Quality of Service Register
SCB0_CRYPTO_IB_WRITE_QOS	CRYPTO Write Quality of Service Register
SCB0_DBG_IB_READ_QOS	DBG Read Quality of Service Register
SCB0_DBG_IB_WRITE_QOS	DBG Write Quality of Service Register
SCB0_DLDMA0_CH0_IB_READ_QOS	DLDMA0 Channel 0 Read Quality of Service Register
SCB0_DLDMA0_CH0_IB_WRITE_QOS	DLDMA0 Channel 0 Write Quality of Service Register
SCB0_DLDMA0_CH1_IB_READ_QOS	DLDMA0 Channel 1 Read Quality of Service Register
SCB0_DLDMA0_CH1_IB_WRITE_QOS	DLDMA0 Channel 1 Write Quality of Service Register
SCB0_DLDMA1_CH0_IB_READ_QOS	DLDMA1 Channel 0 Read Quality of Service Register
SCB0_DLDMA1_CH0_IB_WRITE_QOS	DLDMA1 Channel 0 Write Quality of Service Register
SCB0_DLDMA1_CH1_IB_READ_QOS	DLDMA1 Channel 1 Read Quality of Service Register
SCB0_DLDMA1_CH1_IB_WRITE_QOS	DLDMA1 Channel 1 Write Quality of Service Register
SCB0_EMAC_IB_READ_QOS	EMAC Read Quality of Service Register

Table 48-9: ADSP-SC59x SCB0 Register List (Continued)

Name	Description
SCB0_EMAC_IB_WRITE_QOS	EMAC Write Quality of Service Register
SCB0_ETR_IB_READ_QOS	ETR Read Quality of Service Register
SCB0_ETR_IB_WRITE_QOS	ETR Write Quality of Service Register
SCB0_GIGE_IB_READ_QOS	GIGE Read Quality of Service Register
SCB0_GIGE_IB_WRITE_QOS	GIGE Write Quality of Service Register
SCB0_HSMDMA1_CH0_IB_READ_QOS	HSMDMA1 Channel 0 Read Quality of Service Register
SCB0_HSMDMA1_CH0_IB_WRITE_QOS	HSMDMA1 Channel 0 Write Quality of Service Register
SCB0_HSMDMA1_CH1_IB_READ_QOS	HSMDMA1 Channel 1 Read Quality of Service Register
SCB0_HSMDMA1_CH1_IB_WRITE_QOS	HSMDMA1 Channel 1 Write Quality of Service Register
SCB0_HSMDMA_CH0_IB_READ_QOS	HSMDMA Channel 0 Read Quality of Service Register
SCB0_HSMDMA_CH0_IB_WRITE_QOS	HSMDMA Channel 0 Write Quality of Service Register
SCB0_HSMDMA_CH1_IB_READ_QOS	HSMDMA Channel 1 Read Quality of Service Register
SCB0_HSMDMA_CH1_IB_WRITE_QOS	HSMDMA Channel 1 Write Quality of Service Register
SCB0_LP0_READ_QOS	LP0 Read Quality of Service Register
SCB0_LP0_WRITE_QOS	LP0 Write Quality of Service Register
SCB0_LP1_READ_QOS	LP1 Read Quality of Service Register
SCB0_LP1_WRITE_QOS	LP1 Write Quality of Service Register
SCB0_MLB_READ_QOS	MLB Read Quality of Service Register
SCB0_MLB_WRITE_QOS	MLB Write Quality of Service Register
SCB0_MSMDMA1_CH0_IB_READ_QOS	MSMDMA1 Channel 0 Read Quality of Service Register
SCB0_MSMDMA1_CH0_IB_WRITE_QOS	MSMDMA1 Channel 0 Write Quality of Service Register
SCB0_MSMDMA1_CH1_IB_READ_QOS	MSMDMA1 Channel 1 Read Quality of Service Register
SCB0_MSMDMA1_CH1_IB_WRITE_QOS	MSMDMA1 Channel 1 Write Quality of Service Register
SCB0_MSMDMA_CH0_IB_READ_QOS	MSMDMA Channel 0 Read Quality of Service Register
SCB0_MSMDMA_CH0_IB_WRITE_QOS	MSMDMA Channel 0 Write Quality of Service Register
SCB0_MSMDMA_CH1_IB_READ_QOS	MSMDMA Channel 1 Read Quality of Service Register
SCB0_MSMDMA_CH1_IB_WRITE_QOS	MSMDMA Channel 1 Write Quality of Service Register
SCB0_PL310_M0_IB_READ_QOS	PL310 M0 Read Quality of Service Register
SCB0_PL310_M0_IB_WRITE_QOS	PL310 M0 Write Quality of Service Register
SCB0_PL310_M1_IB_READ_QOS	PL310 M1 Read Quality of Service Register
SCB0_PL310_M1_IB_WRITE_QOS	PL310 M1 Write Quality of Service Register

Table 48-9: ADSP-SC59x SCB0 Register List (Continued)

Name	Description
SCB0_PL310_MMR_IB_READ_QOS	PL310 MMR Read Quality of Service Register
SCB0_PL310_MMR_IB_WRITE_QOS	PL310 MMR Write Quality of Service Register
SCB0_PPI_F0_READ_QOS	PPI F0 Read Quality of Service Register
SCB0_PPI_F0_WRITE_QOS	PPI F0 Write Quality of Service Register
SCB0_PPI_F1_READ_QOS	PPI F1 Read Quality of Service Register
SCB0_PPI_F1_WRITE_QOS	PPI F1 Write Quality of Service Register
SCB0_SCB6_IB_SYNC_MODE	SCB6 Synchronization Mode Register
SCB0_SDIO0_IB_READ_QOS	SDIO0 Read Quality of Service Register
SCB0_SDIO0_IB_WRITE_QOS	SDIO0 Write Quality of Service Register
SCB0_SH0_DPORT_READ_QOS	SH0 DPORT Read Quality of Service Register
SCB0_SH0_DPORT_WRITE_QOS	SH0 DPORT Write Quality of Service Register
SCB0_SH0_FIR_CH0_READ_QOS	SH0 FIR Channel 0Read Quality of Service Register
SCB0_SH0_FIR_CH0_WRITE_QOS	SH0 FIR Channel 0Write Quality of Service Register
SCB0_SH0_FIR_CH1_READ_QOS	SH0 FIR Channel 1Read Quality of Service Register
SCB0_SH0_FIR_CH1_WRITE_QOS	SH0 FIR Channel 1Write Quality of Service Register
SCB0_SH0_IIR_CH0_READ_QOS	SH0 IIR Channel 0 Read Quality of Service Register
SCB0_SH0_IIR_CH0_WRITE_QOS	SH0 IIR Channel 0 Write Quality of Service Register
SCB0_SH0_IIR_CH1_READ_QOS	SH0 IIR Channel 1 Read Quality of Service Register
SCB0_SH0_IIR_CH1_WRITE_QOS	SH0 IIR Channel 1 Write Quality of Service Register
SCB0_SH0_IPORT_READ_QOS	SH0 IPORT Read Quality of Service Register
SCB0_SH0_IPORT_WRITE_QOS	SH0 IPORT Write Quality of Service Register
SCB0_SH0_MMR_IB_READ_QOS	SH0 MMR Read Quality of Service Register
SCB0_SH0_MMR_IB_WRITE_QOS	SH0 MMR Write Quality of Service Register
SCB0_SH1_DPORT_READ_QOS	SH1 DPORT Read Quality of Service Register
SCB0_SH1_DPORT_WRITE_QOS	SH1 DPORT Write Quality of Service Register
SCB0_SH1_FIR_CH0_READ_QOS	SH1 FIR Channel 0 Read Quality of Service Register
SCB0_SH1_FIR_CH0_WRITE_QOS	SH1 FIR Channel 0 Write Quality of Service Register
SCB0_SH1_FIR_CH1_READ_QOS	SH1 FIR Channel 1 Read Quality of Service Register
SCB0_SH1_FIR_CH1_WRITE_QOS	SH1 FIR Channel 1 Write Quality of Service Register
SCB0_SH1_IIR_CH0_READ_QOS	SH1 IIR Channel 0 Read Quality of Service Register
SCB0_SH1_IIR_CH0_WRITE_QOS	SH1 IIR Channel 0 Write Quality of Service Register

Table 48-9: ADSP-SC59x SCB0 Register List (Continued)

Name	Description
SCB0_SH1_IIR_CH1_READ_QOS	SH1 IIR Channel 1 Read Quality of Service Register
SCB0_SH1_IIR_CH1_WRITE_QOS	SH1 IIR Channel 1 Write Quality of Service Register
SCB0_SH1_IPORT_READ_QOS	SH1 IPORT Read Quality of Service Register
SCB0_SH1_IPORT_WRITE_QOS	SH1 IPORT Write Quality of Service Register
SCB0_SH1_MMR_IB_READ_QOS	SH1 MMR Read Quality of Service Register
SCB0_SH1_MMR_IB_WRITE_QOS	SH1 MMR Write Quality of Service Register
SCB0_SP0A_READ_QOS	SP0A Read Quality of Service Register
SCB0_SP0A_WRITE_QOS	SP0A Write Quality of Service Register
SCB0_SP0B_READ_QOS	SP0B Read Quality of Service Register
SCB0_SP0B_WRITE_QOS	SP0B Write Quality of Service Register
SCB0_SP1A_READ_QOS	SP1A Read Quality of Service Register
SCB0_SP1A_WRITE_QOS	SP1A Write Quality of Service Register
SCB0_SP1B_READ_QOS	SP1B Read Quality of Service Register
SCB0_SP1B_WRITE_QOS	SP1B Write Quality of Service Register
SCB0_SP2A_READ_QOS	SP2A Read Quality of Service Register
SCB0_SP2A_WRITE_QOS	SP2A Write Quality of Service Register
SCB0_SP2B_READ_QOS	SP1B Read Quality of Service Register
SCB0_SP2B_WRITE_QOS	SP1B Write Quality of Service Register
SCB0_SP3A_READ_QOS	SP3A Read Quality of Service Register
SCB0_SP3A_WRITE_QOS	SP3A Write Quality of Service Register
SCB0_SP3B_READ_QOS	SP3B Read Quality of Service Register
SCB0_SP3B_WRITE_QOS	SP3B Write Quality of Service Register
SCB0_SP4A_READ_QOS	SP4A Read Quality of Service Register
SCB0_SP4A_WRITE_QOS	SP4A Write Quality of Service Register
SCB0_SP4B_READ_QOS	SP4B Read Quality of Service Register
SCB0_SP4B_WRITE_QOS	SP4B Write Quality of Service Register
SCB0_SP5A_READ_QOS	SP5A Read Quality of Service Register
SCB0_SP5A_WRITE_QOS	SP5A Write Quality of Service Register
SCB0_SP5B_READ_QOS	SP5B Read Quality of Service Register
SCB0_SP5B_WRITE_QOS	SP5B Write Quality of Service Register
SCB0_SP6A_READ_QOS	SP6A Read Quality of Service Register

Table 48-9: ADSP-SC59x SCB0 Register List (Continued)

Name	Description
SCB0_SP6A_WRITE_QOS	SP6A Write Quality of Service Register
SCB0_SP6B_READ_QOS	SP6B Read Quality of Service Register
SCB0_SP6B_WRITE_QOS	SP6B Write Quality of Service Register
SCB0_SP7A_READ_QOS	SP7A Read Quality of Service Register
SCB0_SP7A_WRITE_QOS	SP7A Write Quality of Service Register
SCB0_SP7B_READ_QOS	SP7B Read Quality of Service Register
SCB0_SP7B_WRITE_QOS	SP7B Write Quality of Service Register
SCB0_SPI0RX_READ_QOS	SPI0 RX Read Quality of Service Register
SCB0_SPI0RX_WRITE_QOS	SPI0 RX Write Quality of Service Register
SCB0_SPI0TX_READ_QOS	SPI0 TX Read Quality of Service Register
SCB0_SPI0TX_WRITE_QOS	SPI0 TX Write Quality of Service Register
SCB0_SPI1RX_READ_QOS	SPI1 RX Read Quality of Service Register
SCB0_SPI1RX_WRITE_QOS	SPI1 RX Write Quality of Service Register
SCB0_SPI1TX_READ_QOS	SPI1 TX Read Quality of Service Register
SCB0_SPI1TX_WRITE_QOS	SPI1 TX Write Quality of Service Register
SCB0_SPI2RX_READ_QOS	SPI2 RX Read Quality of Service Register
SCB0_SPI2RX_WRITE_QOS	SPI2 RX Write Quality of Service Register
SCB0_SPI2TX_READ_QOS	SPI2 TX Read Quality of Service Register
SCB0_SPI2TX_WRITE_QOS	SPI2 TX Write Quality of Service Register
SCB0_SPI3RX_READ_QOS	SPI3 RX Read Quality of Service Register
SCB0_SPI3RX_WRITE_QOS	SPI3 RX Write Quality of Service Register
SCB0_SPI3TX_READ_QOS	SPI3 TX Read Quality of Service Register
SCB0_SPI3TX_WRITE_QOS	SPI3 TX Write Quality of Service Register
SCB0_UART0_RX_READ_QOS	UART0 RX Read Quality of Service Register
SCB0_UART0_RX_WRITE_QOS	UART0 RX Write Quality of Service Register
SCB0_UART0_TX_READ_QOS	UART0 TX Read Quality of Service Register
SCB0_UART0_TX_WRITE_QOS	UART0 TX Write Quality of Service Register
SCB0_UART1_RX_READ_QOS	UART1 RX Read Quality of Service Register
SCB0_UART1_RX_WRITE_QOS	UART1 RX Write Quality of Service Register
SCB0_UART1_TX_READ_QOS	UART1 TX Read Quality of Service Register
SCB0_UART1_TX_WRITE_QOS	UART1 TX Write Quality of Service Register

Table 48-9: ADSP-SC59x SCB0 Register List (Continued)

Name	Description
SCB0_UART2_RX_READ_QOS	UART2 RX Read Quality of Service Register
SCB0_UART2_RX_WRITE_QOS	UART2 RX Write Quality of Service Register
SCB0_UART2_TX_READ_QOS	UART2 TX Read Quality of Service Register
SCB0_UART2_TX_WRITE_QOS	UART2 TX Write Quality of Service Register
SCB0_UART3_RX_READ_QOS	UART3 RX Read Quality of Service Register
SCB0_UART3_RX_WRITE_QOS	UART3 RX Write Quality of Service Register
SCB0_UART3_TX_READ_QOS	UART3 TX Read Quality of Service Register
SCB0_UART3_TX_WRITE_QOS	UART3 TX Write Quality of Service Register
SCB0_USB0_IB_READ_QOS	USB0 Read Quality of Service Register
SCB0_USB0_IB_WRITE_QOS	USB0 Write Quality of Service Register

CRC0 Channel 0 Read Quality of Service Register

The SCB0_CRC0_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

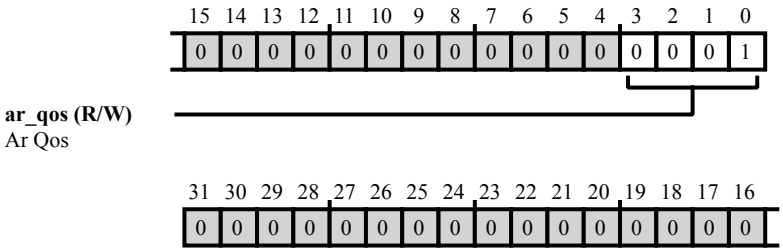


Figure 48-9: SCB0_CRC0_CH0_IB_READ_QOS Register Diagram

Table 48-10: SCB0_CRC0_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

CRC0 Channel 0 Write Quality of Service Register

The SCB0_CRC0_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

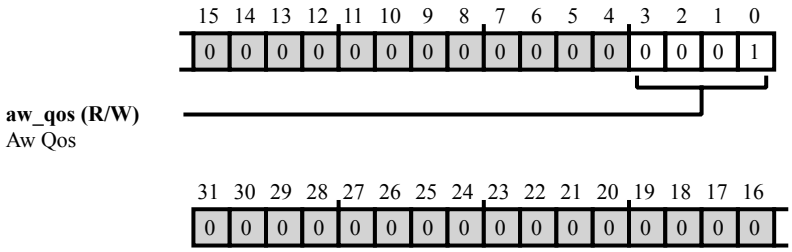


Figure 48-10: SCB0_CRC0_CH0_IB_WRITE_QOS Register Diagram

Table 48-11: SCB0_CRC0_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

CRC0 Channel 1 Read Quality of Service Register

The SCB0_CRC0_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

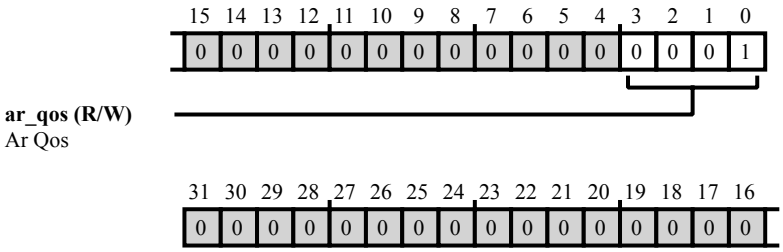


Figure 48-11: SCB0_CRC0_CH1_IB_READ_QOS Register Diagram

Table 48-12: SCB0_CRC0_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

CRC0 Channel 1 Write Quality of Service Register

The SCB0_CRC0_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

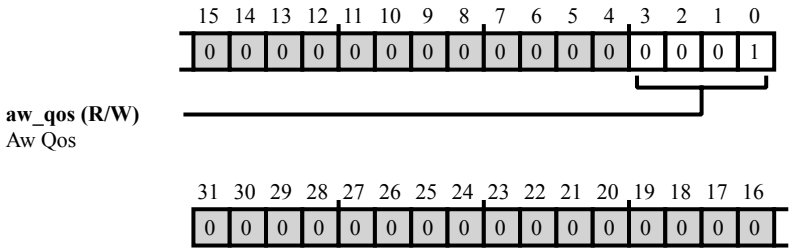


Figure 48-12: SCB0_CRC0_CH1_IB_WRITE_QOS Register Diagram

Table 48-13: SCB0_CRC0_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

CRC1 Channel 0 Read Quality of Service Register

The SCB0_CRC1_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

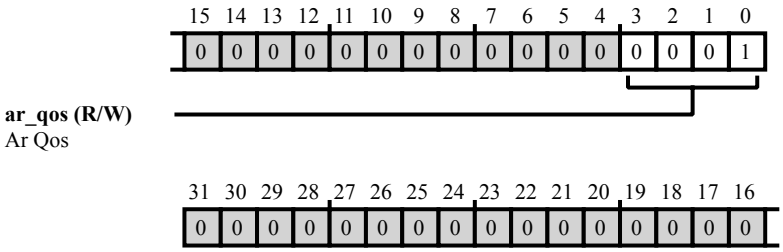


Figure 48-13: SCB0_CRC1_CH0_IB_READ_QOS Register Diagram

Table 48-14: SCB0_CRC1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

CRC1 Channel 0 Write Quality of Service Register

The SCB0_CRC1_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

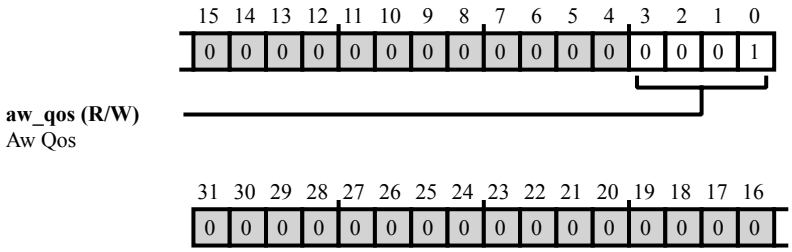


Figure 48-14: SCB0_CRC1_CH0_IB_WRITE_QOS Register Diagram

Table 48-15: SCB0_CRC1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

CRC1 Channel 1 Read Quality of Service Register

The SCB0_CRC1_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

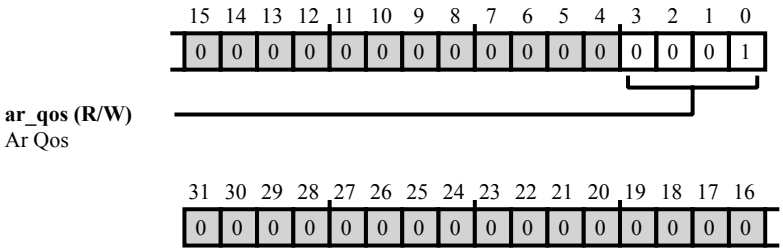


Figure 48-15: SCB0_CRC1_CH1_IB_READ_QOS Register Diagram

Table 48-16: SCB0_CRC1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

CRC1 Channel 1 Write Quality of Service Register

The SCB0_CRC1_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

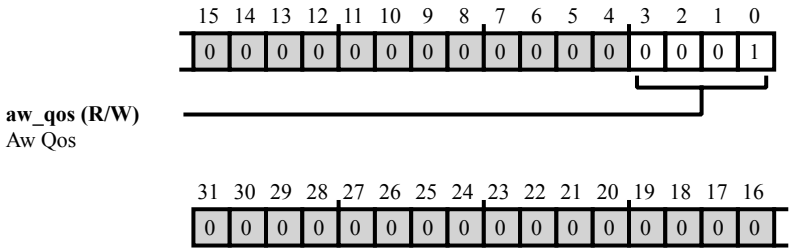


Figure 48-16: SCB0_CRC1_CH1_IB_WRITE_QOS Register Diagram

Table 48-17: SCB0_CRC1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

CRC2 Channel 0 Read Quality of Service Register

The SCB0_CRC2_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

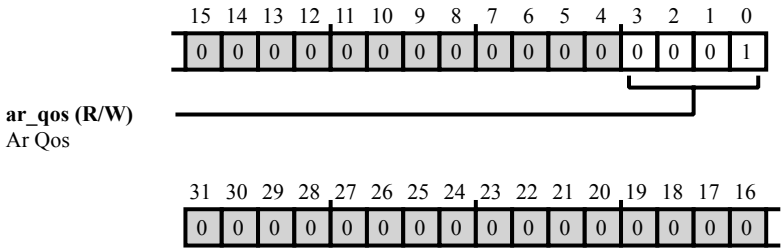


Figure 48-17: SCB0_CRC2_CH0_IB_READ_QOS Register Diagram

Table 48-18: SCB0_CRC2_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

CRC2 Channel 0 Write Quality of Service Register

The SCB0_CRC2_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

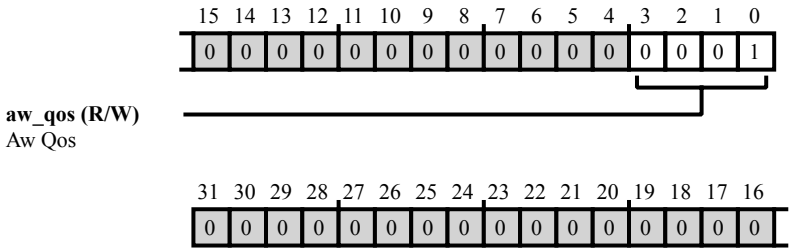


Figure 48-18: SCB0_CRC2_CH0_IB_WRITE_QOS Register Diagram

Table 48-19: SCB0_CRC2_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

CRC2 Channel 1 Read Quality of Service Register

The SCB0_CRC2_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

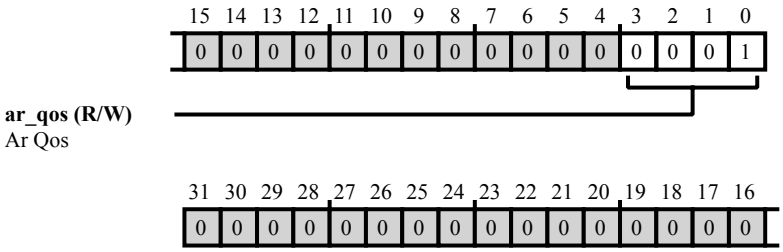


Figure 48-19: SCB0_CRC2_CH1_IB_READ_QOS Register Diagram

Table 48-20: SCB0_CRC2_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

CRC2 Channel 1 Write Quality of Service Register

The SCB0_CRC2_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

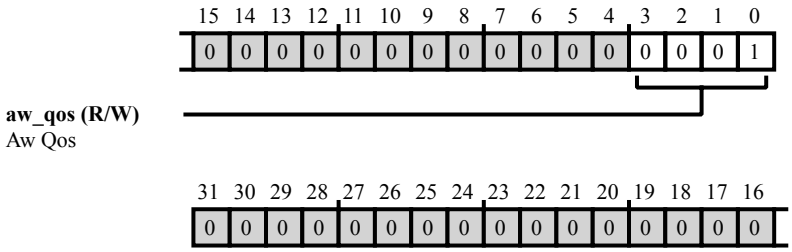


Figure 48-20: SCB0_CRC2_CH1_IB_WRITE_QOS Register Diagram

Table 48-21: SCB0_CRC2_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

CRC3 Channel 0 Read Quality of Service Register

The SCB0_CRC3_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

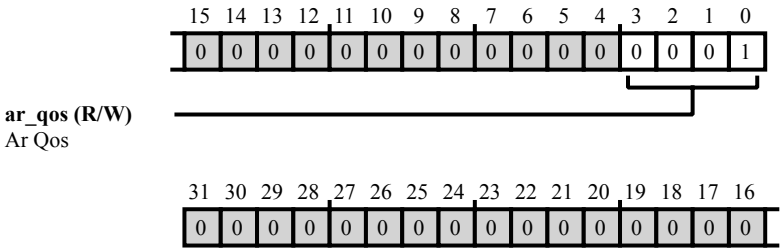


Figure 48-21: SCB0_CRC3_CH0_IB_READ_QOS Register Diagram

Table 48-22: SCB0_CRC3_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

CRC3 Channel 0 Write Quality of Service Register

The SCB0_CRC3_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

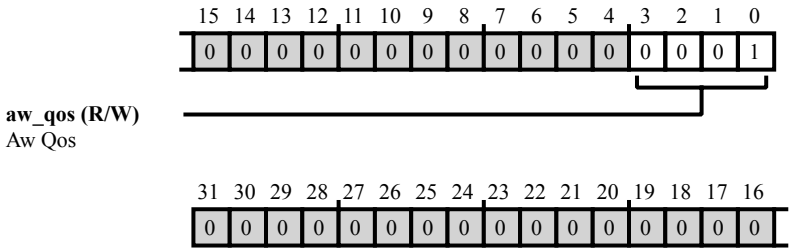


Figure 48-22: SCB0_CRC3_CH0_IB_WRITE_QOS Register Diagram

Table 48-23: SCB0_CRC3_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

CRC3 Channel 1 Read Quality of Service Register

The SCB0_CRC3_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

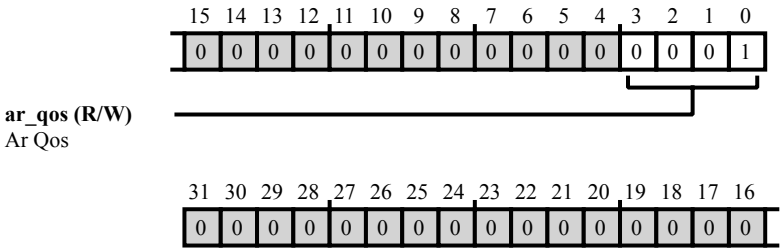


Figure 48-23: SCB0_CRC3_CH1_IB_READ_QOS Register Diagram

Table 48-24: SCB0_CRC3_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

CRC3 Channel 1 Write Quality of Service Register

The SCB0_CRC3_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

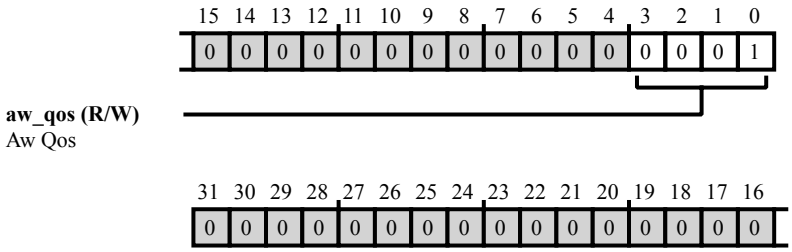


Figure 48-24: SCB0_CRC3_CH1_IB_WRITE_QOS Register Diagram

Table 48-25: SCB0_CRC3_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

CRYPTO Read Quality of Service Register

The SCB0_CRYPTO_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

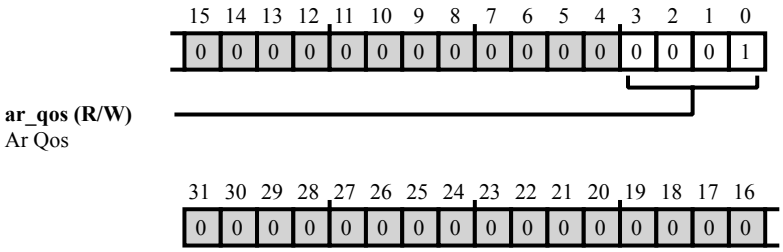


Figure 48-25: SCB0_CRYPTO_IB_READ_QOS Register Diagram

Table 48-26: SCB0_CRYPTO_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

CRYPTO Write Quality of Service Register

The SCB0_CRYPTO_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

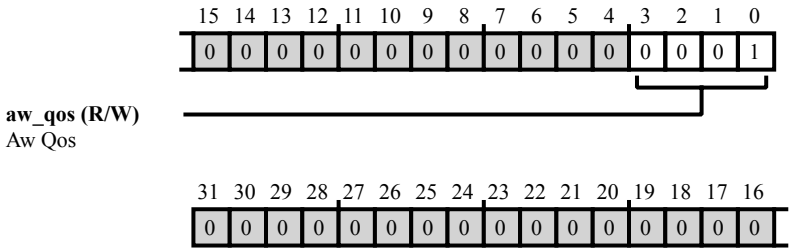


Figure 48-26: SCB0_CRYPTO_IB_WRITE_QOS Register Diagram

Table 48-27: SCB0_CRYPTO_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

DBG Read Quality of Service Register

The SCB0_DBG_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

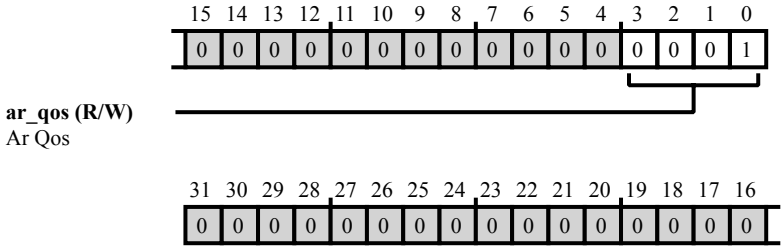


Figure 48-27: SCB0_DBG_IB_READ_QOS Register Diagram

Table 48-28: SCB0_DBG_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

DBG Write Quality of Service Register

The SCB0_DBG_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

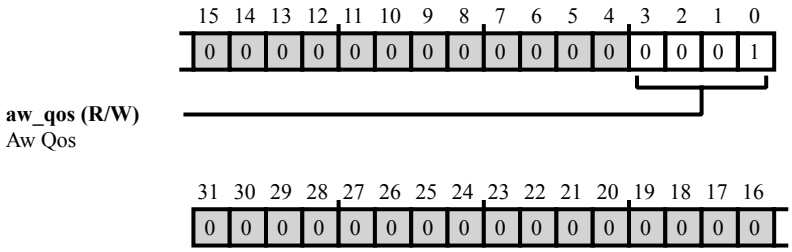


Figure 48-28: SCB0_DBG_IB_WRITE_QOS Register Diagram

Table 48-29: SCB0_DBG_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

DLDMA0 Channel 0 Read Quality of Service Register

The SCB0_DLDMA0_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

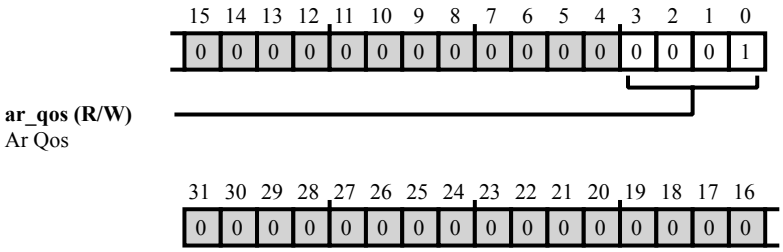


Figure 48-29: SCB0_DLDMA0_CH0_IB_READ_QOS Register Diagram

Table 48-30: SCB0_DLDMA0_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

DLDMA0 Channel 0 Write Quality of Service Register

The SCB0_DLDMA0_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

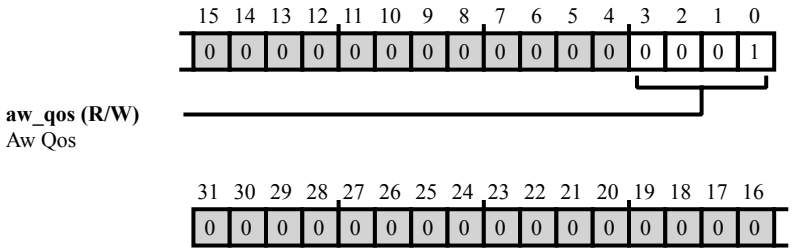


Figure 48-30: SCB0_DLDMA0_CH0_IB_WRITE_QOS Register Diagram

Table 48-31: SCB0_DLDMA0_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

DLDMA0 Channel 1 Read Quality of Service Register

The SCB0_DLDMA0_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

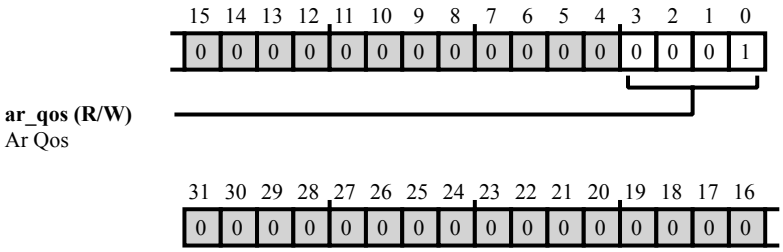


Figure 48-31: SCB0_DLDMA0_CH1_IB_READ_QOS Register Diagram

Table 48-32: SCB0_DLDMA0_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

DLDMA0 Channel 1 Write Quality of Service Register

The SCB0_DLDMA0_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

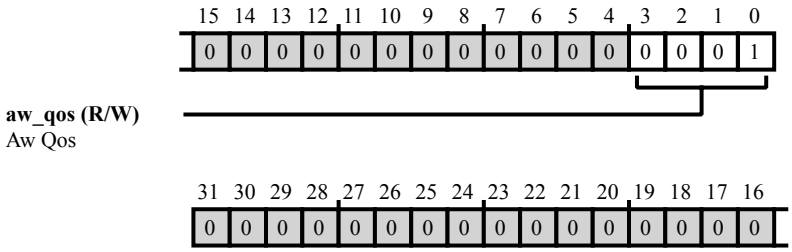


Figure 48-32: SCB0_DLDMA0_CH1_IB_WRITE_QOS Register Diagram

Table 48-33: SCB0_DLDMA0_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

DLDMA1 Channel 0 Read Quality of Service Register

The SCB0_DLDMA1_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

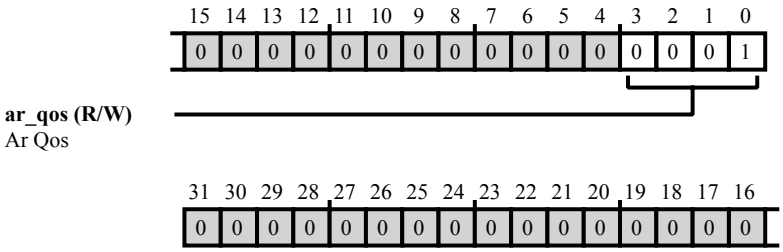


Figure 48-33: SCB0_DLDMA1_CH0_IB_READ_QOS Register Diagram

Table 48-34: SCB0_DLDMA1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

DLDMA1 Channel 0 Write Quality of Service Register

The SCB0_DLDMA1_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

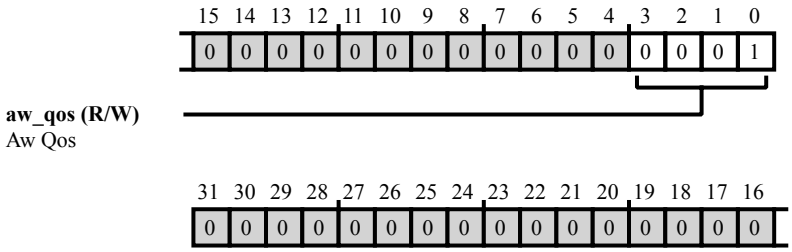


Figure 48-34: SCB0_DLDMA1_CH0_IB_WRITE_QOS Register Diagram

Table 48-35: SCB0_DLDMA1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

DLDMA1 Channel 1 Read Quality of Service Register

The SCB0_DLDMA1_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

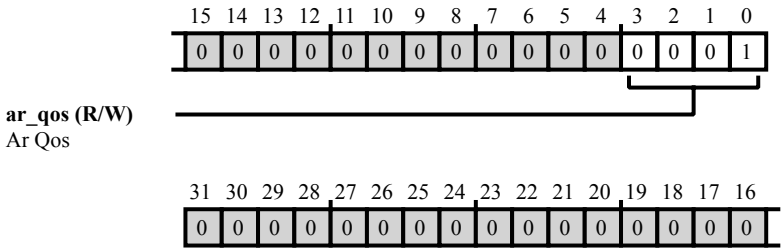


Figure 48-35: SCB0_DLDMA1_CH1_IB_READ_QOS Register Diagram

Table 48-36: SCB0_DLDMA1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

DLDMA1 Channel 1 Write Quality of Service Register

The SCB0_DLDMA1_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

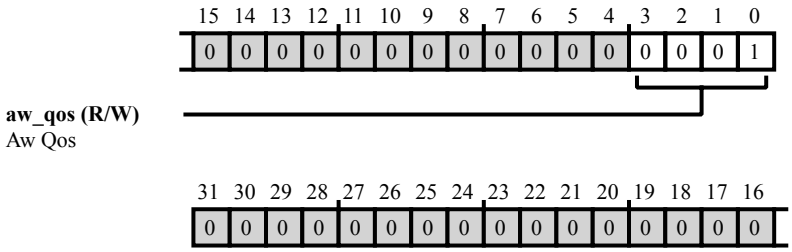


Figure 48-36: SCB0_DLDMA1_CH1_IB_WRITE_QOS Register Diagram

Table 48-37: SCB0_DLDMA1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

EMAC Read Quality of Service Register

The SCB0_EMAC_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

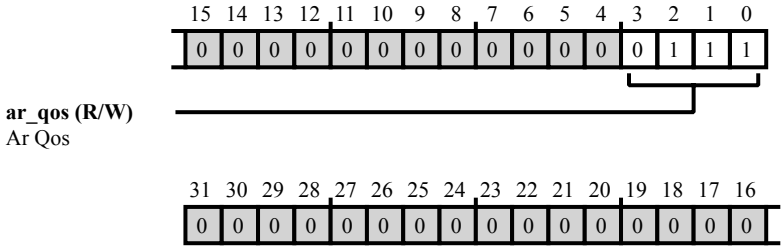


Figure 48-37: SCB0_EMAC_IB_READ_QOS Register Diagram

Table 48-38: SCB0_EMAC_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

EMAC Write Quality of Service Register

The SCB0_EMAC_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

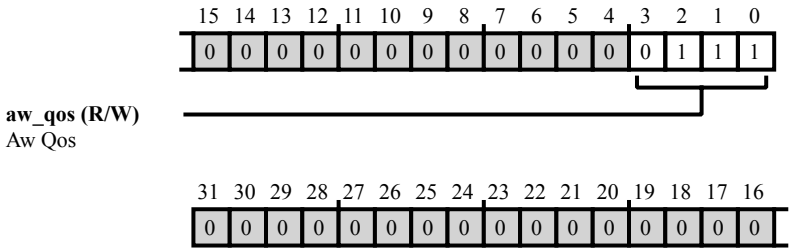


Figure 48-38: SCB0_EMAC_IB_WRITE_QOS Register Diagram

Table 48-39: SCB0_EMAC_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

ETR Read Quality of Service Register

The SCB0_ETR_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

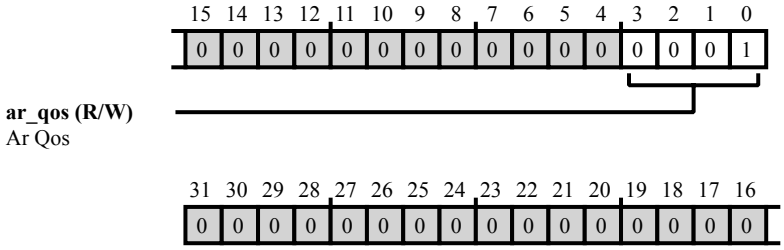


Figure 48-39: SCB0_ETR_IB_READ_QOS Register Diagram

Table 48-40: SCB0_ETR_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

ETR Write Quality of Service Register

The SCB0_ETR_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

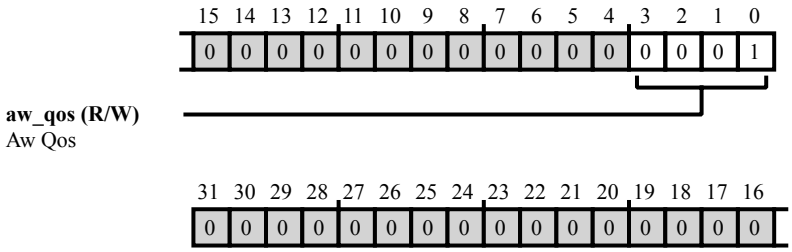


Figure 48-40: SCB0_ETR_IB_WRITE_QOS Register Diagram

Table 48-41: SCB0_ETR_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

GIGE Read Quality of Service Register

The SCB0_GIGE_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

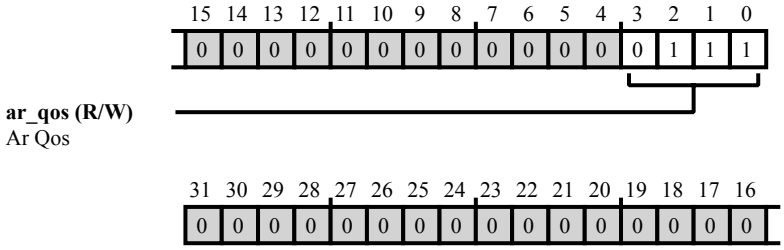


Figure 48-41: SCB0_GIGE_IB_READ_QOS Register Diagram

Table 48-42: SCB0_GIGE_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

GIGE Write Quality of Service Register

The SCB0_GIGE_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

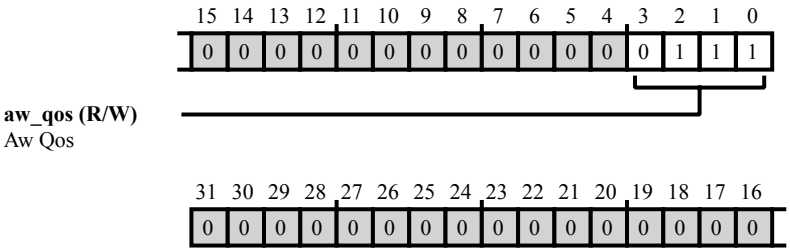


Figure 48-42: SCB0_GIGE_IB_WRITE_QOS Register Diagram

Table 48-43: SCB0_GIGE_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

HSMDMA1 Channel 0 Read Quality of Service Register

The SCB0_HSMDMA1_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

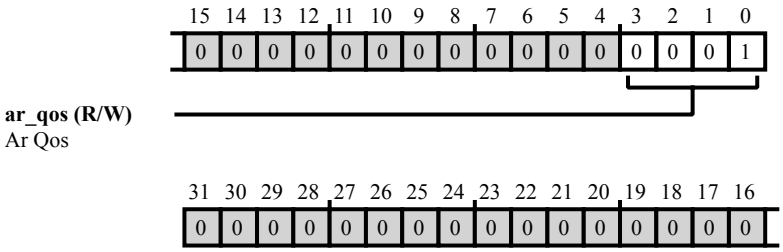


Figure 48-43: SCB0_HSMDMA1_CH0_IB_READ_QOS Register Diagram

Table 48-44: SCB0_HSMDMA1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

HSMDMA1 Channel 0 Write Quality of Service Register

The SCB0_HSMDMA1_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

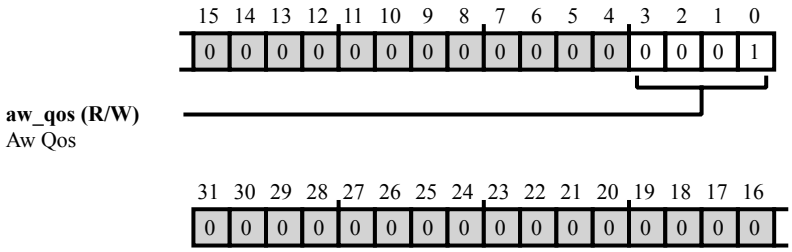


Figure 48-44: SCB0_HSMDMA1_CH0_IB_WRITE_QOS Register Diagram

Table 48-45: SCB0_HSMDMA1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

HSMDMA1 Channel 1 Read Quality of Service Register

The SCB0_HSMDMA1_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

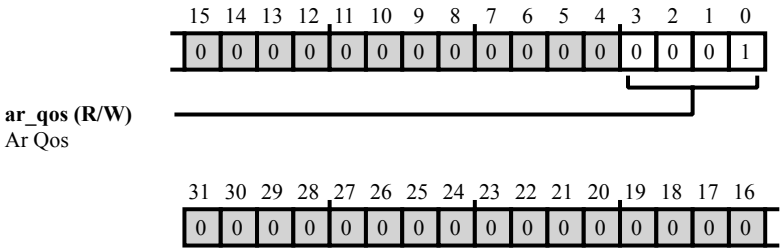


Figure 48-45: SCB0_HSMDMA1_CH1_IB_READ_QOS Register Diagram

Table 48-46: SCB0_HSMDMA1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

HSMDMA1 Channel 1 Write Quality of Service Register

The SCB0_HSMDMA1_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

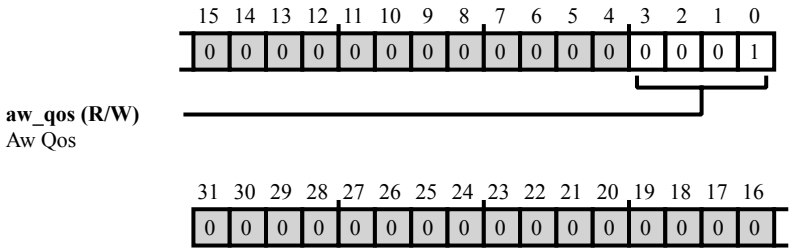


Figure 48-46: SCB0_HSMDMA1_CH1_IB_WRITE_QOS Register Diagram

Table 48-47: SCB0_HSMDMA1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

HSMDMA Channel 0 Read Quality of Service Register

The SCB0_HSMDMA_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

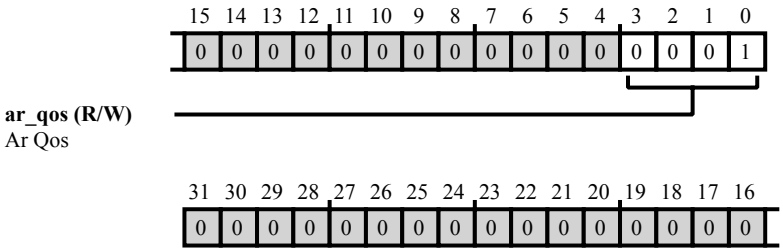


Figure 48-47: SCB0_HSMDMA_CH0_IB_READ_QOS Register Diagram

Table 48-48: SCB0_HSMDMA_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

HSMDMA Channel 0 Write Quality of Service Register

The SCB0_HSMDMA_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

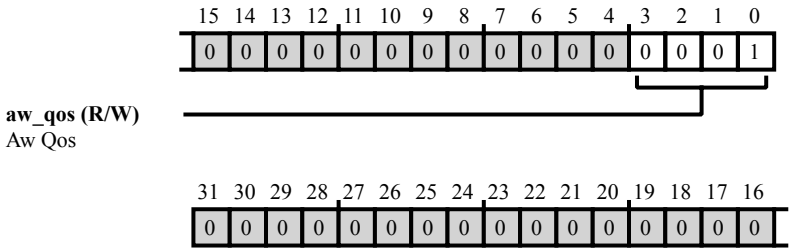


Figure 48-48: SCB0_HSMDMA_CH0_IB_WRITE_QOS Register Diagram

Table 48-49: SCB0_HSMDMA_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

HSMDMA Channel 1 Read Quality of Service Register

The SCB0_HSMDMA_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

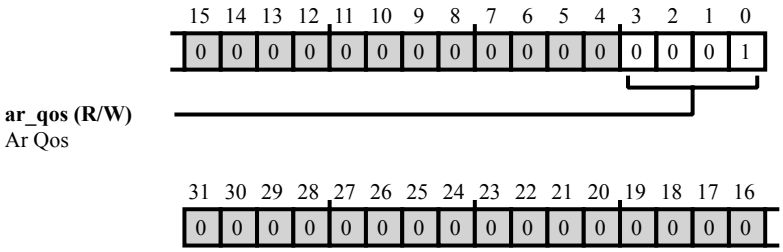


Figure 48-49: SCB0_HSMDMA_CH1_IB_READ_QOS Register Diagram

Table 48-50: SCB0_HSMDMA_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

HSMDMA Channel 1 Write Quality of Service Register

The SCB0_HSMDMA_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

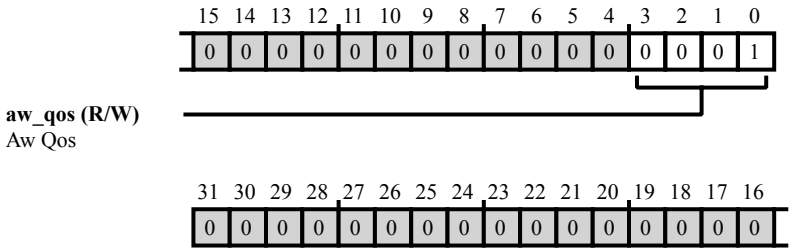


Figure 48-50: SCB0_HSMDMA_CH1_IB_WRITE_QOS Register Diagram

Table 48-51: SCB0_HSMDMA_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

LP0 Read Quality of Service Register

The SCB0_LP0_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

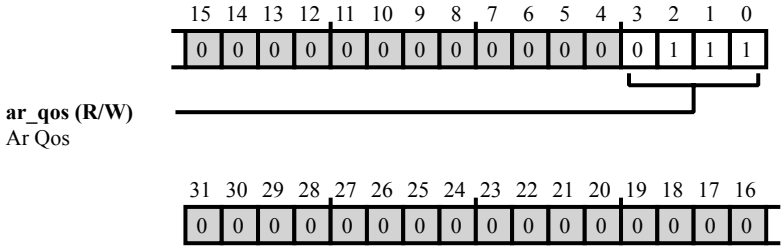


Figure 48-51: SCB0_LP0_READ_QOS Register Diagram

Table 48-52: SCB0_LP0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

LP0 Write Quality of Service Register

The SCB0_LP0_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

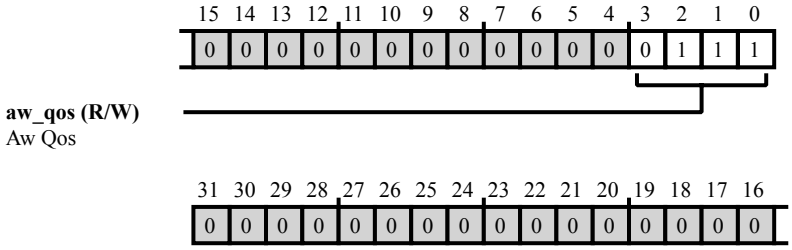


Figure 48-52: SCB0_LP0_WRITE_QOS Register Diagram

Table 48-53: SCB0_LP0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

LP1 Read Quality of Service Register

The SCB0_LP1_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

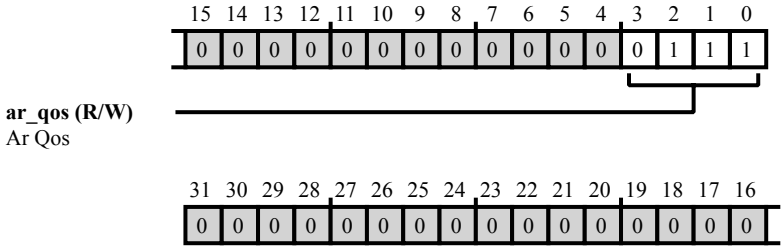


Figure 48-53: SCB0_LP1_READ_QOS Register Diagram

Table 48-54: SCB0_LP1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

LP1 Write Quality of Service Register

The SCB0_LP1_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

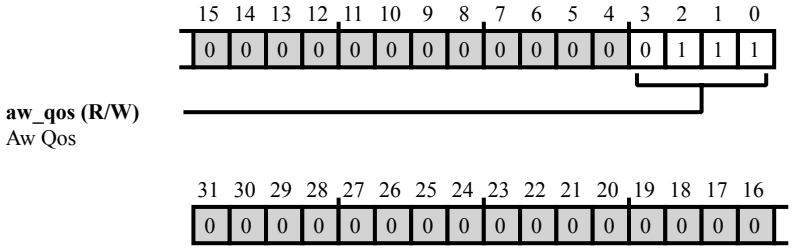


Figure 48-54: SCB0_LP1_WRITE_QOS Register Diagram

Table 48-55: SCB0_LP1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

MLB Read Quality of Service Register

The SCB0_MLB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

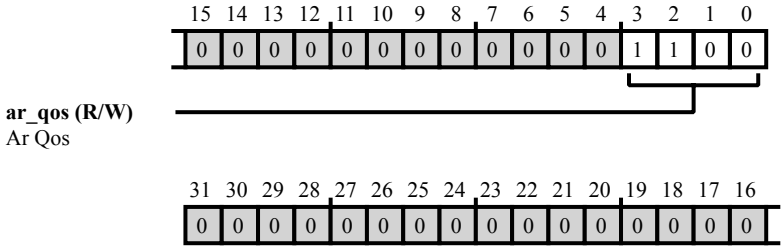


Figure 48-55: SCB0_MLB_READ_QOS Register Diagram

Table 48-56: SCB0_MLB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

MLB Write Quality of Service Register

The SCB0_MLB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

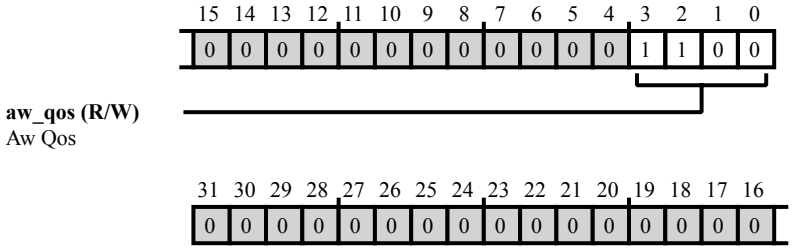


Figure 48-56: SCB0_MLB_WRITE_QOS Register Diagram

Table 48-57: SCB0_MLB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

MSMDMA1 Channel 0 Read Quality of Service Register

The SCB0_MSMDMA1_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

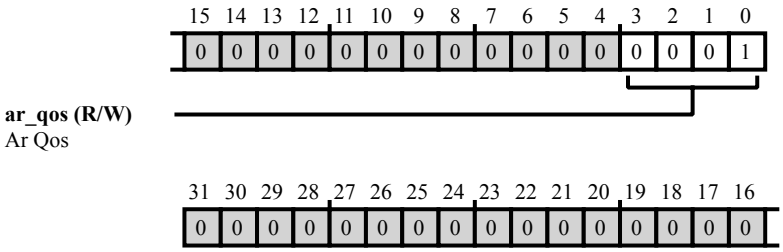


Figure 48-57: SCB0_MSMDMA1_CH0_IB_READ_QOS Register Diagram

Table 48-58: SCB0_MSMDMA1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

MSMDMA1 Channel 0 Write Quality of Service Register

The SCB0_MSMDMA1_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

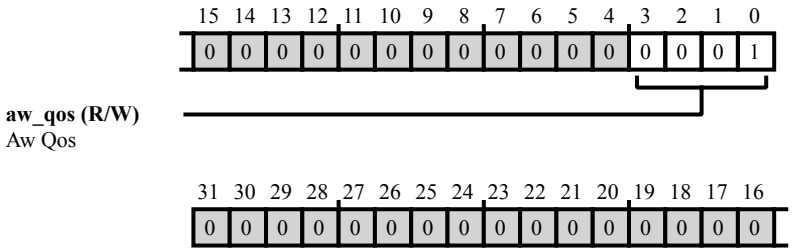


Figure 48-58: SCB0_MSMDMA1_CH0_IB_WRITE_QOS Register Diagram

Table 48-59: SCB0_MSMDMA1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

MSMDMA1 Channel 1 Read Quality of Service Register

The SCB0_MSMDMA1_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

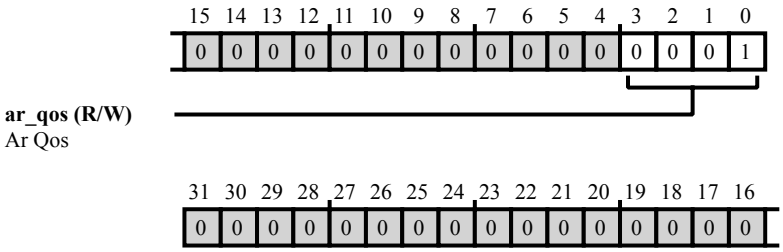


Figure 48-59: SCB0_MSMDMA1_CH1_IB_READ_QOS Register Diagram

Table 48-60: SCB0_MSMDMA1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

MSMDMA1 Channel 1 Write Quality of Service Register

The SCB0_MSMDMA1_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

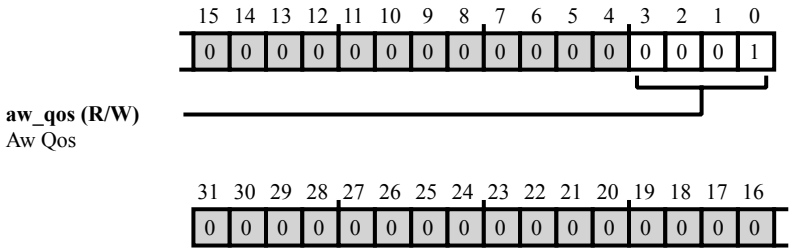


Figure 48-60: SCB0_MSMDMA1_CH1_IB_WRITE_QOS Register Diagram

Table 48-61: SCB0_MSMDMA1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

MSMDMA Channel 0 Read Quality of Service Register

The SCB0_MSMDMA_CH0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

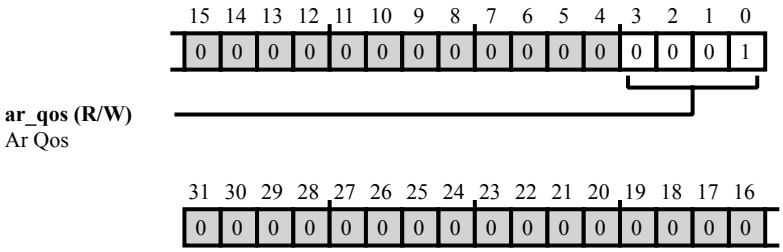


Figure 48-61: SCB0_MSMDMA_CH0_IB_READ_QOS Register Diagram

Table 48-62: SCB0_MSMDMA_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

MSMDMA Channel 0 Write Quality of Service Register

The SCB0_MSMDMA_CH0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

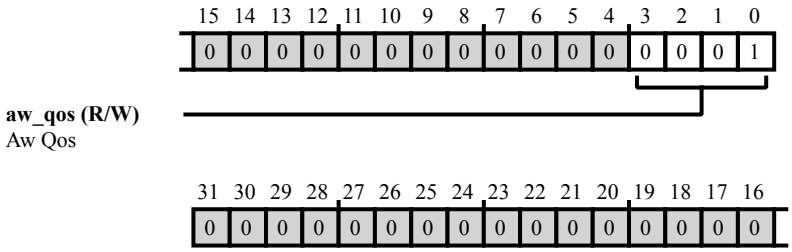


Figure 48-62: SCB0_MSMDMA_CH0_IB_WRITE_QOS Register Diagram

Table 48-63: SCB0_MSMDMA_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

MSMDMA Channel 1 Read Quality of Service Register

The SCB0_MSMDMA_CH1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

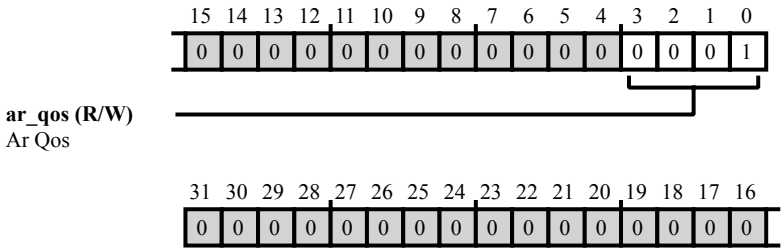


Figure 48-63: SCB0_MSMDMA_CH1_IB_READ_QOS Register Diagram

Table 48-64: SCB0_MSMDMA_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

MSMDMA Channel 1 Write Quality of Service Register

The SCB0_MSMDMA_CH1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

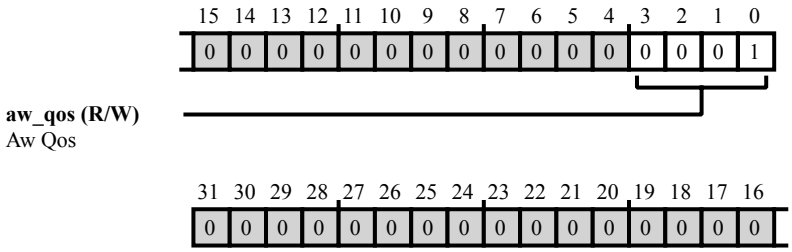


Figure 48-64: SCB0_MSMDMA_CH1_IB_WRITE_QOS Register Diagram

Table 48-65: SCB0_MSMDMA_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

PL310 M0 Read Quality of Service Register

The SCB0_PL310_M0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

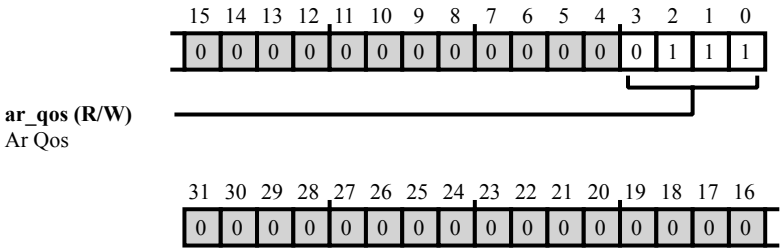


Figure 48-65: SCB0_PL310_M0_IB_READ_QOS Register Diagram

Table 48-66: SCB0_PL310_M0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

PL310 M0 Write Quality of Service Register

The SCB0_PL310_M0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

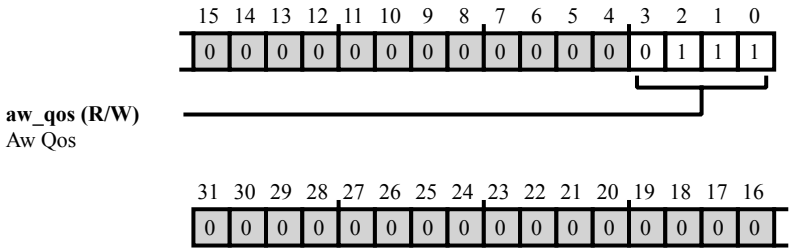


Figure 48-66: SCB0_PL310_M0_IB_WRITE_QOS Register Diagram

Table 48-67: SCB0_PL310_M0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

PL310 M1 Read Quality of Service Register

The SCB0_PL310_M1_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

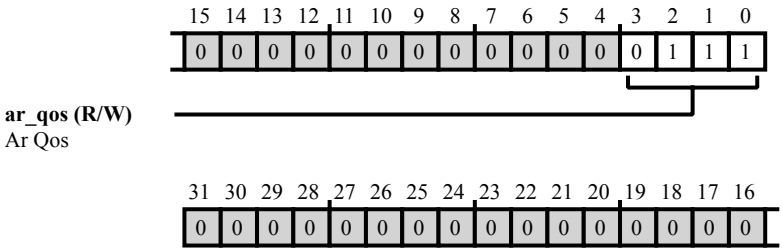


Figure 48-67: SCB0_PL310_M1_IB_READ_QOS Register Diagram

Table 48-68: SCB0_PL310_M1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

PL310 M1 Write Quality of Service Register

The SCB0_PL310_M1_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

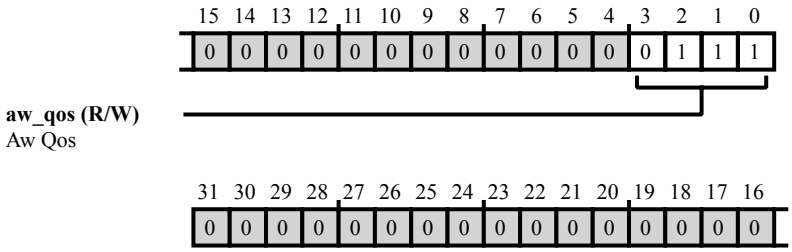


Figure 48-68: SCB0_PL310_M1_IB_WRITE_QOS Register Diagram

Table 48-69: SCB0_PL310_M1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

PL310 MMR Read Quality of Service Register

The SCB0_PL310_MMR_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

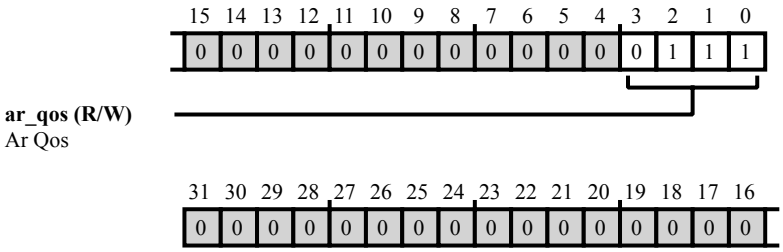


Figure 48-69: SCB0_PL310_MMR_IB_READ_QOS Register Diagram

Table 48-70: SCB0_PL310_MMR_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

PL310 MMR Write Quality of Service Register

The SCB0_PL310_MMR_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

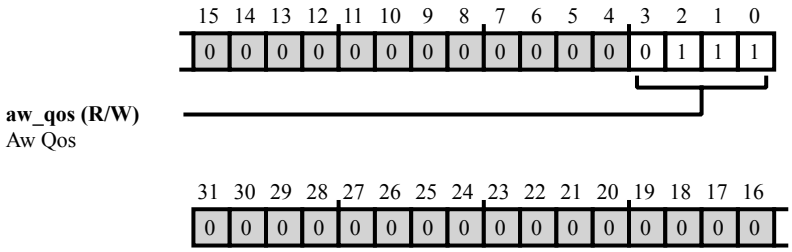


Figure 48-70: SCB0_PL310_MMR_IB_WRITE_QOS Register Diagram

Table 48-71: SCB0_PL310_MMR_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

PPI F0 Read Quality of Service Register

The SCB0_PPI_F0_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

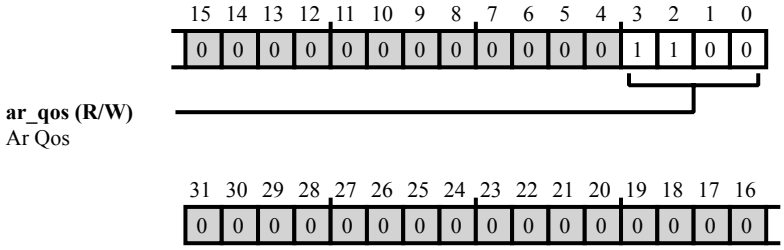


Figure 48-71: SCB0_PPI_F0_READ_QOS Register Diagram

Table 48-72: SCB0_PPI_F0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

PPI F0 Write Quality of Service Register

The SCB0_PPI_F0_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

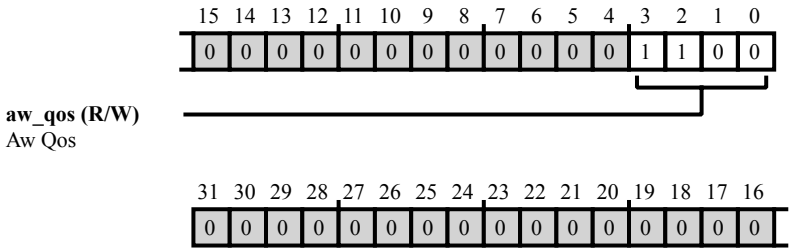


Figure 48-72: SCB0_PPI_F0_WRITE_QOS Register Diagram

Table 48-73: SCB0_PPI_F0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

PPI F1 Read Quality of Service Register

The SCB0_PPI_F1_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

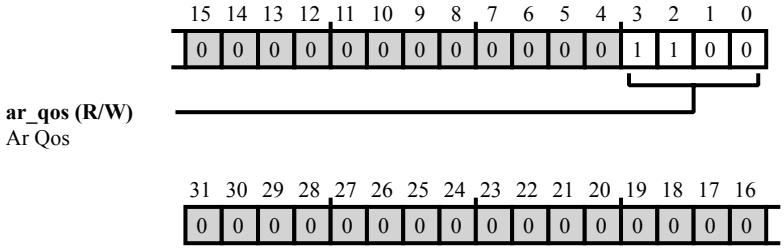


Figure 48-73: SCB0_PPI_F1_READ_QOS Register Diagram

Table 48-74: SCB0_PPI_F1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

PPI F1 Write Quality of Service Register

The SCB0_PPI_F1_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

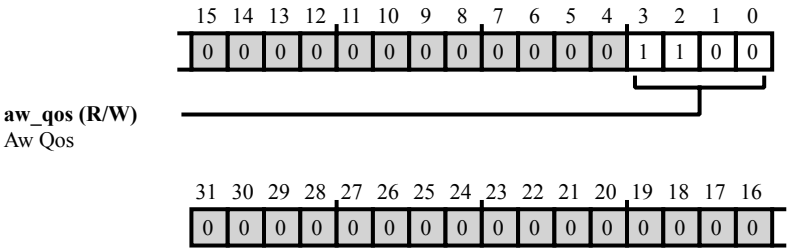


Figure 48-74: SCB0_PPI_F1_WRITE_QOS Register Diagram

Table 48-75: SCB0_PPI_F1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SCB6 Synchronization Mode Register

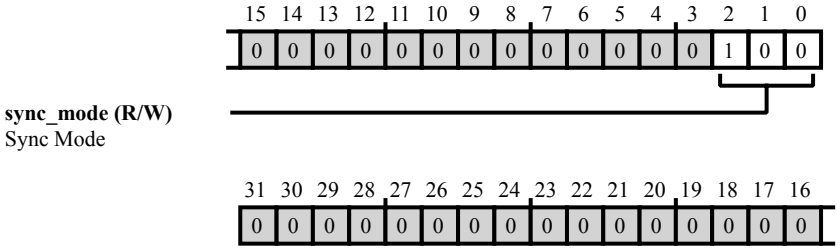


Figure 48-75: SCB0_SCB6_IB_SYNC_MODE Register Diagram

Table 48-76: SCB0_SCB6_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
2:0 (R/W)	SYNC_MODE	Sync Mode.

SDIO0 Read Quality of Service Register

The SCB0_SDIO0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

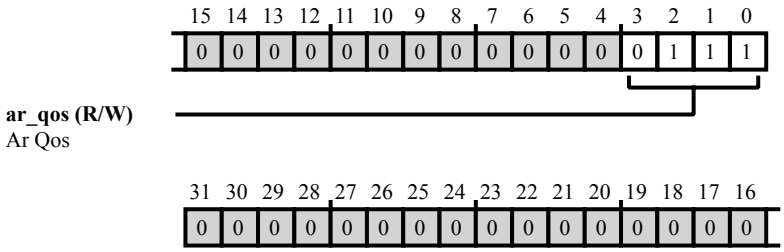


Figure 48-76: SCB0_SDIO0_IB_READ_QOS Register Diagram

Table 48-77: SCB0_SDIO0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SDIO0 Write Quality of Service Register

The SCB0_SDIO0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

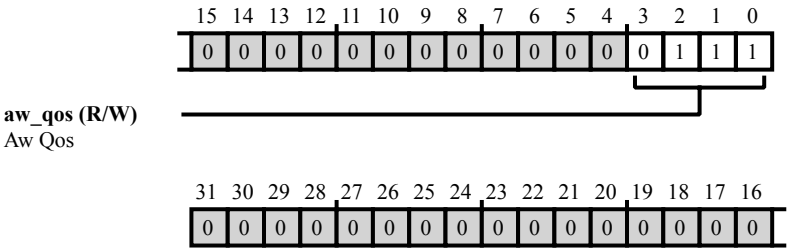


Figure 48-77: SCB0_SDIO0_IB_WRITE_QOS Register Diagram

Table 48-78: SCB0_SDIO0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH0 DPORT Read Quality of Service Register

The SCB0_SH0_DPORT_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

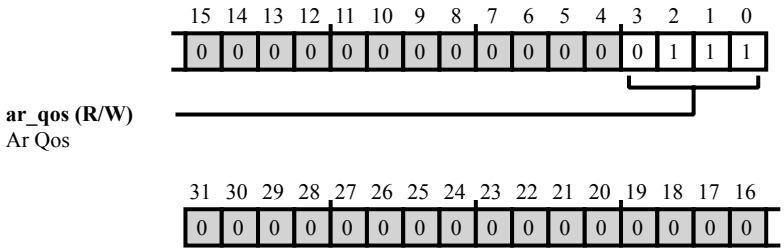


Figure 48-78: SCB0_SH0_DPORT_READ_QOS Register Diagram

Table 48-79: SCB0_SH0_DPORT_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH0 DPORT Write Quality of Service Register

The SCB0_SH0_DPORT_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

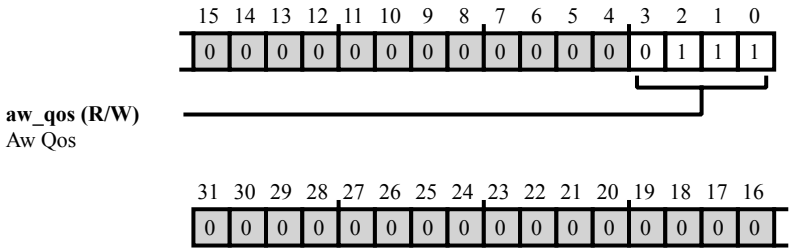


Figure 48-79: SCB0_SH0_DPORT_WRITE_QOS Register Diagram

Table 48-80: SCB0_SH0_DPORT_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH0 FIR Channel 0Read Quality of Service Register

The SCB0_SH0_FIR_CH0_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

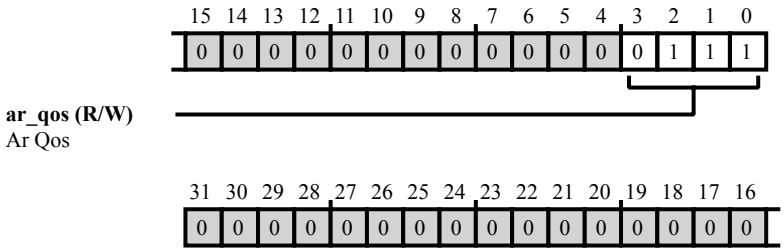


Figure 48-80: SCB0_SH0_FIR_CH0_READ_QOS Register Diagram

Table 48-81: SCB0_SH0_FIR_CH0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH0 FIR Channel 0Write Quality of Service Register

The SCB0_SH0_FIR_CH0_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

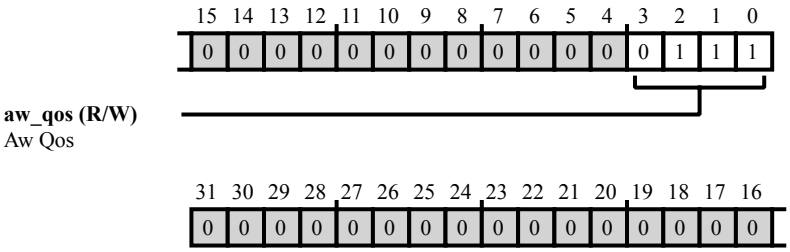


Figure 48-81: SCB0_SH0_FIR_CH0_WRITE_QOS Register Diagram

Table 48-82: SCB0_SH0_FIR_CH0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH0 FIR Channel 1Read Quality of Service Register

The SCB0_SH0_FIR_CH1_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

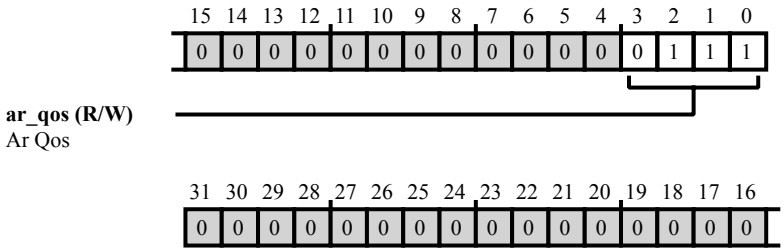


Figure 48-82: SCB0_SH0_FIR_CH1_READ_QOS Register Diagram

Table 48-83: SCB0_SH0_FIR_CH1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH0 FIR Channel 1Write Quality of Service Register

The SCB0_SH0_FIR_CH1_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

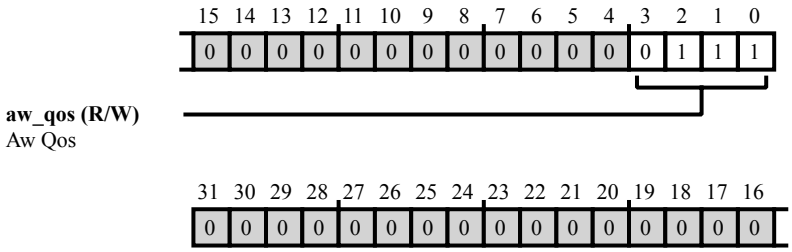


Figure 48-83: SCB0_SH0_FIR_CH1_WRITE_QOS Register Diagram

Table 48-84: SCB0_SH0_FIR_CH1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH0 IIR Channel 0 Read Quality of Service Register

The SCB0_SH0_IIR_CH0_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

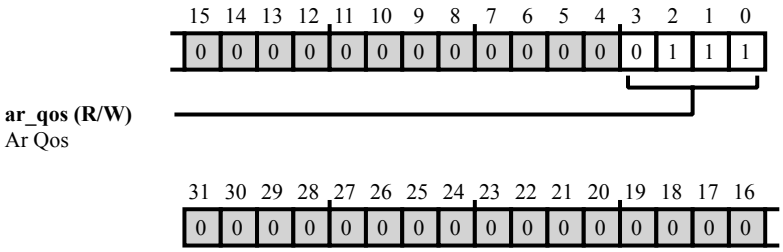


Figure 48-84: SCB0_SH0_IIR_CH0_READ_QOS Register Diagram

Table 48-85: SCB0_SH0_IIR_CH0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH0 IIR Channel 0 Write Quality of Service Register

The SCB0_SH0_IIR_CH0_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

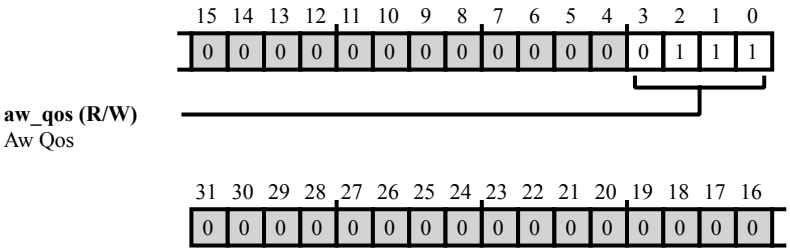


Figure 48-85: SCB0_SH0_IIR_CH0_WRITE_QOS Register Diagram

Table 48-86: SCB0_SH0_IIR_CH0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH0 IIR Channel 1 Read Quality of Service Register

The SCB0_SH0_IIR_CH1_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

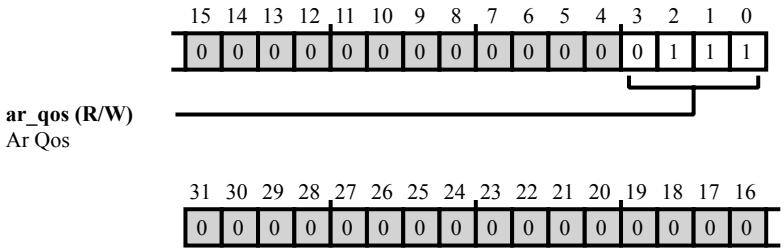


Figure 48-86: SCB0_SH0_IIR_CH1_READ_QOS Register Diagram

Table 48-87: SCB0_SH0_IIR_CH1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH0 IIR Channel 1 Write Quality of Service Register

The SCB0_SH0_IIR_CH1_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

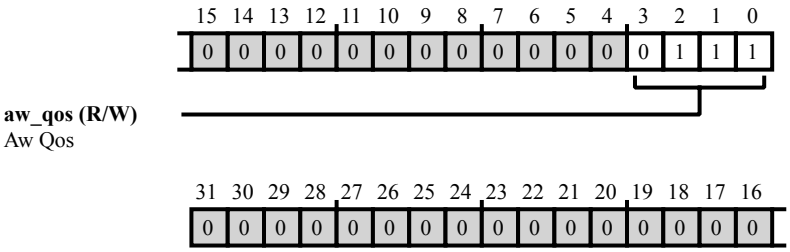


Figure 48-87: SCB0_SH0_IIR_CH1_WRITE_QOS Register Diagram

Table 48-88: SCB0_SH0_IIR_CH1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH0 IPORT Read Quality of Service Register

The SCB0_SH0_IPORT_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

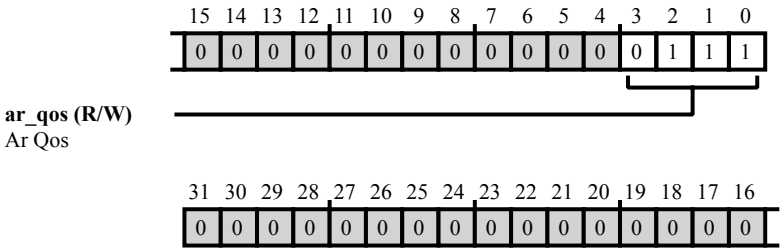


Figure 48-88: SCB0_SH0_IPORT_READ_QOS Register Diagram

Table 48-89: SCB0_SH0_IPORT_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH0 IPORT Write Quality of Service Register

The SCB0_SH0_IPORT_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

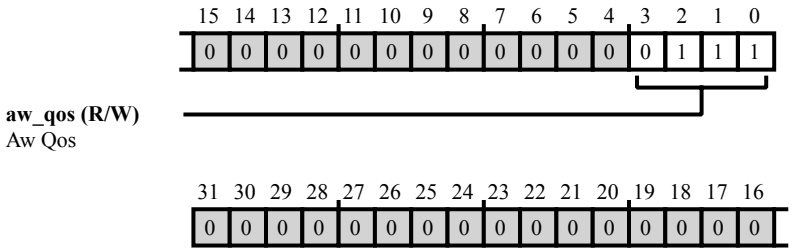


Figure 48-89: SCB0_SH0_IPORT_WRITE_QOS Register Diagram

Table 48-90: SCB0_SH0_IPORT_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH0 MMR Read Quality of Service Register

The SCB0_SH0_MMR_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

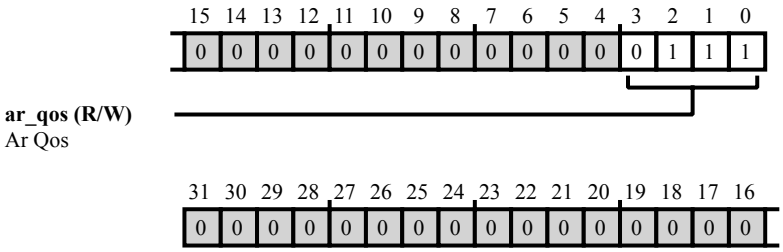


Figure 48-90: SCB0_SH0_MMR_IB_READ_QOS Register Diagram

Table 48-91: SCB0_SH0_MMR_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH0 MMR Write Quality of Service Register

The SCB0_SH0_MMR_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

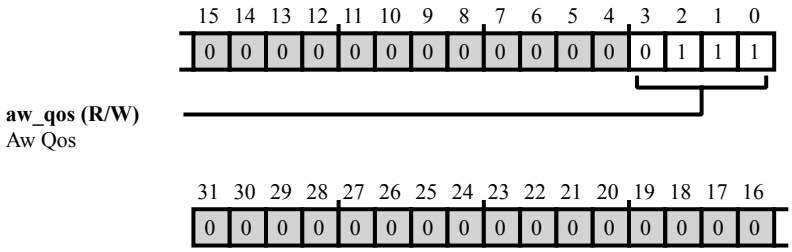


Figure 48-91: SCB0_SH0_MMR_IB_WRITE_QOS Register Diagram

Table 48-92: SCB0_SH0_MMR_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH1 DPORT Read Quality of Service Register

The SCB0_SH1_DPORT_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

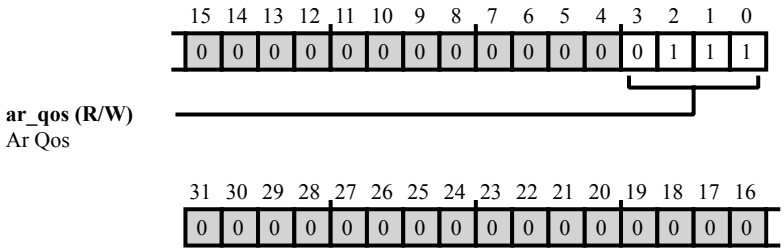


Figure 48-92: SCB0_SH1_DPORT_READ_QOS Register Diagram

Table 48-93: SCB0_SH1_DPORT_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH1 DPORT Write Quality of Service Register

The SCB0_SH1_DPORT_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

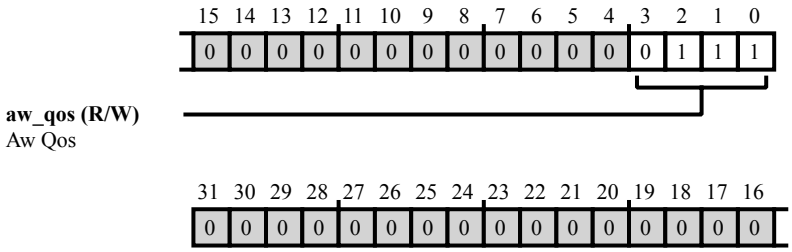


Figure 48-93: SCB0_SH1_DPORT_WRITE_QOS Register Diagram

Table 48-94: SCB0_SH1_DPORT_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH1 FIR Channel 0 Read Quality of Service Register

The SCB0_SH1_FIR_CH0_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

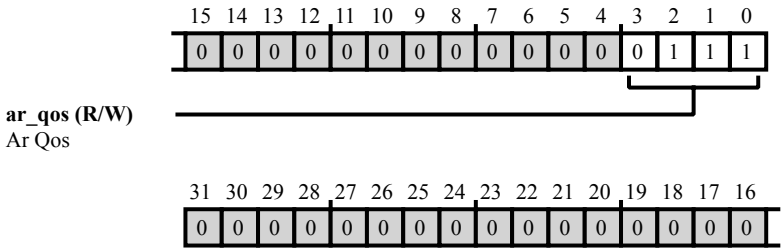


Figure 48-94: SCB0_SH1_FIR_CH0_READ_QOS Register Diagram

Table 48-95: SCB0_SH1_FIR_CH0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH1 FIR Channel 0 Write Quality of Service Register

The SCB0_SH1_FIR_CH0_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

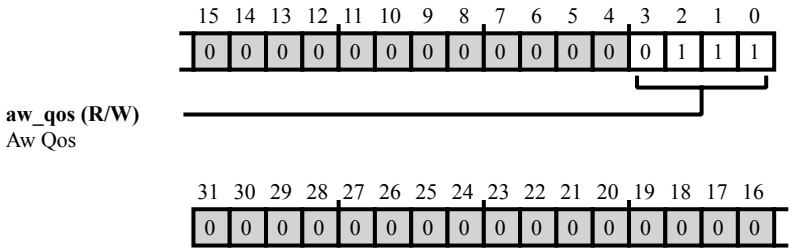


Figure 48-95: SCB0_SH1_FIR_CH0_WRITE_QOS Register Diagram

Table 48-96: SCB0_SH1_FIR_CH0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH1 FIR Channel 1 Read Quality of Service Register

The SCB0_SH1_FIR_CH1_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

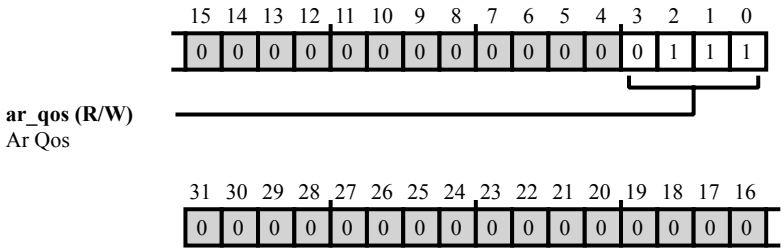


Figure 48-96: SCB0_SH1_FIR_CH1_READ_QOS Register Diagram

Table 48-97: SCB0_SH1_FIR_CH1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH1 FIR Channel 1 Write Quality of Service Register

The SCB0_SH1_FIR_CH1_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

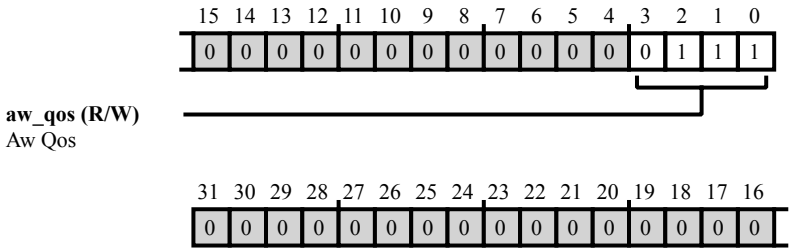


Figure 48-97: SCB0_SH1_FIR_CH1_WRITE_QOS Register Diagram

Table 48-98: SCB0_SH1_FIR_CH1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH1 IIR Channel 0 Read Quality of Service Register

The SCB0_SH1_IIR_CH0_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

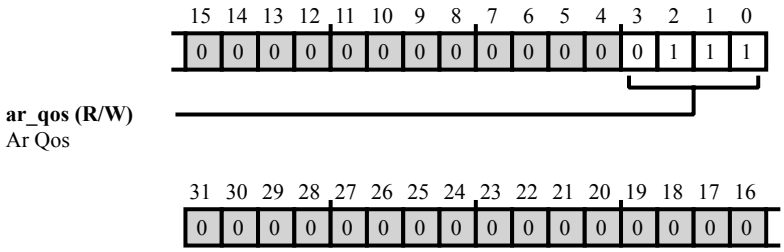


Figure 48-98: SCB0_SH1_IIR_CH0_READ_QOS Register Diagram

Table 48-99: SCB0_SH1_IIR_CH0_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH1 IIR Channel 0 Write Quality of Service Register

The SCB0_SH1_IIR_CH0_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

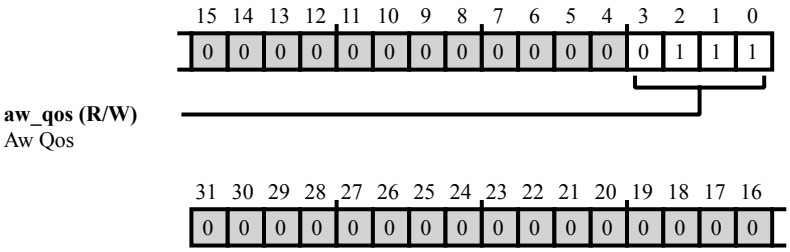


Figure 48-99: SCB0_SH1_IIR_CH0_WRITE_QOS Register Diagram

Table 48-100: SCB0_SH1_IIR_CH0_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH1 IIR Channel 1 Read Quality of Service Register

The SCB0_SH1_IIR_CH1_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

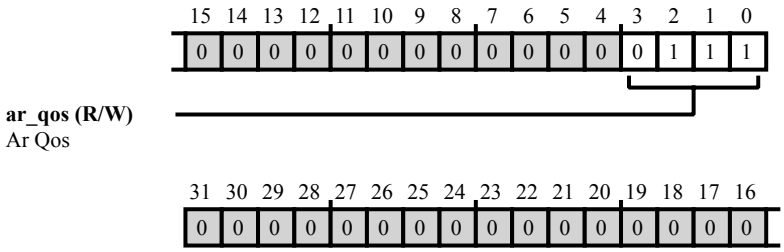


Figure 48-100: SCB0_SH1_IIR_CH1_READ_QOS Register Diagram

Table 48-101: SCB0_SH1_IIR_CH1_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH1 IIR Channel 1 Write Quality of Service Register

The SCB0_SH1_IIR_CH1_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

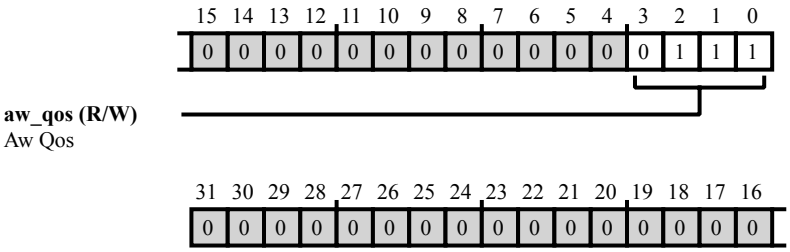


Figure 48-101: SCB0_SH1_IIR_CH1_WRITE_QOS Register Diagram

Table 48-102: SCB0_SH1_IIR_CH1_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH1 IPORT Read Quality of Service Register

The SCB0_SH1_IPORT_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

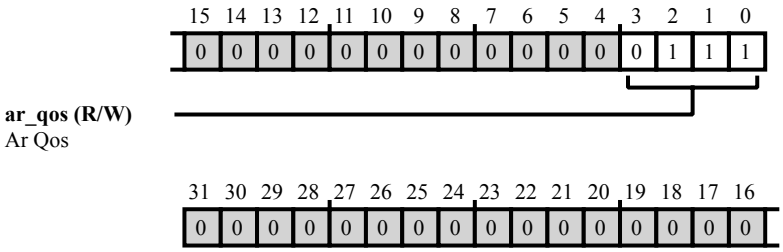


Figure 48-102: SCB0_SH1_IPORT_READ_QOS Register Diagram

Table 48-103: SCB0_SH1_IPORT_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH1 IPORT Write Quality of Service Register

The SCB0_SH1_IPORT_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

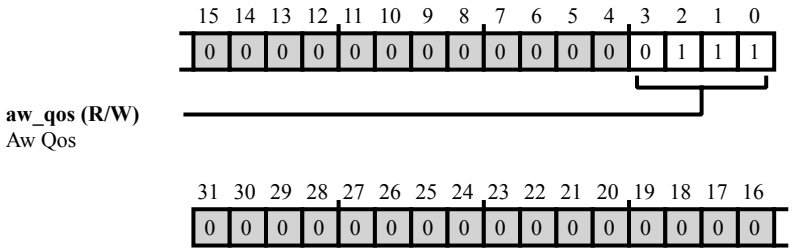


Figure 48-103: SCB0_SH1_IPORT_WRITE_QOS Register Diagram

Table 48-104: SCB0_SH1_IPORT_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SH1 MMR Read Quality of Service Register

The SCB0_SH1_MMR_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

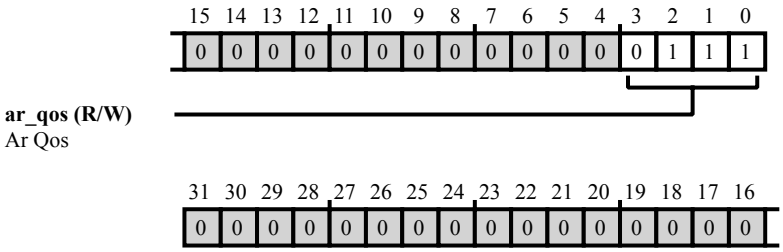


Figure 48-104: SCB0_SH1_MMR_IB_READ_QOS Register Diagram

Table 48-105: SCB0_SH1_MMR_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SH1 MMR Write Quality of Service Register

The SCB0_SH1_MMR_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

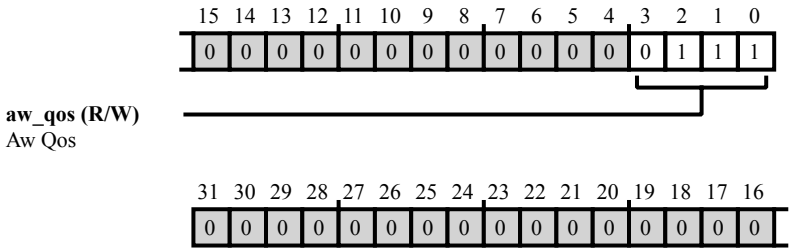


Figure 48-105: SCB0_SH1_MMR_IB_WRITE_QOS Register Diagram

Table 48-106: SCB0_SH1_MMR_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP0A Read Quality of Service Register

The SCB0_SP0A_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

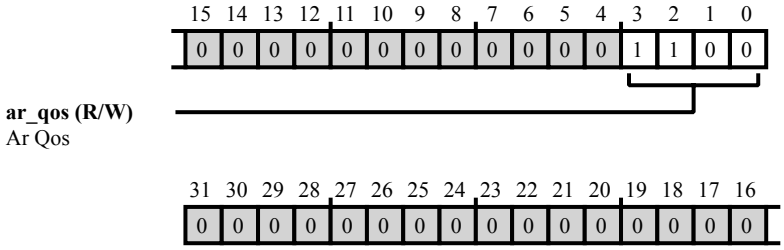


Figure 48-106: SCB0_SP0A_READ_QOS Register Diagram

Table 48-107: SCB0_SP0A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP0A Write Quality of Service Register

The SCB0_SP0A_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

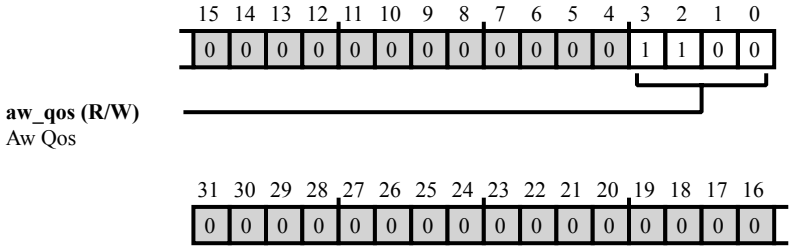


Figure 48-107: SCB0_SP0A_WRITE_QOS Register Diagram

Table 48-108: SCB0_SP0A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP0B Read Quality of Service Register

The SCB0_SP0B_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

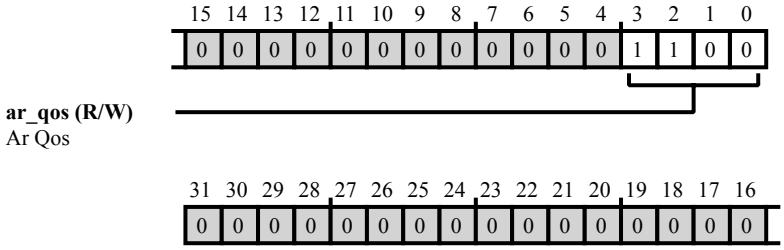


Figure 48-108: SCB0_SP0B_READ_QOS Register Diagram

Table 48-109: SCB0_SP0B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP0B Write Quality of Service Register

The SCB0_SP0B_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

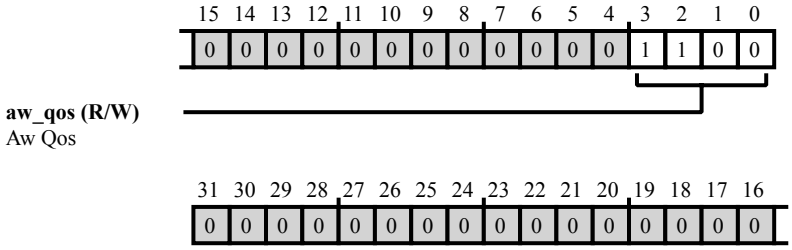


Figure 48-109: SCB0_SP0B_WRITE_QOS Register Diagram

Table 48-110: SCB0_SP0B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP1A Read Quality of Service Register

The SCB0_SP1A_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

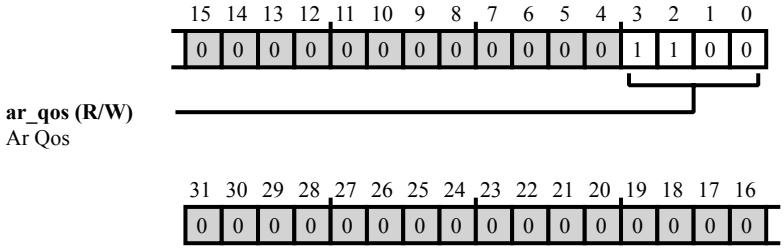


Figure 48-110: SCB0_SP1A_READ_QOS Register Diagram

Table 48-111: SCB0_SP1A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP1A Write Quality of Service Register

The SCB0_SP1A_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

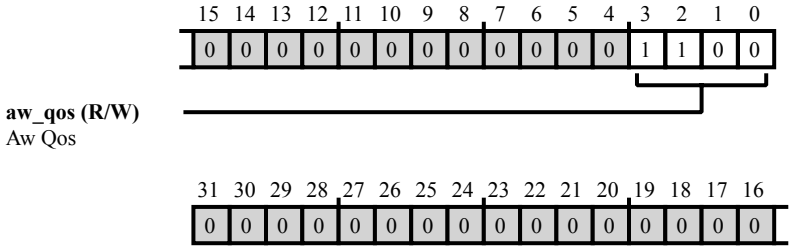


Figure 48-111: SCB0_SP1A_WRITE_QOS Register Diagram

Table 48-112: SCB0_SP1A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP1B Read Quality of Service Register

The SCB0_SP1B_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

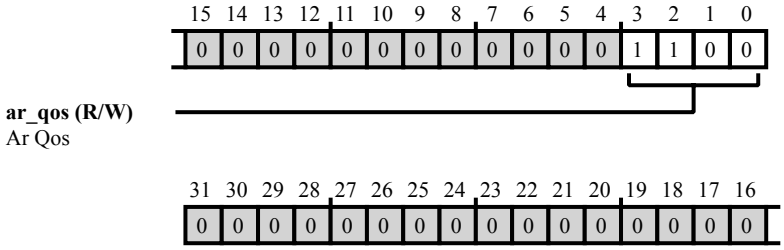


Figure 48-112: SCB0_SP1B_READ_QOS Register Diagram

Table 48-113: SCB0_SP1B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP1B Write Quality of Service Register

The SCB0_SP1B_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

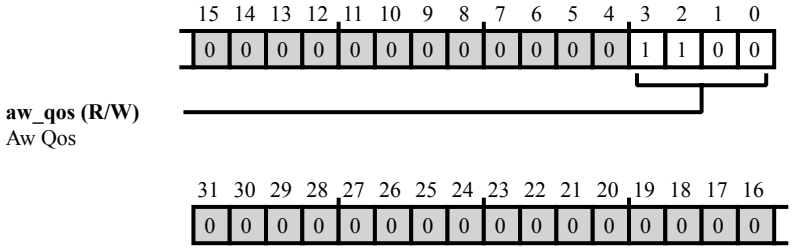


Figure 48-113: SCB0_SP1B_WRITE_QOS Register Diagram

Table 48-114: SCB0_SP1B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP2A Read Quality of Service Register

The SCB0_SP2A_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

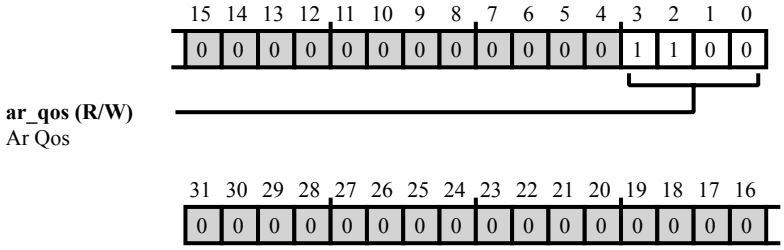


Figure 48-114: SCB0_SP2A_READ_QOS Register Diagram

Table 48-115: SCB0_SP2A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP2A Write Quality of Service Register

The SCB0_SP2A_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

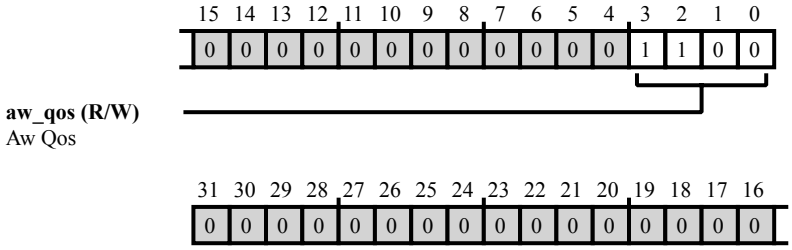


Figure 48-115: SCB0_SP2A_WRITE_QOS Register Diagram

Table 48-116: SCB0_SP2A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP1B Read Quality of Service Register

The SCB0_SP2B_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

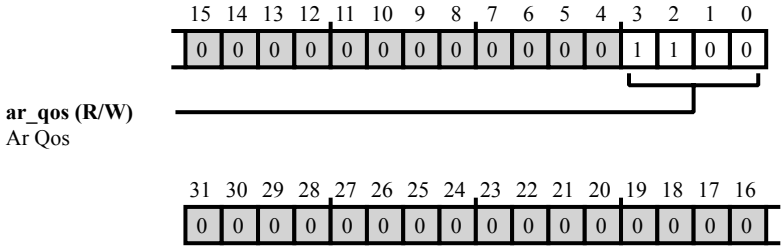


Figure 48-116: SCB0_SP2B_READ_QOS Register Diagram

Table 48-117: SCB0_SP2B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP1B Write Quality of Service Register

The SCB0_SP2B_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

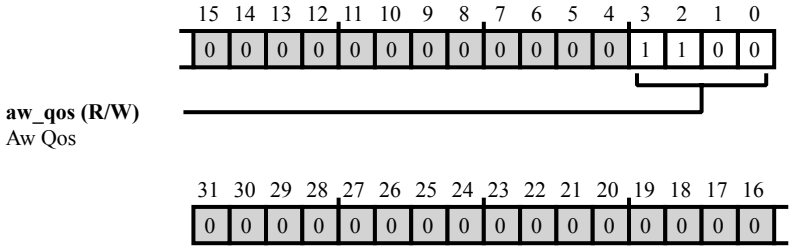


Figure 48-117: SCB0_SP2B_WRITE_QOS Register Diagram

Table 48-118: SCB0_SP2B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP3A Read Quality of Service Register

The SCB0_SP3A_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

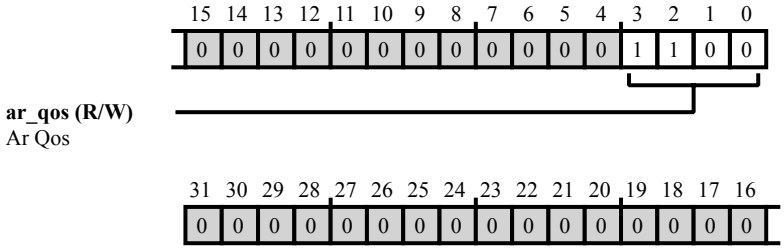


Figure 48-118: SCB0_SP3A_READ_QOS Register Diagram

Table 48-119: SCB0_SP3A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP3A Write Quality of Service Register

The SCB0_SP3A_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

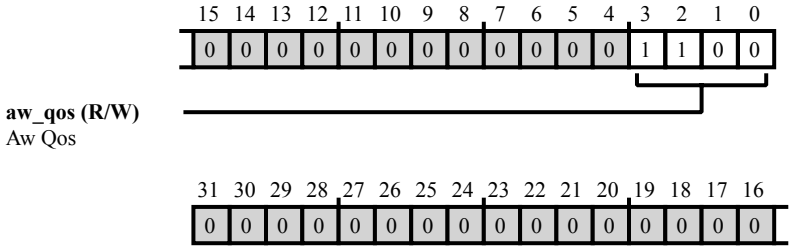


Figure 48-119: SCB0_SP3A_WRITE_QOS Register Diagram

Table 48-120: SCB0_SP3A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP3B Read Quality of Service Register

The SCB0_SP3B_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

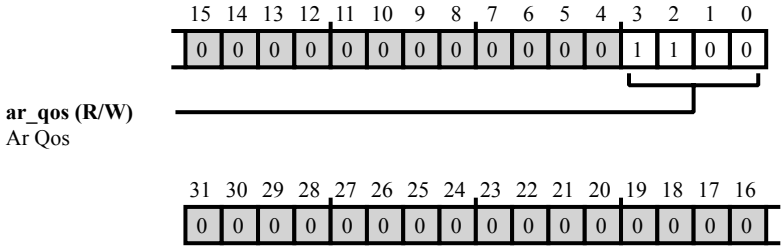


Figure 48-120: SCB0_SP3B_READ_QOS Register Diagram

Table 48-121: SCB0_SP3B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP3B Write Quality of Service Register

The SCB0_SP3B_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

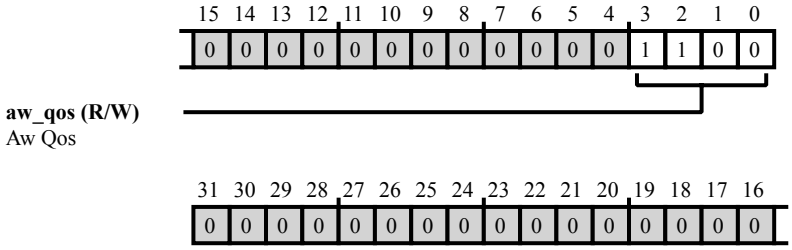


Figure 48-121: SCB0_SP3B_WRITE_QOS Register Diagram

Table 48-122: SCB0_SP3B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP4A Read Quality of Service Register

The SCB0_SP4A_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

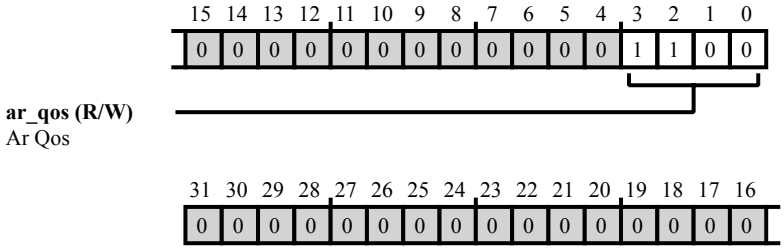


Figure 48-122: SCB0_SP4A_READ_QOS Register Diagram

Table 48-123: SCB0_SP4A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP4A Write Quality of Service Register

The SCB0_SP4A_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

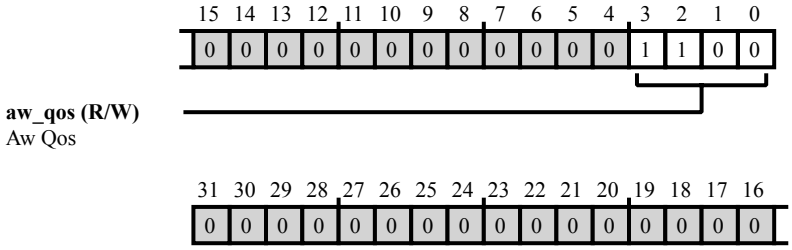


Figure 48-123: SCB0_SP4A_WRITE_QOS Register Diagram

Table 48-124: SCB0_SP4A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP4B Read Quality of Service Register

The SCB0_SP4B_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

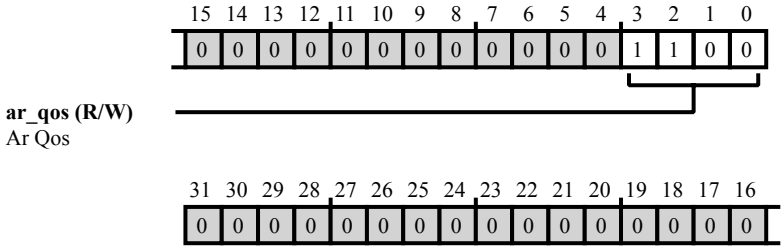


Figure 48-124: SCB0_SP4B_READ_QOS Register Diagram

Table 48-125: SCB0_SP4B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP4B Write Quality of Service Register

The SCB0_SP4B_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

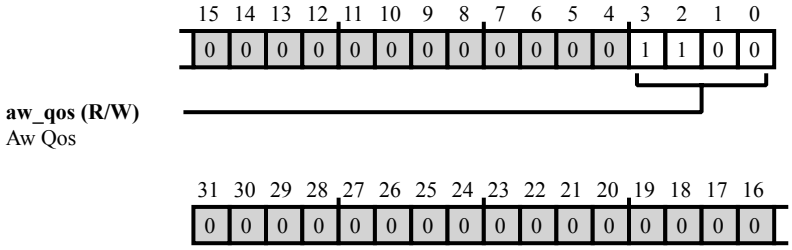


Figure 48-125: SCB0_SP4B_WRITE_QOS Register Diagram

Table 48-126: SCB0_SP4B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP5A Read Quality of Service Register

The SCB0_SP5A_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

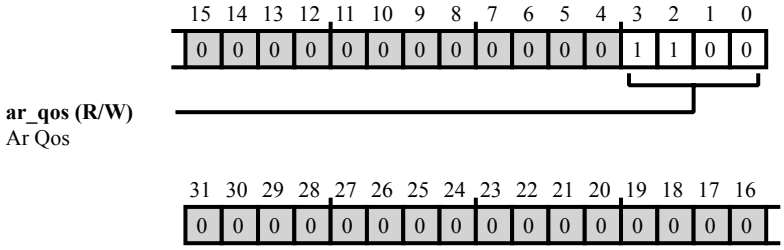


Figure 48-126: SCB0_SP5A_READ_QOS Register Diagram

Table 48-127: SCB0_SP5A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP5A Write Quality of Service Register

The SCB0_SP5A_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

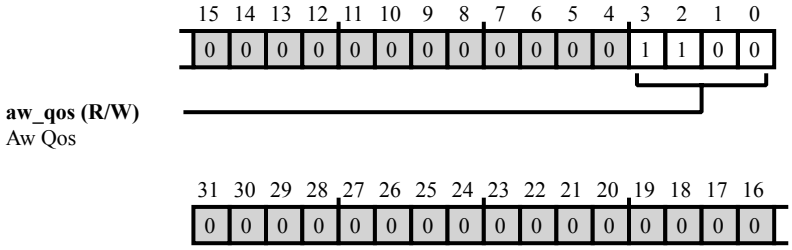


Figure 48-127: SCB0_SP5A_WRITE_QOS Register Diagram

Table 48-128: SCB0_SP5A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP5B Read Quality of Service Register

The SCB0_SP5B_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

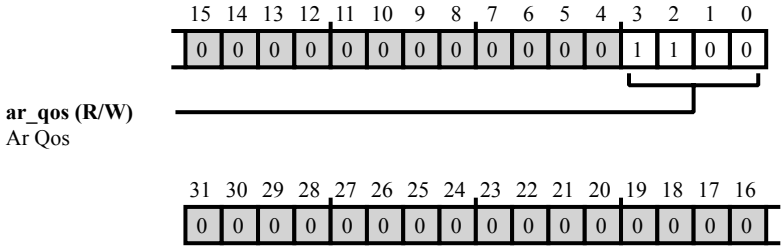


Figure 48-128: SCB0_SP5B_READ_QOS Register Diagram

Table 48-129: SCB0_SP5B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP5B Write Quality of Service Register

The SCB0_SP5B_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

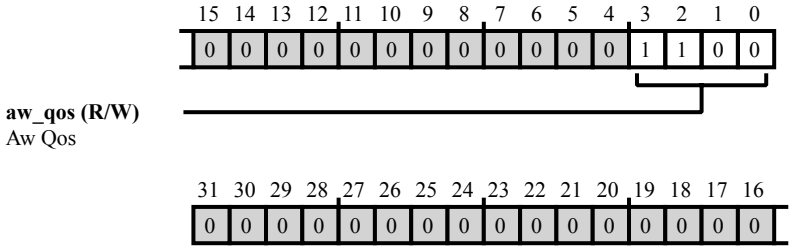


Figure 48-129: SCB0_SP5B_WRITE_QOS Register Diagram

Table 48-130: SCB0_SP5B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP6A Read Quality of Service Register

The SCB0_SP6A_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

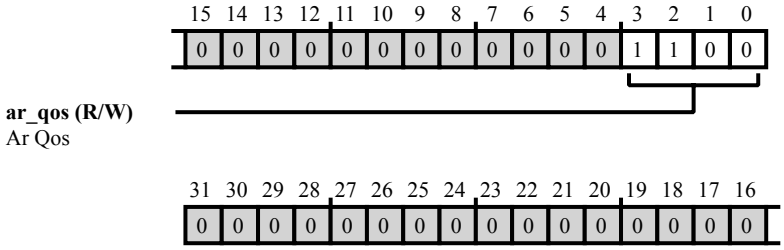


Figure 48-130: SCB0_SP6A_READ_QOS Register Diagram

Table 48-131: SCB0_SP6A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP6A Write Quality of Service Register

The SCB0_SP6A_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

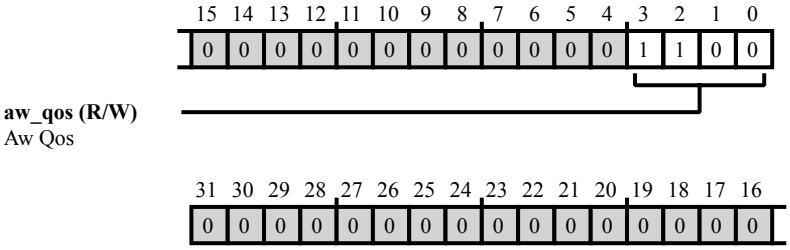


Figure 48-131: SCB0_SP6A_WRITE_QOS Register Diagram

Table 48-132: SCB0_SP6A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP6B Read Quality of Service Register

The SCB0_SP6B_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

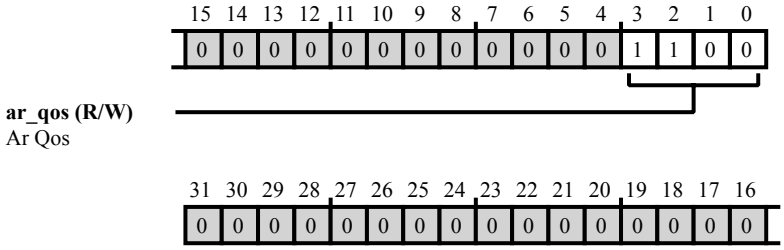


Figure 48-132: SCB0_SP6B_READ_QOS Register Diagram

Table 48-133: SCB0_SP6B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP6B Write Quality of Service Register

The SCB0_SP6B_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

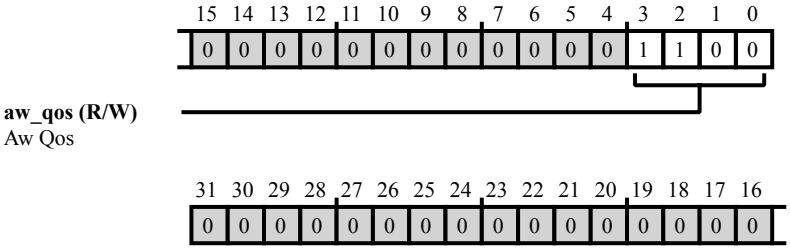


Figure 48-133: SCB0_SP6B_WRITE_QOS Register Diagram

Table 48-134: SCB0_SP6B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP7A Read Quality of Service Register

The SCB0_SP7A_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

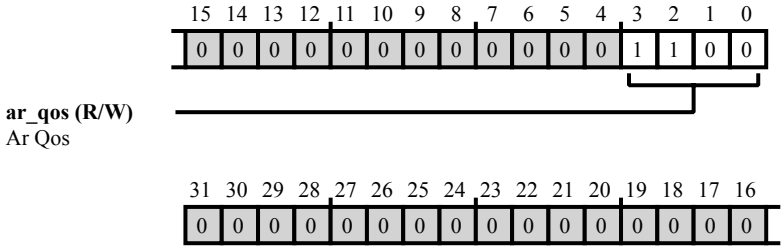


Figure 48-134: SCB0_SP7A_READ_QOS Register Diagram

Table 48-135: SCB0_SP7A_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP7A Write Quality of Service Register

The SCB0_SP7A_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

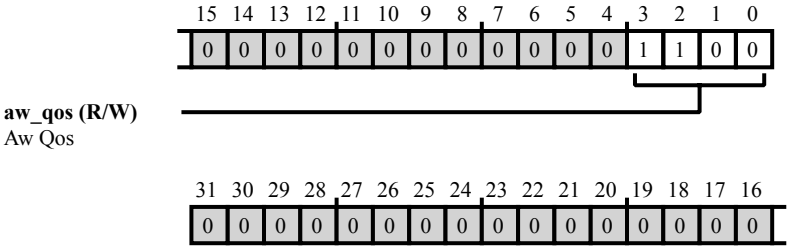


Figure 48-135: SCB0_SP7A_WRITE_QOS Register Diagram

Table 48-136: SCB0_SP7A_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SP7B Read Quality of Service Register

The SCB0_SP7B_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

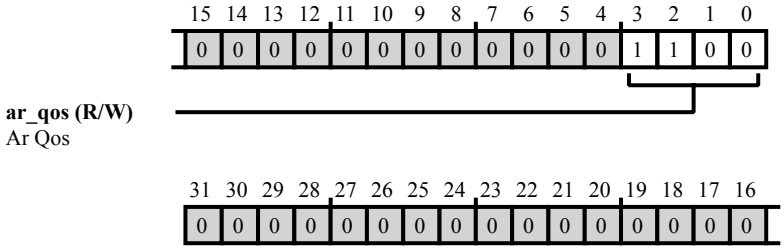


Figure 48-136: SCB0_SP7B_READ_QOS Register Diagram

Table 48-137: SCB0_SP7B_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SP7B Write Quality of Service Register

The SCB0_SP7B_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

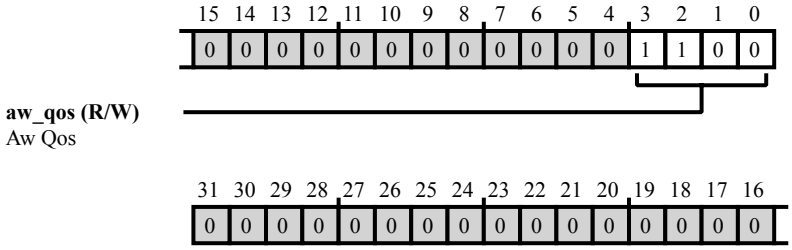


Figure 48-137: SCB0_SP7B_WRITE_QOS Register Diagram

Table 48-138: SCB0_SP7B_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SPI0 RX Read Quality of Service Register

The SCB0_SPI0RX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

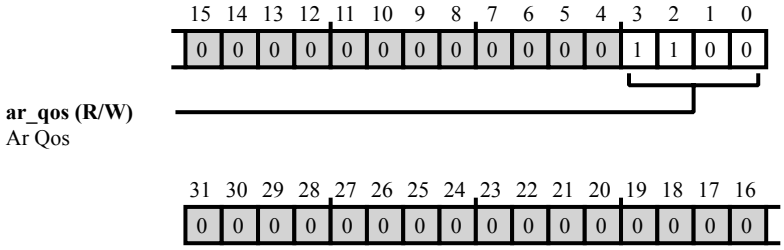


Figure 48-138: SCB0_SPI0RX_READ_QOS Register Diagram

Table 48-139: SCB0_SPI0RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SPI0 RX Write Quality of Service Register

The SCB0_SPI0RX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

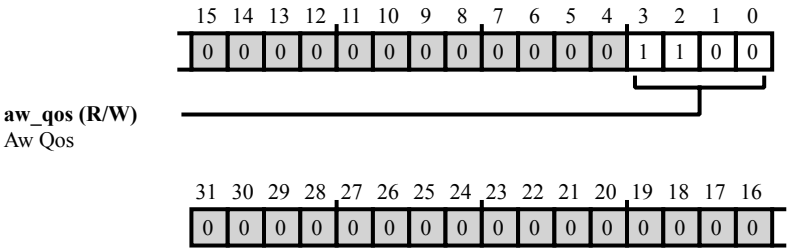


Figure 48-139: SCB0_SPI0RX_WRITE_QOS Register Diagram

Table 48-140: SCB0_SPI0RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SPI0 TX Read Quality of Service Register

The SCB0_SPI0TX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

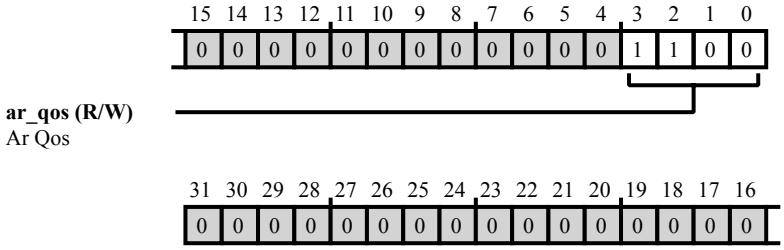


Figure 48-140: SCB0_SPI0TX_READ_QOS Register Diagram

Table 48-141: SCB0_SPI0TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SPI0 TX Write Quality of Service Register

The SCB0_SPI0TX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

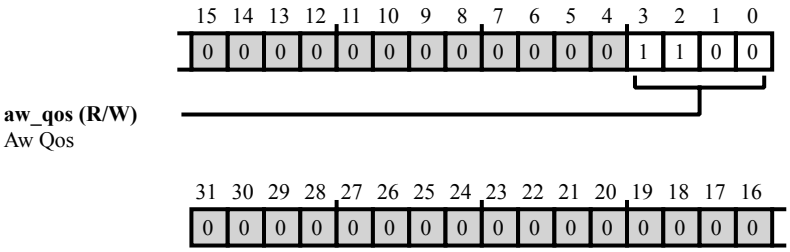


Figure 48-141: SCB0_SPI0TX_WRITE_QOS Register Diagram

Table 48-142: SCB0_SPI0TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SPI1 RX Read Quality of Service Register

The SCB0_SPI1RX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

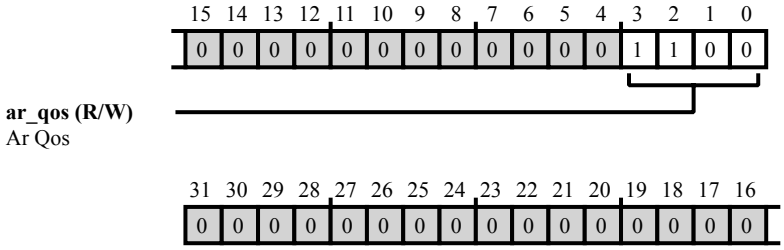


Figure 48-142: SCB0_SPI1RX_READ_QOS Register Diagram

Table 48-143: SCB0_SPI1RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SPI1 RX Write Quality of Service Register

The SCB0_SPI1RX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

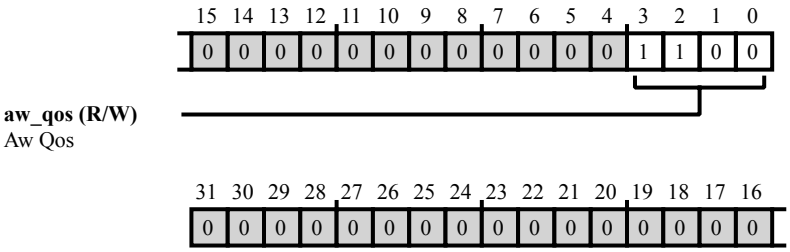


Figure 48-143: SCB0_SPI1RX_WRITE_QOS Register Diagram

Table 48-144: SCB0_SPI1RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SPI1 TX Read Quality of Service Register

The SCB0_SPI1TX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

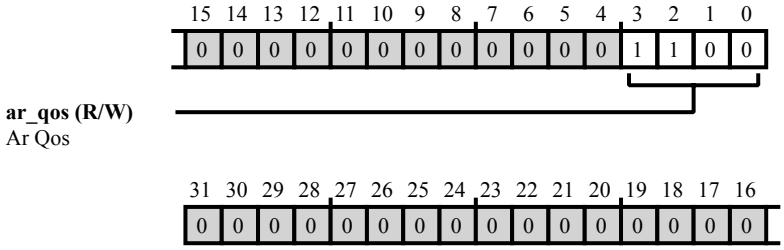


Figure 48-144: SCB0_SPI1TX_READ_QOS Register Diagram

Table 48-145: SCB0_SPI1TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SPI1 TX Write Quality of Service Register

The SCB0_SPI1TX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

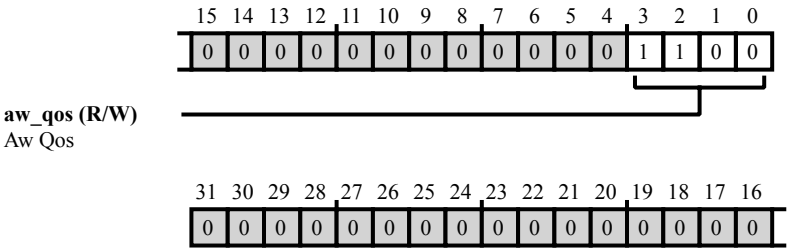


Figure 48-145: SCB0_SPI1TX_WRITE_QOS Register Diagram

Table 48-146: SCB0_SPI1TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SPI2 RX Read Quality of Service Register

The SCB0_SPI2RX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

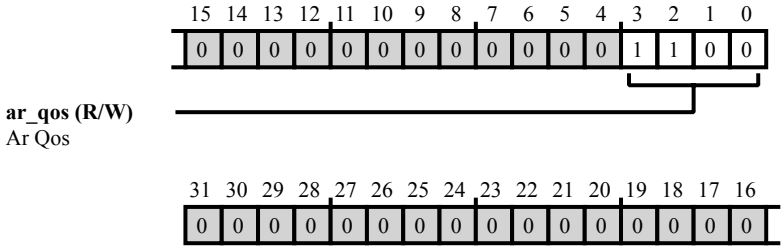


Figure 48-146: SCB0_SPI2RX_READ_QOS Register Diagram

Table 48-147: SCB0_SPI2RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SPI2 RX Write Quality of Service Register

The SCB0_SPI2RX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

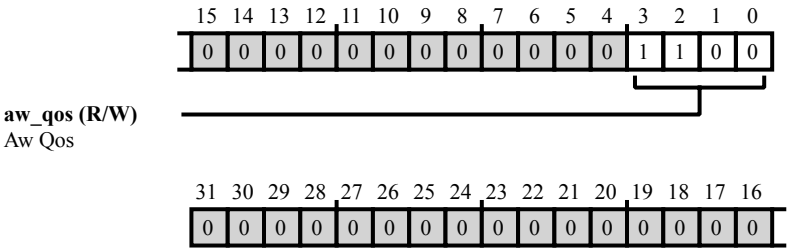


Figure 48-147: SCB0_SPI2RX_WRITE_QOS Register Diagram

Table 48-148: SCB0_SPI2RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SPI2 TX Read Quality of Service Register

The SCB0_SPI2TX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

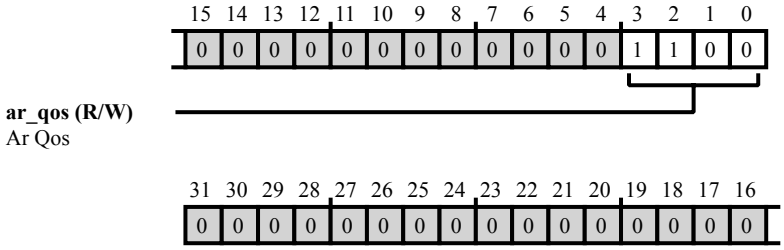


Figure 48-148: SCB0_SPI2TX_READ_QOS Register Diagram

Table 48-149: SCB0_SPI2TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SPI2 TX Write Quality of Service Register

The SCB0_SPI2TX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

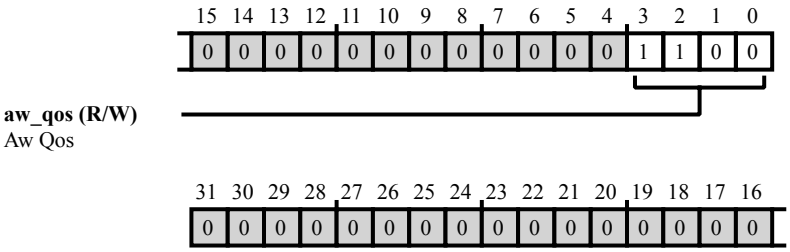


Figure 48-149: SCB0_SPI2TX_WRITE_QOS Register Diagram

Table 48-150: SCB0_SPI2TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SPI3 RX Read Quality of Service Register

The SCB0_SPI3RX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

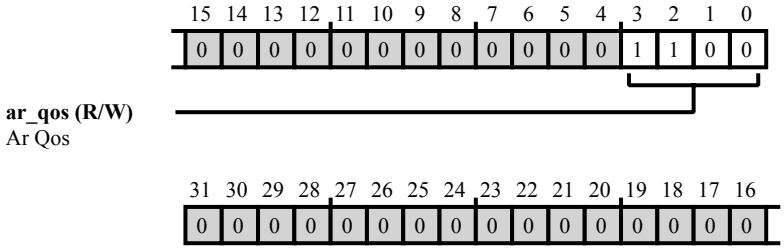


Figure 48-150: SCB0_SPI3RX_READ_QOS Register Diagram

Table 48-151: SCB0_SPI3RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SPI3 RX Write Quality of Service Register

The SCB0_SPI3RX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

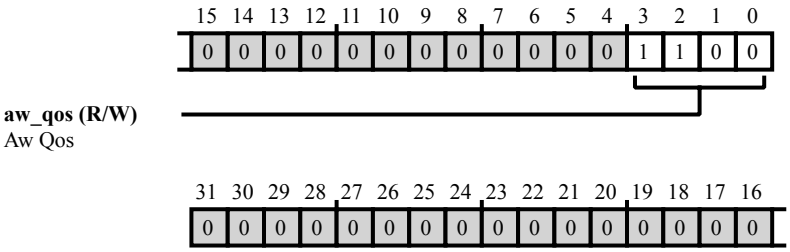


Figure 48-151: SCB0_SPI3RX_WRITE_QOS Register Diagram

Table 48-152: SCB0_SPI3RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

SPI3 TX Read Quality of Service Register

The SCB0_SPI3TX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

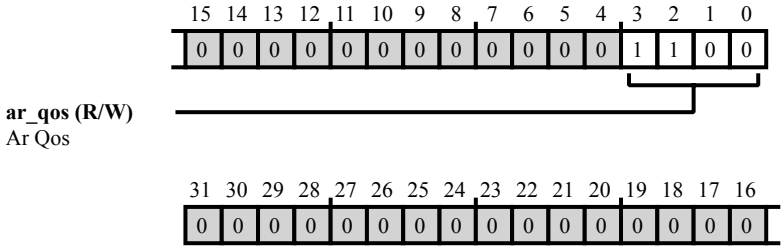


Figure 48-152: SCB0_SPI3TX_READ_QOS Register Diagram

Table 48-153: SCB0_SPI3TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

SPI3 TX Write Quality of Service Register

The SCB0_SPI3TX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

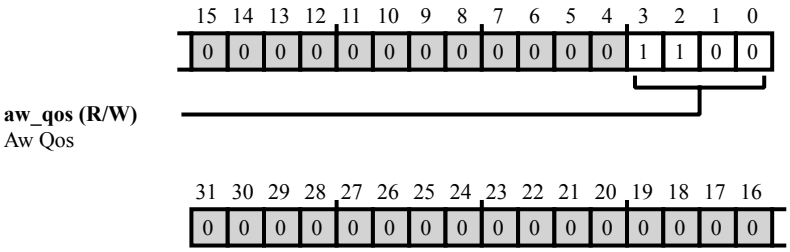


Figure 48-153: SCB0_SPI3TX_WRITE_QOS Register Diagram

Table 48-154: SCB0_SPI3TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

UART0 RX Read Quality of Service Register

The SCB0_UART0_RX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

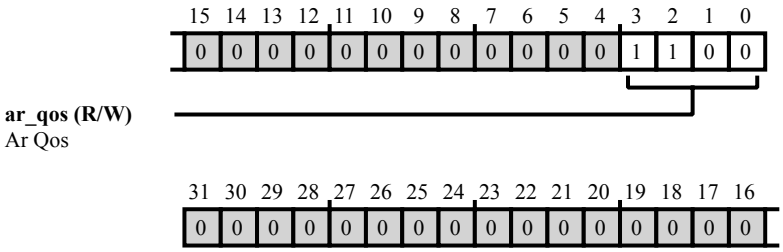


Figure 48-154: SCB0_UART0_RX_READ_QOS Register Diagram

Table 48-155: SCB0_UART0_RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

UART0 RX Write Quality of Service Register

The SCB0_UART0_RX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

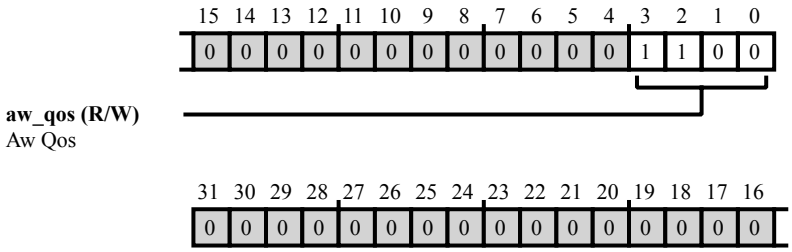


Figure 48-155: SCB0_UART0_RX_WRITE_QOS Register Diagram

Table 48-156: SCB0_UART0_RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

UART0 TX Read Quality of Service Register

The SCB0_UART0_TX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

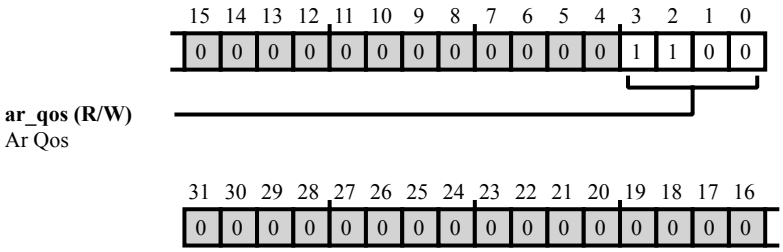


Figure 48-156: SCB0_UART0_TX_READ_QOS Register Diagram

Table 48-157: SCB0_UART0_TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

UART0 TX Write Quality of Service Register

The SCB0_UART0_TX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

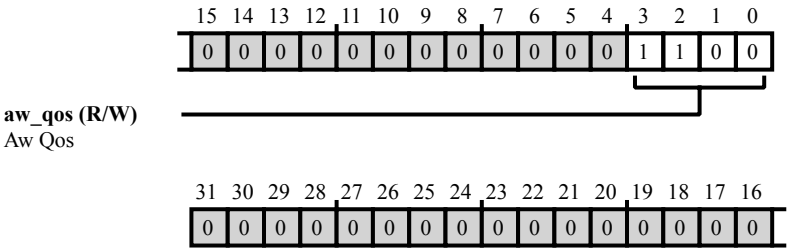


Figure 48-157: SCB0_UART0_TX_WRITE_QOS Register Diagram

Table 48-158: SCB0_UART0_TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

UART1 RX Read Quality of Service Register

The SCB0_UART1_RX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

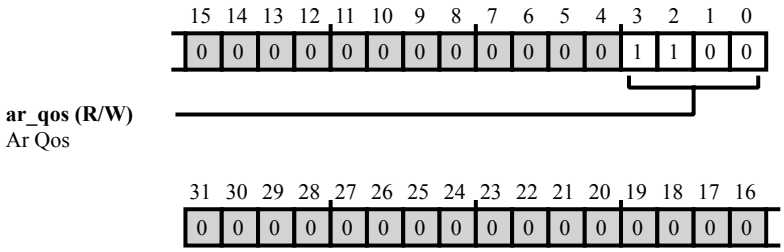


Figure 48-158: SCB0_UART1_RX_READ_QOS Register Diagram

Table 48-159: SCB0_UART1_RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

UART1 RX Write Quality of Service Register

The SCB0_UART1_RX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

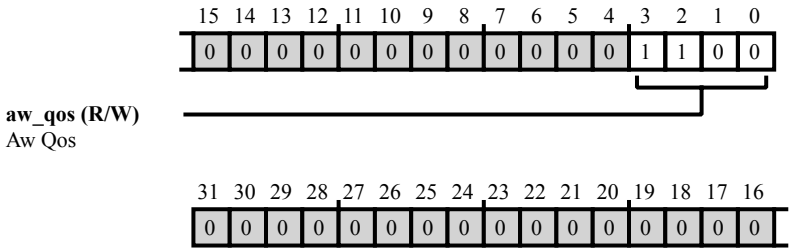


Figure 48-159: SCB0_UART1_RX_WRITE_QOS Register Diagram

Table 48-160: SCB0_UART1_RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

UART1 TX Read Quality of Service Register

The SCB0_UART1_TX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

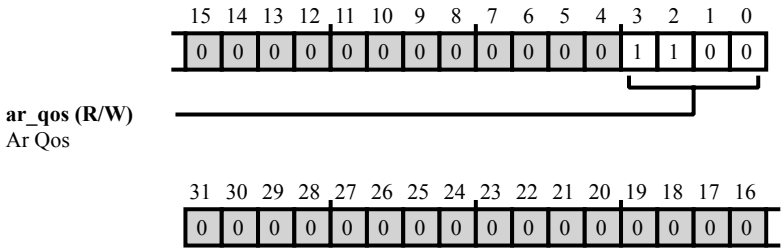


Figure 48-160: SCB0_UART1_TX_READ_QOS Register Diagram

Table 48-161: SCB0_UART1_TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

UART1 TX Write Quality of Service Register

The SCB0_UART1_TX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

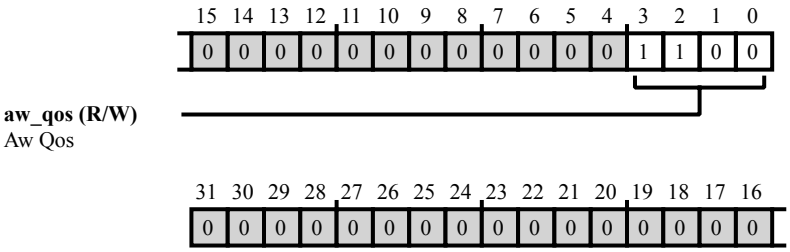


Figure 48-161: SCB0_UART1_TX_WRITE_QOS Register Diagram

Table 48-162: SCB0_UART1_TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

UART2 RX Read Quality of Service Register

The SCB0_UART2_RX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

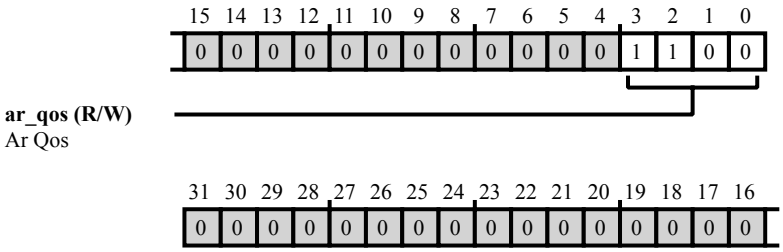


Figure 48-162: SCB0_UART2_RX_READ_QOS Register Diagram

Table 48-163: SCB0_UART2_RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

UART2 RX Write Quality of Service Register

The SCB0_UART2_RX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

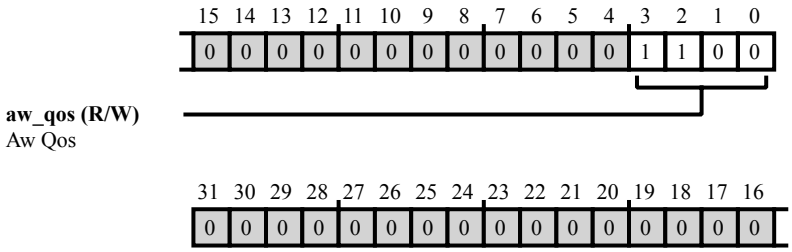


Figure 48-163: SCB0_UART2_RX_WRITE_QOS Register Diagram

Table 48-164: SCB0_UART2_RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

UART2 TX Read Quality of Service Register

The SCB0_UART2_TX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

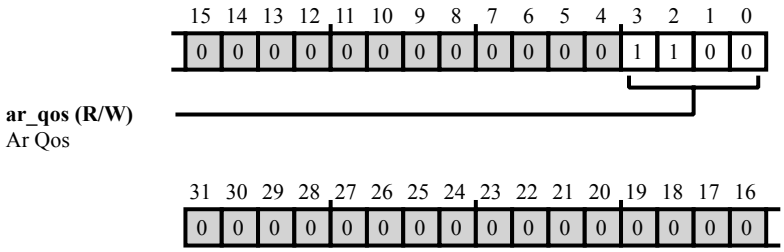


Figure 48-164: SCB0_UART2_TX_READ_QOS Register Diagram

Table 48-165: SCB0_UART2_TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

UART2 TX Write Quality of Service Register

The SCB0_UART2_TX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

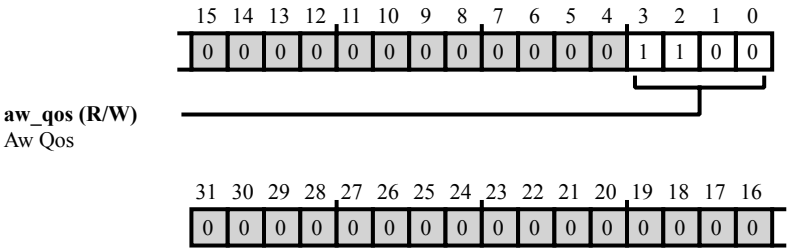


Figure 48-165: SCB0_UART2_TX_WRITE_QOS Register Diagram

Table 48-166: SCB0_UART2_TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

UART3 RX Read Quality of Service Register

The SCB0_UART3_RX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

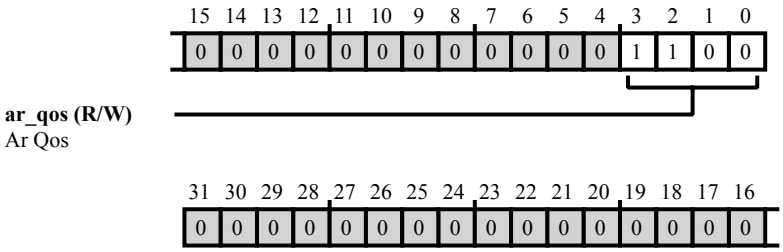


Figure 48-166: SCB0_UART3_RX_READ_QOS Register Diagram

Table 48-167: SCB0_UART3_RX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

UART3 RX Write Quality of Service Register

The SCB0_UART3_RX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

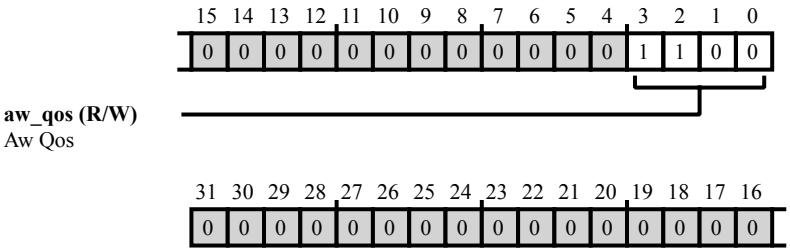


Figure 48-167: SCB0_UART3_RX_WRITE_QOS Register Diagram

Table 48-168: SCB0_UART3_RX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

UART3 TX Read Quality of Service Register

The SCB0_UART3_TX_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

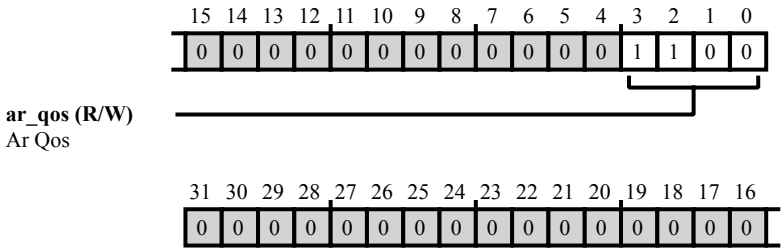


Figure 48-168: SCB0_UART3_TX_READ_QOS Register Diagram

Table 48-169: SCB0_UART3_TX_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

UART3 TX Write Quality of Service Register

The SCB0_UART3_TX_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

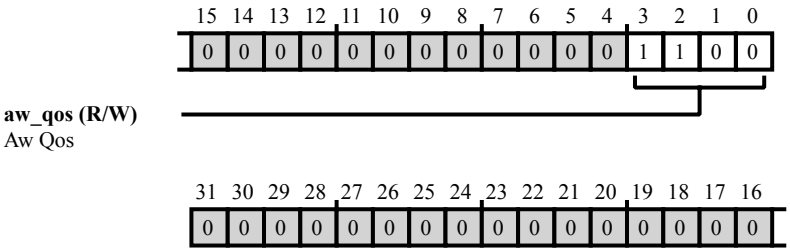


Figure 48-169: SCB0_UART3_TX_WRITE_QOS Register Diagram

Table 48-170: SCB0_UART3_TX_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

USB0 Read Quality of Service Register

The SCB0_USB0_IB_READ_QOS register indicates the read QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting read channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

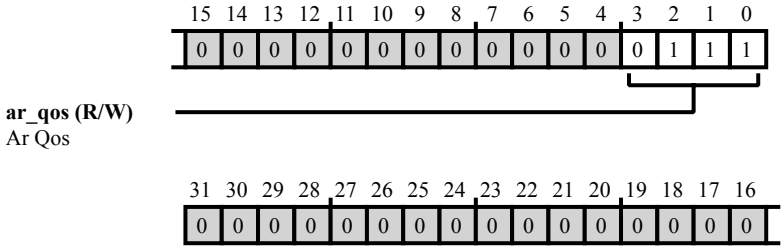


Figure 48-170: SCB0_USB0_IB_READ_QOS Register Diagram

Table 48-171: SCB0_USB0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

USB0 Write Quality of Service Register

The SCB0_USB0_IB_WRITE_QOS register indicates the write QOS or priority value for the indicated bus requester. This value is used by the SCBs at different levels to arbitrate among the ports requesting write channel access. For the mapping of requester IDs to peripherals, see the SCB Bus Requester IDs table in the System Crossbars chapter.

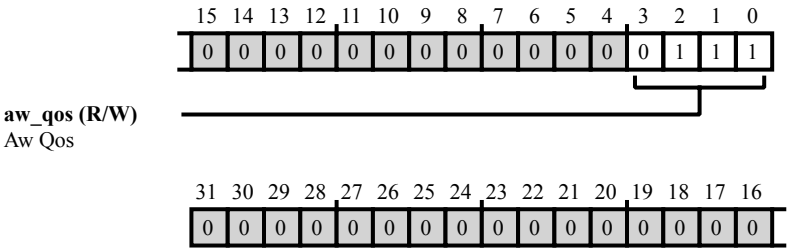


Figure 48-171: SCB0_USB0_IB_WRITE_QOS Register Diagram

Table 48-172: SCB0_USB0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

ADSP-SC59x SCB1 Register Descriptions

(SCB1) contains the following registers.

Table 48-173: ADSP-SC59x SCB1 Register List

Name	Description
SCB1_MST_IB_SYNC_MODE	DMC Fabric (CLK03) Synchronization Mode Register

DMC Fabric (CLK03) Synchronization Mode Register

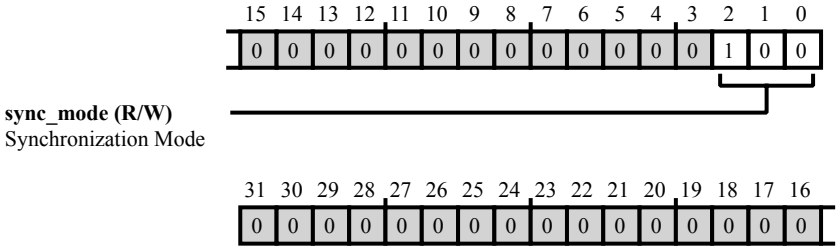


Figure 48-172: SCB1_MST_IB_SYNC_MODE Register Diagram

Table 48-174: SCB1_MST_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	SYNC_MODE	Synchronization Mode. The `SCB1_MST_IB_SYNC_MODE.SYNC_MODE` bit field configures the SYSCLK-DCLK clock domain boundary (CDC) for the DMC fabric.	
		0	Sync 1:1
		1	Sync n:1
		2	Sync 1:n
		3	Sync m:n
		4	Async

ADSP-SC59x SCB3 Register Descriptions

(SCB3) contains the following registers.

Table 48-175: ADSP-SC59x SCB3 Register List

Name	Description
SCB3_APB_CLKO3_IB_SYNC_MODE	DDR MMRG Fabric (CLK03) Synchronization Mode Register
SCB3_APB_CLKO4_IB_SYNC_MODE	DDR MMRG Fabric (CLK04) Synchronization Mode Register
SCB3_APB_CLKO8_IB_SYNC_MODE	DDR MMRG Fabric (CLK08) Synchronization Mode Register

DDR MMRG Fabric (CLK03) Synchronization Mode Register

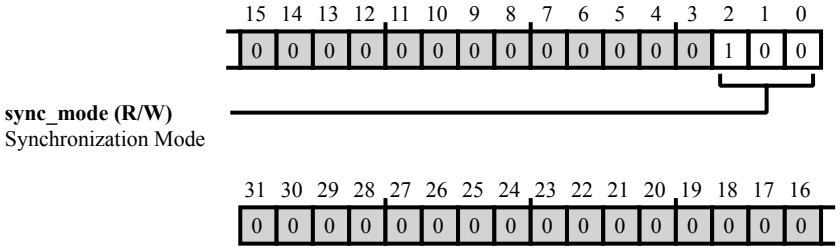


Figure 48-173: SCB3_APB_CLKO3_IB_SYNC_MODE Register Diagram

Table 48-176: SCB3_APB_CLKO3_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	SYNC_MODE	Synchronization Mode. The `SCB3_APB_CLKO3_IB_SYNC_MODE.SYNC_MODE` bit field configures the SYSCLK-DCLK clock domain boundary (CDC) for the MMRG fabric.	
		0	Sync 1:1
		1	Sync n:1
		2	Sync 1:n
		3	Sync m:n
		4	Async

DDR MMRG Fabric (CLK04) Synchronization Mode Register

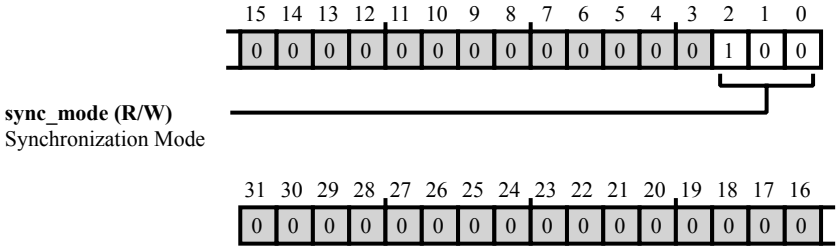


Figure 48-174: SCB3_APB_CLKO4_IB_SYNC_MODE Register Diagram

Table 48-177: SCB3_APB_CLKO4_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	SYNC_MODE	Synchronization Mode. The `SCB3_APB_CLKO4_IB_SYNC_MODE.SYNC_MODE` bit field configures the SYSCLK-DCLK clock domain boundary (CDC) for the MMRG fabric.	
		0	Sync 1:1
		1	Sync n:1
		2	Sync 1:n
		3	Sync m:n
		4	Async

DDR MMRG Fabric (CLK08) Synchronization Mode Register

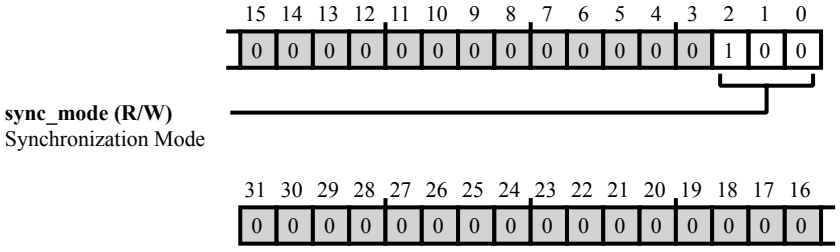


Figure 48-175: SCB3_APB_CLKO8_IB_SYNC_MODE Register Diagram

Table 48-178: SCB3_APB_CLKO8_IB_SYNC_MODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
2:0 (R/W)	SYNC_MODE	Synchronization Mode. The `SCB3_APB_CLKO8_IB_SYNC_MODE.SYNC_MODE` bit field configures the SYSCLK-DCLK clock domain boundary (CDC) for the MMRG fabric.	
		0	Sync 1:1
		1	Sync n:1
		2	Sync 1:n
		3	Sync m:n
		4	Async

ADSP-SC59x SCB4 Register Descriptions

(SCB4) contains the following registers.

Table 48-179: ADSP-SC59x SCB4 Register List

Name	Description
SCB4_FABRIC_ACC_MMR_IB_READ_QOS	Fabric Acc MMR Read Quality of Service Register
SCB4_FABRIC_ACC_MMR_IB_WRITE_QOS	Fabric Acc MMR Write Quality of Service Register
SCB4_FABRIC_S1PORT_IB_READ_QOS	Fabric S1PORT Read Quality of Service Register
SCB4_FABRIC_S1PORT_IB_WRITE_QOS	Fabric S1PORT Write Quality of Service Register
SCB4_FABRIC_S2PORT_IB_READ_QOS	Fabric S2PORT Read Quality of Service Register
SCB4_FABRIC_S2PORT_IB_WRITE_QOS	Fabric S2PORT Write Quality of Service Register
SCB4_FIR_CH0_IB_READ_QOS	FIR Channel 0 Read Quality of Service Register
SCB4_FIR_CH0_IB_WRITE_QOS	FIR Channel 0 Write Quality of Service Register

Table 48-179: ADSP-SC59x SCB4 Register List (Continued)

Name	Description
SCB4_FIR_CH1_IB_READ_QOS	FIR Channel 1 Read Quality of Service Register
SCB4_FIR_CH1_IB_WRITE_QOS	FIR Channel 1 Write Quality of Service Register
SCB4_IIR0_CH0_IB_READ_QOS	IIR0 Channel 0 Read Quality of Service Register
SCB4_IIR0_CH0_IB_WRITE_QOS	IIR0 Channel 0 Write Quality of Service Register
SCB4_IIR0_CH1_IB_READ_QOS	IIR0 Channel 1 Read Quality of Service Register
SCB4_IIR0_CH1_IB_WRITE_QOS	IIR0 Channel 1 Write Quality of Service Register
SCB4_IIR1_CH0_IB_READ_QOS	IIR1 Channel 0 Read Quality of Service Register
SCB4_IIR1_CH0_IB_WRITE_QOS	IIR1 Channel 0 Write Quality of Service Register
SCB4_IIR1_CH1_IB_READ_QOS	IIR1 Channel 1 Read Quality of Service Register
SCB4_IIR1_CH1_IB_WRITE_QOS	IIR1 Channel 1 Write Quality of Service Register
SCB4_IIR2_CH0_IB_READ_QOS	IIR2 Channel 0 Read Quality of Service Register
SCB4_IIR2_CH0_IB_WRITE_QOS	IIR2 Channel 0 Write Quality of Service Register
SCB4_IIR2_CH1_IB_READ_QOS	IIR2 Channel 1 Read Quality of Service Register
SCB4_IIR2_CH1_IB_WRITE_QOS	IIR2 Channel 1 Write Quality of Service Register
SCB4_IIR3_CH0_IB_READ_QOS	IIR3 Channel 0 Read Quality of Service Register
SCB4_IIR3_CH0_IB_WRITE_QOS	IIR3 Channel 0 Write Quality of Service Register
SCB4_IIR3_CH1_IB_READ_QOS	IIR3 Channel 1 Read Quality of Service Register
SCB4_IIR3_CH1_IB_WRITE_QOS	IIR3 Channel 1 Write Quality of Service Register
SCB4_SHARC_DPORT_READ_QOS	DPORT Read Quality of Service Register
SCB4_SHARC_DPORT_WRITE_QOS	DPORT Write Quality of Service Register

Fabric Acc MMR Read Quality of Service Register

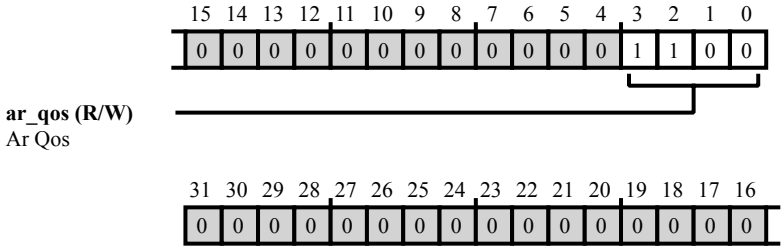


Figure 48-176: SCB4_FABRIC_ACC_MMR_IB_READ_QOS Register Diagram

Table 48-180: SCB4_FABRIC_ACC_MMR_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

Fabric Acc MMR Write Quality of Service Register

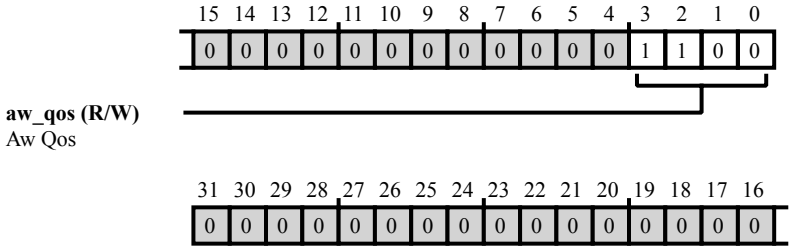


Figure 48-177: SCB4_FABRIC_ACC_MMR_IB_WRITE_QOS Register Diagram

Table 48-181: SCB4_FABRIC_ACC_MMR_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

Fabric S1PORT Read Quality of Service Register

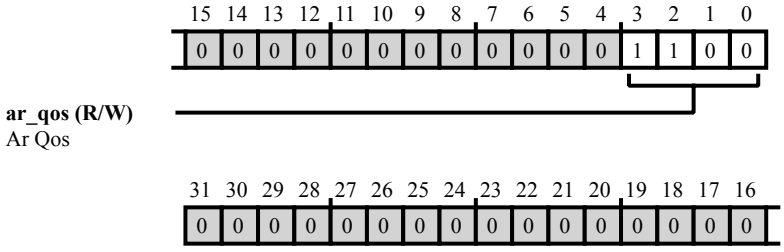


Figure 48-178: SCB4_FABRIC_S1PORT_IB_READ_QOS Register Diagram

Table 48-182: SCB4_FABRIC_S1PORT_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

Fabric S1PORT Write Quality of Service Register

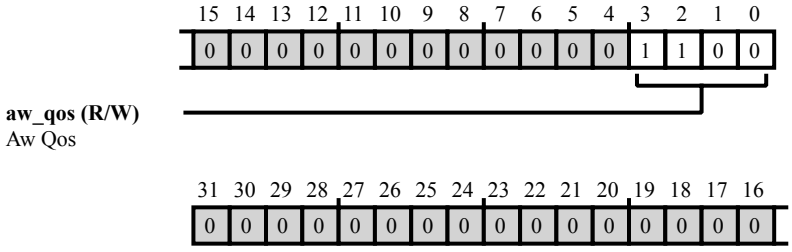


Figure 48-179: SCB4_FABRIC_S1PORT_IB_WRITE_QOS Register Diagram

Table 48-183: SCB4_FABRIC_S1PORT_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

Fabric S2PORT Read Quality of Service Register

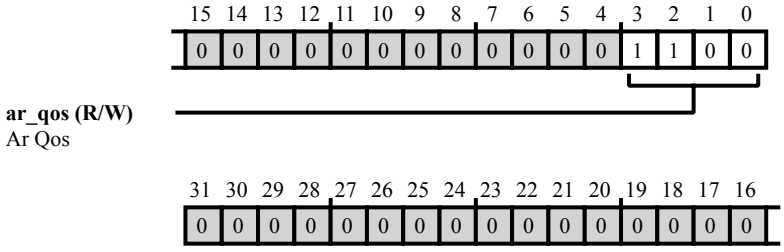


Figure 48-180: SCB4_FABRIC_S2PORT_IB_READ_QOS Register Diagram

Table 48-184: SCB4_FABRIC_S2PORT_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

Fabric S2PORT Write Quality of Service Register

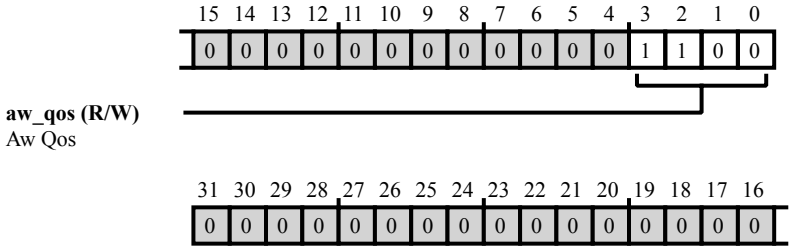


Figure 48-181: SCB4_FABRIC_S2PORT_IB_WRITE_QOS Register Diagram

Table 48-185: SCB4_FABRIC_S2PORT_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

FIR Channel 0 Read Quality of Service Register

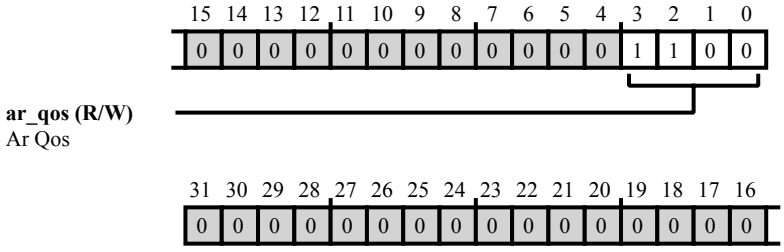


Figure 48-182: SCB4_FIR_CH0_IB_READ_QOS Register Diagram

Table 48-186: SCB4_FIR_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

FIR Channel 0 Write Quality of Service Register

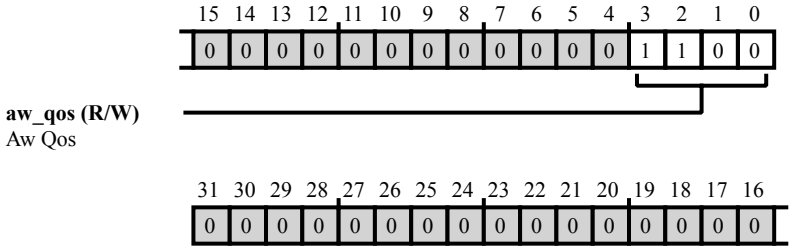


Figure 48-183: SCB4_FIR_CH0_IB_WRITE_QOS Register Diagram

Table 48-187: SCB4_FIR_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

FIR Channel 1 Read Quality of Service Register

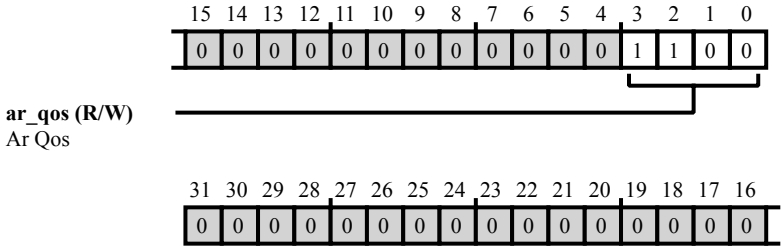


Figure 48-184: SCB4_FIR_CH1_IB_READ_QOS Register Diagram

Table 48-188: SCB4_FIR_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

FIR Channel 1 Write Quality of Service Register

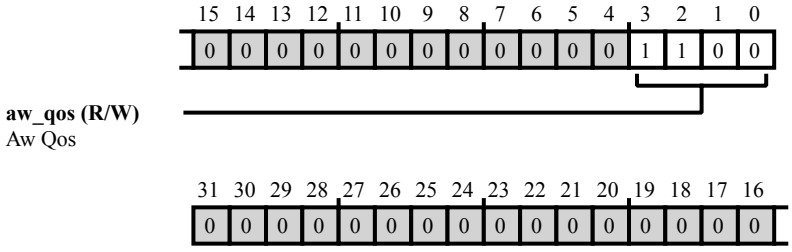


Figure 48-185: SCB4_FIR_CH1_IB_WRITE_QOS Register Diagram

Table 48-189: SCB4_FIR_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

IIR0 Channel 0 Read Quality of Service Register

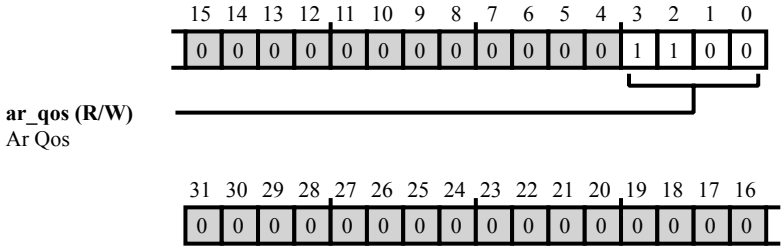


Figure 48-186: SCB4_IIR0_CH0_IB_READ_QOS Register Diagram

Table 48-190: SCB4_IIR0_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

IIR0 Channel 0 Write Quality of Service Register

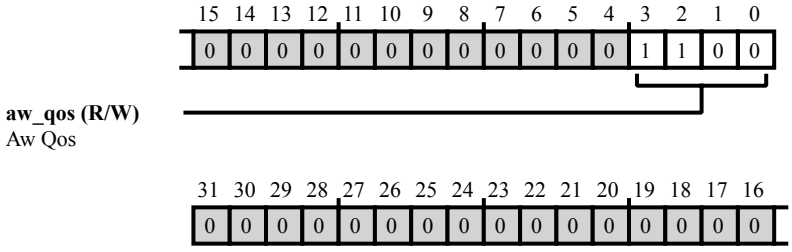


Figure 48-187: SCB4_IIR0_CH0_IB_WRITE_QOS Register Diagram

Table 48-191: SCB4_IIR0_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

IIR0 Channel 1 Read Quality of Service Register

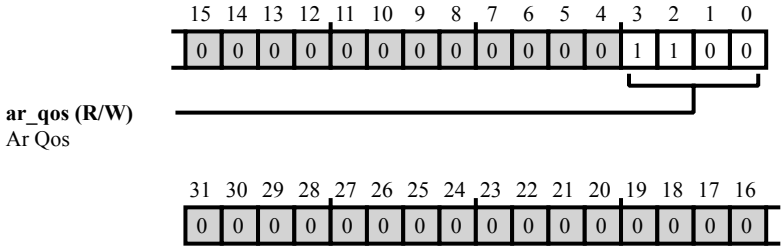


Figure 48-188: SCB4_IIR0_CH1_IB_READ_QOS Register Diagram

Table 48-192: SCB4_IIR0_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

IIR0 Channel 1 Write Quality of Service Register

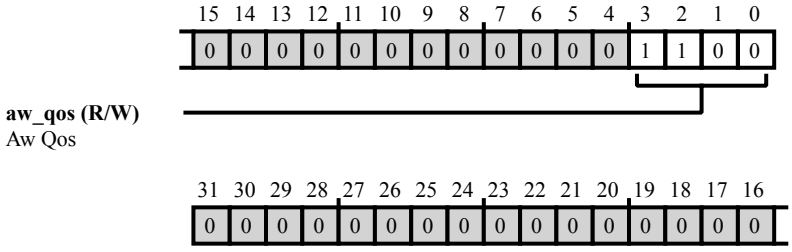


Figure 48-189: SCB4_IIR0_CH1_IB_WRITE_QOS Register Diagram

Table 48-193: SCB4_IIR0_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

IIR1 Channel 0 Read Quality of Service Register

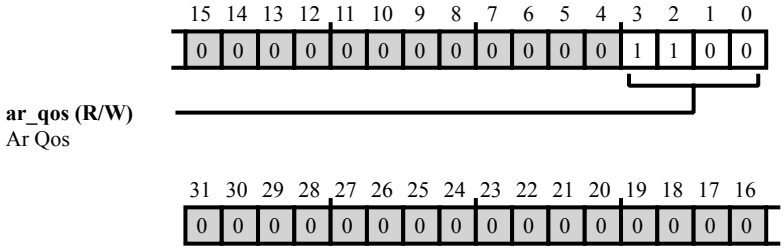


Figure 48-190: SCB4_IIR1_CH0_IB_READ_QOS Register Diagram

Table 48-194: SCB4_IIR1_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

IIR1 Channel 0 Write Quality of Service Register

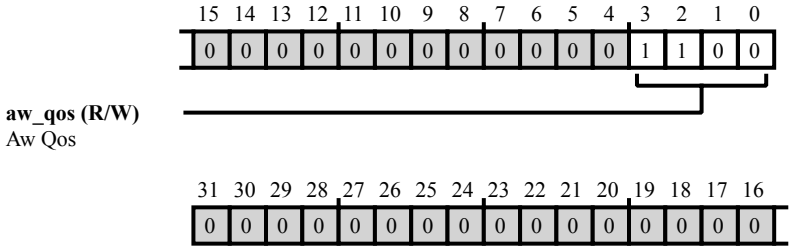


Figure 48-191: SCB4_IIR1_CH0_IB_WRITE_QOS Register Diagram

Table 48-195: SCB4_IIR1_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

IIR1 Channel 1 Read Quality of Service Register

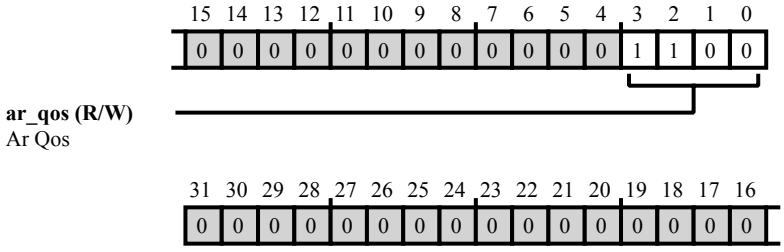


Figure 48-192: SCB4_IIR1_CH1_IB_READ_QOS Register Diagram

Table 48-196: SCB4_IIR1_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

IIR1 Channel 1 Write Quality of Service Register

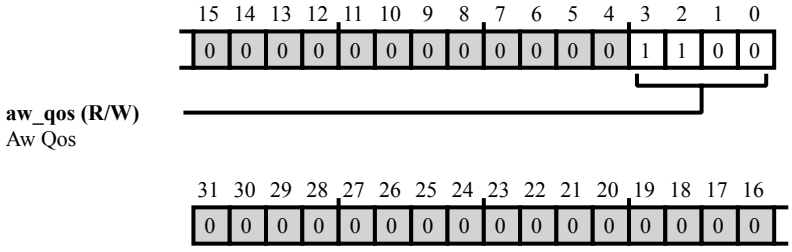


Figure 48-193: SCB4_IIR1_CH1_IB_WRITE_QOS Register Diagram

Table 48-197: SCB4_IIR1_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

IIR2 Channel 0 Read Quality of Service Register

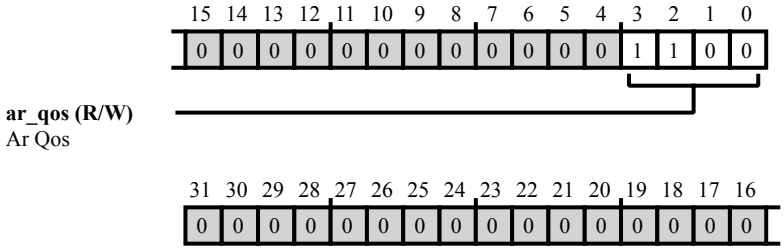


Figure 48-194: SCB4_IIR2_CH0_IB_READ_QOS Register Diagram

Table 48-198: SCB4_IIR2_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

IIR2 Channel 0 Write Quality of Service Register

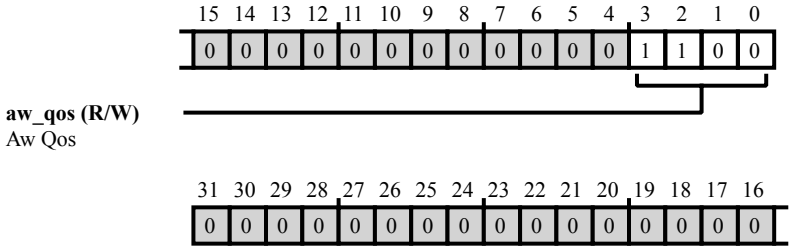


Figure 48-195: SCB4_IIR2_CH0_IB_WRITE_QOS Register Diagram

Table 48-199: SCB4_IIR2_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

IIR2 Channel 1 Read Quality of Service Register

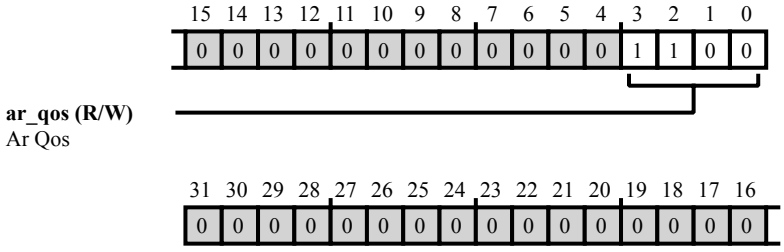


Figure 48-196: SCB4_IIR2_CH1_IB_READ_QOS Register Diagram

Table 48-200: SCB4_IIR2_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

IIR2 Channel 1 Write Quality of Service Register

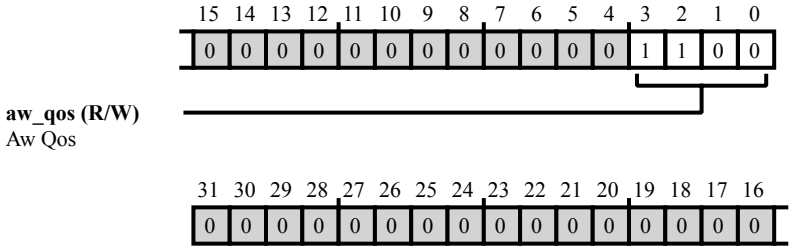


Figure 48-197: SCB4_IIR2_CH1_IB_WRITE_QOS Register Diagram

Table 48-201: SCB4_IIR2_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

IIR3 Channel 0 Read Quality of Service Register

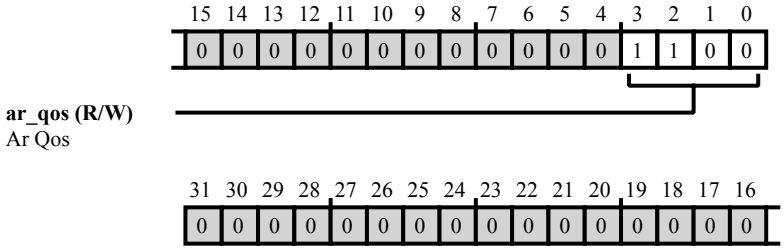


Figure 48-198: SCB4_IIR3_CH0_IB_READ_QOS Register Diagram

Table 48-202: SCB4_IIR3_CH0_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

IIR3 Channel 0 Write Quality of Service Register

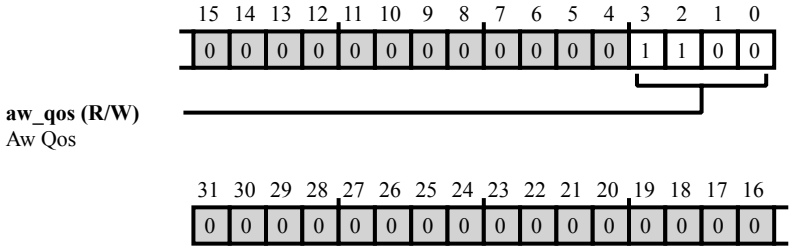


Figure 48-199: SCB4_IIR3_CH0_IB_WRITE_QOS Register Diagram

Table 48-203: SCB4_IIR3_CH0_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

IIR3 Channel 1 Read Quality of Service Register

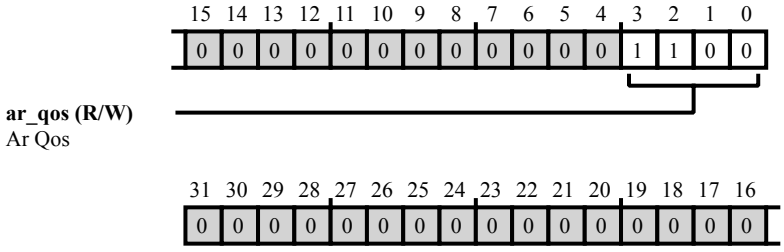


Figure 48-200: SCB4_IIR3_CH1_IB_READ_QOS Register Diagram

Table 48-204: SCB4_IIR3_CH1_IB_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

IIR3 Channel 1 Write Quality of Service Register

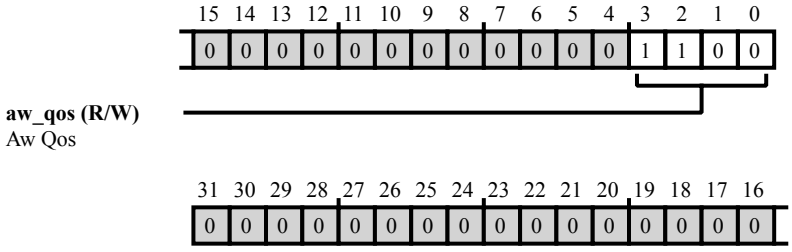


Figure 48-201: SCB4_IIR3_CH1_IB_WRITE_QOS Register Diagram

Table 48-205: SCB4_IIR3_CH1_IB_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

DPORT Read Quality of Service Register

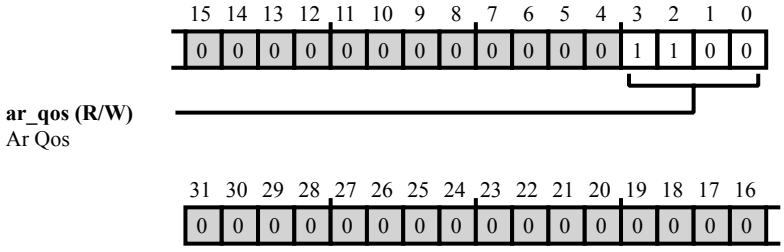


Figure 48-202: SCB4_SHARC_DPORT_READ_QOS Register Diagram

Table 48-206: SCB4_SHARC_DPORT_READ_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AR_QOS	Ar Qos.

DPORT Write Quality of Service Register

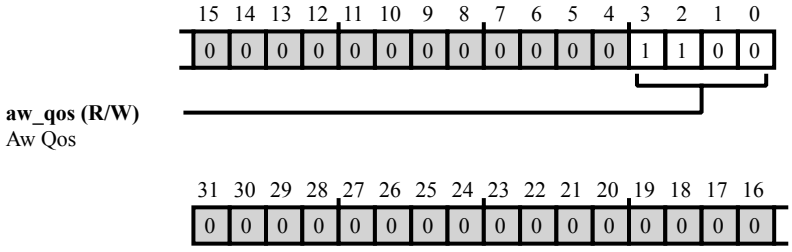


Figure 48-203: SCB4_SHARC_DPORT_WRITE_QOS Register Diagram

Table 48-207: SCB4_SHARC_DPORT_WRITE_QOS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	AW_QOS	Aw Qos.

ADSP-SC59x SCB5 Register Descriptions

System Crossbars (SCB5) contains the following registers.

Table 48-208: ADSP-SC59x SCB5 Register List

Name	Description
SCB5_SPI2_OSPI_REMAP	SPI2/OSPI Memory Map Address Remap Register

SPI2/OSPI Memory Map Address Remap Register

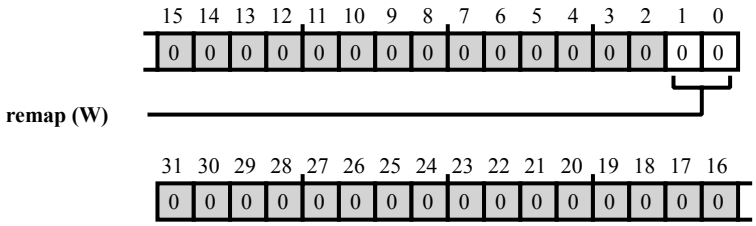


Figure 48-204: SCB5_SPI2_OSPI_REMAP Register Diagram

Table 48-209: SCB5_SPI2_OSPI_REMAP Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1:0 (RX/W)	REMAP	.	
		0	Selects map for SPI2 only 0x6000_0000 - 0x7FFF_FFFF - SPIF
		1	Selects map for OSPI only 0x6000_0000 - 0x7FFF_FFFF - OSPI
		2	Selects map for SPI2/OSPI split sharing 0x6000_0000 - 602F_FFFF - SPIF 0x6030_0000 - 0x7FFF_FFFF - OSPI

ADSP-SC59x SCB6 Register Descriptions

(SCB6) contains the following registers.

Table 48-210: ADSP-SC59x SCB6 Register List

Name	Description
SCB6_A55_M0_IB_FN_MOD	A55 M0 Ib.fn Mod

A55 M0 Ib.fn Mod

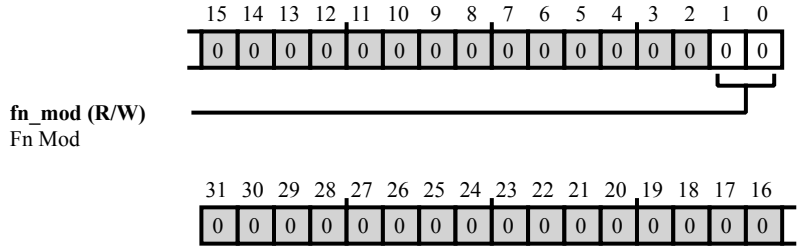


Figure 48-205: SCB6_A55_M0_IB_FN_MOD Register Diagram

Table 48-211: SCB6_A55_M0_IB_FN_MOD Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1:0 (R/W)	FN_MOD	Fn Mod.

49 System Watchpoint Unit (SWU)

The system watchpoint unit (SWU) is a single module used for transaction monitoring. The SWU is attached to each system completer through the system crossbar interface and provides ports for all address channel signals for the system crossbar. The SWU does not have ports for the read/write data channel signals or the low-power interface signals.

Each SWU contains four match groups of registers with associated hardware. These four SWU match groups operate independently, but share common event (interrupt and trigger) outputs. Each match group can monitor either the write or read address channel and can operate in either watchpoint mode or bandwidth mode.

In this chapter, the naming convention for registers omit the numeric reference to any of the SWUs or it's groups. For example, SWU_GCTL represents registers SWU1_GCTL, SWU2_GCTL, and so on. Similarly, SWU_CUR[n] represents SWU1_CUR0/CUR1/CUR2/CUR3, SWU2_CUR0/CUR1/CUR2/CUR3 and so on.

SWU Features

The system watchpoint unit has the following features.

- Four independent match groups for each SWU
- Each match group can operate in either bandwidth mode or watchpoint mode

SWU Functional Description

This section describes the function of the SWU match block, interface block, and MMR block.

NOTE: In the *SWU Interrupt List* table, there are three SWUs, SWU1 (CL2_0), SWU3 (CL2_1) and SWU5 (CL2_2) for L2 Memory Core access and two SWUs, SWU2 (DL2_0) and SWU4 (DL2_1) for L2 Memory DMA access. CL2_2, DL2_0 and DL2_1 have access to the entire L2 range. CL2_0 has access to first 1MB and CL2_1 has access to second 1MB of L2.

ADSP-SC59x SWU Register List

The System Watchpoint Unit (SWU) provides debug and development support through flexible transaction level and bandwidth monitoring and associated event triggering. The SWU can generate events based on monitoring transactions at the system slaves through watchpoint-match groups. The SWU also provides watchpoint event status

reporting, a global lock, and processor reset capability. A set of registers governs SWU operations. For more information on SWU functionality, see the SWU register descriptions.

Table 49-1: ADSP-SC59x SWU Register List

Name	Description
SWU_CNT[n]	Count Register n
SWU_CTL[n]	Control Register n
SWU_CUR[n]	Current Register n
SWU_GCTL	Global Control Register
SWU_GSTAT	Global Status Register
SWU_HIST[n]	Bandwidth History Register n
SWU_ID[n]	ID Register n
SWU_LA[n]	Lower Address Register n
SWU_TARG[n]	Target Register n
SWU_UA[n]	Upper Address Register n

ADSP-SC59x SWU Interrupt List

Table 49-2: ADSP-SC59x SWU Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
314	SWU1_EVT	SWU1 Event L2 Memory Core Port 0	None	
315	SWU2_EVT	SWU2 Event L2 Memory DMA Port 0	None	
316	SWU7_EVT	SWU7 Event SHARC0 Slave Port 1	None	
317	SWU8_EVT	SWU8 Event SHARC0 Slave Port 2	None	
318	SWU9_EVT	SWU9 Event SHARC1 Slave Port 1	None	
319	SWU10_EVT	SWU10 Event SHARC1 Slave Port 2	None	
320	SWU11_EVT	SWU11 Event SMMR	None	
321	SWU12_EVT	SWU12 Event SPI2/OSPI	None	
322	SWU13_EVT	SWU13 Event DMC0	None	
323	SWU3_EVT	SWU3 Event CL2_1	None	
324	SWU4_EVT	SWU4 Event DL2_1	None	
325	SWU5_EVT	SWU5 Event CL2_2	None	

ADSP-SC59x SWU Trigger List

Table 49-3: ADSP-SC59x SWU Trigger List Generators

Trigger ID	Name	Description	Sensitivity
106	SWU1_EVT	SWU1 Event	None
107	SWU2_EVT	SWU2 Event	None
108	SWU7_EVT	SWU7 Event	None
109	SWU8_EVT	SWU8 Event	None
110	SWU9_EVT	SWU9 Event	None
111	SWU10_EVT	SWU10 Event	None
112	SWU11_EVT	SWU11 Event	None
113	SWU12_EVT	SWU12 Event	None
114	SWU13_EVT	SWU13 Event	None
115	SWU3_EVT	SWU3 Event	None
116	SWU4_EVT	SWU4 Event	None
117	SWU5_EVT	SWU5 Event	None
119	SWU1_DBG	SWU1 Debug	Edge
120	SWU2_DBG	SWU2 Debug	Edge
121	SWU7_DBG	SWU7 Debug	Edge
122	SWU8_DBG	SWU8 Debug	Edge
123	SWU9_DBG	SWU9 Debug	Edge
124	SWU10_DBG	SWU10 Debug	Edge
125	SWU11_DBG	SWU11 Debug	Edge
126	SWU12_DBG	SWU12 Debug	Edge
127	SWU13_DBG	SWU13 Debug	Edge
128	SWU3_DBG	SWU3 Debug	Edge
129	SWU4_DBG	SWU4 Debug	Edge
130	SWU5_DBG	SWU5 Debug	Edge

Table 49-4: ADSP-SC59x SWU Trigger List Receivers

Trigger ID	Name	Description	Sensitivity
113	SWU1_EN	SWU1 Enable	Pulse
114	SWU2_EN	SWU2 Enable	Pulse
115	SWU7_EN	SWU7 Enable	Pulse

Table 49-4: ADSP-SC59x SWU Trigger List Receivers (Continued)

Trigger ID	Name	Description	Sensitivity
116	SWU8_EN	SWU8 Enable	Pulse
117	SWU9_EN	SWU9 Enable	Pulse
118	SWU10_EN	SWU10 Enable	Pulse
119	SWU11_EN	SWU11 Enable	Pulse
120	SWU12_EN	SWU12 Enable	Pulse
121	SWU13_EN	SWU13 Enable	Pulse
122	SWU3_EN	SWU3 Enable Event CL2_1	Pulse
123	SWU4_EN	SWU4 Enable Event DL2_1	Pulse
124	SWU5_EN	SWU5 Enable Event CL2_2	Pulse

SWU Definitions

The following definitions are helpful when using the SWU module.

Watchpoint Mode

Mode in which transactions are recognized on an exact match. Actions can be configured to be taken after a specified number of matches have occurred.

Bandwidth Mode

Mode in which transactions are recognized and counted inside sampling window.

SWU Architectural Concepts

The information in this section provides basic module design concepts.

SWU-to-SCB Interface

The SWU system crossbar interface block latches all transactions on the system crossbar read and write address channels when the SWU_GCTL.EN register enable bit is set.

SWU Block Diagram

The ***System Watchpoint Unit Top-Level Block Diagram*** figure shows the SWU block diagram.

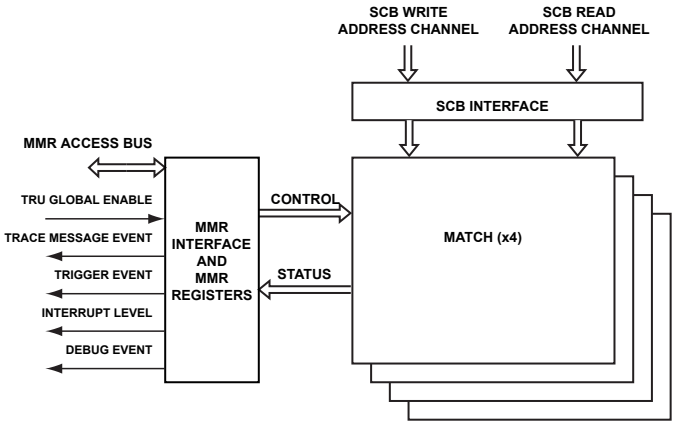


Figure 49-1: System Watchpoint Unit Top-Level Block Diagram

SCB Interface Block

The SWU system crossbar (SCB) latches all transactions on the SCB read and write address channels when the `SWU_GCTL.EN` bit is set.

MMR Interface Block

The SWU MMR block contains the peripheral bus interface and the SWU MMR registers. It also merges all interrupt requests and events from each match block into common outputs.

SWU Operating Modes

There are two operating modes supported by the SWU: bandwidth mode and watchpoint mode.

Bandwidth Mode

In bandwidth mode, the SWU module counts transactions which match the properties specified in the `SWU_CTL[n]` register during a sampling window determined by the respective `SWU_CNT[n]` register. At the end of the sampling window, the SWU stores results in the `SWU_HIST[n]` register. If the sampled bandwidth falls outside a programmed range, then the programmed action occurs.

Watchpoint Mode

In watchpoint mode, if the `SWU_CTL[n].CNTEN` bit is set, the SWU module decrements the `SWU_CUR[n]` register for each match, until it equals zero, at which point any programmed actions occur. The `SWU_CUR[n]` register is then reloaded from the `SWU_CNT[n]` register (if the `SWU_CTL[n].CNTRPTEN` bit is set), and the cycle repeats. If the `SWU_CTL[n].CNTRPTEN` bit is not set, any programmed actions happen on every match.

Match Block

There are four match blocks for each SWU. Each SWU match block can monitor either the read or write address channel, selected by the `SWU_CTL[n].DIR` bit. The SWU match block can operate in either watchpoint or bandwidth mode, as selected by the `SWU_CTL[n].BWEN` bit.

Scaling

Scaling allows the SWU to count more transactions by scaling the number of transactions and the number of clock cycles in bandwidth window (CNTn register) by 10,100 or 1000. This functionality is applicable only in bandwidth mode (SWU_CTL[n].BWEN==1).

Consider a case where the SWU_TARG[n].BWMAX bit field is programmed to 2. In the absence of any scaling, bandwidth overflow occurs when the SWU_CUR[n].CURBW bit field value > the SWU_TARG[n].BWMAX bit field value (2). With scaling set to 1:100 and after 275 transactions, the SWU_CUR[n].CURBW value is still 2 and equal to SWU_TARG[n].BWMAX. This event triggers a bandwidth overflow as the actual number of transactions is greater than 200 (2 × 100). The code can be rerun with a smaller scaling selected to analyze the cause for the overflow.

The counter increments after every group of N transactions with scaling enabled, where N is either 10,100 or 1000. (For N = 10, 0–9 transactions == 0 scaled transaction, 10–19 transactions == 1 scaled transaction, 20–29 transactions == 2 scaled transactions, and so on).

Fractional counts with scaling enabled are discarded and not rolled over from one bandwidth window to the next. For example, consider a case where scaling by 1000 is configured and the first window has 1200 transactions. The second window has 2800 transactions. The bandwidth for the first window is read as 1 and the second as 2. The 200 transactions from the first window do not get carried forward to the next window.

Do not use scaling by 10 with the SWU_CTL[n].BLENINC bit enabled if any of the requesters accessing the completer can launch a transaction of burst length 16.

SWU Event Control

The SWU can generate the following events when a match occurs and when the event is enabled by configuring the proper bits in the control register.

1. Trace Message
2. Trigger
3. Interrupt request
4. Debug

SWU Interrupts

All interrupt requests and events from each match block are merged into common outputs.

SWU Status and Errors

SWU status and errors are reported in the SWU_GSTAT register. The SWU records an address error when a write or read attempt is made to the MMR address space of the SWU and the register does not exist. This error is the only one the SWU records. The register contains bits that perform the following functions.

- Indicate whether a particular match group sampled a transaction that is below a minimum target or above a maximum target in bandwidth mode.
- Indicate whether a watchpoint match occurred for each match group.
- Indicate whether an interrupt request was triggered due to a match event from one of the match groups.

Triggers

The SWU can be either a trigger generator or a trigger receiver depending on the trigger routing unit (TRU) configuration. As a trigger generator, programs must set the `SWU_CTL[n].TRGEN` bit so that when a match condition is met, a trigger event is generated. Each SWU in the system can also be a trigger receiver when mapped as one in the TRU.

When the SWU is a receiver, a trigger event activates the SWU by automatically setting the `SWU_GCTL.EN` bit. Since the SWU can be automatically enabled through a trigger event, programs must pre-configure the SWU before enabling the TRU. Furthermore, although a trigger event can enable the SWU as a generator, to disable the SWU, programs must manually clear the `SWU_GCTL.EN` bit.

SWU Programming Model

Program the appropriate registers to use the SWU. Each control register configures aspects such as:

- The direction of monitoring (reads or writes)
- Whether SWU uses bandwidth mode or watchpoint mode
- The setup of events that are triggered when a condition is met while monitoring using the SWU

Configure supplemental registers such as the lower (`SWU_LA[n]`) and upper (`SWU_UA[n]`) address boundaries before enabling the SWU.

Once the SWU has been enabled and the monitoring conditions are met, events are generated when configured.

The global status register (`SWU_GSTAT`) can be read to observe the status of the units.

The ***SWU Logical Flow*** diagram shows the logical program flow of the SWU.

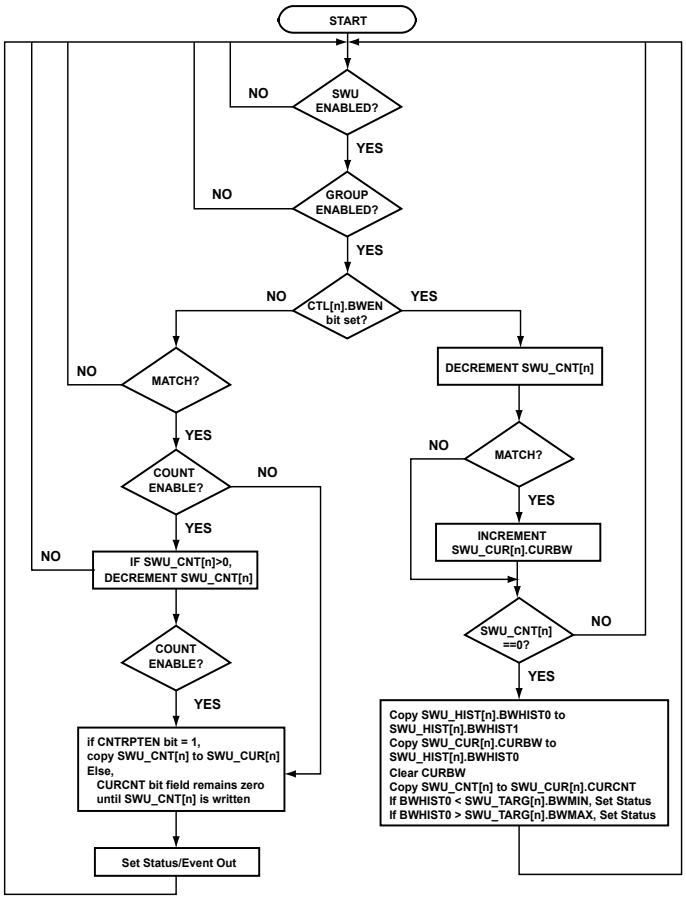


Figure 49-2: SWU Logical Flow

SWU Mode Configuration

The following sections show the steps for configuring SWU bandwidth mode and watchpoint mode.

Configuring the SWU for Bandwidth Mode

In bandwidth mode, the SWU counts transactions which match during a sampling window. At the end of the sampling window, the SWU stores the results. An action can be taken if the sampled bandwidth goes above or falls below a programmed range.

1. Configure the `SWU_CTL[n].DIR` bit to test the match on writes or reads.
2. Configure the `SWU_CTL[n].ACMPM` bits to address comparisons, exact match, matches inside a range or matches outside a range.
3. If ID comparison is desired, set the `SWU_CTL[n].IDCMPEN` bit.
4. Set the `SWU_CTL[n].BLENINC` bit to increment by burst length or clear it to increment by 1.
5. Configure the `SWU_CTL[n].MAXACT` and `SWU_CTL[n].MINACT` bits to enable actions taken when the bandwidth goes above the maximum, or falls below the minimum, respectively.

6. Set the SWU_CTL[n].BWEN=1 to enable bandwidth mode.
7. Program the lower address register, SWU_LA[n], and upper address register, SWU_UA[n], to define the memory range for comparison.
8. If ID comparison is enabled, program the ID register, SWU_ID[n].
9. Program the count register, SWU_CNT[n], with the number of clock cycles for which the SWU counts the number of matches.
10. If the SWU is set to respond when the bandwidth measurement underflows or overflows, program the min and max values into the SWU_TARG[n] register.
11. Enable the SWU

The SWU counts the number of matches in a pre-defined number of clock cycles as programmed. As an option, it can define lower and upper limits. If the matches fall outside the limits, an action can be taken.

Configuring the SWU for Watchpoint Mode

In watchpoint mode, the SWU can trigger a programmed action after every match or after a number of matches. This sequence can be automatically reset.

1. Set the SWU_CTL[n].DIR bit to test the match on writes or reads.
2. Configure the SWU_CTL[n].ACMPM bits for address comparisons, exact match, matches inside a range or matches outside a range.
3. If ID comparison is desired, set the SWU_CTL[n].IDCMPEN.
4. Set the SWU_CTL[n].CNTEN bit to enable the events to be triggered when the count decrements to zero.
5. If needed, set the SWU_CTL[n].CNTRPTEN bit to automatically reload the counter after it has decremented to zero to start another match sequence.
6. Clear the SWU_CTL[n].BWEN = 0 to configure watchpoint mode.
7. Configure the lower address register, SWU_LA[n], and upper address register, SWU_UA[n], to define the memory range for comparison.
8. If ID comparison is enabled, configure the ID register, SWU_ID[n].
9. Configure the count register, SWU_CNT[n], to determine how many matches occur before the watchpoint group responds.
10. Enable the SWU.

The SWU detects and counts down the number of match occurrences. When the counter expires, an action is taken.

ADSP-SC59x SWU Register Descriptions

System Watchpoint Unit (SWU) contains the following registers.

Table 49-5: ADSP-SC59x SWU Register List

Name	Description
SWU_CNT[n]	Count Register n
SWU_CTL[n]	Control Register n
SWU_CUR[n]	Current Register n
SWU_GCTL	Global Control Register
SWU_GSTAT	Global Status Register
SWU_HIST[n]	Bandwidth History Register n
SWU_ID[n]	ID Register n
SWU_LA[n]	Lower Address Register n
SWU_TARG[n]	Target Register n
SWU_UA[n]	Upper Address Register n

Count Register n

The SWU count registers (SWU_CNT[n]) contain a 16-bit count field (SWU_CNT[n].COUNT) whose usage differs depending on the mode of the watchpoint group. In bandwidth mode, the SWU_CNT[n].COUNT field value defines the number of clock cycles in a bandwidth period. In watchpoint mode, when the cycle count is enabled, the SWU_CNT[n].COUNT field value determines how many matches occur before the watchpoint group takes action.

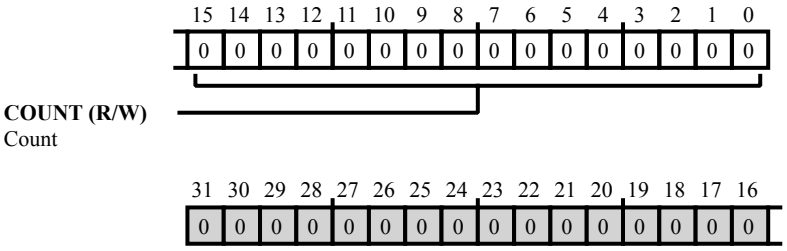


Figure 49-3: SWU_CNT[n] Register Diagram

Table 49-6: SWU_CNT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	COUNT	Count. The SWU_CNT[n].COUNT field value defines the number of clock cycles in a bandwidth period. In watchpoint mode, when the cycle count is enabled, the SWU_CNT[n].COUNT field value determines how many matches occur before the watchpoint group takes action.

Control Register n

The SWU control registers (SWU_CTL[n]) contain watchpoint attribute controls for all four watchpoint groups. These controls include enabling watchpoints, selecting the transaction direction for match, selecting address comparison mode, enabling ID comparison, enabling security comparison, enabling cycle count, enabling count repeat, enabling debug events, enabling interrupts, enabling triggers, enabling trace messages, enabling bandwidth mode, selecting the burst length increment, and enabling bandwidth underflow and overflow detection.

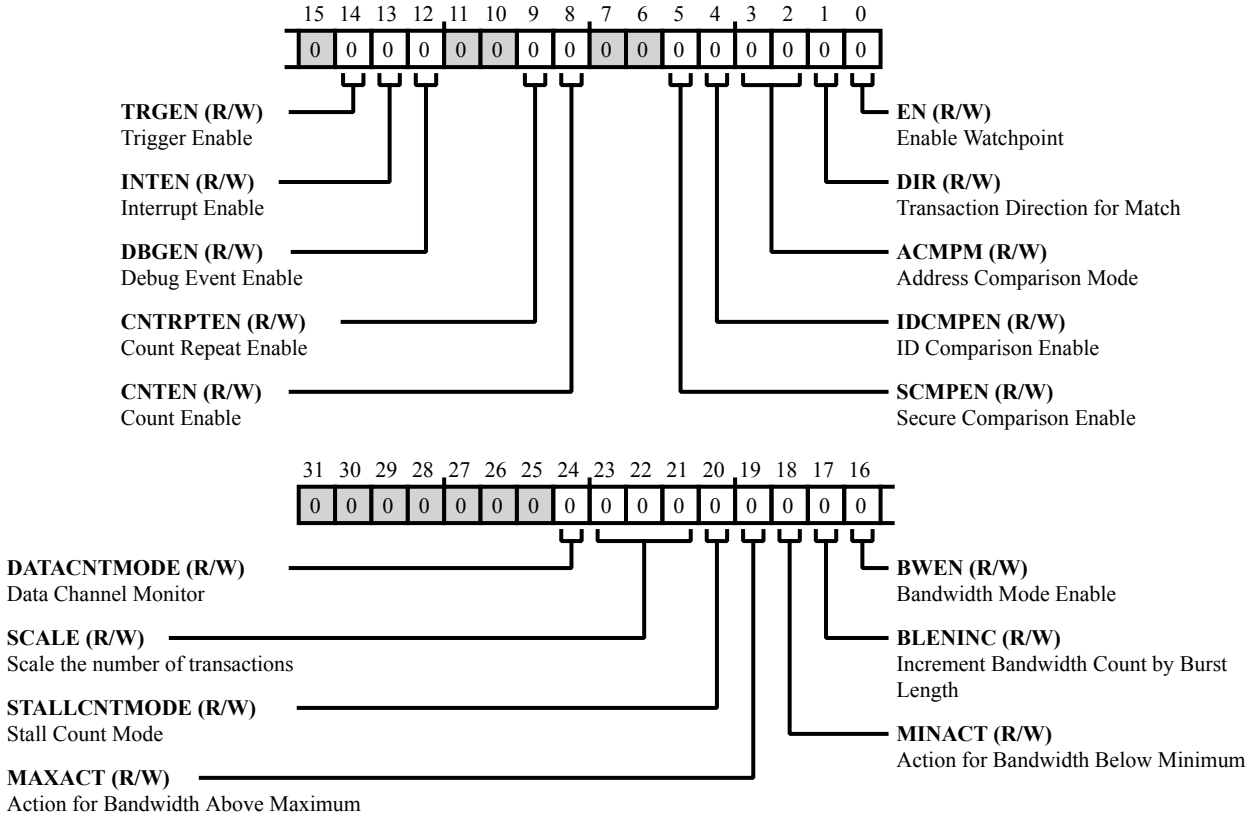


Figure 49-4: SWU_CTL[n] Register Diagram

Table 49-7: SWU_CTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
24 (R/W)	DATACNTMODE	Data Channel Monitor. The SWU_CTL[n].DATACNTMODE bit determines whether an address channel or a data channel is monitored. Note that in data channel only ID,READY and VALID signals are monitored and hence other comparisons (Address, Lock, Secure) will be ignored even if enabled. SWU_CTL[n].STALLCNTMODE and SWU_CTL[n].DIR can be used in conjunction with this bit	
		0	Monitor address channel
		1	Monitor data channel
23:21 (R/W)	SCALE	Scale the number of transactions. The SWU_CTL[n].SCALE bit field allows a program to count more transactions by scaling the number of transactions and also the number of clock cycles in the bandwidth window (SWU_CNT[n] register) by 10,100 or 1000. This is applicable only in bandwidth mode (SWU_CTL[n].BWEN==1).	
		0	No scaling
		1	1:10
		2	1:100
		3	Reserved
		4	1:1000
		5-7	Reserved
20 (R/W)	STALLCNTMODE	Stall Count Mode. The SWU_CTL[n].STALLCNTMODE bit determines whether the number of stalls are counted or whether the number of transactions are counted. This feature is only valid in bandwidth mode.	
		0	Count number of transactions
		1	Count number of stalls
19 (R/W)	MAXACT	Action for Bandwidth Above Maximum. Each SWU_CTL[n].MAXACT bit determines whether a watchpoint group takes action on bandwidth overflow. This feature is only valid in bandwidth mode.	
		0	No Action
		1	Take Action
18 (R/W)	MINACT	Action for Bandwidth Below Minimum. Each SWU_CTL[n].MINACT bit determines whether a watchpoint group takes action on bandwidth underflow. This feature is only valid in bandwidth mode.	
		0	No Action
		1	Take Action

Table 49-7: SWU_CTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
17 (R/W)	BLENINC	Increment Bandwidth Count by Burst Length. Each SWU_CTL[n].BLENINC bit controls how a watchpoint group's bandwidth count is incremented in the SWU_CUR[n] register's SWU_CUR[n].CURBW field. If the SWU_CTL[n].BLENINC bit is cleared (= 0), the SWU increments the bandwidth count by 1 for each matching transaction. If the SWU_CTL[n].BLENINC bit is set (=1), the SWU increments the bandwidth count by the burst length of the transaction for each matching transaction. This feature is only valid for bandwidth mode (SWU_CTL[n].BWEN bit == 1). Note that if the address range match is enabled (SWU_CTL[n].ACMPM bits) and if any address of a burst falls within the address range, the SWU_CUR[n].CURBW field is incremented by the burst length even if some of the burst address fall outside of the range. Also, note that the burst size of the transaction is not included in the increment, only the burst length of the transaction. This increment operation provides an approximate (not exact) number of bus cycles consumed during the bandwidth.	
		0	Increment by 1
		1	Burst Length Increment for Bandwidth Count
16 (R/W)	BWEN	Bandwidth Mode Enable. Each SWU_CTL[n].BWEN bit controls whether a watchpoint group operates in watchpoint mode or bandwidth mode. In watchpoint mode, the SWU_CTL[n].CNTEN and (optionally) SWU_CTL[n].CNTRPTEN registers control usage of the cycle count for watchpoint group operations. In bandwidth mode, the SWU_CTL[n].BLENINC, SWU_TARG[n], and SWU_HIST[n] registers control usage of watchpoint matches for watchpoint group operations.	
		0	Watchpoint Mode
		1	Bandwidth Mode
14 (R/W)	TRGEN	Trigger Enable. Each SWU_CTL[n].TRGEN bit controls whether a match for a watchpoint group generates a trigger event. This feature is valid in both bandwidth and watchpoint modes.	
		0	Disable
		1	Enable
13 (R/W)	INTEN	Interrupt Enable. Each SWU_CTL[n].INTEN bit controls whether a match for a watchpoint group generates an interrupt. This feature is valid in both bandwidth and watchpoint modes.	
		0	Disable
		1	Enable

Table 49-7: SWU_CTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
12 (R/W)	DBGEN	Debug Event Enable. Each SWU_CTL[n].DBGEN bit controls debug event comparison for a watchpoint group, permitting matches based on debug status.	
		0	Disable
		1	Enable
9 (R/W)	CNTRPTEN	Count Repeat Enable. Each SWU_CTL[n].CNTRPTEN bit controls whether the watchpoint group's cycle count is reloaded and repeated after cycle countdown. If the SWU_CTL[n] register's SWU_CTL[n].CNTRPTEN bit is set, the SWU_CUR[n] register's SWU_CUR[n].CURCNT field is reloaded from SWU_CNT[n] register's SWU_CNT[n].COUNT field, and the countdown starts again. If SWU_CTL[n].CNTRPTEN bit is cleared, the expired count remains zero, and no further events are signalled. (See the SWU_CTL[n].CNTEN bit description for information regarding the countdown setup.)	
		0	Disable
		1	Enable
8 (R/W)	CNTEN	Count Enable. Each SWU_CTL[n].CNTEN bit controls whether the cycle count in the watchpoint group's SWU_CNT[n] register is decremented each cycle until it reaches zero. This feature is only valid in watchpoint mode (SWU_CTL[n].BWEN bit == 0).When the count reaches zero, any enabled watchpoint events are triggered. (See the SWU_CTL[n].CNTRPTEN bit description for optional actions at that may occur at the end of the countdown.)	
		0	Disable
		1	Enable
5 (R/W)	SCMPEN	Secure Comparison Enable. Each SWU_CTL[n].SCMPEN bit controls secure transaction comparison operation of an SWU watchpoint group, permitting matches based on transaction security.	
		0	Match on all transaction
		1	Match only secure transactions
4 (R/W)	IDCMPEN	ID Comparison Enable. Each SWU_CTL[n].IDCMPEN bit controls the ID comparison operation of an SWU watchpoint group. The ID match is based on comparison with the value in the SWU_ID[n] register.	

Table 49-7: SWU_CTL[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3:2 (R/W)	ACMPM	Address Comparison Mode. Each set of SWU_CTL[n].ACMPM bits control the address comparison operation of an SWU watchpoint group. The address within range for comparison is defined as (SWU_LA[n] register <= address < SWU_UA[n] register). The address outside range for comparison is defined as (address < SWU_LA[n]) or (SWU_UA[n] <= address).	
		0	No address comparison
		1	Exact match on LAn
		2	Match on address within range
		3	Match on address outside range
1 (R/W)	DIR	Transaction Direction for Match. Each SWU_CTL[n].DIR bit determines whether the SWU check reads or writes for watchpoint matches.	
		0	Match on reads only
		1	Match on writes only
0 (R/W)	EN	Enable Watchpoint. Each SWU_CTL[n].EN bit controls the operation of one SWU watchpoint group. Clearing the SWU_CTL[n].EN bit halts the execution of watchpoint or bandwidth tracking operations in the watchpoint group without resetting status or configuration registers. Setting the SWU_CTL[n].EN bit enables the SWU watchpoint group to begin or resume operation with the current configuration and status.	
		0	Disable
		1	Enable

Current Register n

The SWU current register (SWU_CUR[n]) operation varies depending whether the watchpoint group is in bandwidth mode or watchpoint mode. In both modes, the watchpoint count begins when the SWU loads the register's SWU_CUR[n].CURCNT field from the SWU_CNT[n] register's SWU_CNT[n].COUNT field when the watchpoint count is enabled (SWU_CTL[n] register, SWU_CTL[n].CNTEN bit =1).

In bandwidth mode, the current count field (SWU_CUR[n].CURCNT) contains the cycle count remaining within the current watchpoint period. The SWU decrements this value every cycle until the count reaches zero. At that point, the SWU reloads the SWU_CUR[n].CURCNT field from SWU_CNT[n] register's SWU_CNT[n].COUNT field. In bandwidth mode, the current bandwidth field (SWU_CUR[n].CURBW) contains the count of watchpoint matches (bandwidth) accumulated in the current watchpoint period.

In watchpoint mode, the current count field (SWU_CUR[n].CURCNT) contains the watchpoint match count remaining within the current watchpoint period. The SWU decrements this value with every watchpoint match until the count reaches zero. At that point, the SWU reloads the SWU_CUR[n].CURCNT field from SWU_CNT[n] register's SWU_CNT[n].COUNT field if the SWU_CTL[n] register's SWU_CTL[n].CNTRPTEN bit is set (=1). In watchpoint mode, the current bandwidth field (SWU_CUR[n].CURBW) is undefined.

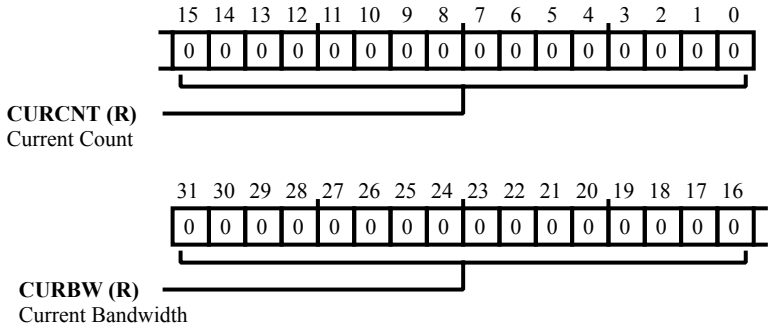


Figure 49-5: SWU_CUR[n] Register Diagram

Table 49-8: SWU_CUR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/NW)	CURBW	Current Bandwidth.
15:0 (R/NW)	CURCNT	Current Count.

Global Control Register

The SWU global control register (SWU_GCTL) provides SWU reset and enable.

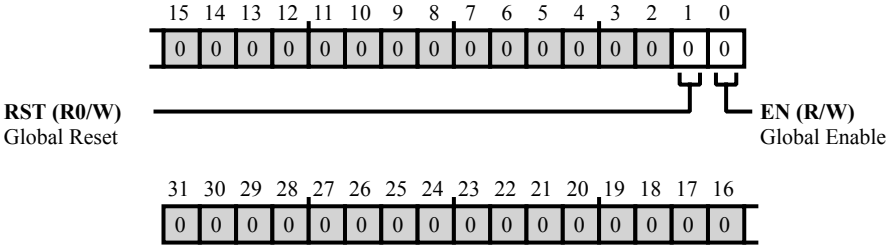


Figure 49-6: SWU_GCTL Register Diagram

Table 49-9: SWU_GCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R0/W)	RST	Global Reset. The SWU_GCTL.RST) is write-1-action/read zero and controls the SWU operational state. Setting SWU_GCTL.RST resets all SWU registers to their default values and halts all SWU operations.	
		0	No Action
		1	Reset
0 (R/W)	EN	Global Enable. The SWU_GCTL.EN controls the SWU operational state. Clearing SWU_GCTL.EN halts the execution of all watchpoint and bandwidth tracking operations without resetting status registers or associated signals. Setting SWU_GCTL.EN enables the SWU to begin/resume operation with the current configuration and status.	
		0	Disable
		1	Enable

Global Status Register

The SWU global status register (SWU_GSTAT) contains status bits for all four watchpoint groups.

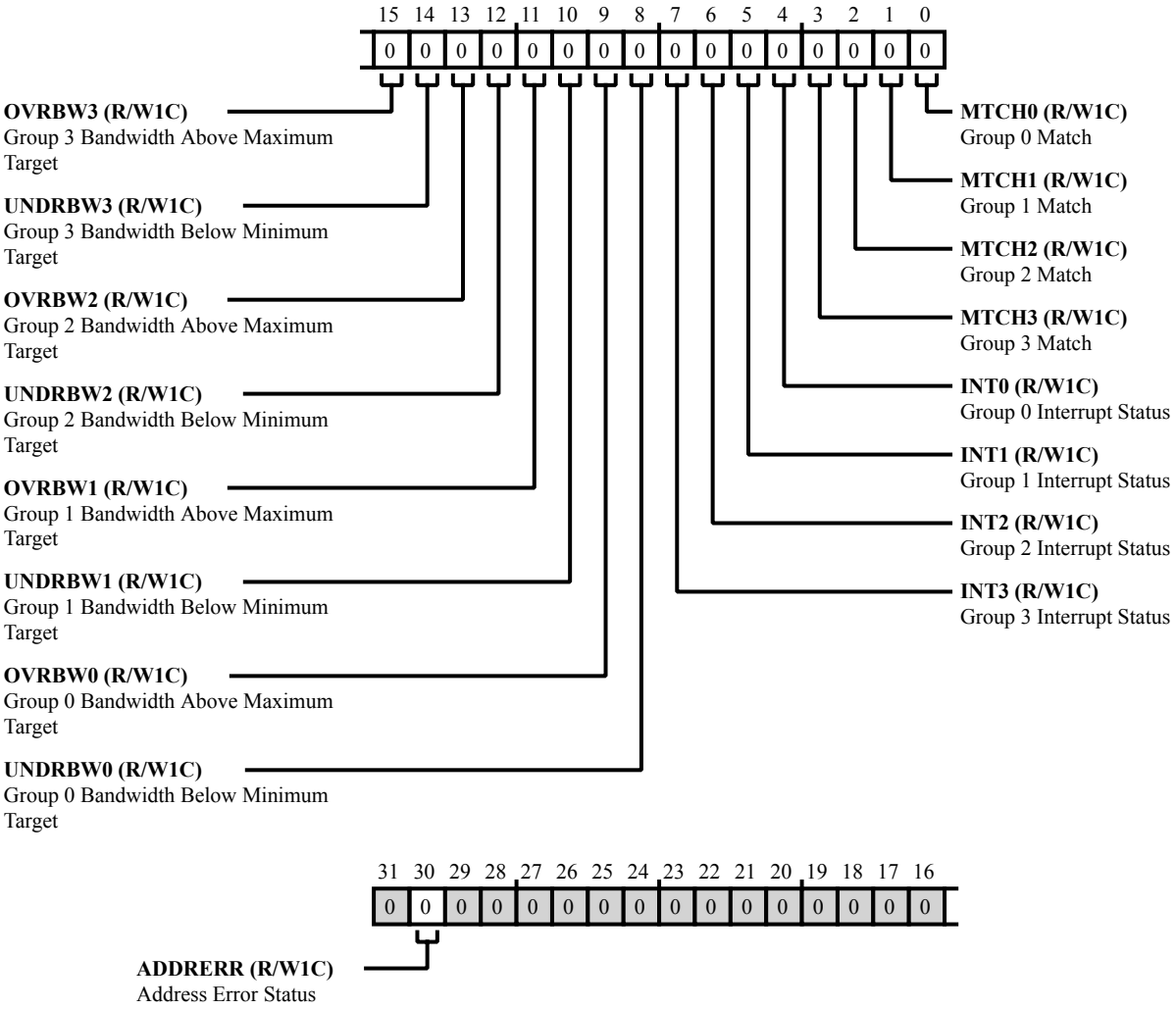


Figure 49-7: SWU_GSTAT Register Diagram

Table 49-10: SWU_GSTAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
30 (R/W1C)	ADDRERR	Address Error Status. The SWU_GSTAT.ADDRERR indicates that the SWU generated an address error. This status bit is sticky; write-1-to-clear it.	
		0	Inactive
		1	Active

Table 49-10: SWU_GSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
15 (R/W1C)	OVRBW3	Group 3 Bandwidth Above Maximum Target. See SWU_GSTAT.OVRBW0 description.	
		0	Group 3 was not above maximum bandwidth
		1	Group 3 was above maximum bandwidth
14 (R/W1C)	UNDRBW3	Group 3 Bandwidth Below Minimum Target. See SWU_GSTAT.UNDRBW0 description.	
		0	Group 3 was not below minimum bandwitdth
		1	Group 3 was below minimum bandwitdth
13 (R/W1C)	OVRBW2	Group 2 Bandwidth Above Maximum Target. See SWU_GSTAT.OVRBW0 description.	
		0	Group 2 was not above maximum bandwidth
		1	Group 2 was above maximum bandwidth
12 (R/W1C)	UNDRBW2	Group 2 Bandwidth Below Minimum Target. See SWU_GSTAT.UNDRBW0 description.	
		0	Group 2 was not below minimum bandwidth
		1	Group 2 was below minimum bandwidth
11 (R/W1C)	OVRBW1	Group 1 Bandwidth Above Maximum Target. See SWU_GSTAT.OVRBW0 description.	
		0	Group 1 was not above maximum bandwidth
		1	Group 1 was above maximum bandwidth
10 (R/W1C)	UNDRBW1	Group 1 Bandwidth Below Minimum Target. See SWU_GSTAT.UNDRBW0 description.	
		0	Group 1 was not below minimum bandwidth
		1	Group 1 was below minimum bandwidth
9 (R/W1C)	OVRBW0	Group 0 Bandwidth Above Maximum Target. The SWU_GSTAT.OVRBW0 - SWU_GSTAT.OVRBW3 -- Group 0 through 3 watch-point bandwidth over maximum target bits. Each maximum bandwidth bit indicate (for each group)s that the measured bandwidth over the period defined by the SWU_CNT[n] register was over the maximum target. This status bit is sticky; write-1-to-clear it.	
		0	Group 0 was not above maximum bandwidth
		1	Group 0 was above maximum bandwidth

Table 49-10: SWU_GSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
8 (R/W1C)	UNDRBW0	Group 0 Bandwidth Below Minimum Target. The SWU_GSTAT.UNDRBW0 - SWU_GSTAT.UNDRBW3 -- Group 0 through 3 watchpoint bandwidth below minimum target bits. Each minimum bandwidth bit indicates (for each group) that the measured bandwidth over the period defined by the SWU_CNT[n] register was below the minimum target. This status bit is sticky; write-1-to-clear it.	
		0	Group 0 was not below minimum bandwidth
		1	Group 0 was below minimum bandwidth
7 (R/W1C)	INT3	Group 3 Interrupt Status. See SWU_GSTAT.INT0 description.	
		0	No Interrupt
		1	Interrupt Occurred
6 (R/W1C)	INT2	Group 2 Interrupt Status. See SWU_GSTAT.INT0 description.	
		0	No Interrupt
		1	Interrupt Occurred
5 (R/W1C)	INT1	Group 1 Interrupt Status. See SWU_GSTAT.INT0 description.	
		0	No Interrupt
		1	Interrupt Occurred
4 (R/W1C)	INT0	Group 0 Interrupt Status. The SWU_GSTAT.INT0 - SWU_GSTAT.INT3 -- Group 0 through 3 interrupt bits. Each interrupt bit indicates (for each group) whether a watchpoint group is contributing to the SWU's interrupt output. This status bit is sticky; write-1-to-clear it.	
		0	No interrupt
		1	Interrupt Occurred
3 (R/W1C)	MTCH3	Group 3 Match. See SWU_GSTAT.MTCH0 description.	
		0	No Match
		1	Group 3 Watchpoint Match

Table 49-10: SWU_GSTAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
2 (R/W1C)	MTCH2	Group 2 Match. See SWU_GSTAT.MTCH0 description.	
		0	No match
		1	Group 2 Watchpoint Match
1 (R/W1C)	MTCH1	Group 1 Match. See SWU_GSTAT.MTCH0 description.	
		0	No match
		1	Group 1 Watchpoint Match
0 (R/W1C)	MTCH0	Group 0 Match. The SWU_GSTAT.MTCH0 - SWU_GSTAT.MTCH3 -- Group 0 through 3 match bits. Each match bit indicates (for each group) whether a watchpoint match has occurred in a SWU watchpoint group, as controlled by the group's related watchpoint control register (SWU_CTL[n]). This status bit is sticky; write-1-to-clear it.	
		0	No match
		1	Group 0 Watchpoint Match

Bandwidth History Register n

The SWU bandwidth history registers (SWU_HIST[n]) contain data copied from a watchpoint group's current bandwidth value (SWU_CUR[n] register, SWU_CUR[n].CURBW bits) at the end of the last two watchpoint periods. At the end of each watchpoint period, the SWU copies the previous bandwidth value from the SWU_HIST[n].BWHIST0 field to the SWU_HIST[n].BWHIST1 field and copies the new bandwidth value from the SWU_CUR[n].CURBW field to the SWU_HIST[n].BWHIST0 field.

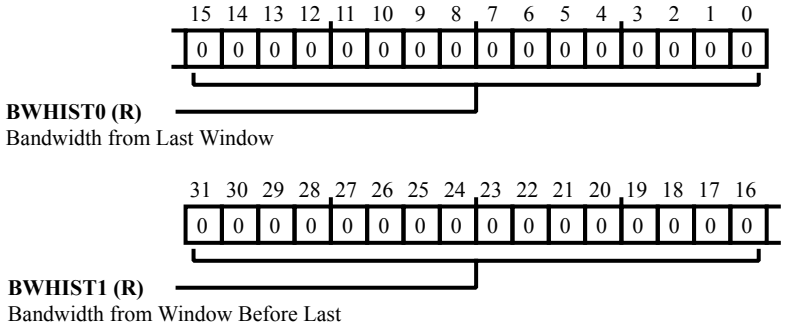


Figure 49-8: SWU_HIST[n] Register Diagram

Table 49-11: SWU_HIST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/NW)	BWHIST1	Bandwidth from Window Before Last.
15:0 (R/NW)	BWHIST0	Bandwidth from Last Window.

ID Register n

The SWU ID registers (SWU_ID[n]) contain a 16-bit ID field (SWU_ID[n].ID) and a 16-bit ID mask field (SWU_ID[n].IDMASK) that watchpoint groups use for ID comparison. The ID on the bus is AND'ed with the SWU_ID[n].IDMASK field, then the watchpoint group compares the result against the SWU_ID[n].ID field.

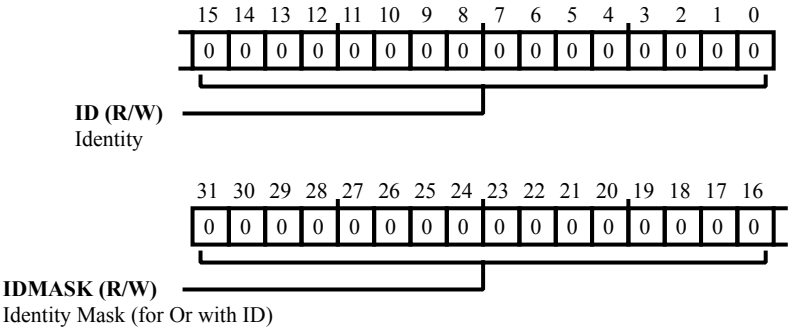


Figure 49-9: SWU_ID[n] Register Diagram

Table 49-12: SWU_ID[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	IDMASK	Identity Mask (for Or with ID).
15:0 (R/W)	ID	Identity.

Lower Address Register n

The SWU lower address registers (SWU_LA[n]) contain each watchpoint group's lower address for address match comparison. In exact match on SWU_LA[n] address mode (SWU_CTL[n].ACMPM bits =01), the watchpoint group uses only this address for match comparison.

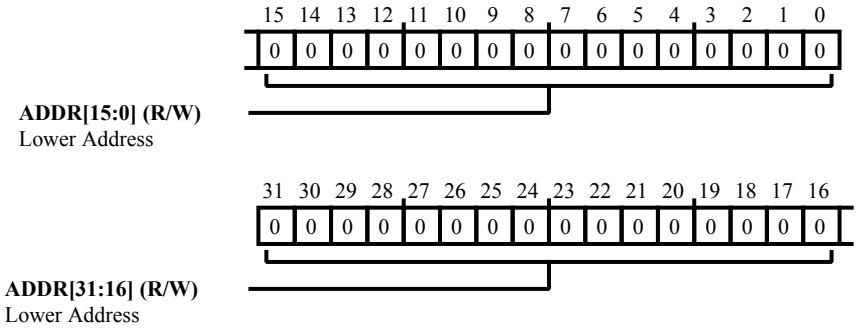


Figure 49-10: SWU_LA[n] Register Diagram

Table 49-13: SWU_LA[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDR	Lower Address.

Target Register n

The SWU target registers (SWU_TARG[n]) contain a minimum value field (SWU_TARG[n].BWMIN) and maximum value field (SWU_TARG[n].BWMAX) of bandwidth targets used by watchpoint groups in bandwidth mode. When the bandwidth period expires, if the current bandwidth value (SWU_CUR[n] register, SWU_CUR[n].CURBW bits) is below the minimum target or above the maximum target, the watchpoint group takes action as enabled by the SWU_CTL[n] register's SWU_CTL[n].MINACT or SWU_CTL[n].MAXACT bits.

In bandwidth mode, note that the watchpoint group increments its count of either data bus transactions or address bus transactions (bursts) as selected by the SWU_CTL[n].BLENINC bit. Keep this mode selection in mind when programming the bandwidth target values.

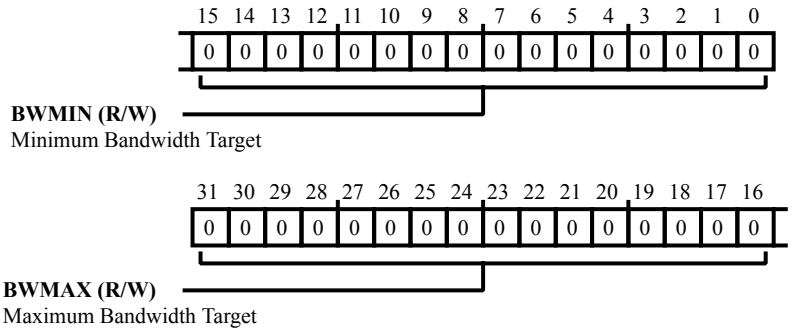


Figure 49-11: SWU_TARG[n] Register Diagram

Table 49-14: SWU_TARG[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:16 (R/W)	BWMAX	Maximum Bandwidth Target.
15:0 (R/W)	BWMIN	Minimum Bandwidth Target.

Upper Address Register n

The SWU upper address registers (SWU_UA[n]) contain each watchpoint group's upper address for address match comparison. In exact match on SWU_LA[n] address mode (SWU_CTL[n].ACMPM bits =01), the SWU_UA[n] is not used for match comparison.

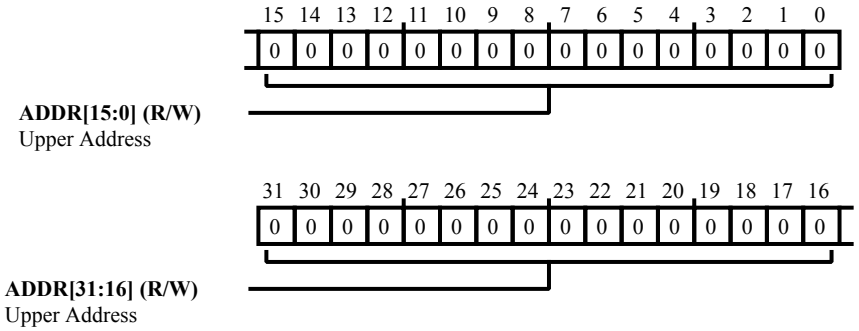


Figure 49-12: SWU_UA[n] Register Diagram

Table 49-15: SWU_UA[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDR	Upper Address.

50 Memory Error Protection Unit (MEPU)

Memory Error Protection Unit (MEPU) handles single-bit and double-bit memory error detection and correction across the memories in the chip and controls routing of their interrupts/triggers.

It consists of following blocks:

- Memory Error Controller (MEC)
- Parity Controller (PCTL)

Memory Error Controller (MEC)

The Memory Error Controller (MEC) manages memory parity, ECC errors and warning inputs from various cores and peripherals and sends out interrupt and trigger outputs. It is a generic N x M multiplexer and interrupt controller for error or warning inputs to interrupt and trigger outputs. It has control features such as enable and disable and interrupt masking, and status signaling for input/outputs. For more information on MEC functionality, see the MEC registers description.

MEC Features

The MEC unit has the following features:

- Interrupt controller for memory parity, ECC errors and warnings
- N x M multiplexer for error inputs to interrupt and trigger outputs
- Peripheral bus completer interface for register read and write
- Control and status registers
    - Control for enabling or disabling each input error status and masking its interrupt
    - Control for enabling or disabling each output error interrupt
    - Input error status bits are sticky and write-1-to-clear
    - Output error interrupt status bits are read only
    - Lock feature for control registers write. Status registers capture lock write error.

- Peripheral and component identification registers
- Capability to retain parity, ECC error and warning status even after any reset for fault source debug
- Fully-configurable design

Parity Controller (PCTL)

The Parity Controller (PCTL) handles parity encoding and decoding to and from memory data for single bit memory error detection.

PCTL Features

The PCTL unit has the following features:

- Generic memory parity controller
    - Generation of write parity bits from write data bits
    - Concatenation of parity bits with data bits for memory write data
    - Generation of write enable mask for parity bits to memory
    - Extraction of read data bits from memory read data
    - Detection of read parity error from memory read data
- Even parity logic for parity generation and error detection
- Provision for parity bit interleaving
- Single parity error output per memory instance
- Memory initialization control logic
- Fully-configurable design

MEC Functional Description

The MEC works with the PCTL to detect and correct memory error across the memories in the chip. It also controls the routing of the generated interrupts and triggers.

ADSP-SC59x MEC Register List

The Memory Error Controller (MEC) manages memory parity/ inputs from the core and peripherals and sends interrupt/trigger outputs. It is a generic N x M multiplexer and interrupt controller for error to interrupt/trigger outputs. It has control features such as enable/disable and interrupt masking, and status signaling for inputs/outputs. For more information on MEC functionality, see the MEC register descriptions.

Table 50-1: ADSP-SC59x MEC Register List

Name	Description
MEC_A55EIRQ_GCTL[n]	A55 Core Memory Errors Interrupt Request Global Control Register
MEC_A55EIRQ_GSTAT[n]	A55 Error Interrupt Request Global Status Register
MEC_A55EIRQ_STAT[n]	A55 Error Status Register
MEC_A55ERR_CTL[n]	A55 Error Control Register
MEC_A55ERR_IMASK[n]	A55 Error Interrupt Mask Register
MEC_CID0	Component ID0 Register
MEC_CID1	Component ID1 Register
MEC_CID2	Component ID2 Register
MEC_CID3	Component ID3 Register
MEC_CLR	Clear Register
MEC_ECCERR_CTL[n]	ECC Error Control Register
MEC_ECCERR_IMASK[n]	ECC Error Interrupt Mask Register
MEC_ECCERR_STAT[n]	ECC Error Status Register
MEC_EEIRQ_GCTL[n]	ECC Error Interrupt Request Global Control Register
MEC_EEIRQ_GSTAT[n]	ECC Error Interrupt Request Global Status Register
MEC_PEIRQ_GCTL[n]	Parity Error Interrupt Request Global Control Register
MEC_PEIRQ_GSTAT[n]	Parity Error Interrupt Request Global Status Register
MEC_PERR_CTL0	Parity Error Control Register
MEC_PERR_CTL1	Parity Error Control Register
MEC_PERR_IMASK0	Parity Error Interrupt Mask Register
MEC_PERR_IMASK1	Parity Error Interrupt Mask Register
MEC_PERR_STAT0	Parity Error Status Register
MEC_PERR_STAT1	Parity Error Status Register
MEC_PID0	Peripheral ID0 Register
MEC_PID1	Peripheral ID1 Register
MEC_PID2	Peripheral ID2 Register
MEC_PID3	Peripheral ID3 Register
MEC_PID4	Peripheral ID4 Register

MEPU Block Diagram

The ***MEPU Block Diagram*** shows the functional blocks within the Memory Error Protection Unit (MEPU) unit.

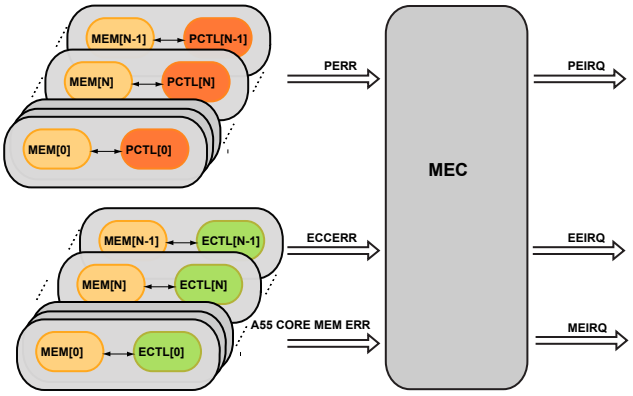


Figure 50-1: MEPU Unit Block Diagram

MEC Block Diagram

The *MEC Interface Block Diagram* shows the functional blocks within the MEC module.

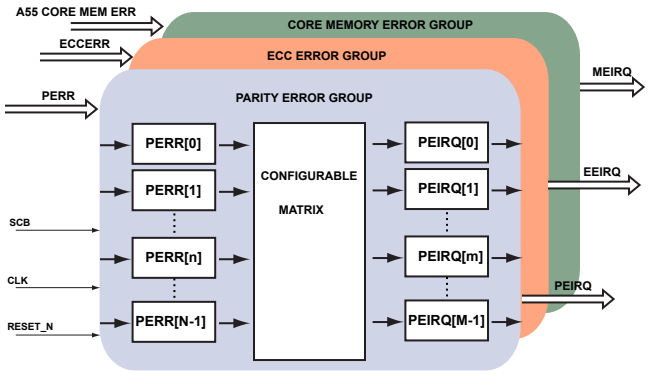


Figure 50-2: MEC Unit Block Diagram

PCTL Block Diagram

The *PCTL Block Diagram* shows the functional blocks within the PCTL module.

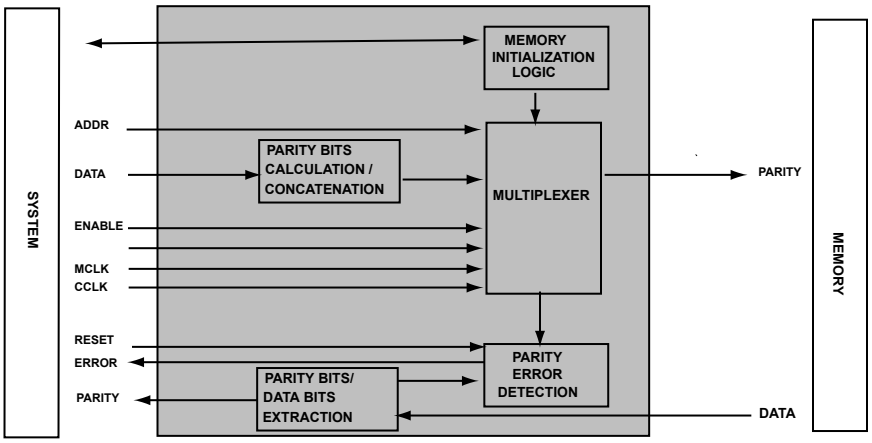


Figure 50-3: PCTL Block Diagram

MECx Input and Output Block Diagram

The *MECx Input and Output Block Diagram* shows the input and output routing to various cores.

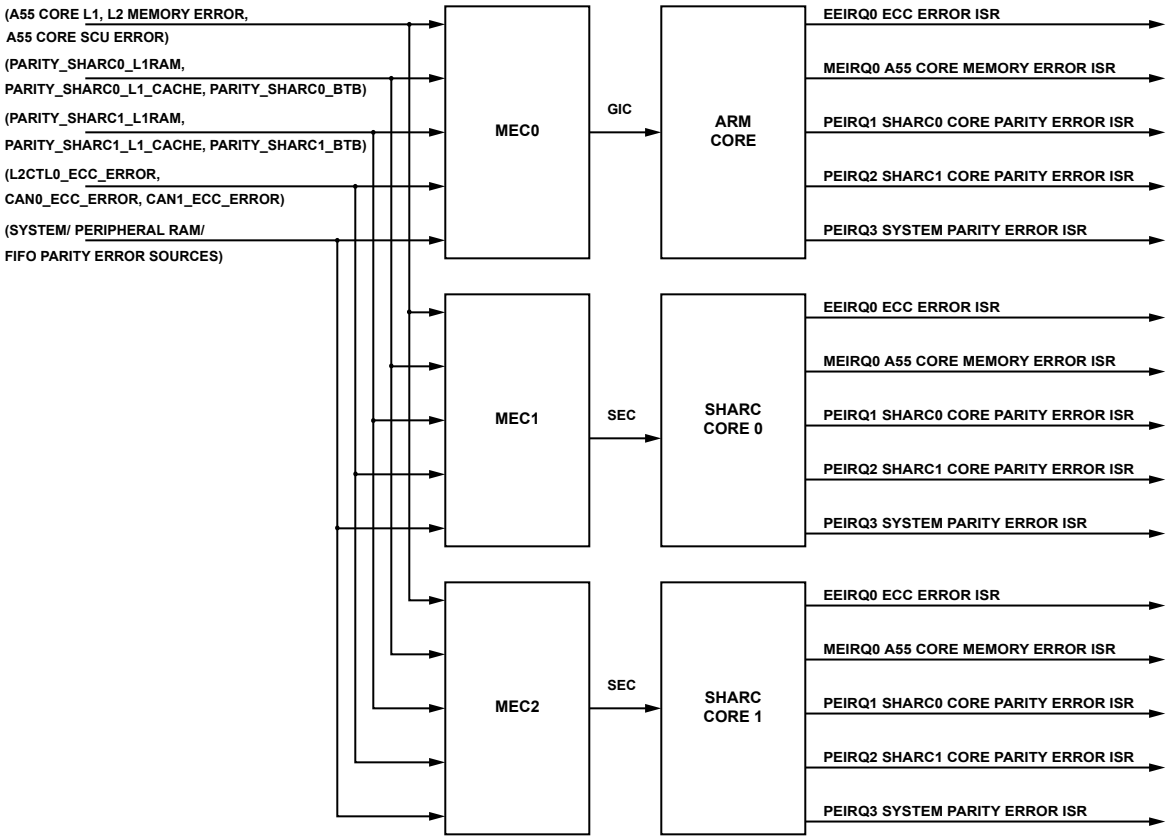


Figure 50-4: MECx Input and Output Block Diagram

MEC Architectural Concepts

The MEC unit communicates with the Cortex-A55 core directly through the subsystem's SYS bus crossbar. It services MMR addresses associated with the MEC function unit and translates the addresses into specific transcendental MEC functions. The results are read from common result registers shared by all the functions in single-precision floating-point format. The block stalls all read accesses until the result is ready.

MEC Configuration

The following tables describe the range of values for configuration and the mapping of parity errors.

Table 50-2: Mapping of Input Parity Errors (PERR)

PERR ID	Name	Description
0	Reserved	Not used
1	Reserved	Not used
2	PERR_C1_L1R	Core1 (SHARC0) L1 RAM Parity Error
3	PERR_C1_L1CR	Core1 (SHARC0) L1 Cache RAM Parity Error
4	PERR_C1_BPR	Core1 (SHARC0) Branch Predictor RAM Parity Error
5	PERR_C2_L1R	Core2 (SHARC1) L1 RAM Parity Error
6	PERR_C2_L1CR	Core2 (SHARC1) L1 Cache RAM Parity Error
7	PERR_C2_BPR	Core2 (SHARC1) Branch Predictor RAM Parity Error
8	PERR_ASRC0_FR	ASRC0 FIFO RAM Parity Error
9	PERR_ASRC1_FR	ASRC1 FIFO RAM Parity Error
10	PERR_ASRC2_FR	ASRC2 FIFO RAM Parity Error
11	PERR_ASRC3_FR	ASRC3 FIFO RAM Parity Error
12	PERR_ASRC4_FR	ASRC3 FIFO RAM Parity Error
13	PERR_ASRC5_FR	ASRC3 FIFO RAM Parity Error
14	PERR_ASRC6_FR	ASRC3 FIFO RAM Parity Error
15	PERR_ASRC7_FR	ASRC3 FIFO RAM Parity Error
16	PERR_C0_IIR0_R	SH0 IIR0 RAM Parity Error
17	PERR_C0_IIR1_R	SH0 IIR1 RAM Parity Error
18	PERR_C0_IIR2_R	SH0 IIR2 RAM Parity Error
19	PERR_C0_IIR3_R	SH0 IIR3 RAM Parity Error
20	PERR_C0_FIR0_R	FIR0 RAM Parity Error
21	PERR_C1_IIR0_R	SH0 IIR0 RAM Parity Error
22	PERR_C1_IIR1_R	SH0 IIR1 RAM Parity Error
23	PERR_C1_IIR2_R	SH0 IIR2 RAM Parity Error

Table 50-2: Mapping of Input Parity Errors (PERR) (Continued)

PERR ID	Name	Description
24	PERR_C1_IIR3_R	SH0 IIR3 RAM Parity Error
25	PERR_C1_FIR0_R	FIR0 RAM Parity Error
26	PERR_USB0_FR	USB0 FIFO RAM Parity Error
27	PERR_TRNG0_DBR	TRNG0 Data Buffer RAM Parity Error
28	PERR_PKA0_DR	PKA0 Data RAM Parity Error
29	PERR_SPE0_BR	SPE0 Buffer RAM Parity Error
30	PERR_SPE0_AR	SPE0 ARC4 RAM Parity Error
31	PERR_EMAC0_TFR	EMAC0 Transmit FIFO RAM Parity Error
32	PERR_EMAC0_RFR	EMAC0 Receive FIFO RAM Parity Error
33	PERR_MLB0_DBR	MLB0 Data Buffer RAM Parity Error
34	PERR_MLB0_CTR	MLB0 Channel Table RAM Parity Error
35	PERR_TMC0_TDR	TMC0 Trace Data RAM Parity Error
36	PERR_EMAC1_TFR	EMAC1 Transmit FIFO RAM Parity Error
37	PERR_EMAC1_RFR	EMAC1 Receive FIFO RAM Parity Error
38	PERR_EMAC0_TMI	EMAC0 TMI FIFO RAM Parity Error
39	PERR_ EMAC0_EST	EMAC0 EST FIFO RAM Parity Error
40	PERR_EMSI0	EMSI0 FIFO RAM Parity Error

Table 50-3: Mapping of Output Parity Errors (PERR)

PERR ID	Name	Description
0	PEIRQ_C0	Core0 (Arm) Parity Error Interrupt
1	PEIRQ_C1	Core1 (SHARC0) Parity Error Interrupt
2	PEIRQ_C2	Core2 (SHARC1) Parity Error Interrupt
3	PEIRQ_SYS	System Peripheral Parity Error Interrupt

Table 50-4: Mapping of Input ECC Errors (ECCERR)

PERR ID	Name	Description
0	ECCERR_L2CTL0	L2CTL0 ECC Error
1	ECCERR_CAN0	CAN0 ECC Error
2	ECCERR_CAN1	CAN1 ECC Error

Table 50-5: Mapping of Output ECC Errors (EEIRQ)

PERR ID	Name	Description
0	EEIRQ	ECC Error Interrupt

Table 50-6: Mapping of Input A55 Core Memory Errors (A55ERR)

PERR ID	Name	Description
0	A55_SCU_ERR	A55 SCU Error
1	A55_L1L2_ERR	A55 L1,L2 Error

Table 50-7: Mapping of Output A55 Core Memory Errors (A55EIRQ)

PERR ID	Name	Description
0	A55EIRQ	A55 Core Memory Error Interrupt

PCTL Integration

There is one PCTL instance per port per memory instance (for example, one PCTL instance for single port memory and two PCTL instances for dual port memories).

For A55 core memories initialization is done by built in hardware. So software initialization is not required.

ADSP-SC59x MEC Register Descriptions

Memory Error Controller (MEC) contains the following registers.

Table 50-8: ADSP-SC59x MEC Register List

Name	Description
MEC_A55EIRQ_GCTL[n]	A55 Core Memory Errors Interrupt Request Global Control Register
MEC_A55EIRQ_GSTAT[n]	A55 Error Interrupt Request Global Status Register
MEC_A55EIRQ_STAT[n]	A55 Error Status Register
MEC_A55ERR_CTL[n]	A55 Error Control Register
MEC_A55ERR_IMASK[n]	A55 Error Interrupt Mask Register
MEC_CID0	Component ID0 Register
MEC_CID1	Component ID1 Register
MEC_CID2	Component ID2 Register
MEC_CID3	Component ID3 Register
MEC_CLR	Clear Register
MEC_ECCERR_CTL[n]	ECC Error Control Register
MEC_ECCERR_IMASK[n]	ECC Error Interrupt Mask Register

Table 50-8: ADSP-SC59x MEC Register List (Continued)

Name	Description
MEC_ECCERR_STAT[n]	ECC Error Status Register
MEC_EEIRQ_GCTL[n]	ECC Error Interrupt Request Global Control Register
MEC_EEIRQ_GSTAT[n]	ECC Error Interrupt Request Global Status Register
MEC_PEIRQ_GCTL[n]	Parity Error Interrupt Request Global Control Register
MEC_PEIRQ_GSTAT[n]	Parity Error Interrupt Request Global Status Register
MEC_PERR_CTL0	Parity Error Control Register
MEC_PERR_CTL1	Parity Error Control Register
MEC_PERR_IMASK0	Parity Error Interrupt Mask Register
MEC_PERR_IMASK1	Parity Error Interrupt Mask Register
MEC_PERR_STAT0	Parity Error Status Register
MEC_PERR_STAT1	Parity Error Status Register
MEC_PID0	Peripheral ID0 Register
MEC_PID1	Peripheral ID1 Register
MEC_PID2	Peripheral ID2 Register
MEC_PID3	Peripheral ID3 Register
MEC_PID4	Peripheral ID4 Register

A55 Core Memory Errors Interrupt Request Global Control Register

The MEC_A55EIRQ_GCTL[n] register bits control the core memory error interrupt and trigger outputs.

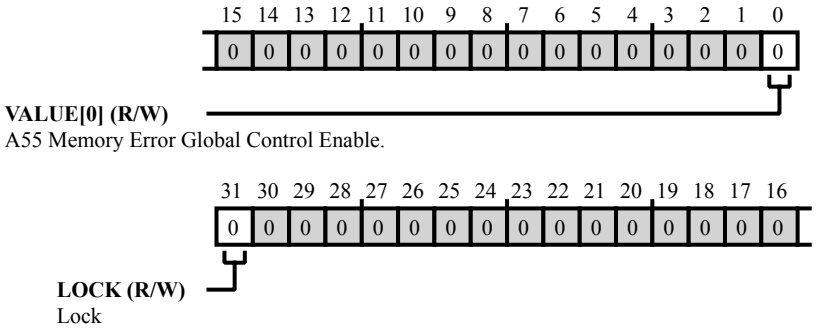


Figure 50-5: MEC_A55EIRQ_GCTL[n] Register Diagram

Table 50-9: MEC_A55EIRQ_GCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_A55EIRQ_GCTL[n].LOCK bit is enabled, the MEC_A55EIRQ_GCTL[n] register is read only.	
		0	Unlock
		1	Lock
0 (R/W)	VALUE	A55 Memory Error Global Control Enable.. The MEC_A55EIRQ_GCTL[n].VALUE bit enables the A55 memory error global control.	
		0	Disable
		1	Enable

A55 Error Interrupt Request Global Status Register

The MEC_A55EIRQ_GSTAT[n] register bits reflect the status of the core memory error interrupt and trigger outputs.

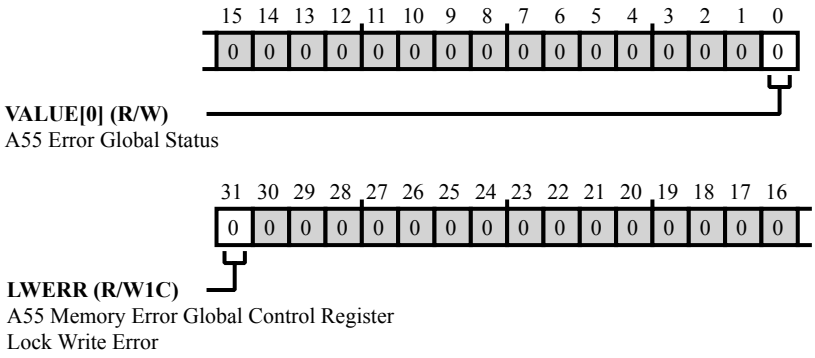


Figure 50-6: MEC_A55EIRQ_GSTAT[n] Register Diagram

Table 50-10: MEC_A55EIRQ_GSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	A55 Memory Error Global Control Register Lock Write Error. When set (=1), the MEC_A55EIRQ_GSTAT[n].LWERR bit indicates that there was an attempted write to an MEC register while the MEC_EEIRQ_GCTL[n].LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
0 (R/W)	VALUE	A55 Error Global Status. When set (=1), the MEC_A55EIRQ_GSTAT[n].VALUE bit indicates the assertion of an interrupt/trigger. When cleared (=0), there is no interrupt or trigger.	
		0	No Interrupt/Trigger
		1	Interrupt/Trigger Asserted

A55 Error Status Register

The MEC_A55EIRQ_STAT[n] register bits reflect the status of the core memory error inputs.

Writing '1' to these bits clears the corresponding error status. (W1C)

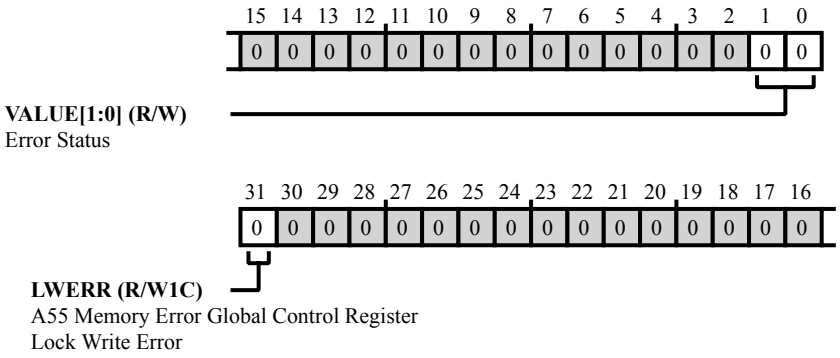


Figure 50-7: MEC_A55EIRQ_STAT[n] Register Diagram

Table 50-11: MEC_A55EIRQ_STAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	A55 Memory Error Global Control Register Lock Write Error. The MEC_A55EIRQ_STAT[n].LWERR bit indicates (when set) there was an attempted write to an MEC register while the MEC_EEIRQ_GCTL[n].LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
1:0 (R/W)	VALUE	Error Status. The MEC_A55EIRQ_STAT[n].VALUE field indicates the status of an interrupt/trigger. When set (=1), an interrupt or trigger is asserted. When cleared (=0), there is no interrupt or trigger.	
		1	A55 SCU Error
		2	A55 L1,L2 Error

A55 Error Control Register

The MEC_A55ERR_CTL[n] register bits control the core memory error inputs and whether their status will be reflected in the register.

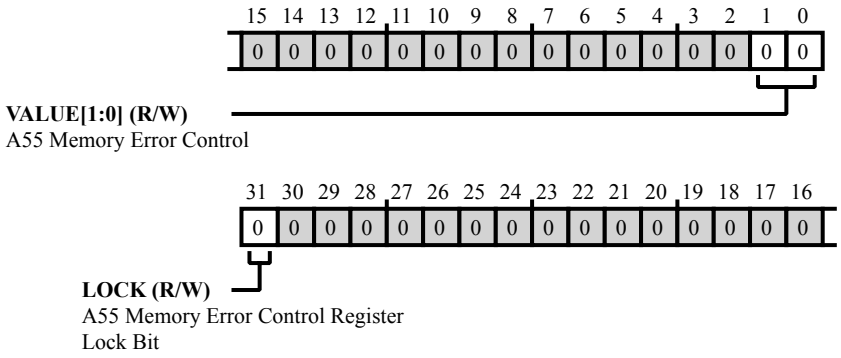


Figure 50-8: MEC_A55ERR_CTL[n] Register Diagram

Table 50-12: MEC_A55ERR_CTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	A55 Memory Error Control Register Lock Bit. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_A55ERR_CTL[n].LOCK bit is enabled, the MEC_A55ERR_CTL[n] register is read only.	
		0	Unlock
		1	Lock
1:0 (R/W)	VALUE	A55 Memory Error Control. The MEC_A55ERR_CTL[n].VALUE field enables A55 memory error control. When set (=1), control is enabled. When cleared (=0), control is disabled.	
		1	A55 SCU Err
		2	A55 L1,L2 Err

A55 Error Interrupt Mask Register

The MEC_A55ERR_IMASK[n] register bits control interrupt masks for memory error inputs from the core.

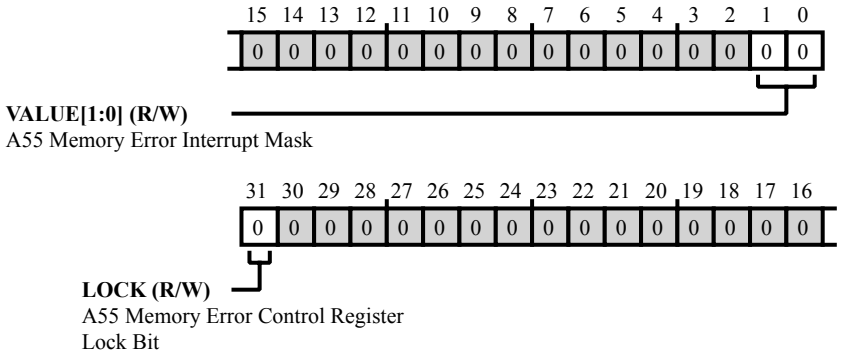


Figure 50-9: MEC_A55ERR_IMASK[n] Register Diagram

Table 50-13: MEC_A55ERR_IMASK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	A55 Memory Error Control Register Lock Bit. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_A55ERR_IMASK[n].LOCK bit is enabled, the MEC_A55ERR_IMASK[n] register is read only.	
		0	Unlock
		1	Lock
1:0 (R/W)	VALUE	A55 Memory Error Interrupt Mask. The MEC_A55ERR_IMASK[n].VALUE field indicates whether an interrupt is masked (when set (=1)) or unmasked (=0).	
		1	A55 SCU Err
		2	A55 L1,L2 Err

Component ID0 Register

The MEC_CID0 register is a component ID register.

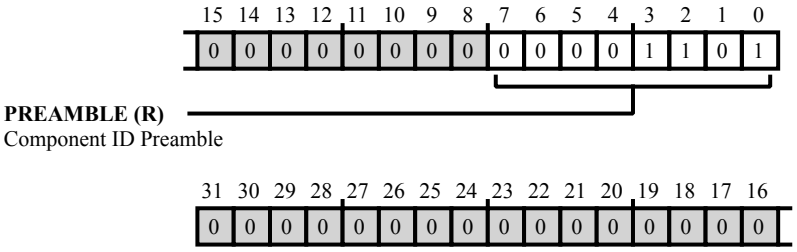


Figure 50-10: MEC_CID0 Register Diagram

Table 50-14: MEC_CID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PREAMBLE	Component ID Preamble. The MEC_CID0.PREAMBLE field indicates the component ID preamble.

Component ID1 Register

The MEC_CID1 register is a component ID register.

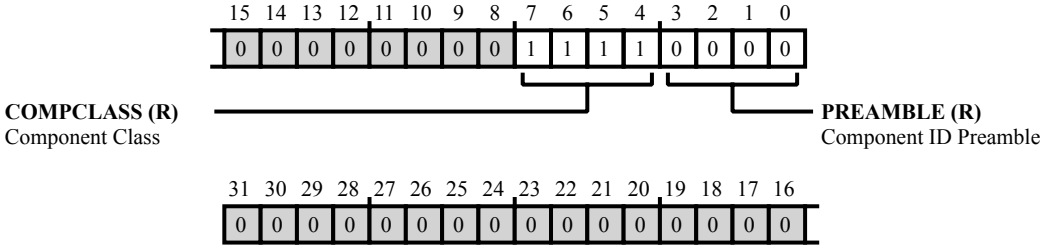


Figure 50-11: MEC_CID1 Register Diagram

Table 50-15: MEC_CID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
7:4 (R/NW)	COMPCLASS	Component Class. The MEC_CID1.COMPCLASS field indicates the component class. Dedicated debug blocks (core debug access port, Program Trace, etc.) should identify as CoreSight (i.e. 0x9) and implement the full compliment of CoreSight registers including DEVTYPE. All other ADI components should identify as System (i.e. 0xF) components. See CoreSight Architecture Specification for details.	
		9	CoreSight
		15	System
3:0 (R/NW)	PREAMBLE	Component ID Preamble. The MEC_CID1.PREAMBLE field indicates the component ID preamble.	

Component ID2 Register

The MEC_CID2 register is a component ID register.

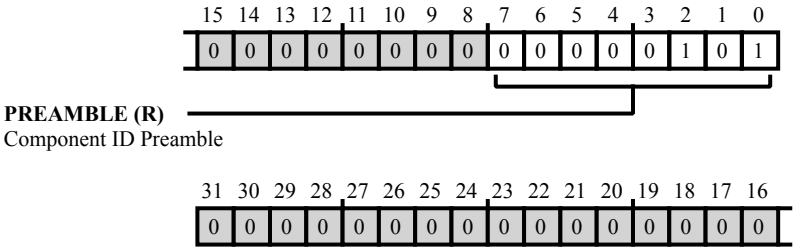


Figure 50-12: MEC_CID2 Register Diagram

Table 50-16: MEC_CID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PREAMBLE	Component ID Preamble. The MEC_CID2.PREAMBLE field indicates the component ID preamble.

Component ID3 Register

The MEC_CID3 register is a component ID register.

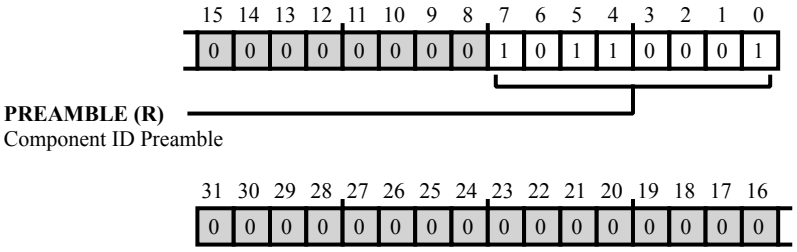


Figure 50-13: MEC_CID3 Register Diagram

Table 50-17: MEC_CID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PREAMBLE	Component ID Preamble. The MEC_CID3.PREAMBLE field indicates the component ID preamble.

Clear Register

The MEC_CLR register clears all the MEC status registers. Before enabling any of the MEC control registers, write to this register in the software.

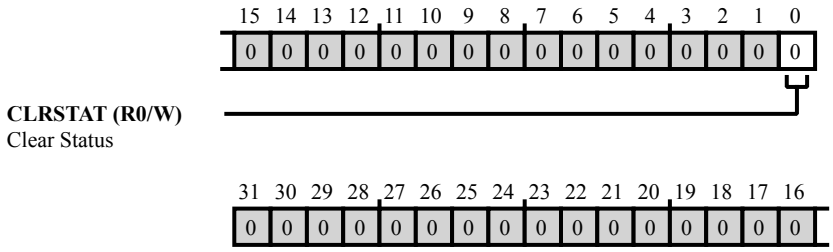


Figure 50-14: MEC_CLR Register Diagram

Table 50-18: MEC_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R0/W)	CLRSTAT	Clear Status. Write of '1' to the MEC_CLR.CLRSTAT bit clears all status registers of MEC

ECC Error Control Register

MEC_ECCERR_CTL[n] register bits control enable/disable for ECC error inputs from various cores/peripherals and decide whether their status will be reflected in ECC error status register bits.

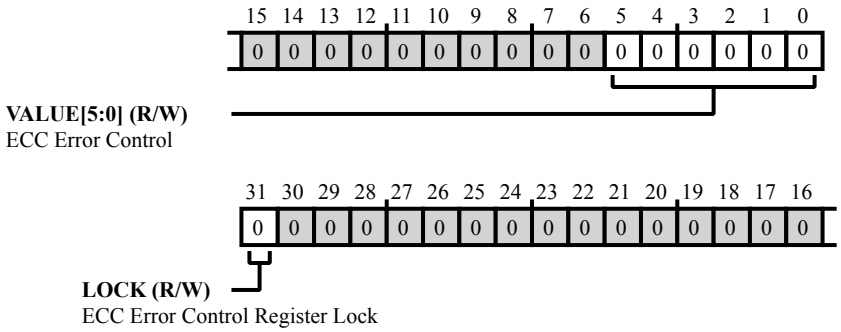


Figure 50-15: MEC_ECCERR_CTL[n] Register Diagram

Table 50-19: MEC_ECCERR_CTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	ECC Error Control Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_ECCERR_CTL[n].LOCK bit is enabled, the MEC_ECCERR_CTL[n] register is read only.	
		0	Unlock
		1	Lock
5:0 (R/W)	VALUE	ECC Error Control. The MEC_ECCERR_CTL[n].VALUE bit indicates whether ECC control is enabled (=1) or disabled (=0).	
		1	L2 CTL ECC Error
		2	CAN0 ECC Error
		4	CAN1 ECC Error
		8	Reserved

ECC Error Interrupt Mask Register

MEC_ECCERR_IMASK[n] register bits control interrupt masks for ECC error inputs from various cores/peripherals.

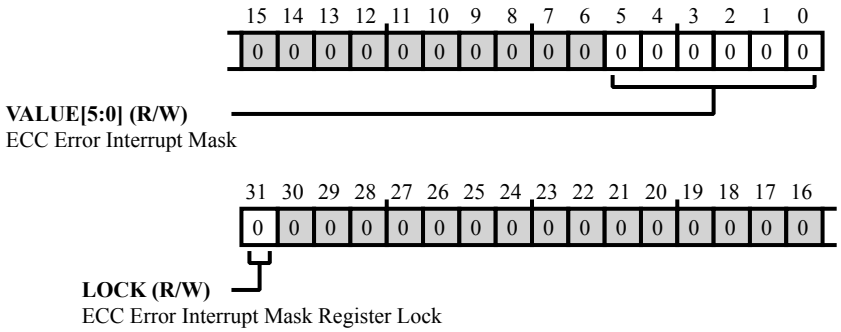


Figure 50-16: MEC_ECCERR_IMASK[n] Register Diagram

Table 50-20: MEC_ECCERR_IMASK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	ECC Error Interrupt Mask Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_ECCERR_IMASK[n].LOCK bit is enabled, the MEC_ECCERR_IMASK[n] register is read only.	
		0	Unlock
		1	Lock
5:0 (R/W)	VALUE	ECC Error Interrupt Mask. The MEC_ECCERR_IMASK[n].VALUE bit indicates when the ECC error interrupt is masked (=1) or unmasked (=0).	
		1	L2 CTL ECC Error
		2	CAN0 ECC Error
		4	CAN1 ECC Error
		8	Reserved

ECC Error Status Register

MEC_ECCERR_STAT[n] register bits reflect status for ECC error inputs from various cores/peripherals. Writing '1' to these bits clear corresponding ECC error status.

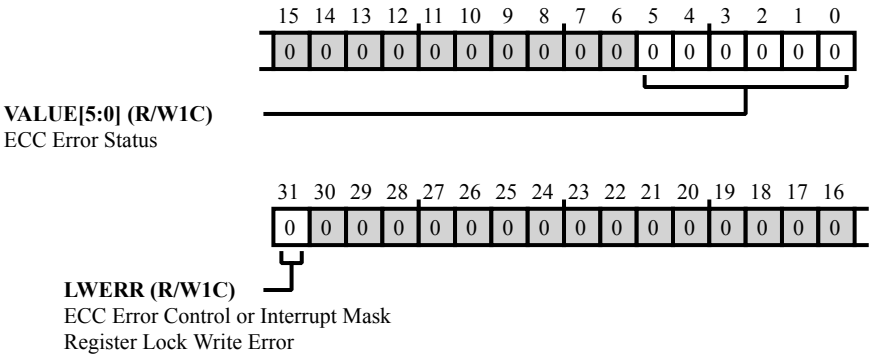


Figure 50-17: MEC_ECCERR_STAT[n] Register Diagram

Table 50-21: MEC_ECCERR_STAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	ECC Error Control or Interrupt Mask Register Lock Write Error. The MEC_ECCERR_STAT[n].LWERR bit indicates (when set) there was an attempted write to an MEC register while the MEC_ECCERR_CTL[n].LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK bit =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
5:0 (R/W1C)	VALUE	ECC Error Status. '1' indicates error and '0' indicates no error. Sticky bit, write '1' to clear.	
		1	L2 CTL ECC Error
		2	CAN0 ECC Error
		4	CAN1 ECC Error
		8	Reserved

ECC Error Interrupt Request Global Control Register

MEC_EEIRQ_GCTL[n] register bits control enable/disable of ECC error interrupt/trigger outputs.

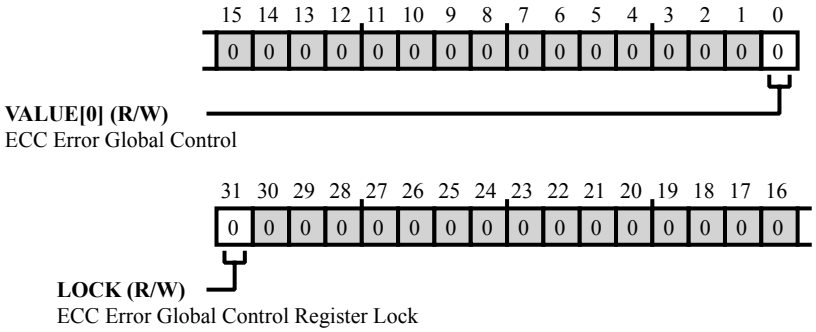


Figure 50-18: MEC_EEIRQ_GCTL[n] Register Diagram

Table 50-22: MEC_EEIRQ_GCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	ECC Error Global Control Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_EEIRQ_GCTL[n].LOCK bit is enabled, the MEC_EEIRQ_GCTL[n] register is read only.	
		0	Unlock
		1	Lock
0 (R/W)	VALUE	ECC Error Global Control. '1' indicates enable and '0' indicates disable.	

ECC Error Interrupt Request Global Status Register

MEC_EEIRQ_GSTAT[n] register bits reflect status of ECC error interrupt/trigger outputs.

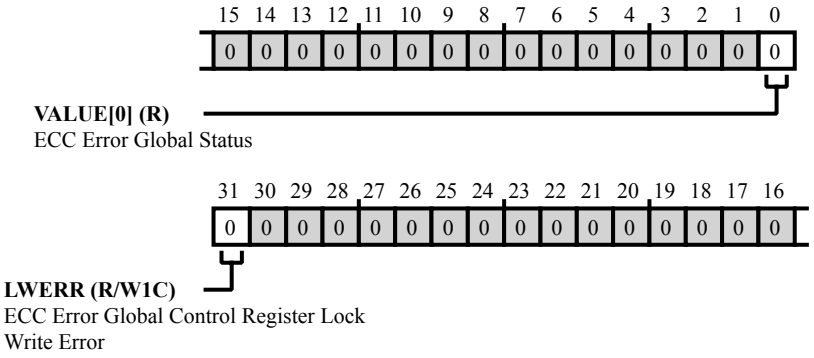


Figure 50-19: MEC_EEIRQ_GSTAT[n] Register Diagram

Table 50-23: MEC_EEIRQ_GSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	ECC Error Global Control Register Lock Write Error. The MEC_EEIRQ_GSTAT[n].LWERR bit indicates (when set) there was an attempted write to an MEC register while the MEC_EEIRQ_GCTL[n].LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK bit =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
0 (R/NW)	VALUE	ECC Error Global Status. '1' indicates assertion of interrupt/trigger and '0' indicates no interrupt/trigger.	

Parity Error Interrupt Request Global Control Register

MEC_PEIRQ_GCTL[n] register bits control enable/disable of parity error interrupt/trigger outputs.

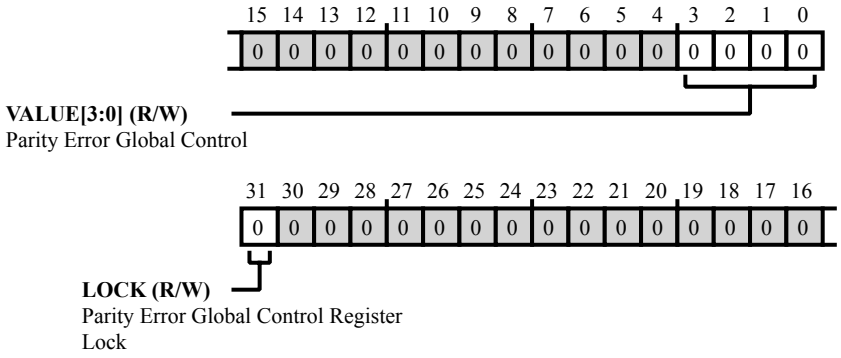


Figure 50-20: MEC_PEIRQ_GCTL[n] Register Diagram

Table 50-24: MEC_PEIRQ_GCTL[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Parity Error Global Control Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PEIRQ_GCTL[n].LOCK bit is enabled, the MEC_PEIRQ_GCTL[n] register is read only.	
		0	Unlock
		1	Lock
3:0 (R/W)	VALUE	Parity Error Global Control. The MEC_PEIRQ_GCTL[n].VALUE field indicates parity error global control. '1' indicates enable and '0' indicates disable.	

Parity Error Interrupt Request Global Status Register

MEC_PEIRQ_GSTAT[n] register bits reflect status of parity error interrupt/trigger outputs.

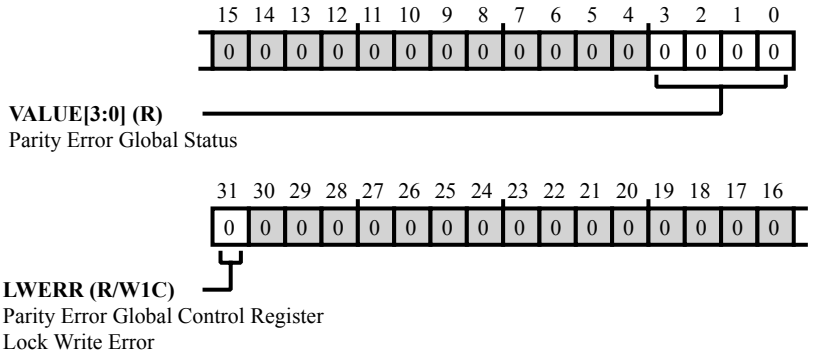


Figure 50-21: MEC_PEIRQ_GSTAT[n] Register Diagram

Table 50-25: MEC_PEIRQ_GSTAT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Parity Error Global Control Register Lock Write Error. The MEC_PEIRQ_GSTAT[n].LWERR bit indicates (when set) there was an attempted write to an MEC register while the MEC_PEIRQ_GCTL[n].LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK bit =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
3:0 (R/NW)	VALUE	Parity Error Global Status. The MEC_PEIRQ_GSTAT[n].VALUE field indicates the parity error global status. '1' indicates assertion of interrupt/trigger, '0' indicates no interrupt/trigger.	

Parity Error Control Register

MEC_PERR_CTL0 register bits control enable/disable for parity error inputs from various cores/peripherals and decide whether their status will be reflected in parity error status register bits.

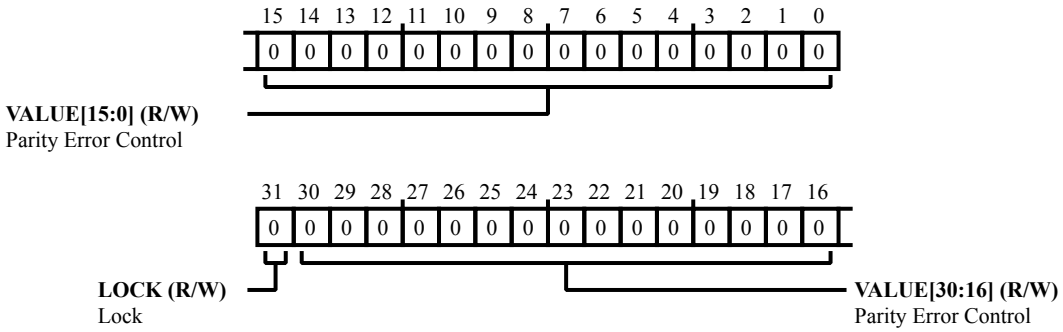


Figure 50-22: MEC_PERR_CTL0 Register Diagram

Table 50-26: MEC_PERR_CTL0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PERR_CTL0.LOCK bit is enabled, the MEC_PERR_CTL0 register is read only.	
		0	Unlock
		1	Lock
30:0 (R/W)	VALUE	Parity Error Control. 1 indicates enable and 0 indicates disable.	
		4	Core 1 L1 RAM Parity Error
		8	Core 1 L1 Cache Parity Error
		16	Core 1 Branch Predictor Parity Error
		32	Core 2 L1 RAM Parity Error
		64	Core 2 L1 Cache Parity Error
		128	Core 2 Branch Predictor Parity Error
		256	ASRC0 FIFO Parity Error
		512	ASRC1 FIFO Parity Error
		1024	ASRC2 FIFO Parity Error
		2048	ASRC3 FIFO Parity Error
		4096	ASRC4 FIFO Parity Error

Table 50-26: MEC_PERR_CTL0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		8192	ASRC5 FIFO Parity Error
		16384	ASRC6 FIFO Parity Error
		32768	ASRC7 FIFO Parity Error
		65536	SHARC 0 IIR0 RAM Parity Error
		131072	SHARC 0 IIR1 RAM Parity Error
		262144	SHARC 0 IIR2 RAM Parity Error
		524288	SHARC 0 IIR3 RAM Parity Error
		1048576	SHARC 0 FIR0 RAM Parity Error
		2097152	SHARC 1 IIR0 RAM Parity Error
		4194304	SHARC 1 IIR1 RAM Parity Error
		8388608	SHARC 1 IIR2 RAM Parity Error
		16777216	SHARC 1 IIR3 RAM Parity Error
		33554432	SHARC 1 FIR0 RAM Parity Error
		67108864	USB0 FIFO RAM Parity Error
		134217728	TRNG0 Data Buffer Parity Error
		268435456	PKA0 Data RAM Parity Error
		536870912	SPE0 Buffer RAM Parity Error
		1073741824	SPE0 ARC4 RAM Parity Error

Parity Error Control Register

MEC_PERR_CTL1 register bits control enable/disable for parity error inputs from various cores/peripherals and decide whether their status will be reflected in parity error status register bits.

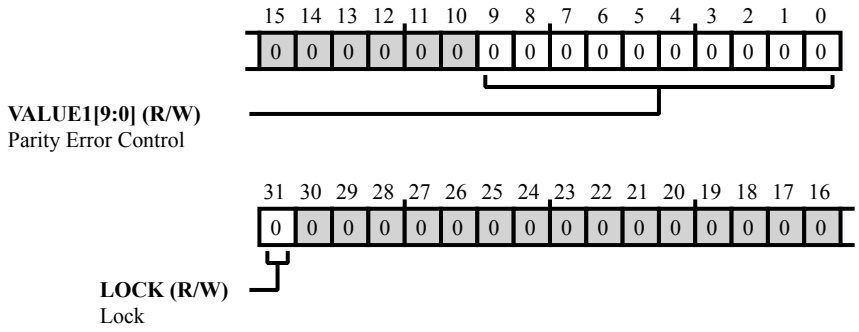


Figure 50-23: MEC_PERR_CTL1 Register Diagram

Table 50-27: MEC_PERR_CTL1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PERR_CTL1.LOCK bit is enabled, the MEC_PERR_CTL1 register is read only.	
		0	Unlock
		1	Lock
9:0 (R/W)	VALUE1	Parity Error Control. The MEC_PERR_CTL1.VALUE1 field indicates the parity error inputs. 1 indicates enable and 0 indicates disable.	
		1	EMAC0 Transmit FIFO RAM Parity Error
		2	EMAC0 Receive FIFO RAM Parity Error
		4	MLB0 Data Buffer RAM Parity Error
		8	MLB0 Channel Table RAM Parity Error
		16	TMC0 Trace Data RAM Parity Error
		32	EMAC1 Transmit FIFO RAM Parity Error
		64	EMAC1 Receive FIFO RAM Parity Error
		128	EMAC0 TMI FIFO RAM Parity Error
		256	EMAC0 EST FIFO RAM Parity Error
		512	EMSI0 FIFO RAM Parity Error

Parity Error Interrupt Mask Register

MEC_PERR_IMASK0 register bits control interrupt masks for parity error inputs from various cores/peripherals.

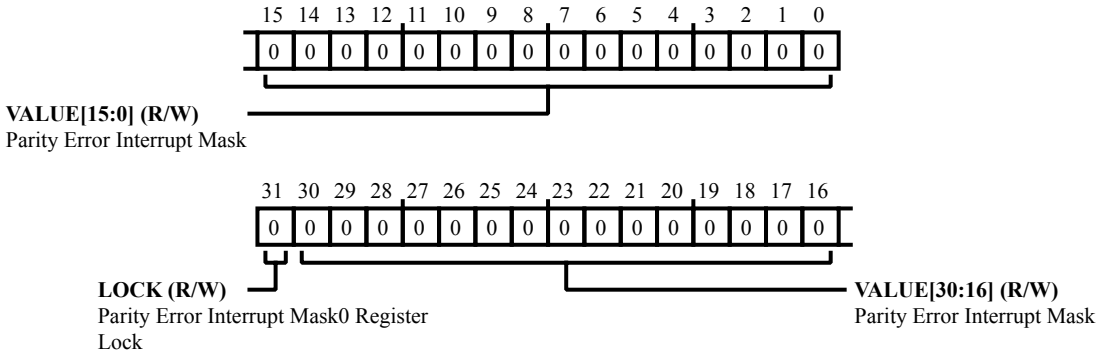


Figure 50-24: MEC_PERR_IMASK0 Register Diagram

Table 50-28: MEC_PERR_IMASK0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Parity Error Interrupt Mask0 Register Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PERR_IMASK0.LOCK bit is enabled, the MEC_PERR_IMASK0 register is read only.	
		0	Unlock
		1	Lock
30:0 (R/W)	VALUE	Parity Error Interrupt Mask. '1' indicates interrupt masked and '0' indicates interrupt unmasked.	
		4	Core 1 L1 RAM Parity Error
		8	Core 1 L1 Cache Parity Error
		16	Core 1 Branch Predictor Parity Error
		32	Core 2 L1 RAM Parity Error
		64	Core 2 L1 Cache Parity Error
		128	Core 2 Branch Predictor Parity Error
		256	ASRC0 FIFO Parity Error
		512	ASRC1 FIFO Parity Error
		1024	ASRC2 FIFO Parity Error
		2048	ASRC3 FIFO Parity Error
		4096	ASRC4 FIFO Parity Error

Table 50-28: MEC_PERR_IMASK0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		8192	ASRC5 FIFO Parity Error
		16384	ASRC6 FIFO Parity Error
		32768	ASRC7 FIFO Parity Error
		65536	SHARC 0 IIR0 RAM Parity Error
		131072	SHARC 0 IIR1 RAM Parity Error
		262144	SHARC 0 IIR2 RAM Parity Error
		524288	SHARC 0 IIR3 RAM Parity Error
		1048576	SHARC 0 FIR0 RAM Parity Error
		2097152	SHARC 1 IIR0 RAM Parity Error
		4194304	SHARC 1 IIR1 RAM Parity Error
		8388608	SHARC 1 IIR2 RAM Parity Error
		16777216	SHARC 1 IIR3 RAM Parity Error
		33554432	SHARC1 FIR0 RAM Parity Error
		67108864	USB0 FIFO RAM Parity Error
		134217728	TRNG0 Data Buffer Parity Error
		268435456	PKA0 Data RAM Parity Error
		536870912	SPE0 Buffer RAM Parity Error
		1073741824	SPE0 ARC4 RAM Parity Error

Parity Error Interrupt Mask Register

MEC_PERR_IMASK1 register bits control interrupt masks for parity error inputs from various cores/peripherals.

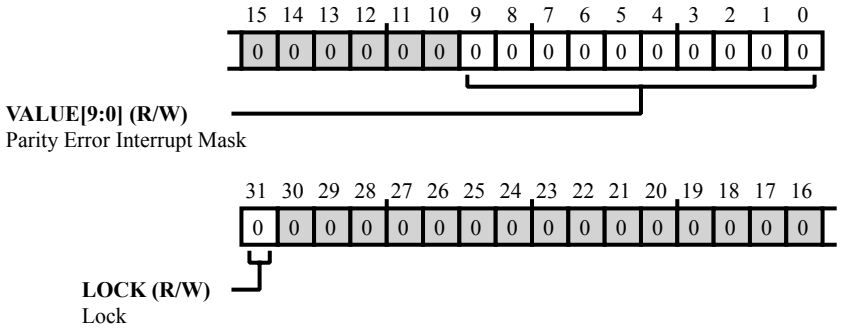


Figure 50-25: MEC_PERR_IMASK1 Register Diagram

Table 50-29: MEC_PERR_IMASK1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W)	LOCK	Lock. If the global lock is enabled (SPU_CTL.GLCK =1) and the MEC_PERR_IMASK1.LOCK bit is enabled, the MEC_PERR_IMASK1 register is read only.	
		0	Unlock
		1	Lock
9:0 (R/W)	VALUE	Parity Error Interrupt Mask. '1' indicates interrupt masked and '0' indicates interrupt unmasked.	
		1	EMAC0 Transmit FIFO RAM Parity Error
		2	EMAC0 Receive FIFO RAM Parity Error
		4	MLB0 Data Buffer RAM Parity Error
		8	MLB0 Channel Table RAM Parity Error
		16	TMC0 Trace Data RAM Parity Error
		32	EMAC1 Transmit FIFO RAM Parity Error
		64	EMAC1 Receive FIFO RAM Parity Error
		128	EMAC0 TMI FIFO RAM Parity Error
		256	EMAC0 EST FIFO RAM Parity Error
		512	EMSI0 FIFO RAM Parity Error

Parity Error Status Register

MEC_PERR_STAT0 register bits reflect status for parity error inputs from various cores/peripherals. Writing '1' to these bits clear corresponding parity error status.

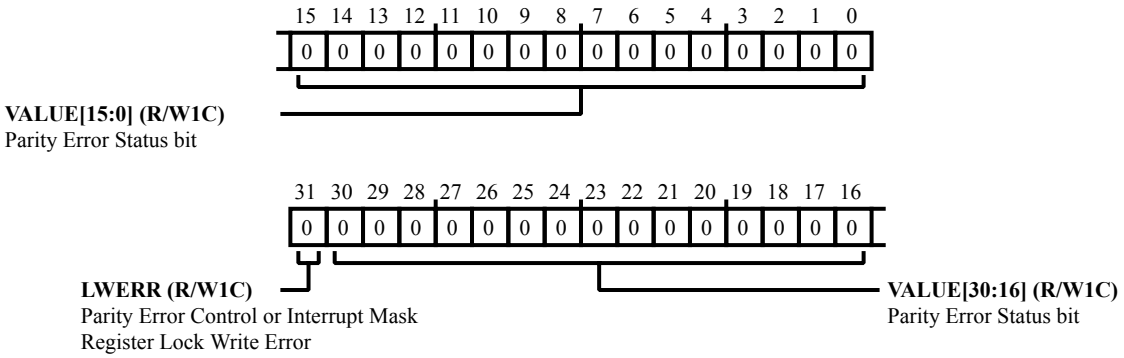


Figure 50-26: MEC_PERR_STAT0 Register Diagram

Table 50-30: MEC_PERR_STAT0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Parity Error Control or Interrupt Mask Register Lock Write Error. The MEC_PERR_STAT0.LWERR bit indicates (when set) there was an attempted write to an MEC register while the MEC_PERR_CTL0.LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
30:0 (R/W1C)	VALUE	Parity Error Status bit. '1' indicates error and '0' indicates no error. Sticky bit, write '1' to clear.	
		4	Core 1 L1 RAM Parity Error
		8	Core 1 L1 Cache Parity Error
		16	Core 1 Branch Predictor Parity Error
		32	Core 2 L1 RAM Parity Error
		64	Core 2 L1 Cache Parity Error
		128	Core 2 Branch Predictor Parity Error
		256	ASRC0 FIFO Parity Error
		512	ASRC1 FIFO Parity Error
		1024	ASRC2 FIFO Parity Error
		2048	ASRC3 FIFO Parity Error

Table 50-30: MEC_PERR_STAT0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		4096	ASRC4 FIFO Parity Error
		8192	ASRC5 FIFO Parity Error
		16384	ASRC6 FIFO Parity Error
		32768	ASRC7 FIFO Parity Error
		65536	SHARC 0 IIR0 RAM Parity Error
		131072	SHARC 0 IIR1 RAM Parity Error
		262144	SHARC 0 IIR2 RAM Parity Error
		524288	SHARC 0 IIR3 RAM Parity Error
		1048576	SHARC 0 FIR0 RAM Parity Error
		2097152	SHARC 1 IIR0 RAM Parity Error
		4194304	SHARC 1 IIR1 RAM Parity Error
		8388608	SHARC 1 IIR2 RAM Parity Error
		16777216	SHARC 1 IIR3 RAM Parity Error
		33554432	SHARC 1 FIR0 RAM Parity Error
		67108864	USB0 FIFO RAM Parity Error
		134217728	TRNG0 Data Buffer Parity Error
		268435456	PKA0 Data RAM Parity Error
		536870912	SPE0 Buffer RAM Parity Error
		1073741824	SPE0 ARC4 RAM Parity Error

Parity Error Status Register

MEC_PERR_STAT1 register bits reflect status for parity error inputs from various cores/peripherals. Writing '1' to these bits clear corresponding parity error status.

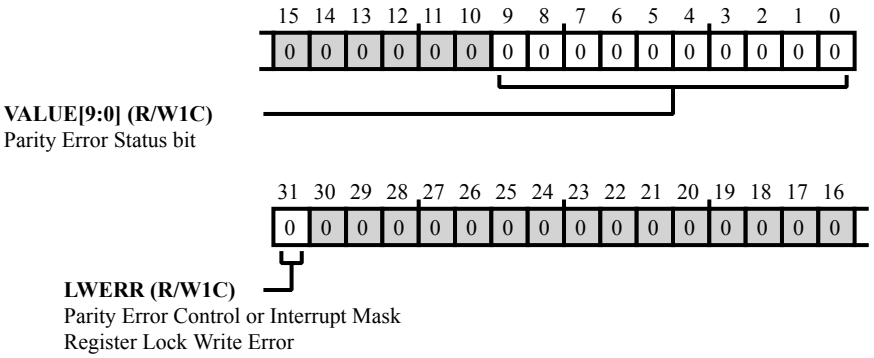


Figure 50-27: MEC_PERR_STAT1 Register Diagram

Table 50-31: MEC_PERR_STAT1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31 (R/W1C)	LWERR	Parity Error Control or Interrupt Mask Register Lock Write Error. The MEC_PERR_STAT1.LWERR bit indicates (when set) there was an attempted write to an MEC register while the MEC_PERR_CTL1.LOCK bit was set and while the global lock bit was enabled (SPU_CTL.GLCK =1). This status bit is sticky; write-1-to-clear it.	
		0	No Error
		1	Error Occurred
9:0 (R/W1C)	VALUE	Parity Error Status bit. 1 indicates error and 0 indicates no error. Sticky bit, write 1 to clear.	
		1	EMAC0 Transmit FIFO RAM Parity Error
		2	EMAC0 Receive FIFO RAM Parity Error
		4	MLB0 Data Buffer RAM Parity Error
		8	MLB0 Channel Table RAM Parity Error
		16	TMC0 Trace Data RAM Parity Error
		32	EMAC1 Transmit FIFO RAM Parity Error
		64	EMAC1 Receive FIFO RAM Parity Error
		128	EMAC0 TMI FIFO RAM Parity Error
		256	EMAC0 EST FIFO RAM Parity Error
		512	EMSI0 FIFO RAM Parity Error

Peripheral ID0 Register

The MEC_PID0 register is a peripheral ID register.

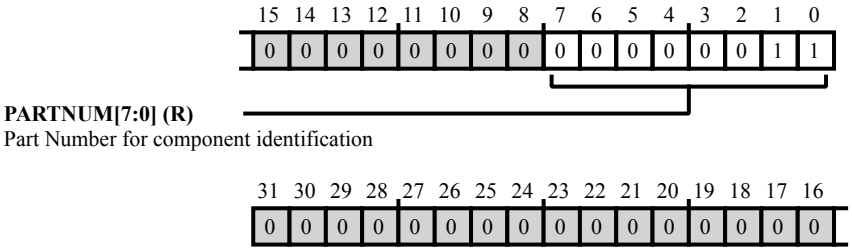


Figure 50-28: MEC_PID0 Register Diagram

Table 50-32: MEC_PID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PARTNUM	Part Number for component identification.

Peripheral ID1 Register

The MEC_PID1 register is a peripheral ID register.

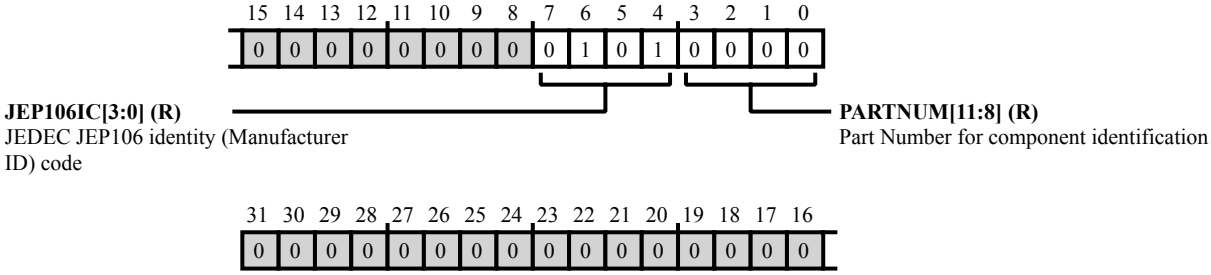


Figure 50-29: MEC_PID1 Register Diagram

Table 50-33: MEC_PID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	JEP106IC	JEDEC JEP106 identity (Manufacturer ID) code.
3:0 (R/NW)	PARTNUM	Part Number for component identification.

Peripheral ID2 Register

The MEC_PID2 register is a peripheral ID register.

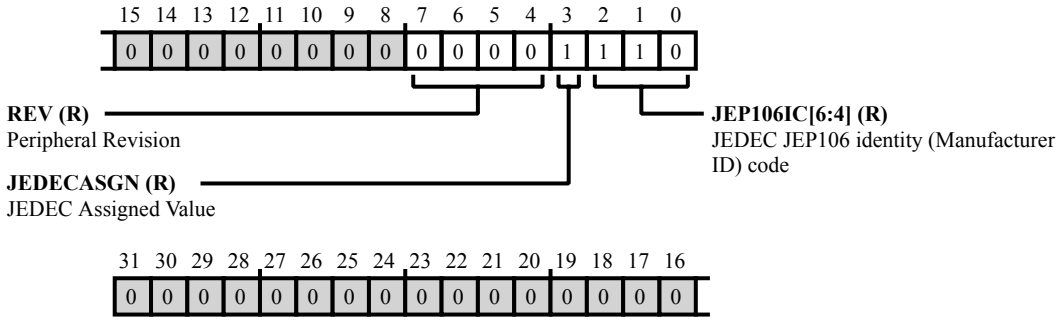


Figure 50-30: MEC_PID2 Register Diagram

Table 50-34: MEC_PID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REV	Peripheral Revision.
3 (R/NW)	JEDECASGN	JEDEC Assigned Value.
2:0 (R/NW)	JEP106IC	JEDEC JEP106 identity (Manufacturer ID) code.

Peripheral ID3 Register

The MEC_PID3 register is a peripheral ID register.

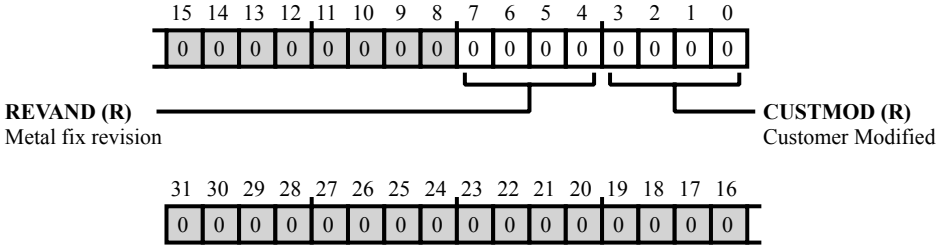


Figure 50-31: MEC_PID3 Register Diagram

Table 50-35: MEC_PID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REVAND	Metal fix revision.
3:0 (R/NW)	CUSTMOD	Customer Modified.

Peripheral ID4 Register

The MEC_PID4 register is a peripheral ID register.

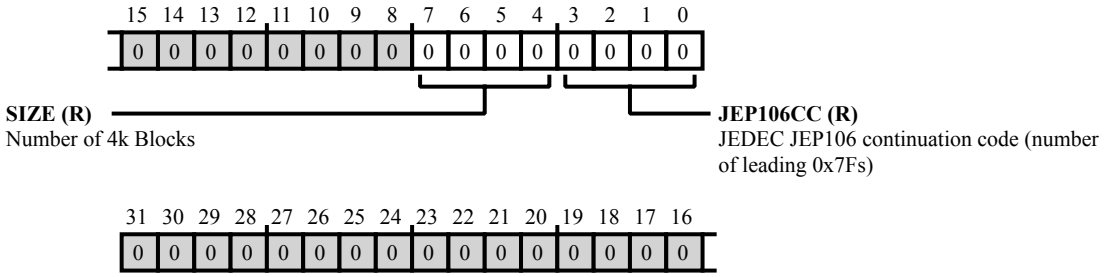


Figure 50-32: MEC_PID4 Register Diagram

Table 50-36: MEC_PID4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	SIZE	Number of 4k Blocks. Size of component 4k chunks minus 1 (i.e. 0=4k)
3:0 (R/NW)	JEP106CC	JEDEC JEP106 continuation code (number of leading 0x7Fs).

51 System Debug and Trace Unit (DBG)

The system debug and trace unit is based on Arm Core Sight technology. CoreSight is a set of architecture specifications defining debug and trace architecture. The processor uses CoreSight infrastructure to provide industry standard debug and trace capabilities.

http://infocenter.arm.com/help/

The applicable documentation for more details about the ArmCoreSight feature includes:

- CoreSight PFT Architecture Specification, ARM IHI 0035B (PFT)
- System Trace Macrocell, Programmers' Model Architecture Specification, ARM IHI 0054A (STM)
- CoreSight Trace Memory Controller, ARM DDI0461B (TMC)
- CoreSight Components Technical Reference Manual, ARM DDI 0314H (TPIU)
- Embedded Cross Trigger Technical Reference Manual, ARM DDI 0291A
- ETM Architecture Specification, ARM IHI0014Q

DBG Features

The debug unit has the following features.

- System JTAG TAP controller for system debug features, boundary scan, and public JTAG features
- A debug interface to core, and other system resources
- Direct and run-time access to the memory system and system MMRs
- Direct control over system reset
- Support for debug immediately after reset (boot debug)
- Group halt (debug event immediately halts all specified endpoints)
- Real-time on-chip visibility is made available to all developers, including software developers

Debug Functional Description

The following sections provide functional descriptions of the DBG unit.

ADSP-SC59x CSPFT Register List

The CoreSight (TM) Program Flow Trace unit (CSPFT) provides debug features. A set of registers governs CSPFT operations. For more information on CSPFT functionality, see the CSPFT register descriptions.

Table 51-1: ADSP-SC59x CSPFT Register List

Name	Description
CSPFT_ACTR[n]	Address Comparator Access Type Register
CSPFT_ACVR[n]	Address Comparator Value Register
CSPFT_AUTHSTATUS	Authentication Status Register
CSPFT_CCER	Configuration Code Extension Register
CSPFT_CID0	Component ID0 Register
CSPFT_CID1	Component ID1 Register
CSPFT_CID2	Component ID2 Register
CSPFT_CID3	Component ID3 Register
CSPFT_CIDCMR	Context ID Comparator Mask Register
CSPFT_CIDCVR[n]	Context ID Comparator Value
CSPFT_CLAIMCLR	Claim Tag Clear Register
CSPFT_CLAIMSET	Claim Tag Set Register
CSPFT_CNTENR[n]	Counter Enable Event Register
CSPFT_CNTRLDEVR[n]	Counter Reload Event Register
CSPFT_CNTRLDVR[n]	Counter Reload Value Register
CSPFT_CNTVR[n]	Counter Value Register
CSPFT_CTL	Main Control Register
CSPFT_DEVTYPE	Device Type Identifier Register
CSPFT_EXTOUTEVR[n]	External Output Event Register
CSPFT_HWFEAT	Hardware Feature Register
CSPFT_LAR	Lock Access Register
CSPFT_LSR	Lock Status Register
CSPFT_PID0	Peripheral ID0 Register
CSPFT_PID1	Peripheral ID1 Register
CSPFT_PID2	Peripheral ID2 Register

Table 51-1: ADSP-SC59x CSPFT Register List (Continued)

Name	Description
CSPFT_PID3	Peripheral ID3 Register
CSPFT_PID4	Peripheral ID4 Register
CSPFT_STAT	Status Register
CSPFT_SYNCFR	Synchronization Frequency Register
CSPFT_TRACEIDR	CoreSight Trace ID Register

ADSP-SC59x TAPC Register List

The Test Access Port Controller (TAPC) provides access to debug features. A set of registers governs TAPC operations. For more information on TAPC functionality, see the TAPC register descriptions.

Table 51-2: ADSP-SC59x TAPC Register List

Name	Description
TAPC_DBGCTL	Debug Control Register
TAPC_IDCODE	IDCODE Register
TAPC_SDBGKEY0	Secure Debug Key 0 Register
TAPC_SDBGKEY1	Secure Debug Key 1 Register
TAPC_SDBGKEY2	Secure Debug Key 2 Register
TAPC_SDBGKEY3	Secure Debug Key 3 Register
TAPC_SDBGKEY_CTL	Secure Debug Key Control Register
TAPC_SDBGKEY_STAT	Secure Debug Key Status Register
TAPC_USERCODE	USERCODE Register

DBG Block Diagram

The DBG block diagram is shown below.

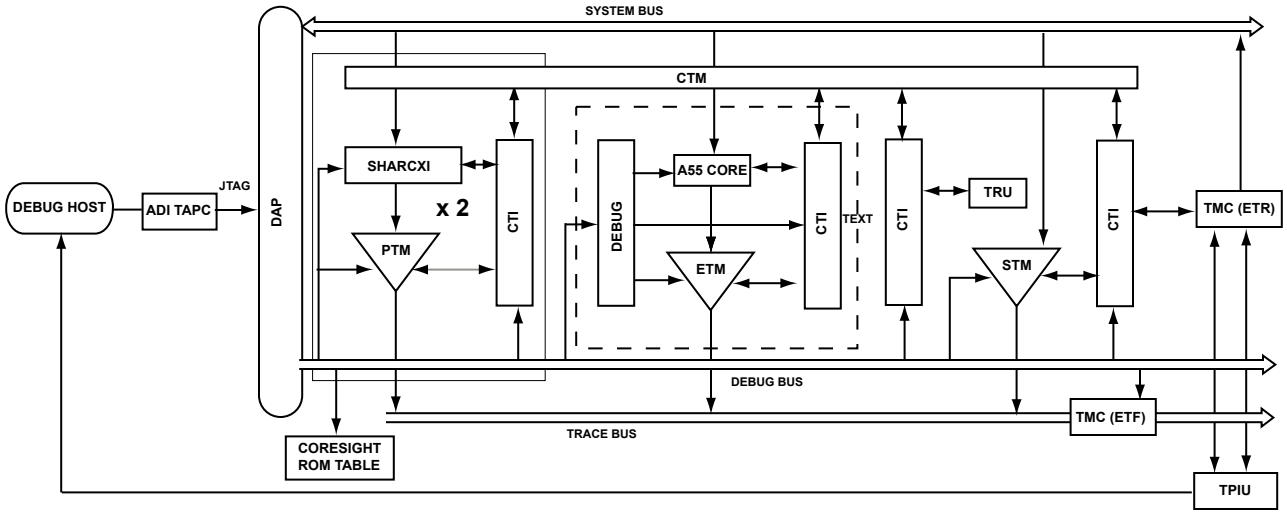


Figure 51-1: Coresight in ADSP-SC598 Family

DBG Definitions

The following terms are useful when working with the debug features of the processor and programming tools.

Test Access Port Controller (TAPC)

Provides IDCODE and SDBGKEY features.

Debug Access Port (DAP)

Core Sight Interface providing a single port for two debug options: JTAG-DP (JTAG Debug Port), SW-DP (Serial Wire Debug Port)

Program Trace Macrocell (PTM)

Provides DSP () Core Trace.

Standard Trace Macrocell (STM)

Provides capability to trace up to 32 hardware events and supports 32 software stimulus for data transfer between the user and emulator.

Embedded Cross Trigger (ECT)

The ECTs are responsible managing events and triggers as follows.

- CTI (Cross Trigger Interface) is a CoreSight component for enabling cross triggering of events across a system. From the view of the ECT, it is responsible for combining and mapping trigger requests.
- CTM (Cross Trigger Matrix) is a CoreSight component for connecting multiple CTIs. From the view of the ECT, it is responsible for connecting CTIs and distribution of events.

NOTE: An embedded cross trigger is not the same as the requester and completer trigger in the trigger routing unit.

Trace Capture Devices

The trace capture devices capture and format the trace data. There are two trace capture devices in the system:

- ETF - Embedded Trace FIFO - Provides a buffer for burst trace data.
- ETR - Embedded Trace Router - Provides interface for trace data to be stored in system memories.

Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the on-chip trace data, with separate IDs, to a data stream. It encapsulates IDs, when required, that the Trace Port Analyzer (TPA) captures.

Test Access Port Controller (TAPC)

TAPC is an independent component daisy-chained to the DAP in the JTAG scan path. The TAPC component provides the product IDCODE (Chip ID) and a security feature.

The TAPC component provides security features to the chip using a debug key match features. This feature permits only those components which have the SDBGKEY to connect and debug the chip. Initially, a 128-bits user key is programmed in the SDBGKEY registers (`TAPC_SDBGKEY0`, `TAPC_SDBGKEY1`, `TAPC_SDBGKEY2`, `TAPC_SDBGKEY3`) in TAPC by a secure requester. The `TAPC_SDBGKEY_CTL.VALID` bit is set.

Then, a user key is entered through the emulator for a match to the initially programmed-entered user key. On a successful match, the chip connects to the emulator. It can be debugged on a failed user key match. Further attempts at keys matching are disabled. JTAG reset or system reset is required to reenable the user key match logic.

Embedded Trace Macrocell (ETM)

ETM is the standard trace support provided for the Arm processor. For details of programming and functionality, refer to the Arm documentation.

Debug Access Ports

The Debug Access Port (DAP) is an implementation of an Arm Debug Interface version 6 (ADIv6) comprising a number of components supplied in a single configuration. All the supplied components fit into the various architectural components for Debug Ports (DPs). The components are used to access the DAP from an external debugger and APs, to access on-chip system resources.

Trace Unit

The trace module provides instruction, data tracing, and system activity tracing for the processor. The program trace module on the processor is similar to the embedded trace macrocell module provided by the Arm processor. System trace is provided using the system trace macrocell as part of the CoreSight debug and trace interface. The trace module uses an interface based on the AMBA Trace Bus (ATB) standard to output its trace data. The trace data can be

either exported to an off-chip trace port analyzer or captured on an on-chip buffer. The PFT and STM modules capture information on the processor both before and after a specific event. The modules add no burden to the processor performance when it runs at full speed.

- Programmable Flow Trace (CSPFT)
- System Trace Module (STM)

Programmable Flow Trace (CSPFT)

When tracing processor execution, trace information can be generated for every instruction the processor executes. This information would be easy to interpret, but would require a prohibitively high trace bandwidth to get the trace data off the chip. With program flow tracing, only branch points are traced. The debugger uses the source code to infer the rest of the executed code.

Certain instructions in the program and events are identified as waypoints. A waypoint is a point where instruction execution involves a change of program flow. The CSPFT only traces those waypoints. These waypoints are:

- All indirect branches
- All direct branches
- Exceptions or interrupts
- Emulator debug entry and exit

When a waypoint occurs, trace data is generated to describe it. From this data and the source code, a trace decompressor can determine what instructions were executed and recreate the instruction flow. To allow the decompressor to calculate where it is in the source code, conditional instructions are marked as waypoints, regardless of whether they pass or fail their condition test. Events like interrupts or debug entry and exit can be promoted from non-waypoint instructions to waypoints to trace the interrupted program flow.

Tracing a waypoint implies the execution of all instructions from the target address of the previous waypoint up to the current waypoint. Non-waypoint instructions are not explicitly traced but the debugger must infer them using the source code. The concept of an instruction block is used throughout this manual and refers to the contiguous block of instructions between two waypoints.

The programming model and function is a subset of the PTM (Programmable Trace Module) of Arm.

System Trace Module (STM)

The STM is a trace source that is integrated into a CoreSight system, and is designed primarily for high-bandwidth trace of instrumentation embedded into software. The STM enables tracing of system activity from various sources:

- Instrumented software, using memory-mapped stimulus ports
- Hardware events

The STM supports the following features:

- Multiple software requesters writing software instrumentation independently. Each requester can use multiple stimulus ports.

- Time stamping of the system activity. The time stamp is a global time stamp which can be shared with other trace sources in the system to enable correlation of activity from multiple trace sources.
- Indication that specific events have occurred, such as a particular hardware event or a piece of software instrumentation. These events are known as triggers and can be indicated in the trace stream, or through signals to other system components.

Embedded Cross Trigger (ECT)

ECT provides an interface to the CoreSight debug system enabling the subsystems to interact (cross trigger) with each other. ECT provides a mechanism to forward debug events from one connected subsystem to another connected subsystem. The different subsystems connected to the ECT depend on the processor design. For example, in a multiprocessor system, the interface can be connected to each of the cores and one to the trace subsystem. For a uniprocessor system, the interface can include just the core and trace subsystem connection.

- CTI cross trigger interface. A CoreSight component for enabling cross triggering of events across a system.
- CTM cross trigger matrix. A CoreSight component for connecting multiple cross trigger interfaces.

The main function of the ECT (CTI and CTM) is to pass debug events from one connected subsystem to another connected subsystem.

For example, the ECT can communicate debug state information from the core to trace subsystem for a single processor system or to another core in a multiprocessor-based system. Program execution on both the subsystem can be stopped at the same time.

The ***Trigger Flow*** figure shows a simple debug trigger flow sequence. On each CTI, there are four channel, eight input, and eight output debug triggers. All the eight inputs and outputs can be mapped to a single channel or different channels based on the debug trigger to channel mapping. When a trigger input occurs, it creates a channel event. The channel event causes all the output debug triggers to be triggered. The embedded cross trigger depends on the debug trigger it connects to.

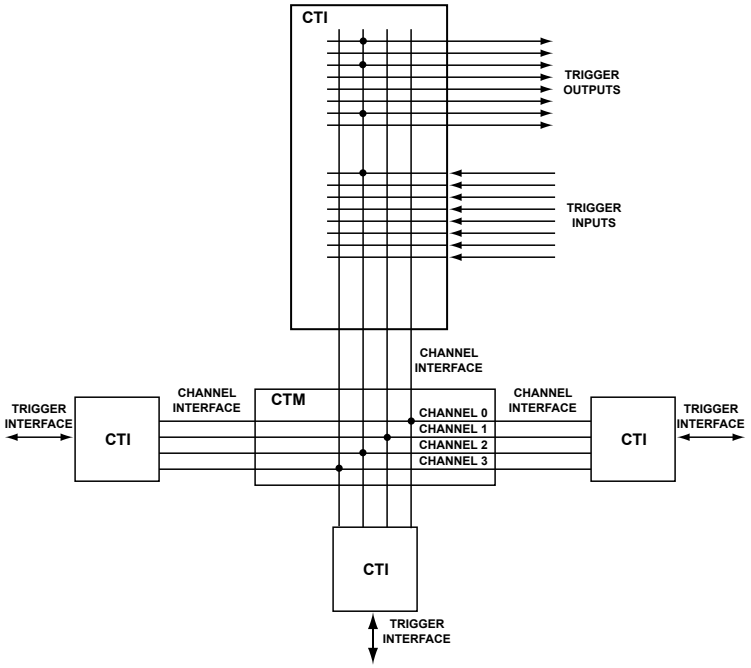


Figure 51-2: Trigger Flow

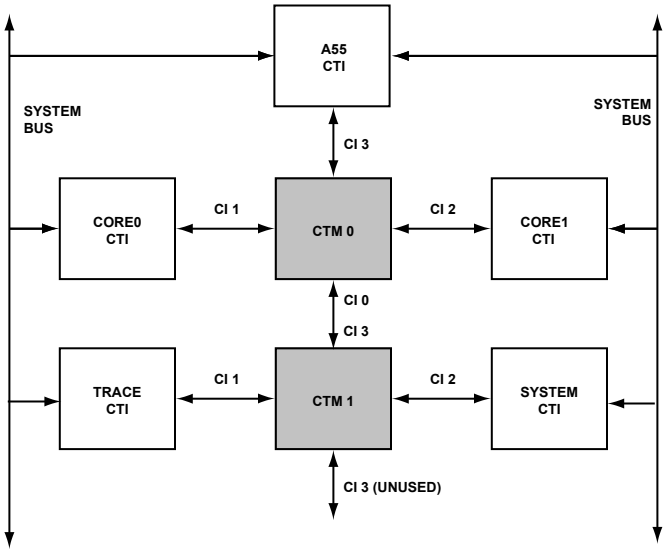


Figure 51-3: ECT Integration

CTI Debug Trigger Tables

The ***System CTI Trigger Connection*** tables show the debug trigger that connects to each CTI.

Table 51-3: System CTI Trigger Connection

CTI Port	Direction	System Signal	CTI Signal Description
EVENTIN[7:3]	Input	From TRU	Trigger In
EVENTIN[2]	Input	$\overline{\text{RCU_SYSRSTB}}$	
EVENTIN[1:0]	Input	From TRU	
EVENTOUT[13]	Output	To A55 core sample request input	Trigger Out
EVENTOUT[12]	Output	To A55 core PMU snapshot request input	
EVENTOUT[11]	Output	To DAP DP event status input	
EVENTOUT[10]	Output	To system time tsgen restart input	
EVENTOUT[9]	Output	To system time tsgen halt input	
EVENTOUT[8]	Output	To Coresight tsgen restart input	
EVENTOUT[7]	Output	To TRU and as DBGRESTART to CTITRIGOU-TACK[1] and CTITRIGOUTACK[0]	
EVENTOUT[6:3]	Output	To TRU	
EVENTOUT[2]	Output	To RCU as system reset request	
EVENTOUT[1]	Output	To TRU and system peripheral halt	
EVENTOUT[0]	Output	To TRU and system fabric halt	

Table 51-4: ADSP-SC598-Core 1/2 CTI Trace Connection

CTI Port	Direction	SHARC-XI Signal	CTI Signal Description
CTITRIGIN[7]	Input	Tied low (Unused)	Trigger in
CTITRIGIN[6]	Input	PTM TRIGGER	
CTITRIGIN[5:2]	Input	PTM EXTOUT[3:0	
CTITRIGIN[1]	Input	Tied low (Unused)	
CTITRIGIN[0]	Input	DBGTRIGGER	
CTITRIGINACK[7]	Output	Unused	Trigger in acknowledge
CTITRIGINACK[6]	Output	PTM TRIGGER ACK	
CTITRIGINACK[5:1]	Output	Unused	
CTITRIGINACK[0]	Output	DBGTRIGGER ACK	
CTITRIGOUT[7]	Output	DBGRESTART	Trigger out
CTITRIGOUT[6]	Output	SEC	
CTITRIGOUT[5]	Output	Unused	
CTITRIGOUT[4:1]	Output	PTM EXTIN[3:0]	
CTITRIGOUT[0]	Output	EDBGRQ	

Table 51-4: ADSP-SC598-Core 1/2 CTI Trace Connection (Continued)

CTI Port	Direction	SHARC-XI Signal	CTI Signal Description
CTITRIGOUTACK[7]	Input	DBGRESTARTED	Trigger out acknowledge
CTITRIGOUTACK[6:0]	Input	Tied Low	

Table 51-5: ADSP-SC598 CTI Trace Connection

CTI Port	Direction	Trace Block Signal	CTI Signal Description
EVENTIN[10]	Input	STM ASYNCOUT	Trigger in
EVENTIN[9]	Input	STM TRIGOUTHETE	
EVENTIN[8]	Input	STM TRIGOUTSW	
EVENTIN[7]	Input	STM TRIGOUTSPTE	
EVENTIN[6]	Input	TPIU FLUSHCOMP	
EVENTIN[5]	Input	ETR FLUSHCOMP	
EVENTIN[4]	Input	ETR ACQCOMP	
EVENTIN[3]	Input	ETR FULL	
EVENTIN[2]	Input	ETF FLUSHCOMP	
EVENTIN[1]	Input	ETF ACQCOMP	
EVENTIN[0]	Input	ETF FULL	
EVENTOUT[7:6]	Output	trigout_cti_to_stm[1:0]	Trigger out
EVENTOUT[5]	Output	TPIU FLUSHIN	
EVENTOUT[4]	Output	TPIU TRIGIN	
EVENTOUT[3]	Output	ETR FLUSHIN	
EVENTOUT[2]	Output	ETR TRIGIN	
EVENTOUT[1]	Output	ETF FLUSHIN	
EVENTOUT[0]	Output	ETF TRIGIN	

Table 51-6: SC59x Arm Core CTI Connection

CTI Port	Input	CTI Port	Output
CTITRIGIN[7:4]	EXT_EXTOUT[3:0]	CTITRIGOUT[7:4]	ETM EXTIN[3:0]
CTITRIGIN[3]	Unused	CTITRIGOUT[3]	Unused
CTITRIGIN[2]	Unused	CTITRIGOUT[2]	GIC Interrupt Request
CTITRIGIN[1]	PMU IRQ	CTITRIGOUT[1]	DBGRESTART
CTITRIGIN[0]	DBGTRIGGER	CTITRIGOUT[0]	EDBGRQ

Table 51-7: Trigger Descriptions

Signal Name	Description
STM ASYNCOUT	Alignment synchronization output. This signal is asserted for one clock cycle when an ASYNC-VERSION-FREQ sequence is completely output on the ATB, and can be used for cross-triggering.
STM TRIGOUTHETE	Trigger output. This signal is asserted for one clock cycle when a trigger event is detected on match using STMHETER.
STM TRIGOUTSW	Trigger output. This signal is asserted for one clock cycle when a trigger event is generated on writes to a TRIG location in the extended stimulus port registers.
STM TRIGOUTSPTE	Trigger output. This signal is asserted for one clock cycle when a trigger event is detected on match using STMSPTER.
ETR / ETF FULL	This output indicates the value of the full bit in the ETR/ETF status register. A full bit indicates the amount of data in ETF/ETR. An output signal indicating when the circular buffer or FIFO is full, or within a programmable amount of being full.
ETR / ETF ACQCOMP	This output indicates the value of the FtEmpty bit in the ETR/ETF status register. The bit it is set when trace capture has stopped. An output signal indicating when trace capture has stopped, usually following a trigger condition.
ETR/ETF/ TPIU TRIGIN	This input can cause a trigger event (Start /Stop Trigger). An input signal indicating when a trigger condition has occurred.
ETR/ETF /TPIU FLUSHIN	This input can cause a Trace flush. An input signal indicating a flush request
PTM/ETM TRIGGER	Trigger Input. Trigger event specifies the conditions that must be met to generate a trigger.
PTM/ETM EXTOUT[3:0]	PTM Output.
PTM/ETM EXTIN[3:0]	PTM Input
SEC	CTI Interrupt
DBGRESTART	This is an output from the CTI to a processor core or to system to return from debug mode.
DBGTRIGGER	This is a processor core output signal indicating that the core has moved to debug mode. If the CTIs are setup for synchronous halt, it will generate EDBGRQ to everyone else.
EDBGRQ	This is an output from CTI and an input (as debug halt) to a processor core and to the system in general. It can assert as a result of another core going to emulation space (DBGTRIGGER) or by setting the corresponding bit in the CTI.
SYS_DBGRESTART	This is an output from the CTI to system to return from debug mode.
To TRU	TRU Controller Event
From TRU	TRU Target Event

ADSP-SC59x CSPFT Register Descriptions

Program Flow Trace (CSPFT) contains the following registers.

Table 51-8: ADSP-SC59x CSPFT Register List

Name	Description
CSPFT_ACTR[n]	Address Comparator Access Type Register
CSPFT_ACVR[n]	Address Comparator Value Register
CSPFT_AUTHSTATUS	Authentication Status Register
CSPFT_CCER	Configuration Code Extension Register
CSPFT_CID0	Component ID0 Register
CSPFT_CID1	Component ID1 Register
CSPFT_CID2	Component ID2 Register
CSPFT_CID3	Component ID3 Register
CSPFT_CIDCMR	Context ID Comparator Mask Register
CSPFT_CIDCVR[n]	Context ID Comparator Value
CSPFT_CLAIMCLR	Claim Tag Clear Register
CSPFT_CLAIMSET	Claim Tag Set Register
CSPFT_CNTENR[n]	Counter Enable Event Register
CSPFT_CNTRLDEVR[n]	Counter Reload Event Register
CSPFT_CNTRLDVR[n]	Counter Reload Value Register
CSPFT_CNTVR[n]	Counter Value Register
CSPFT_CTL	Main Control Register
CSPFT_DEVTYPE	Device Type Identifier Register
CSPFT_EXTOUTEVR[n]	External Output Event Register
CSPFT_HWFEAT	Hardware Feature Register
CSPFT_LAR	Lock Access Register
CSPFT_LSR	Lock Status Register
CSPFT_PID0	Peripheral ID0 Register
CSPFT_PID1	Peripheral ID1 Register
CSPFT_PID2	Peripheral ID2 Register
CSPFT_PID3	Peripheral ID3 Register
CSPFT_PID4	Peripheral ID4 Register
CSPFT_STAT	Status Register
CSPFT_SYNCFR	Synchronization Frequency Register

Table 51-8: ADSP-SC59x CSPFT Register List (Continued)

Name	Description
CSPFT_TRACEIDR	CoreSight Trace ID Register

Address Comparator Access Type Register

The `CSPFT_ACTR[n]` register specifies whether the context ID needs to match.

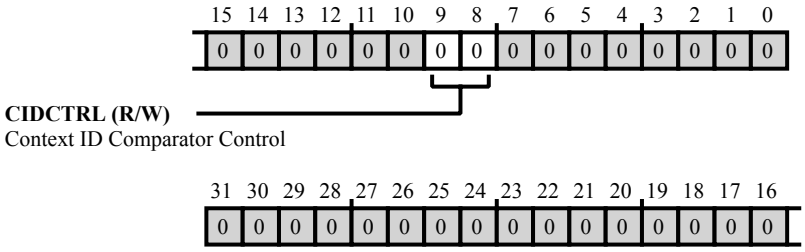


Figure 51-4: CSPFT_ACTR[n] Register Diagram

Table 51-9: CSPFT_ACTR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	CIDCTRL	Context ID Comparator Control. The `CSPFT_ACTR[n].CIDCTRL` contains the ID comparator control value.	
		0	Ignore Context ID
		1	Match if Context ID Comparator 0 Matches
		2	Match if Context ID Comparator 1 Matches
		3	Match if Context ID Comparator 2 Matches

Address Comparator Value Register

The `CSPFT_ACVR[n]` register holds an address for comparison.

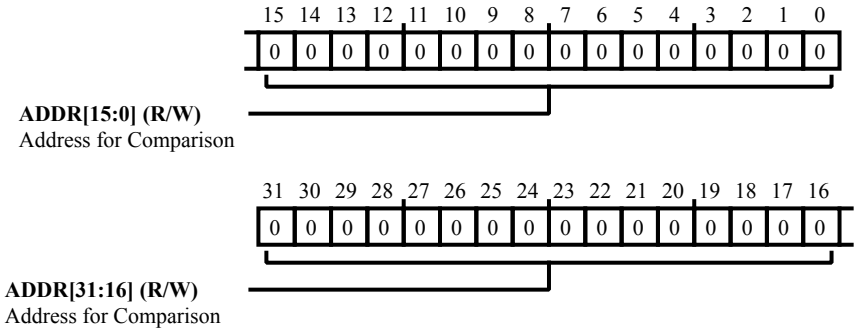


Figure 51-5: CSPFT_ACVR[n] Register Diagram

Table 51-10: CSPFT_ACVR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	ADDR	Address for Comparison. The `CSPFT_ACVR[n].ADDR` bit field contains the address used for comparison.

Authentication Status Register

The CSPFT_AUTHSTATUS register reports the level of tracing currently permitted based on the DBGEN signal.

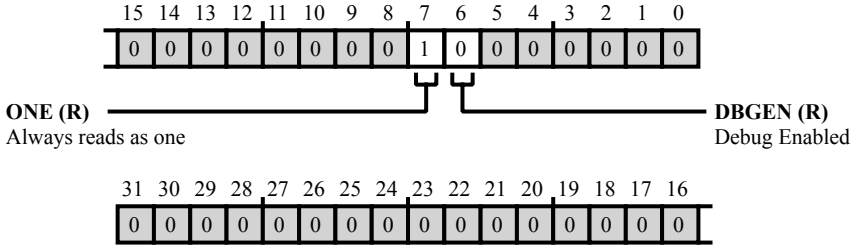


Figure 51-6: CSPFT_AUTHSTATUS Register Diagram

Table 51-11: CSPFT_AUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7 (R/NW)	ONE	Always reads as one.
6 (R/NW)	DBGEN	Debug Enabled. The CSPFT_AUTHSTATUS.DBGEN bit indicates that invasive debug is enabled. Normally, NIDEN is used in conjunction with a signal that enables invasive debug, DBGEN. Non-invasive debug is disabled only if both NIDEN and DBGEN signals are LOW. In a PTM, typically these signals are ORed together and the result is used to determine whether non-invasive debug is enabled.

Configuration Code Extension Register

The CSPFT_CCER register holds extra feature information. (See CSPFT_HWFEAT.)

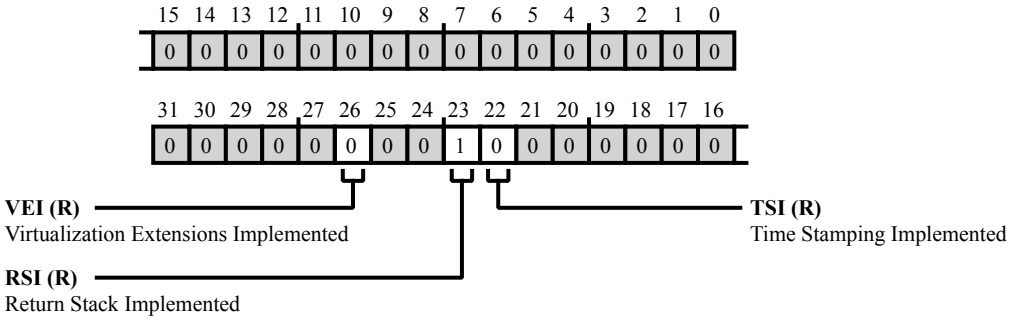


Figure 51-7: CSPFT_CCER Register Diagram

Table 51-12: CSPFT_CCER Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
26 (R/NW)	VEI	Virtualization Extensions Implemented. The CSPFT_CCER.VEI bit indicates if the virtualization extensions are implemented	
		0	Not Implemented
		1	Implemented
23 (R/NW)	RSI	Return Stack Implemented. The CSPFT_CCER.RSI bit indicates if a return stack is implemented.	
		0	Not Implemented
		1	Implemented
22 (R/NW)	TSI	Time Stamping Implemented. The CSPFT_CCER.TSI bit indicates if time stamping is implemented.	
		0	Disabled
		1	Enabled

Component ID0 Register

The `CSPFT_CID0` register holds sections of the CoreSight Component ID for CSPFT.

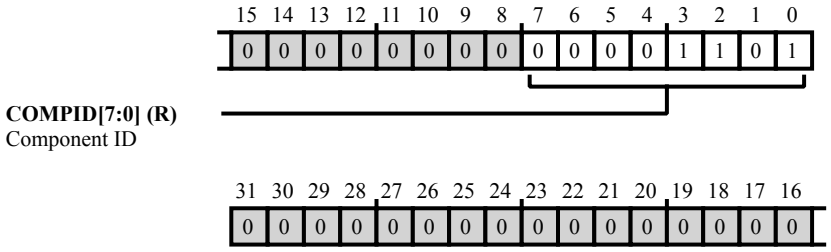


Figure 51-8: CSPFT_CID0 Register Diagram

Table 51-13: CSPFT_CID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMPID	Component ID. The `CSPFT_CID0.COMPID` bit field identifies this component as a CoreSight component.

Component ID1 Register

The CSPFT_CID1 register holds sections of the CoreSight Component ID for CSPFT.

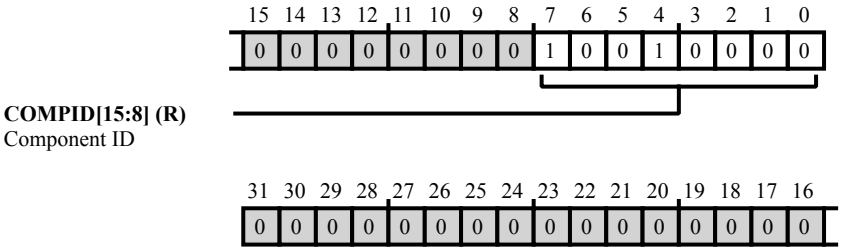


Figure 51-9: CSPFT_CID1 Register Diagram

Table 51-14: CSPFT_CID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMPID	Component ID. The CSPFT_CID1.COMPID bit field identifies this component as a CoreSight component.

Component ID2 Register

The `CSPFT_CID2` register holds sections of the CoreSight Component ID for CSPFT.

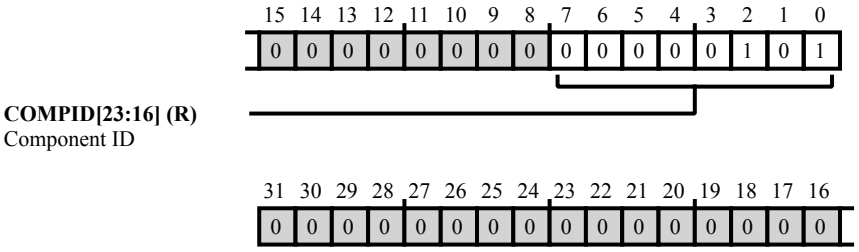


Figure 51-10: CSPFT_CID2 Register Diagram

Table 51-15: CSPFT_CID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMPID	Component ID. The `CSPFT_CID2.COMPID` bit field identifies this component as a CoreSight component.

Component ID3 Register

The CSPFT_CID3 register holds sections of the CoreSight Component ID for CSPFT.

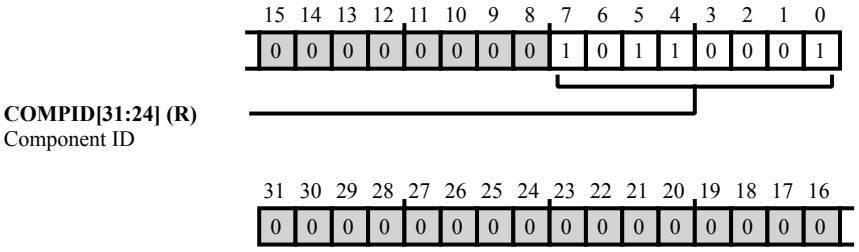


Figure 51-11: CSPFT_CID3 Register Diagram

Table 51-16: CSPFT_CID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	COMPID	Component ID. The CSPFT_CID3.COMPID bit field identifies this component as a CoreSight component.

Context ID Comparator Mask Register

The CSPFT_CIDCMR register holds a 32-bit mask for use for all context ID comparisons.

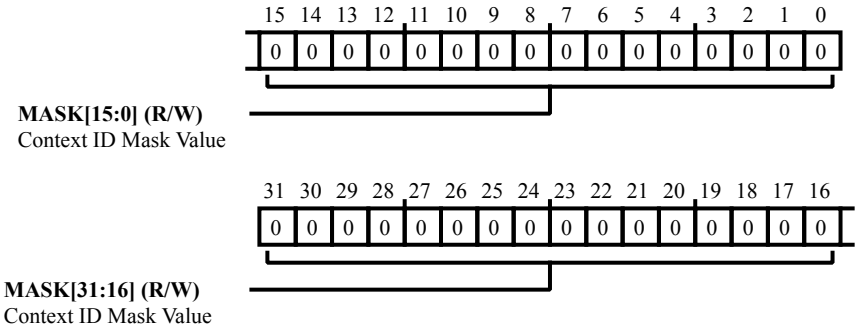


Figure 51-12: CSPFT_CIDCMR Register Diagram

Table 51-17: CSPFT_CIDCMR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	MASK	Context ID Mask Value. The CSPFT_CIDCMR.MASK bit field holds a 32-bit mask for use in all context ID comparisons.

Context ID Comparator Value

The `CSPFT_CIDCVR[n]` register holds a context ID value for comparison.

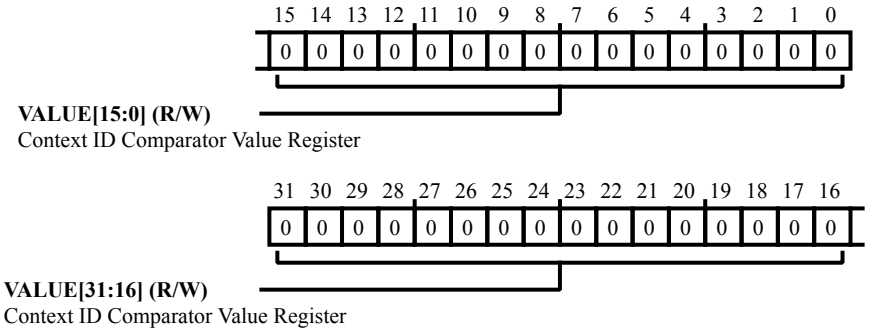


Figure 51-13: CSPFT_CIDCVR[n] Register Diagram

Table 51-18: CSPFT_CIDCVR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Context ID Comparator Value Register. The `CSPFT_CIDCVR[n].VALUE` bit field holds a context ID value for comparison.

Claim Tag Clear Register

The CSPFT_CLAIMCLR register is used to clear bits in the claim tag or get the current value of the claim tag.

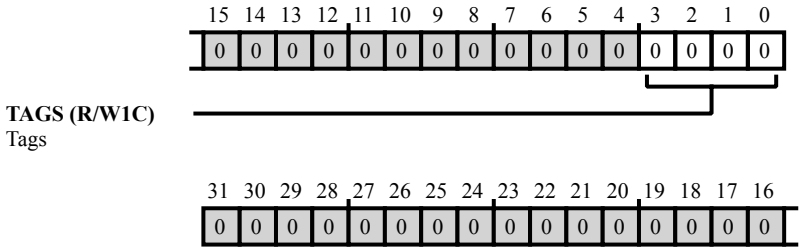


Figure 51-14: CSPFT_CLAIMCLR Register Diagram

Table 51-19: CSPFT_CLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W1C)	TAGS	Tags. A read of the CSPFT_CLAIMCLR.TAGS bit field returns the current value, a write clears bits.

Claim Tag Set Register

The CSPFT_CLAIMSET register is used to set bits in the claim tag and find the number of bits supported by the claim tag.

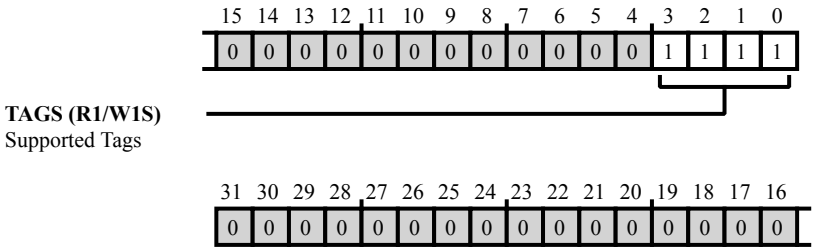


Figure 51-15: CSPFT_CLAIMSET Register Diagram

Table 51-20: CSPFT_CLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R1/W1S)	TAGS	Supported Tags. The CSPFT_CLAIMSET.TAGS bit field sets bits in the claim tag and finds the number of bits supported by the claim tag.

Counter Enable Event Register

The CSPFT_CNTENR[n] register describes the event that enables the corresponding counter.

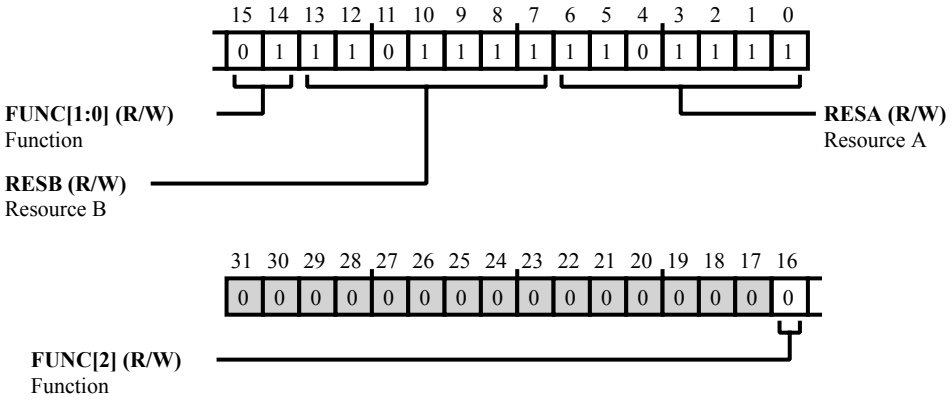


Figure 51-16: CSPFT_CNTENR[n] Register Diagram

Table 51-21: CSPFT_CNTENR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16:14 (R/W)	FUNC	Function. The CSPFT_CNTENR[n].FUNC bit field specifies the logical operation that combines the two resources that define the event.	
		0	A
		1	NOT(A)
		2	A AND B
		3	NOT(A) AND B
		4	NOT(A) AND NOT(B)
		5	A OR B
		6	NOT(A) OR B
		7	NOT(A) OR NOT(B)
13:7 (R/W)	RESB	Resource B. The CSPFT_CNTENR[n].RESB bit field specifies one of the two resources that can be combined by the logical operation specified in the CSPFT_CNTENR[n].FUNC field (See CSPFT_CNTENR[n].RESA for list of possible values).	
6:0 (R/W)	RESA	Resource A. The CSPFT_CNTENR[n].RESA bit field specifies one of the two resources that can be combined by the logical operation specified in the CSPFT_CNTENR[n].FUNC field.	

Table 51-21: CSPFT_CNTENR[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	Single Addr Comparator 0
		1	Single Addr Comparator 1
		2	Single Addr Comparator 2
		3	Single Addr Comparator 3
		4	Single Addr Comparator 4
		5	Single Addr Comparator 5
		6	Single Addr Comparator 6
		7	Single Addr Comparator 7
		8	Single Addr Comparator 8
		9	Single Addr Comparator 9
		10	Single Addr Comparator 10
		11	Single Addr Comparator 11
		12	Single Addr Comparator 12
		13	Single Addr Comparator 13
		14	Single Addr Comparator 14
		15	Single Addr Comparator 15
		16	Addr Range Comparator 0
		17	Addr Range Comparator 1
		18	Addr Range Comparator 2
		19	Addr Range Comparator 3
		20	Addr Range Comparator 4
		21	Addr Range Comparator 5
		22	Addr Range Comparator 6
		23	Addr Range Comparator 7
		64	Counter 0 at Zero
		65	Counter 1 at Zero
		66	Counter 2 at Zero
		67	Counter 3 at Zero
		88	Context ID Comparator 0
		89	Context ID Comparator 1

Table 51-21: CSPFT_CNTENR[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		90	Context ID Comparator 2
		95	TraceEnable Start/Stop Resource 0 or 1
		96	External Inputs 0
		97	External Inputs 1
		98	External Inputs 2
		99	External Inputs 3
		110	Trace Prohibited
		111	Always TRUE

Counter Reload Event Register

The CSPFT_CNTRLDEVR[n] register defines the event that causes the corresponding counter to be reloaded with the value held in the corresponding CSPFT_CNTRLDVR[n] register.

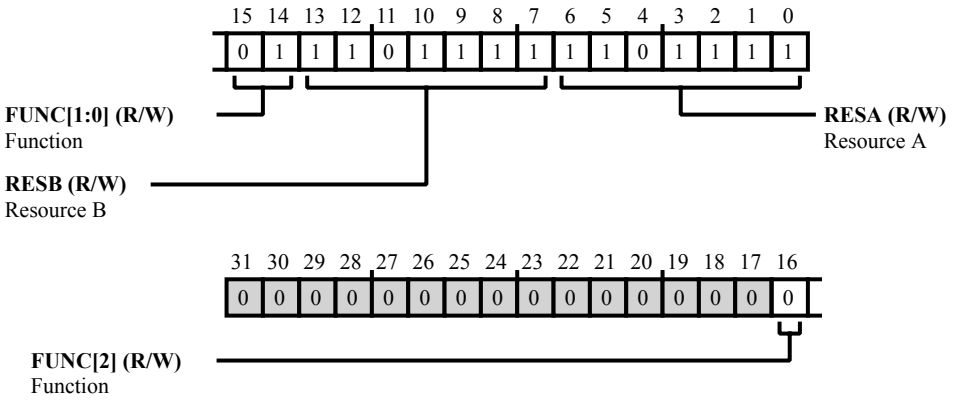


Figure 51-17: CSPFT_CNTRLDEVR[n] Register Diagram

Table 51-22: CSPFT_CNTRLDEVR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16:14 (R/W)	FUNC	Function. The CSPFT_CNTRLDEVR[n].FUNC bit field specifies the logical operation that combines the two resources that define the event.	
		0	A
		1	NOT(A)
		2	A AND B
		3	NOT(A) AND B
		4	NOT(A) AND NOT(B)
		5	A OR B
		6	NOT(A) OR B
		7	NOT(A) OR NOT(B)
13:7 (R/W)	RESB	Resource B. The CSPFT_CNTRLDEVR[n].RESB bit field specifies one of the two resources that can be combined by the logical operation specified in the CSPFT_CNTRLDEVR[n].FUNC field (See CSPFT_CNTRLDEVR[n].RESA for list of possible values).	

Table 51-22: CSPFT_CNTRLDEVR[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:0 (R/W)	RESA	Resource A. The `CSPFT_CNTRLDEVR[n].RESA` bit field specifies one of the two resources that can be combined by the logical operation specified in the `CSPFT_CNTRLDEVR[n].FUNC` field.	
		0	Single Addr Comparator 0
		1	Single Addr Comparator 1
		2	Single Addr Comparator 2
		3	Single Addr Comparator 3
		4	Single Addr Comparator 4
		5	Single Addr Comparator 5
		6	Single Addr Comparator 6
		7	Single Addr Comparator 7
		8	Single Addr Comparator 8
		9	Single Addr Comparator 9
		10	Single Addr Comparator 10
		11	Single Addr Comparator 11
		12	Single Addr Comparator 12
		13	Single Addr Comparator 13
		14	Single Addr Comparator 14
		15	Single Addr Comparator 15
		16	Addr Range Comparator 0
		17	Addr Range Comparator 1
		18	Addr Range Comparator 2
		19	Addr Range Comparator 3
		20	Addr Range Comparator 4
		21	Addr Range Comparator 5
		22	Addr Range Comparator 6
		23	Addr Range Comparator 7
		64	Counter 0 at zero
		65	Counter 1 at zero
		66	Counter 2 at zero

Table 51-22: CSPFT_CNTRLDEVR[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		67	Counter 3 at zero
		88	Context ID comparator 0
		89	Context ID comparator 1
		90	Context ID comparator 2
		95	TraceEnable start/stop resource 0 or 1
		96	External Inputs 0
		97	External Inputs 1
		98	External Inputs 2
		99	External Inputs 3
		110	Trace prohibited
		111	Always TRUE

Counter Reload Value Register

The CSPFT_CNTRLDVR[n] register specifies the starting value of the corresponding counter.

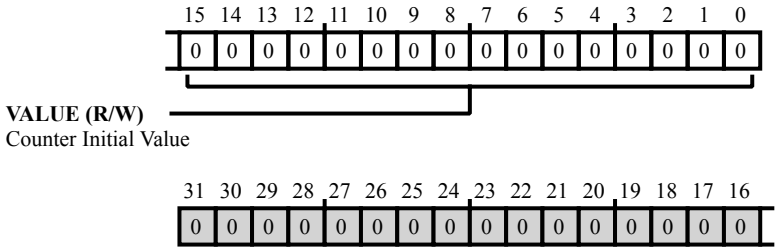


Figure 51-18: CSPFT_CNTRLDVR[n] Register Diagram

Table 51-23: CSPFT_CNTRLDVR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Counter Initial Value. The CSPFT_CNTRLDVR[n].VALUE bit field specifies the starting value of the corresponding counter.

Counter Value Register

The `CSPFT_CNTVR[n]` register holds the current value of the corresponding counter.

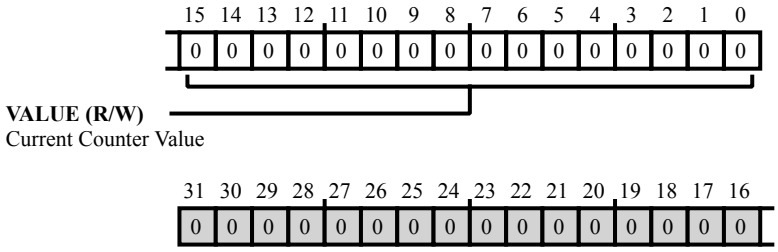


Figure 51-19: CSPFT_CNTVR[n] Register Diagram

Table 51-24: CSPFT_CNTVR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15:0 (R/W)	VALUE	Current Counter Value. The `CSPFT_CNTVR[n].VALUE` bit field specifies the current value of the corresponding counter.

Main Control Register

The CSPFT_CTL register controls general operation of the PTM, such as whether tracing is enabled.

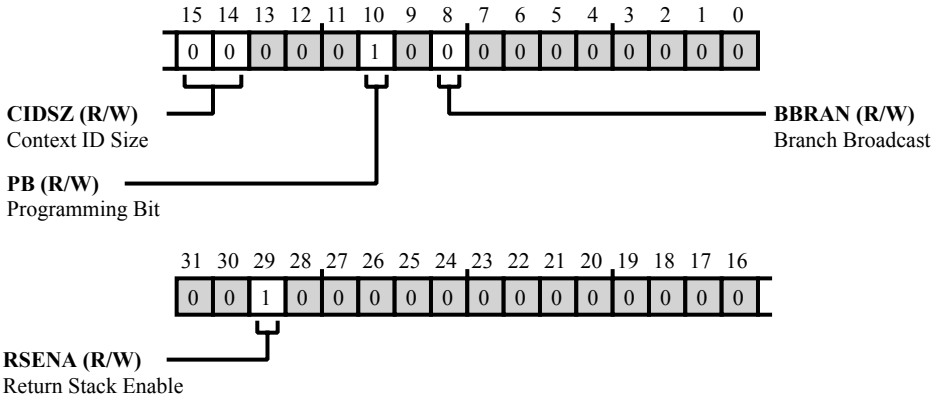


Figure 51-20: CSPFT_CTL Register Diagram

Table 51-25: CSPFT_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	RSENA	Return Stack Enable. When the CSPFT_CTL.RSENA bit is set, the first indirect branch back to an address generates a branch without exception packet, and subsequent branches back to the same address generate E Atoms. This compresses inner loops of HW loops and code that indirectly branches back.	
15:14 (R/W)	CIDSZ	Context ID Size. The CSPFT_CTL.CIDSZ bit field specifies the byte size to trace. Only the bytes specified are traced, even if the new Context ID value is larger than this.	
		0	No Context ID Tracing
		1	One byte Traced
		2	Two Bytes Traced
		3	Three Bytes Traced

Table 51-25: CSPFT_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
10 (R/W)	PB	Programming Bit. To program the CSPFT, use the following procedure. 1. Set the `CSPFT_CTL.PB` bit to disable all trace functionality. 2. Poll the `CSPFT_STAT.PB` bit waiting for it to be 1 (FIFO drained, trace halted). 3. Program the trace registers, counter and other registers, as required. 4. Set this bit to 0. 5. Poll the `CSPFT_STAT.PB` bit until it reads 0 (trace status reset, trace restarted). When the `CSPFT_CTL.PB` bit is set, the FIFO is drained and no more trace is produced. All counters are held in their present state and the external outputs are forced low. After the FIFO is drained, the `CSPFT_STAT.PB` is set to reflect that the part is ready to program. When this bit is cleared, the trace status is cleared and trace is restarted.	
		0	Trace Enabled
		1	Trace Disabled
8 (R/W)	BBRAN	Branch Broadcast. Set the `CSPFT_CTL.BBRAN` bit to 1 to enable branch broadcasting. Branch broadcasting traces the address of direct branch instructions rather than producing E atoms.	

Device Type Identifier Register

The CSPFT_DEVTYPE register is read-only. It provides a debugger with information about the component when the part number field is not recognized. The debugger can then report this information.

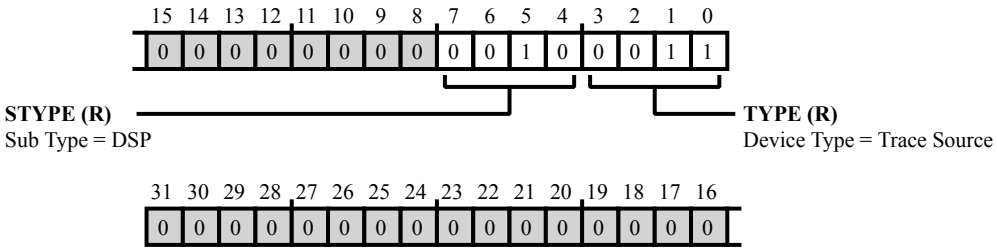


Figure 51-21: CSPFT_DEVTYPE Register Diagram

Table 51-26: CSPFT_DEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	STYPE	Sub Type = DSP.
3:0 (R/NW)	TYPE	Device Type = Trace Source.

External Output Event Register

The CSPFT_EXTOUTEVR[n] register defines the event that controls the corresponding EXTOUT external output signal.

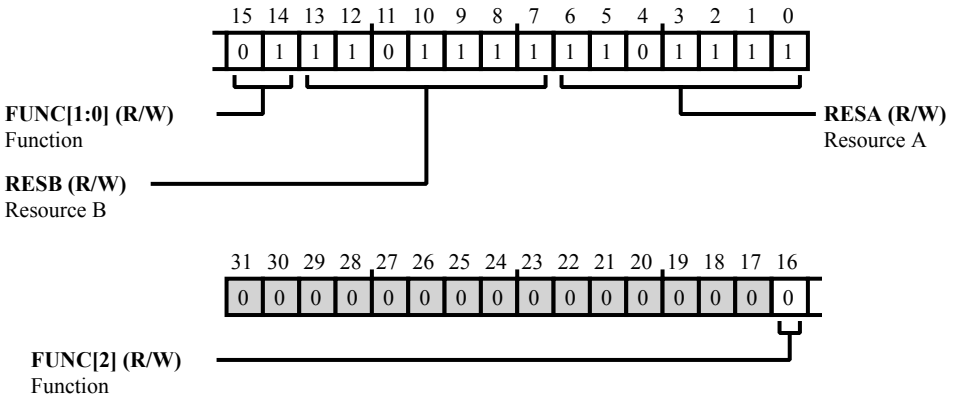


Figure 51-22: CSPFT_EXTOUTEVR[n] Register Diagram

Table 51-27: CSPFT_EXTOUTEVR[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
16:14 (R/W)	FUNC	Function. The CSPFT_EXTOUTEVR[n].FUNC bit field specifies the logical operation that combines the two resources that define the event.	
		0	A
		1	NOT(A)
		2	A AND B
		3	NOT(A) AND B
		4	NOT(A) AND NOT(B)
		5	A OR B
		6	NOT(A) OR B
		7	NOT(A) OR NOT(B)
13:7 (R/W)	RESB	Resource B. The CSPFT_EXTOUTEVR[n].RESB bit field specifies one of the two resources that can be combined by the logical operation specified in the CSPFT_EXTOUTEVR[n].FUNC field (See CSPFT_EXTOUTEVR[n].RESA for list of possible values).	

Table 51-27: CSPFT_EXTOUTEVR[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
6:0 (R/W)	RESA	Resource A. The `CSPFT_EXTOUTEVR[n].RESA` bit field specifies one of the two resources that can be combined by the logical operation specified in the `CSPFT_EXTOUTEVR[n].FUNC` field.	
		0	Single Addr Comparator 0
		1	Single Addr Comparator 1
		2	Single Addr Comparator 2
		3	Single Addr Comparator 3
		4	Single Addr Comparator 4
		5	Single Addr Comparator 5
		6	Single Addr Comparator 6
		7	Single Addr Comparator 7
		8	Single Addr Comparator 8
		9	Single Addr Comparator 9
		10	Single Addr Comparator 10
		11	Single Addr Comparator 11
		12	Single Addr Comparator 12
		13	Single Addr Comparator 13
		14	Single Addr Comparator 14
		15	Single Addr Comparator 15
		16	Addr Range Comparator 0
		17	Addr Range Comparator 1
		18	Addr Range Comparator 2
		19	Addr Range Comparator 3
		20	Addr Range Comparator 4
		21	Addr Range Comparator 5
		22	Addr Range Comparator 6
		23	Addr Range Comparator 7
		64	Counter 0 at Zero
		65	Counter 1 at Zero
		66	Counter 2 at Zero

Table 51-27: CSPFT_EXTOUTEVR[n] Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		67	Counter 3 at Zero
		88	Context ID Comparator 0
		89	Context ID Comparator 1
		90	Context ID Comparator 2
		95	TraceEnable Start/Stop Resource 0 or 1
		96	External Inputs 0
		97	External Inputs 1
		98	External Inputs 2
		99	External Inputs 3
		110	Trace Prohibited
		111	Always TRUE

Hardware Feature Register

The CSPFT_HWFEAT register enables software to read the implementation defined configuration of the PTM, giving the number of each type of hardware resource. Each field indicates the number of instances of a particular resource, zero indicates that there are no implemented resources of that type.

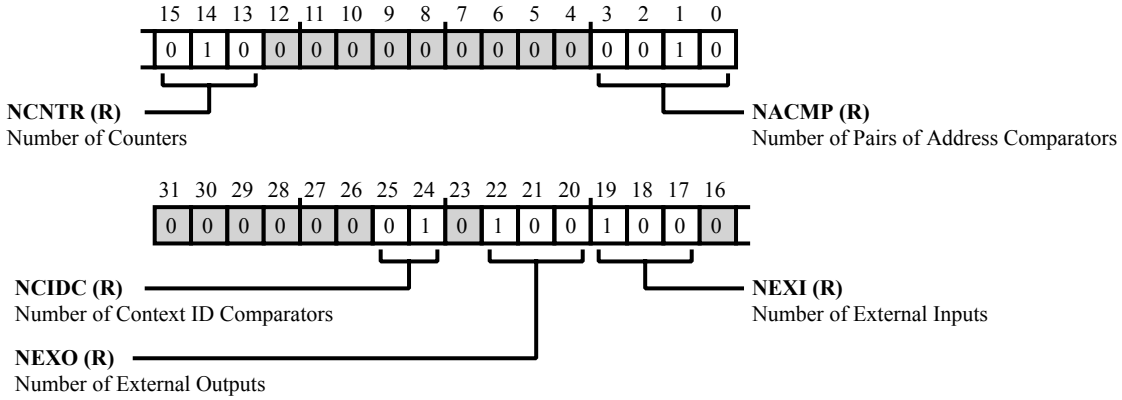


Figure 51-23: CSPFT_HWFEAT Register Diagram

Table 51-28: CSPFT_HWFEAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:24 (R/NW)	NCIDC	Number of Context ID Comparators. The CSPFT_HWFEAT.NCIDC bit field identifies the number of context ID comparators.
22:20 (R/NW)	NEXO	Number of External Outputs. The CSPFT_HWFEAT.NEXO bit field identifies the number of external outputs (up to four).
19:17 (R/NW)	NEXI	Number of External Inputs. The CSPFT_HWFEAT.NEXI bit field identifies the number of external inputs (up to four).
15:13 (R/NW)	NCNTR	Number of Counters. The CSPFT_HWFEAT.NCNTR bit field identifies the number of counters (up to four) that are configured using the counter registers.
3:0 (R/NW)	NACMP	Number of Pairs of Address Comparators. The CSPFT_HWFEAT.NACMP bit field identifies the number of pairs of address comparators as address range comparators (ARCs). In this case, two adjacent address comparators form the ARC, so you can use address comparators 1 and 2 to define the first ARC. An ARC matches when any instruction in the specified range is committed for execution, regardless of whether the instruction passes its condition code test.

Table 51-28: CSPFT_HWFEAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	No Address Comparators
		1	1 Pair of Address Comparators
		2	2 Pairs of Address Comparators
		3	3 Pairs of Address Comparators
		4	4 Pairs of Address Comparators
		5	5 Pairs of Address Comparators
		6	6 Pairs of Address Comparators
		7	7 Pairs of Address Comparators
		8	8 Pairs of Address Comparators

Lock Access Register

The CSPFT_LAR register is used to provide lock and unlock access to all other CSPFT registers.

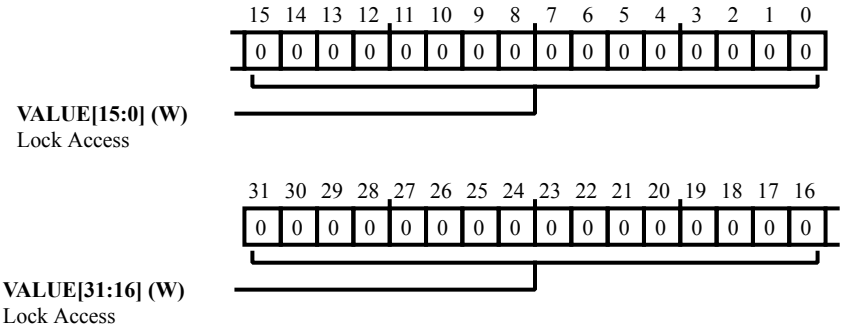


Figure 51-24: CSPFT_LAR Register Diagram

Table 51-29: CSPFT_LAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Lock Access. Write 0xC5ACCE55 to the CSPFT_LAR.VALUE bit field to unlock. Write any other value to lock.

Lock Status Register

The CSPFT_LSR register is used to detect if the lock registers are implemented and if they are currently locked.

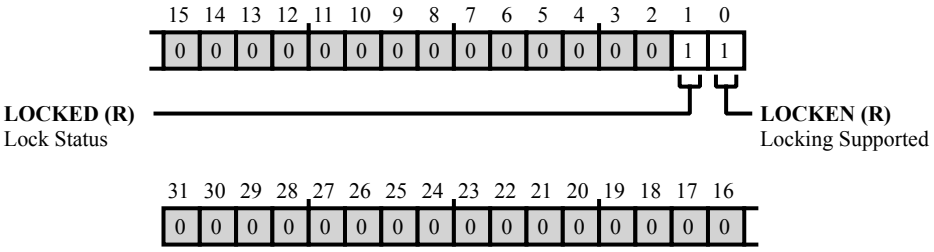


Figure 51-25: CSPFT_LSR Register Diagram

Table 51-30: CSPFT_LSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	LOCKED	Lock Status. The CSPFT_LSR.LOCKED bit indicates whether the PFT is locked.	
		0	Writes are permitted
		1	Locked. Writes are ignored
0 (R/NW)	LOCKEN	Locking Supported. The CSPFT_LSR.LOCKEN bit indicates whether the lock registers are implemented for this interface.	
		0	Locking is Not Required. This access is from an interface that ignores the lock registers.
		1	Locking is Required. This access is from an interface that requires the PFT to be unlocked.

Peripheral ID0 Register

The CSPFT_PID0 register holds peripheral identification information.

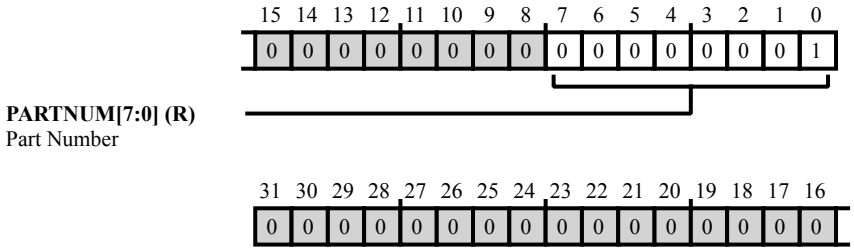


Figure 51-26: CSPFT_PID0 Register Diagram

Table 51-31: CSPFT_PID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PARTNUM	Part Number. The CSPFT_PID0.PARTNUM bit field holds the peripheral identification number.

Peripheral ID1 Register

The CSPFT_PID1 register holds peripheral identification information.

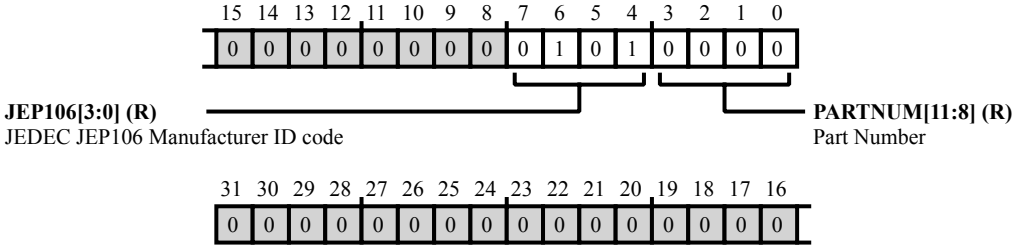


Figure 51-27: CSPFT_PID1 Register Diagram

Table 51-32: CSPFT_PID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	JEP106	JEDEC JEP106 Manufacturer ID code.
3:0 (R/NW)	PARTNUM	Part Number. The CSPFT_PID1.PARTNUM bit field holds the peripheral identification number.

Peripheral ID2 Register

The `CSPFT_PID2` register holds peripheral identification information.

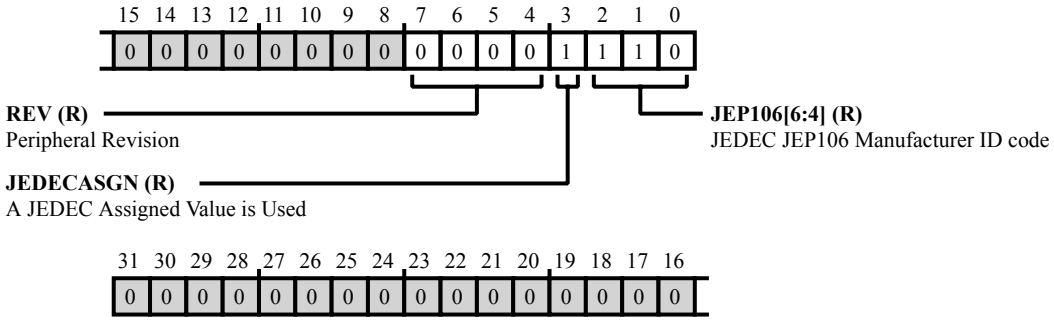


Figure 51-28: CSPFT_PID2 Register Diagram

Table 51-33: CSPFT_PID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REV	Peripheral Revision.
3 (R/NW)	JEDECASGN	A JEDEC Assigned Value is Used. The `CSPFT_PID2.JEDECASGN` bit indicates that a JEDEC assigned value is used.
2:0 (R/NW)	JEP106	JEDEC JEP106 Manufacturer ID code.

Peripheral ID3 Register

The CSPFT_PID3 register holds peripheral identification information.

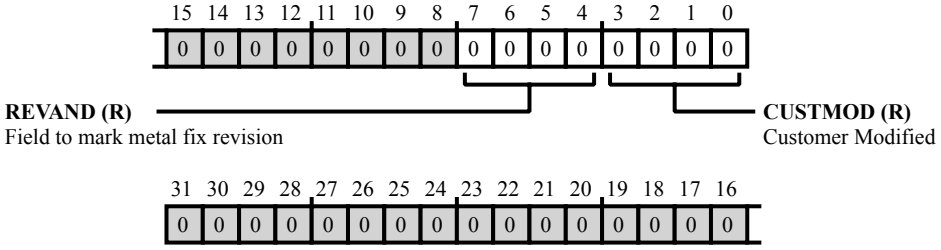


Figure 51-29: CSPFT_PID3 Register Diagram

Table 51-34: CSPFT_PID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	REVAND	Field to mark metal fix revision.
3:0 (R/NW)	CUSTMOD	Customer Modified.

Peripheral ID4 Register

The `CSPFT_PID4` register holds peripheral identification information.

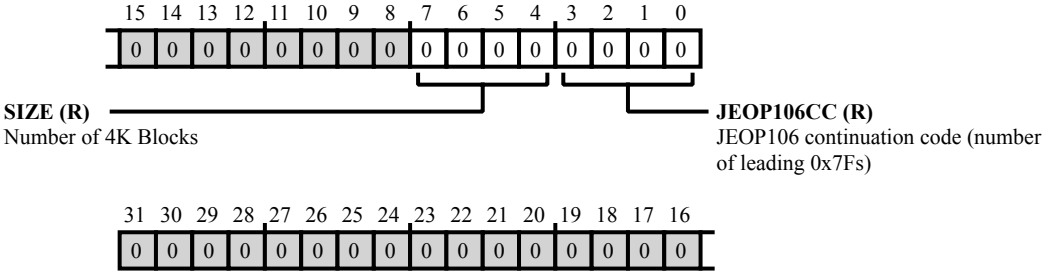


Figure 51-30: CSPFT_PID4 Register Diagram

Table 51-35: CSPFT_PID4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:4 (R/NW)	SIZE	Number of 4K Blocks. The `CSPFT_PID4.SIZE` bit field contains the size of the component in 4K chunks minus 1 (for example 0=4K).
3:0 (R/NW)	JEOP106CC	JEOP106 continuation code (number of leading 0x7Fs).

Status Register

The CSPFT_STAT register provides information about the current status of the trace and trigger logic.

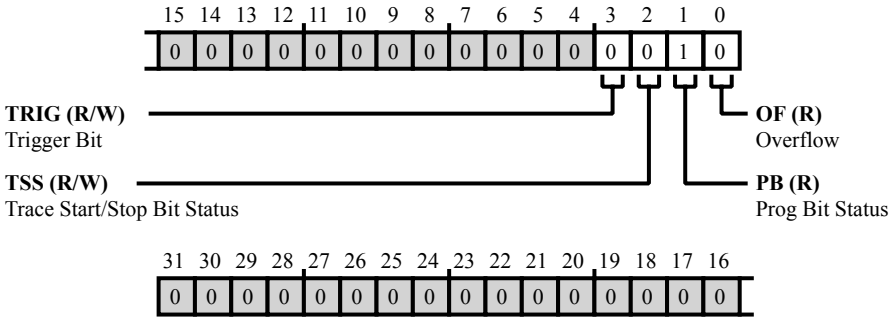


Figure 51-31: CSPFT_STAT Register Diagram

Table 51-36: CSPFT_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W)	TRIG	Trigger Bit. The CSPFT_STAT.TRIG bit is set when the trigger occurs, and prevents the trigger from being output until the CSPFT is next programmed. This bit is reset when the CSPFT_CTL.PB bit transitions from 1 to 0.
2 (R/W)	TSS	Trace Start/Stop Bit Status. The CSPFT_STAT.TSS bit holds the current status of the trace start/stop resource. If = 1, it indicates that a trace start address has been matched, without a corresponding trace stop address match. This bit =0 when trace is restarted (the CSPFT_CTL.PB bit transitions from 1 to 0).
1 (R/NW)	PB	Prog Bit Status. The CSPFT_STAT.PB bit indicates the current effective value of the CSPFT_CTL.PB bit. The program must wait for this bit to =1 before programming the CSPFT. (See the CSPFT_CTL.PB bit description).
0 (R/NW)	OF	Overflow. If the CSPFT_STAT.OF bit is =1, there is an overflow. This bit is cleared =0 when the trace is restarted (CSPFT_CTL.PB transitions from 1 to 0)

Synchronization Frequency Register

The CSPFT_SYNCFR register holds the trace synchronization frequency value.

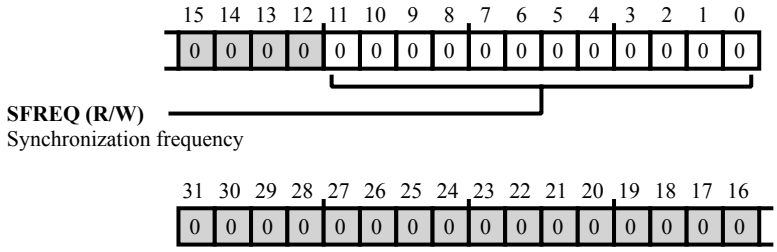


Figure 51-32: CSPFT_SYNCFR Register Diagram

Table 51-37: CSPFT_SYNCFR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
11:0 (R/W)	SFREQ	Synchronization frequency. The CSPFT_SYNCFR.SFREQ bit field is the number of 128-byte blocks of trace data after which you want to drop an address synchronization packet. If the circular buffer size is 16k, ensure that there are a few A-syncs in the buffer, so setting this to 16 means that every 2k there is an A-Sync packet.

CoreSight Trace ID Register

The CSPFT_TRACEIDR register defines the 7-bit trace ID, for output to the trace bus.

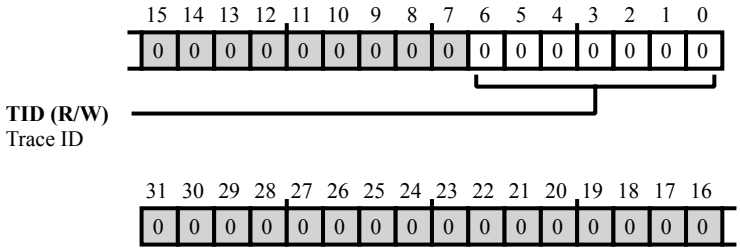


Figure 51-33: CSPFT_TRACEIDR Register Diagram

Table 51-38: CSPFT_TRACEIDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
6:0 (R/W)	TID	Trace ID. Defines the 7-bit trace ID, for output to the trace bus. Used in systems where multiple trace sources are present and tracing simultaneously. For example, when outputs trace onto the AMBA 3 Advanced Trace Bus, a unique ID is required for each trace source.

ADSP-SC59x TAPC Register Descriptions

TAPC (TAPC) contains the following registers.

Table 51-39: ADSP-SC59x TAPC Register List

Name	Description
TAPC_DBGCTL	Debug Control Register
TAPC_IDCODE	IDCODE Register
TAPC_SDBGKEY0	Secure Debug Key 0 Register
TAPC_SDBGKEY1	Secure Debug Key 1 Register
TAPC_SDBGKEY2	Secure Debug Key 2 Register
TAPC_SDBGKEY3	Secure Debug Key 3 Register
TAPC_SDBGKEY_CTL	Secure Debug Key Control Register
TAPC_SDBGKEY_STAT	Secure Debug Key Status Register
TAPC_USERCODE	USERCODE Register

Debug Control Register

The TAPC_DBGCTL register creates authentication signals to all CoreSight components in the system such as DBGEN, NIDEN, SPNIDEN and SPIDEN.

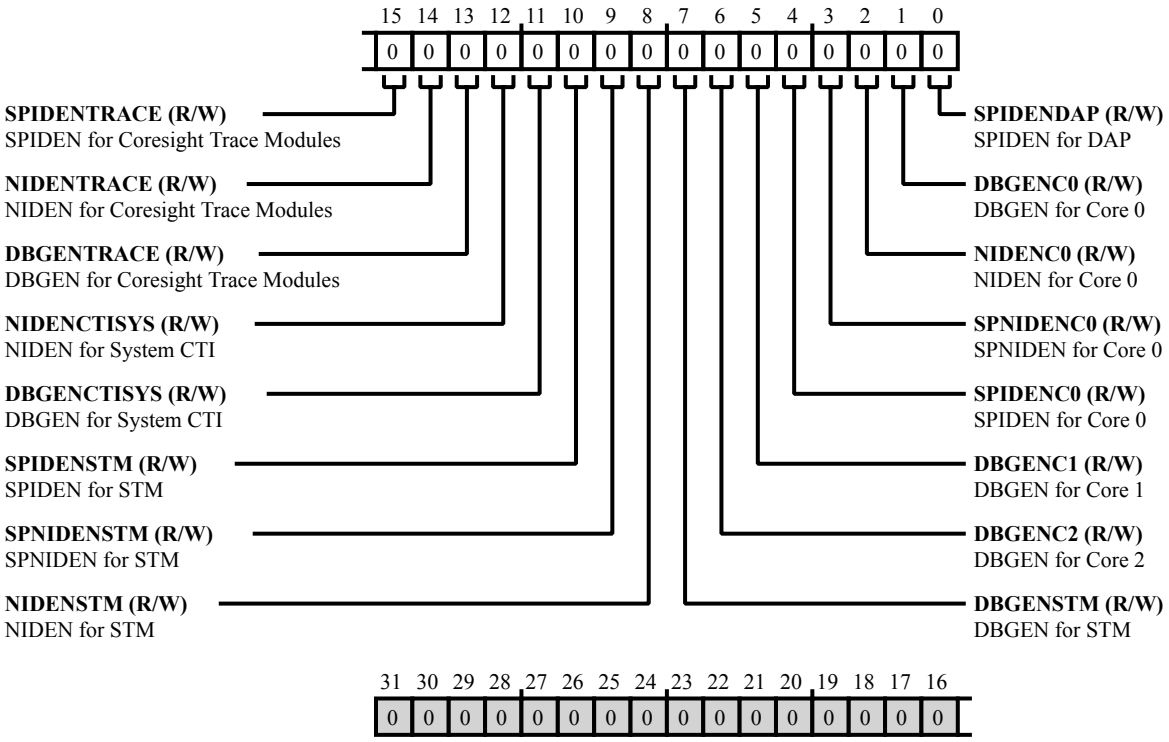


Figure 51-34: TAPC_DBGCTL Register Diagram

Table 51-40: TAPC_DBGCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
15 (R/W)	SPIDENTRACE	SPIDEN for Coresight Trace Modules.
14 (R/W)	NIDENTRACE	NIDEN for Coresight Trace Modules.
13 (R/W)	DBGENTRACE	DBGEN for Coresight Trace Modules.
12 (R/W)	NIDENCTISYS	NIDEN for System CTI.
11 (R/W)	DBGENCTISYS	DBGEN for System CTI.

Table 51-40: TAPC_DBGCTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (R/W)	SPIDENSTM	SPIDEN for STM.
9 (R/W)	SPNIDENSTM	SPNIDEN for STM.
8 (R/W)	NIDENSTM	NIDEN for STM.
7 (R/W)	DBGENSTM	DBGEN for STM.
6 (R/W)	DBGENC2	DBGEN for Core 2.
5 (R/W)	DBGENC1	DBGEN for Core 1.
4 (R/W)	SPIDENC0	SPIDEN for Core 0.
3 (R/W)	SPNIDENC0	SPNIDEN for Core 0.
2 (R/W)	NIDENC0	NIDEN for Core 0.
1 (R/W)	DBGENC0	DBGEN for Core 0.
0 (R/W)	SPIDENDAP	SPIDEN for DAP.

IDCODE Register

The TAPC_IDCODE register holds the IDCODE. The bit field is defined as follows.

IDCODE[31:28] = 0x1* – REVID

IDCODE[27:12] = 0x280B – JTAG ID

IDCODE[11:1] = 0x65 – Manufacturer ID

IDCODE[0] = 0x1

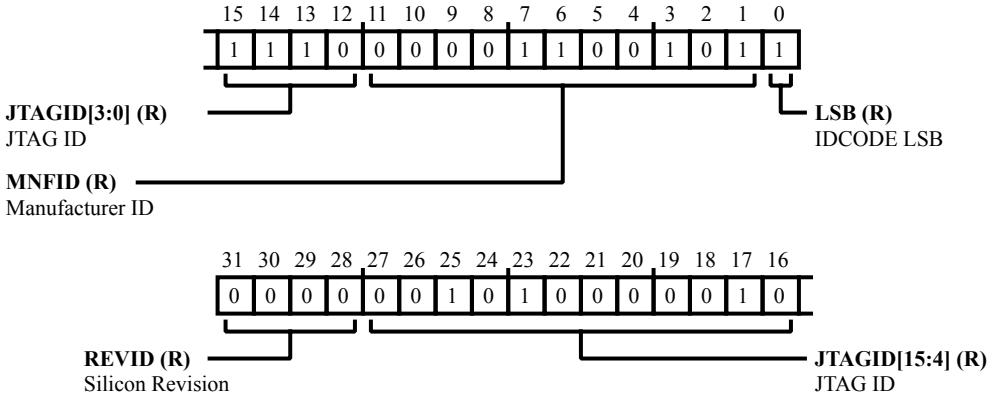


Figure 51-35: TAPC_IDCODE Register Diagram

Table 51-41: TAPC_IDCODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:28 (R/NW)	REVID	Silicon Revision. The TAPC_IDCODE.REVID bit field holds the silicon revision. See the processor anomaly list for details.
27:12 (R/NW)	JTAGID	JTAG ID.
11:1 (R/NW)	MNFID	Manufacturer ID.
0 (R/NW)	LSB	IDCODE LSB.

Secure Debug Key 0 Register

The TAPC_SDBGKEY0 register allows a locked part to unlock debug access through the JTAG or SWD interfaces. A debug key of 128 bits needs to be written into the Secure Debug Key registers (TAPC_SDBGKEY0, TAPC_SDBGKEY1, TAPC_SDBGKEY2, TAPC_SDBGKEY3) in the TAPC through the peripheral bus interface.

These registers hold the value of the key against which a matching key provided by the debug user is compared to enable a debug session. The task of writing these registers is performed (initially) by boot ROM code which copies a customer-selected key from the Flash memory info block to these registers.

An SDBGKEY value of all 0's is always an invalid key, a value of all 1's match the default value of the Secure Debug Key Compare registers and requires no entry in these registers. It is recommended programs have a significant number of 0's and 1's in a pseudo-random pattern throughout the 128-bit code for maximum protection.

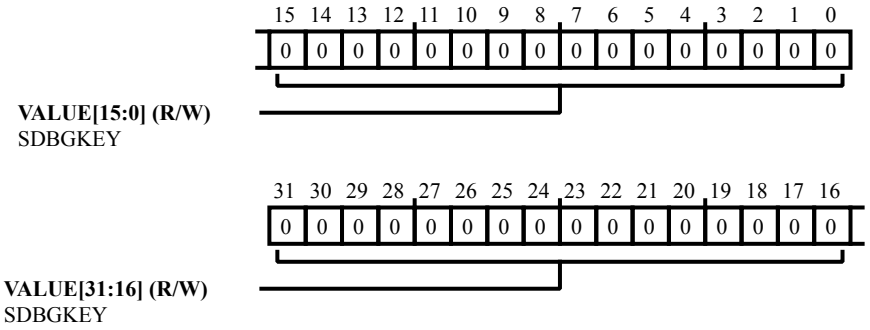


Figure 51-36: TAPC_SDBGKEY0 Register Diagram

Table 51-42: TAPC_SDBGKEY0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SDBGKEY. The TAPC_SDBGKEY0.VALUE bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 1 Register

The TAPC_SDBGKEY1 register allows a locked part to unlock debug access through the JTAG or SWD interfaces. See the TAPC_SDBGKEY0 register description for more information.

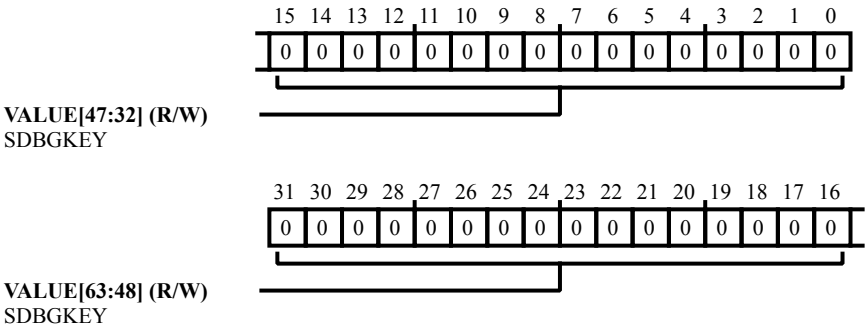


Figure 51-37: TAPC_SDBGKEY1 Register Diagram

Table 51-43: TAPC_SDBGKEY1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SDBGKEY. The TAPC_SDBGKEY1.VALUE bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 2 Register

The TAPC_SDBGKEY2 register allows a locked part to unlock debug access through the JTAG or SWD interfaces. See the TAPC_SDBGKEY0 register description for more information.

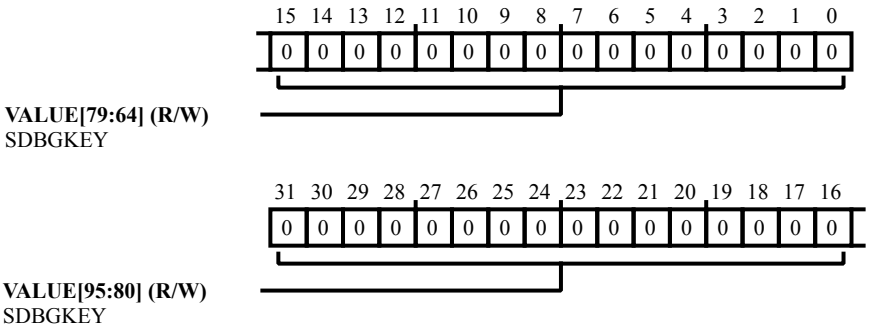


Figure 51-38: TAPC_SDBGKEY2 Register Diagram

Table 51-44: TAPC_SDBGKEY2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SDBGKEY. The TAPC_SDBGKEY2.VALUE bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key 3 Register

The TAPC_SDBGKEY3 register allows a locked part to unlock debug access through the JTAG or SWD interfaces. See the TAPC_SDBGKEY0 register description for more information.

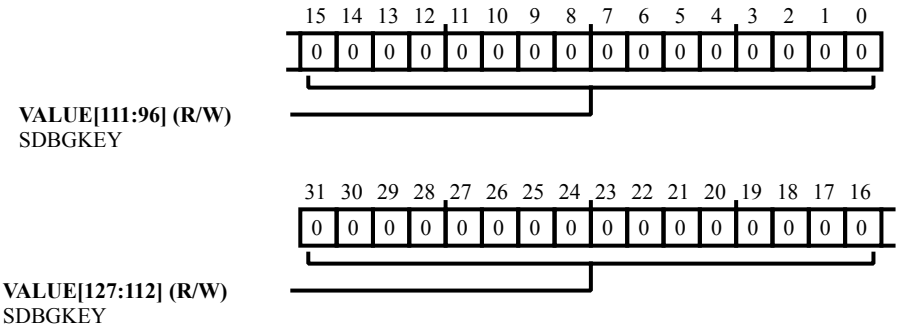


Figure 51-39: TAPC_SDBGKEY3 Register Diagram

Table 51-45: TAPC_SDBGKEY3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SDBGKEY. The TAPC_SDBGKEY3.VALUE bit field holds the value of the key against which a matching key provided by the debug user is compared to enable a debug session.

Secure Debug Key Control Register

The TAPC_SDBGKEY_CTL register contains a bit to configure configure validity for the secure debug keys.

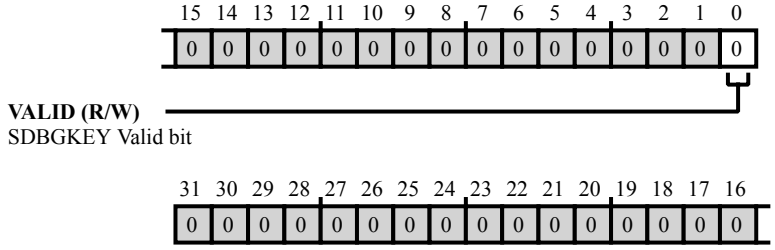


Figure 51-40: TAPC_SDBGKEY_CTL Register Diagram

Table 51-46: TAPC_SDBGKEY_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	VALID	SDBGKEY Valid bit.

Secure Debug Key Status Register

The TAPC_SDBGKEY_STAT register contains the PASS/FAIL status of the secure debug key match.

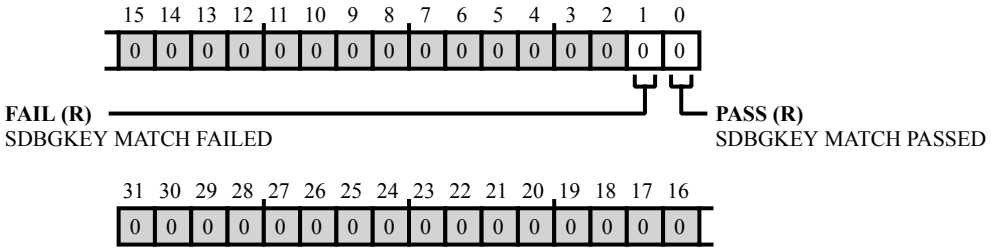


Figure 51-41: TAPC_SDBGKEY_STAT Register Diagram

Table 51-47: TAPC_SDBGKEY_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	FAIL	SDBGKEY MATCH FAILED.
0 (R/NW)	PASS	SDBGKEY MATCH PASSED.

USERCODE Register

The TAPC_USERCODE register stores the 32-bit USERCODE value for the SoC debug unit.

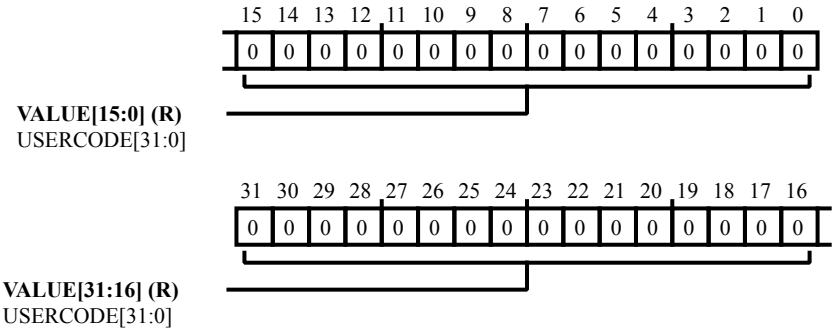


Figure 51-42: TAPC_USERCODE Register Diagram

Table 51-48: TAPC_USERCODE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/NW)	VALUE	USERCODE[31:0].

ADSP-SC59x CSCTI Register Descriptions

CSCTI_Register_Definitions (CSCTI) contains the following registers.

Table 51-49: ADSP-SC59x CSCTI Register List

Name	Description
CSCTI_ASICCTL	External Multiplexor Control Register
CSCTI_AUTHSTATUS	Authentication Status
CSCTI_CLAIMCLR	Claim Tag Clear Register
CSCTI_CLAIMSET	Claim Tag Set Register
CSCTI_COMPID0	Component ID0
CSCTI_COMPID1	Component ID1
CSCTI_COMPID2	Component ID2
CSCTI_COMPID3	Component ID3
CSCTI_CTIAPPCLEAR	CTI Application Trigger Clear Register
CSCTI_CTIAPPPULSE	CTI Application Pulse Register
CSCTI_CTIAPPSET	CTI Application Trigger Set Register

Table 51-49: ADSP-SC59x CSCTI Register List (Continued)

Name	Description
CSCTI_CTICHINSTATUS	CTI Channel In Status Register
CSCTI_CTICHOUTSTATUS	CTI Channel Out Status Register
CSCTI_CTICONTROL	CTI Control Register
CSCTI_CTIGATE	Enable CTI Channel Gate Register
CSCTI_CTIINEN0	CTI Trigger 0 to Channel Enable Register
CSCTI_CTIINEN1	CTI Trigger 1 to Channel Enable Register
CSCTI_CTIINEN2	CTI Trigger 2 to Channel Enable Register
CSCTI_CTIINEN3	CTI Trigger 3 to Channel Enable Register
CSCTI_CTIINEN4	CTI Trigger 4 to Channel Enable Register
CSCTI_CTIINEN5	CTI Trigger 5 to Channel Enable Register
CSCTI_CTIINEN6	CTI Trigger 6 to Channel Enable Register
CSCTI_CTIINEN7	CTI Trigger 7 to Channel Enable Register
CSCTI_CTIINTACK	CTI Interrupt Acknowledge Register
CSCTI_CTIOUTEN0	CTI Channel to Trigger 0 Enable Register
CSCTI_CTIOUTEN1	CTI Channel to Trigger 1 Enable Register
CSCTI_CTIOUTEN2	CTI Channel to Trigger 2 Enable Register
CSCTI_CTIOUTEN3	CTI Channel to Trigger 3 Enable Register
CSCTI_CTIOUTEN4	CTI Channel to Trigger 4 Enable Register
CSCTI_CTIOUTEN5	CTI Channel to Trigger 5 Enable Register
CSCTI_CTIOUTEN6	CTI Channel to Trigger 6 Enable Register
CSCTI_CTIOUTEN7	CTI Channel to Trigger 7 Enable Register
CSCTI_CTITRIGINSTATUS	CTI Trigger In Status Register
CSCTI_CTITRIGOUTSTATUS	CTI Trigger Out Status Register
CSCTI_DEVID	Device ID
CSCTI_DEVTYPE	Device Type
CSCTI_ITCHIN	Integration Test Channel Input
CSCTI_ITCHINACK	ITCHINACK
CSCTI_ITCHOUT	Integration Test Channel Output
CSCTI_ITCHOUTACK	ITCHOUTACK
CSCTI_ITCTRL	Integration Mode Control Register
CSCTI_ITTRIGIN	Integration Test Trigger Input

Table 51-49: ADSP-SC59x CSCTI Register List (Continued)

Name	Description
CSCTI_ITTRIGINACK	ITTRIGINACK
CSCTI_ITTRIGOUT	Integration Test Trigger Output
CSCTI_ITTRIGOUTACK	ITTRIGOUTACK
CSCTI_LAR	Lock Access Register
CSCTI_LSR	Lock Status Register
CSCTI_PERIPHID0	Peripheral ID0
CSCTI_PERIPHID1	Peripheral ID1
CSCTI_PERIPHID2	Peripheral ID2
CSCTI_PERIPHID3	Peripheral ID3
CSCTI_PERIPHID4	Peripheral ID4
CSCTI_PERIPHID5	Peripheral ID5
CSCTI_PERIPHID6	Peripheral ID6
CSCTI_PERIPHID7	Peripheral ID7

External Multiplexor Control Register

The CSCTI_ASICCTL register is for I/O port control.

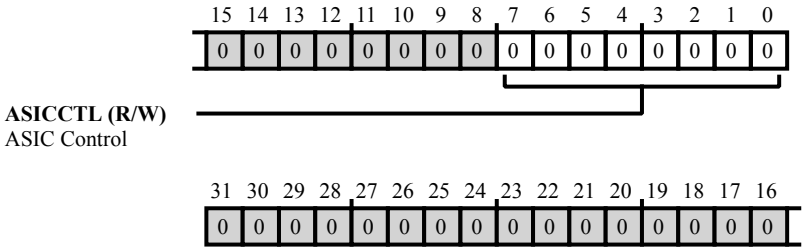


Figure 51-43: CSCTI_ASICCTL Register Diagram

Table 51-50: CSCTI_ASICCTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	ASICCTL	ASIC Control. Set and clear external output signal. 0: Clear output bit to 0. 1: Set output bit to 1.

Authentication Status

The CSCTI_AUTHSTATUS register reports the current status of the authentication control signals.

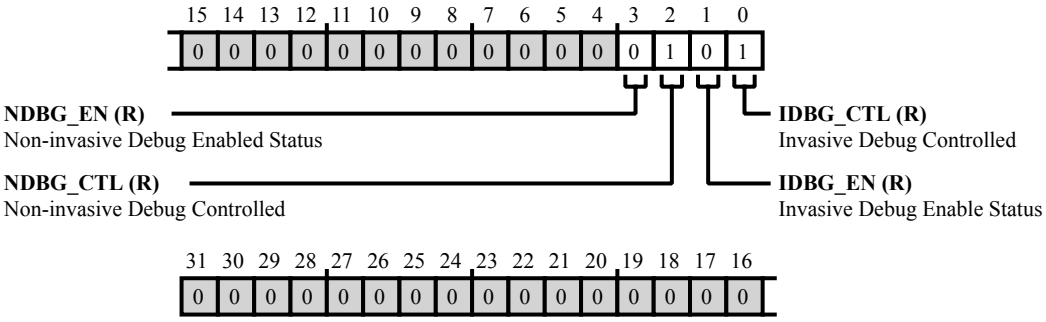


Figure 51-44: CSCTI_AUTHSTATUS Register Diagram

Table 51-51: CSCTI_AUTHSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/NW)	NDBG_EN	Non-invasive Debug Enabled Status.
2 (R/NW)	NDBG_CTL	Non-invasive Debug Controlled.
1 (R/NW)	IDBG_EN	Invasive Debug Enable Status.
0 (R/NW)	IDBG_CTL	Invasive Debug Controlled.

Claim Tag Clear Register

The CSCTI_CLAIMCLR register forms one half of the claim tag value. On writes, this location enables individual bits to be cleared. On reads, it returns the current claim tag value.

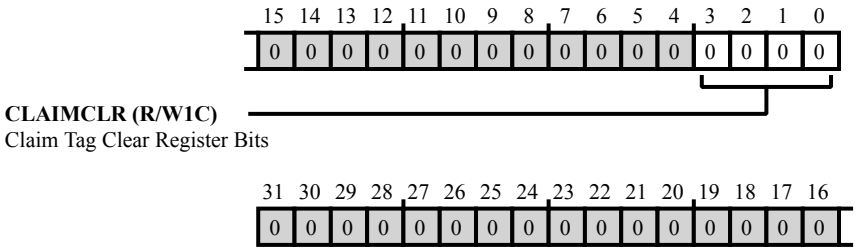


Figure 51-45: CSCTI_CLAIMCLR Register Diagram

Table 51-52: CSCTI_CLAIMCLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W1C)	CLAIMCLR	Claim Tag Clear Register Bits. A bit-programmable register bank that clears the claim tag value. It is zero at reset. It is used by software agents to signal to each other ownership of the hardware. It has no direct effect on the hardware itself.

Claim Tag Set Register

The CSCTI_CLAIMSET register forms one half of the claim tag value. On writes, this location enables individual bits to be set. On reads, it returns the number of bits that can be set.

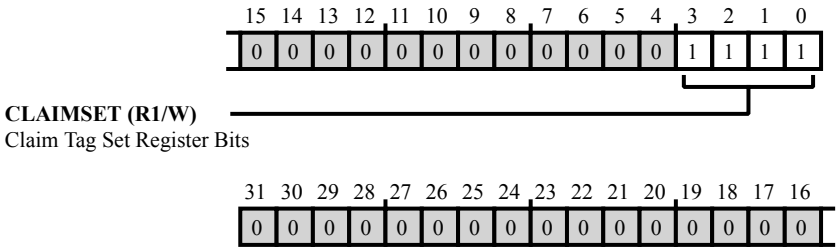


Figure 51-46: CSCTI_CLAIMSET Register Diagram

Table 51-53: CSCTI_CLAIMSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R1/W)	CLAIMSET	Claim Tag Set Register Bits. A bit-programmable register bank that sets the claim tag value. A read returns a logic 1 for all implemented locations.

Component ID0

The CSCTI_COMPID0 register is part of the set of component identification registers.

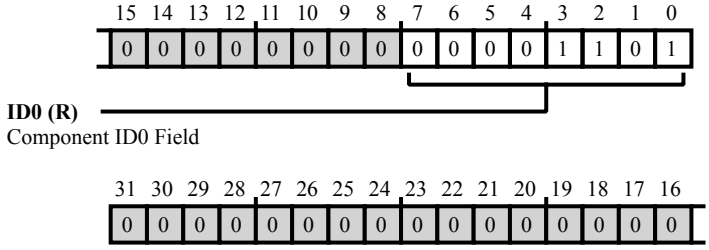


Figure 51-47: CSCTI_COMPID0 Register Diagram

Table 51-54: CSCTI_COMPID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	ID0	Component ID0 Field. Preamble. Returns 0x0D.

Component ID1

The CSCTI_COMPID1 register is part of the set of component identification registers.

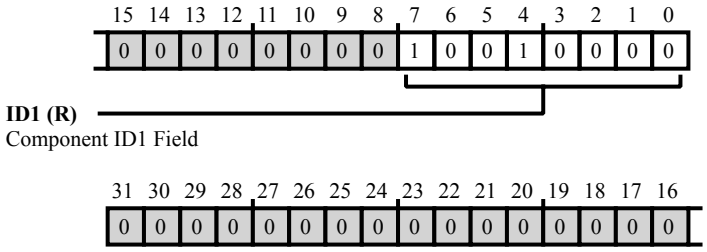


Figure 51-48: CSCTI_COMPID1 Register Diagram

Table 51-55: CSCTI_COMPID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	ID1	Component ID1 Field. Bits[7:4] Component class. Returns 0x9, indicating this is a CoreSight component. Bits[3:0] Preamble. Returns 0x0.

Component ID2

The CSCTI_COMPID2 register is part of the set of component identification registers.

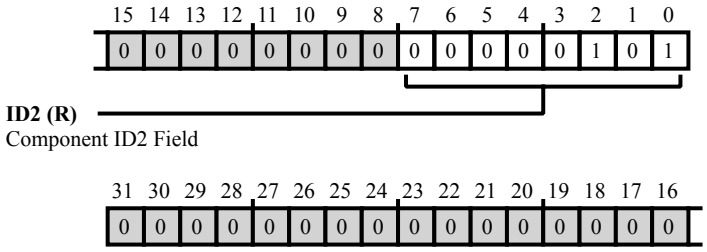


Figure 51-49: CSCTI_COMPID2 Register Diagram

Table 51-56: CSCTI_COMPID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	ID2	Component ID2 Field. Preamble. Returns 0x05.

Component ID3

The CSCTI_COMPID3 register is part of the set of component identification registers.

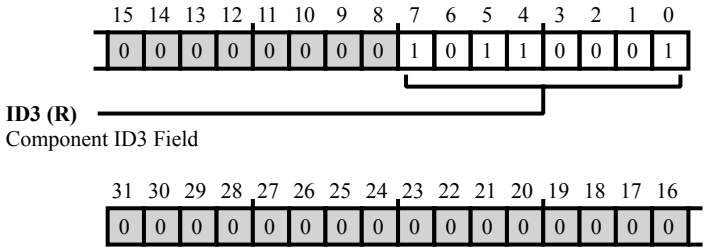


Figure 51-50: CSCTI_COMPID3 Register Diagram

Table 51-57: CSCTI_COMPID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	ID3	Component ID3 Field. Preamble. Returns 0xB1.

CTI Application Trigger Clear Register

The CSCTI_CTIAPPCLEAR register allows software to clear any channel output. Software can use this register to clear a channel event in place of a hardware source on a trigger input. This register must not be used in a system where all events are sent as single-cycle pulses. It is only retained for compatibility with older systems and software. The register is implemented before the CTIGATE register. Therefore, for the channel event to propagate outside the CTI, the corresponding CTIGATE bit must be set to 1.

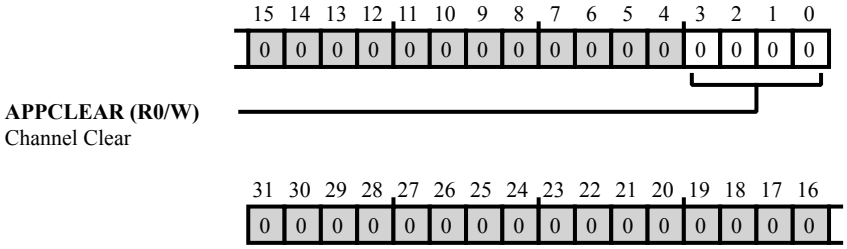


Figure 51-51: CSCTI_CTIAPPCLEAR Register Diagram

Table 51-58: CSCTI_CTIAPPCLEAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R0/W)	APPCLEAR	Channel Clear. The CSCTI_CTIAPPCLEAR.APPCLEAR bit is for clearing the corresponding internal channel flag. 0: No effect. 1: Clears the channel output.

CTI Application Pulse Register

The CSCTI_CTIAPPPULSE register allows software to pulse any channel output. This register can be used by software to pulse a channel event in place of a hardware source on a trigger input. The register is implemented before the CSCTI_CTIGATE register so, for the channel event to propagate outside the CTI, it is necessary for the corresponding bit to be 1.

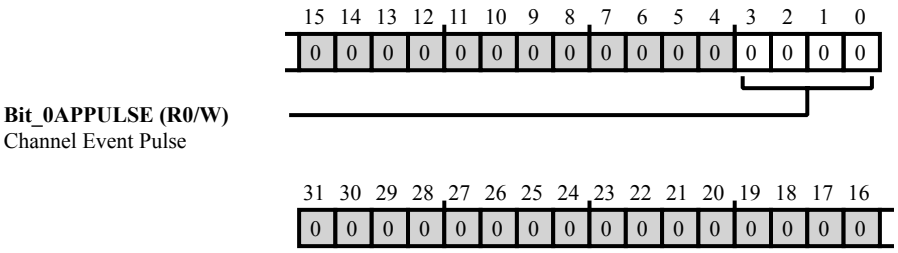


Figure 51-52: CSCTI_CTIAPPPULSE Register Diagram

Table 51-59: CSCTI_CTIAPPPULSE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R0/W)	BIT_0APPULSE	Channel Event Pulse. Pulses the channel outputs. 0: No effect. 1: Pulse channel event for one clk cycle.

CTI Application Trigger Set Register

The CSCTI_CTIAPPSET register allows software to set any channel output. Software can use this register to generate a channel event in place of a hardware source on a trigger input. This register must not be used in a system where all events are sent as single-cycle pulses. It is only retained for compatibility with older systems and software. The register is implemented before the CSCTI_CTIGATE register. Therefore, for the channel event to propagate outside the CTI, the corresponding CTIGATE bit must be set to 1.

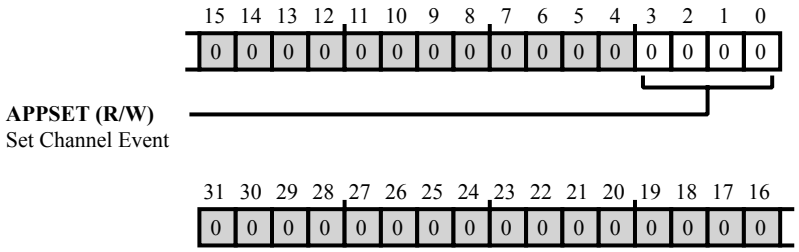


Figure 51-53: CSCTI_CTIAPPSET Register Diagram

Table 51-60: CSCTI_CTIAPPSET Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	APPSET	Set Channel Event. Sets the corresponding internal channel flag: 0: Read: application channel is inactive. Write: has no effect. 1: Read: application channel is active. Write: sets the channel output.

CTI Channel In Status Register

If the channel input is driven by a source that generates single cycle pulses, the `CSCTI_CTICHINSTATUS` register is generally read as 0.

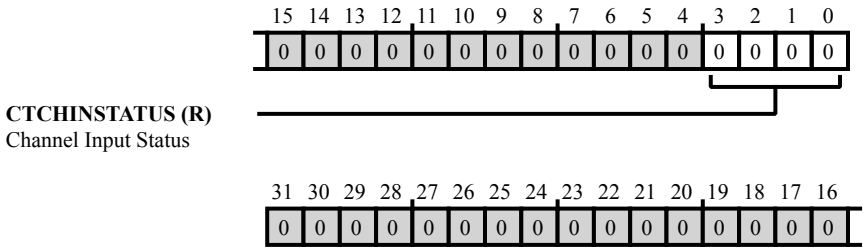


Figure 51-54: CSCTI_CTICHINSTATUS Register Diagram

Table 51-61: CSCTI_CTICHINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/NW)	CTCHINSTATUS	Channel Input Status. Channel input status. 0: One bit per channel input. 0 means that the input is LOW. 1: One bit per channel input. 1 means that the input is HIGH.

CTI Channel Out Status Register

The CSCTI_CTICHOUTSTATUS register only has meaning if the trigger source drives static levels, rather than pulses.

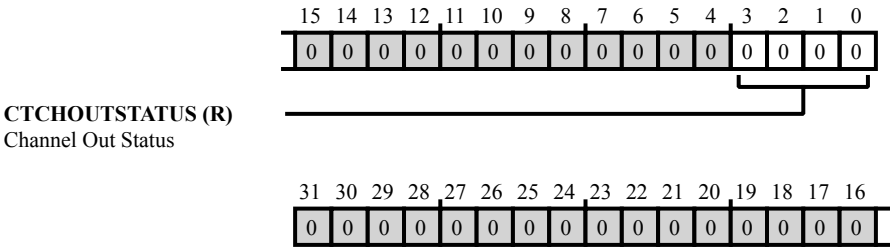


Figure 51-55: CSCTI_CTICHOUTSTATUS Register Diagram

Table 51-62: CSCTI_CTICHOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/NW)	CTCHOUTSTATUS	Channel Out Status. Channel output status. 0: One bit per channel output. 0 means that the output is LOW. 1: One bit per channel output. 1 means that the output is HIGH.

CTI Control Register

The CSCTI_CTICONTROL register enables and disables the CTI.

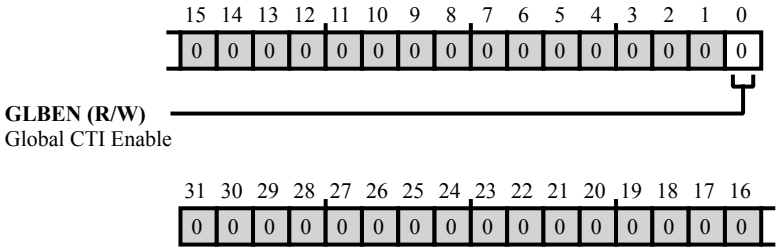


Figure 51-56: CSCTI_CTICONTROL Register Diagram

Table 51-63: CSCTI_CTICONTROL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
0 (R/W)	GLBEN	Global CTI Enable.	
		0	Disable
		1	Enable

Enable CTI Channel Gate Register

The CSCTI_CTIGATE register enables the propagation of channel events out of the CTI, one bit per channel.

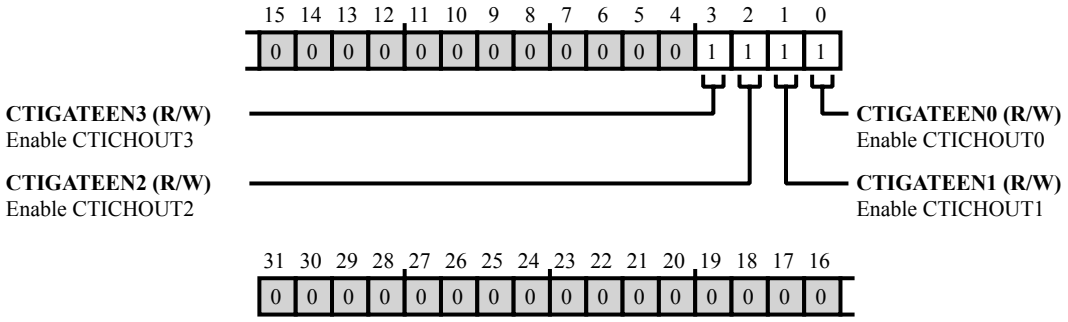


Figure 51-57: CSCTI_CTIGATE Register Diagram

Table 51-64: CSCTI_CTIGATE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3 (R/W)	CTIGATEEN3	Enable CTICHOUT3. 0: Disable channel 3 from propagating. 1: Enable channel 3 propagation.
2 (R/W)	CTIGATEEN2	Enable CTICHOUT2. 0: Disable channel 2 from propagating. 1: Enable channel 2 propagation.
1 (R/W)	CTIGATEEN1	Enable CTICHOUT1. 0: Disable channel 1 from propagating. 1: Enable channel 1 propagation.
0 (R/W)	CTIGATEEN0	Enable CTICHOUT0. 0: Disable channel 0 from propagating. 1: Enable channel 0 propagation.

CTI Trigger 0 to Channel Enable Register

The CSCTI_CTIINEN0 register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (0xF). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

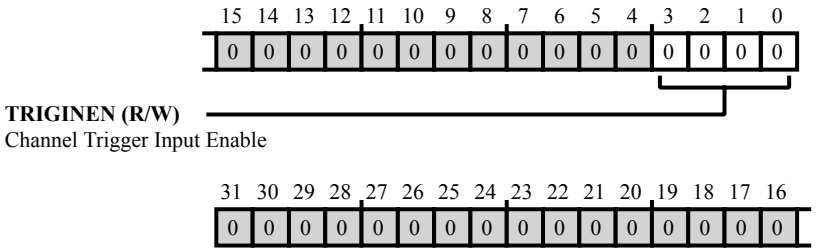


Figure 51-58: CSCTI_CTIINEN0 Register Diagram

Table 51-65: CSCTI_CTIINEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGINEN	Channel Trigger Input Enable. Trigger input to channel mapping. 0: Input trigger 0 events are ignored by the corresponding channel. 1: When an event is received on event_in[0], generate an event on the channel corresponding to this bit.

CTI Trigger 1 to Channel Enable Register

The CSCTI_CTIINEN1 register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (0xF). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

CTIINEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

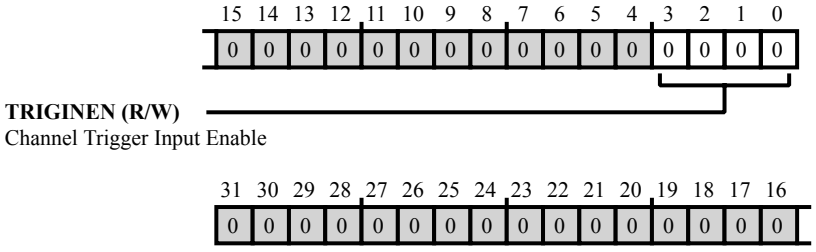


Figure 51-59: CSCTI_CTIINEN1 Register Diagram

Table 51-66: CSCTI_CTIINEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGINEN	Channel Trigger Input Enable. Trigger input to channel mapping. 0: Input trigger 1 events are ignored by the corresponding channel. 1: When an event is received on event_in[1], generate an event on the channel corresponding to this bit.

CTI Trigger 2 to Channel Enable Register

The CSCTI_CTIINEN2 register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (0xF). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

CTIINEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

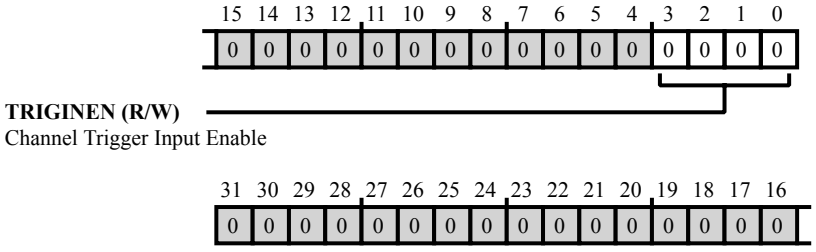


Figure 51-60: CSCTI_CTIINEN2 Register Diagram

Table 51-67: CSCTI_CTIINEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGINEN	Channel Trigger Input Enable. Trigger input to channel mapping. 0: Input trigger 2 events are ignored by the corresponding channel. 1: When an event is received on event_in[2], generate an event on the channel corresponding to this bit.

CTI Trigger 3 to Channel Enable Register

The CSCTI_CTIINEN3 register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (0xF). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

CTIINEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

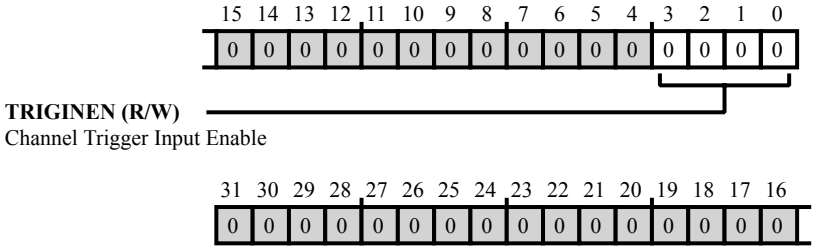


Figure 51-61: CSCTI_CTIINEN3 Register Diagram

Table 51-68: CSCTI_CTIINEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGINEN	Channel Trigger Input Enable. Trigger input to channel mapping. 0: Input trigger 3 events are ignored by the corresponding channel. 1: When an event is received on event_in[3], generate an event on the channel corresponding to this bit.

CTI Trigger 4 to Channel Enable Register

The CSCTI_CTIINEN4 register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (0xF). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

CTIINEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

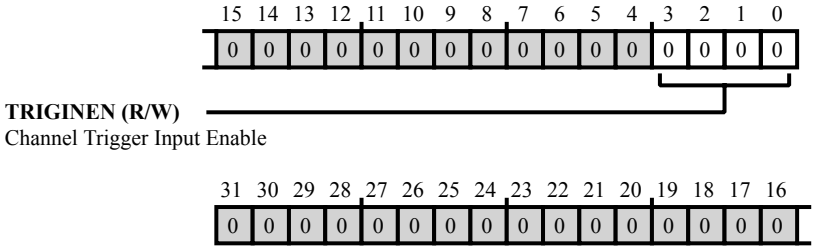


Figure 51-62: CSCTI_CTIINEN4 Register Diagram

Table 51-69: CSCTI_CTIINEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGINEN	Channel Trigger Input Enable. Trigger input to channel mapping. 0: Input trigger 4 events are ignored by the corresponding channel. 1: When an event is received on event_in[4], generate an event on the channel corresponding to this bit.

CTI Trigger 5 to Channel Enable Register

The CSCTI_CTIINEN5 register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (0xF). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

CTIINEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

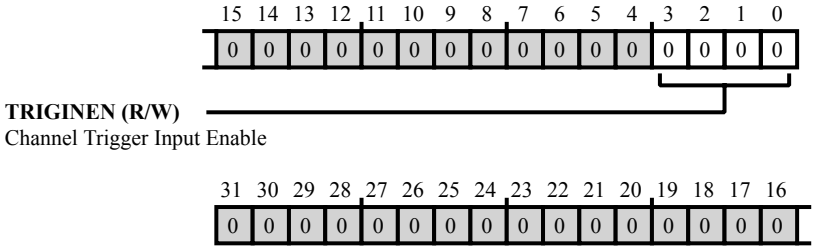


Figure 51-63: CSCTI_CTIINEN5 Register Diagram

Table 51-70: CSCTI_CTIINEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGINEN	Channel Trigger Input Enable. Trigger input to channel mapping. 0: Input trigger 5 events are ignored by the corresponding channel. 1: When an event is received on event_in[5], generate an event on the channel corresponding to this bit.

CTI Trigger 6 to Channel Enable Register

The CSCTI_CTIINEN6 register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (0xF). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

CTIINEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

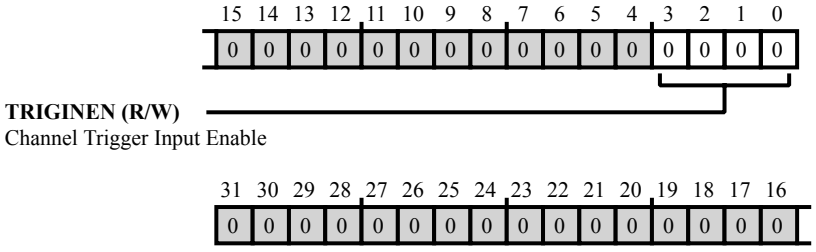


Figure 51-64: CSCTI_CTIINEN6 Register Diagram

Table 51-71: CSCTI_CTIINEN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGINEN	Channel Trigger Input Enable. Trigger input to channel mapping. 0: Input trigger 6 events are ignored by the corresponding channel. 1: When an event is received on event_in[6], generate an event on the channel corresponding to this bit.

CTI Trigger 7 to Channel Enable Register

The CSCTI_CTIINEN7 register maps trigger inputs to channels in the cross trigger system. The CTIINEN registers are bit maps that allow the trigger input to be mapped to any channel output, including none (0x0) and all (0xF). There is one register per trigger input, so it is possible to map any combination of trigger inputs to any channel outputs.

CTIINEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

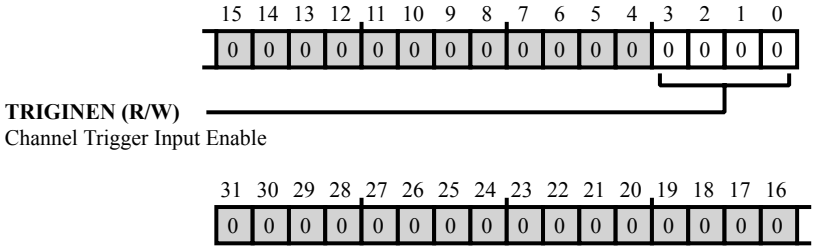


Figure 51-65: CSCTI_CTIINEN7 Register Diagram

Table 51-72: CSCTI_CTIINEN7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGINEN	Channel Trigger Input Enable. Trigger input to channel mapping. 0: Input trigger 7 events are ignored by the corresponding channel. 1: When an event is received on event_in[7], generate an event on the channel corresponding to this bit.

CTI Interrupt Acknowledge Register

The CSCTI_CTIINTACK register acknowledges trigger outputs.

It is a bit map that allows selective clearing of trigger output events. If the SW_HANDSHAKE parameter for a trigger output is set, indicating that the output latches HIGH when an event is sent to that output, then the output remains HIGH until the corresponding INTACK bit is written to a 1. A write of a bit to 0 has no effect. This allows different software threads to be responsible for clearing different trigger outputs without needing to perform a read-modify-write operation to find which bits are set.

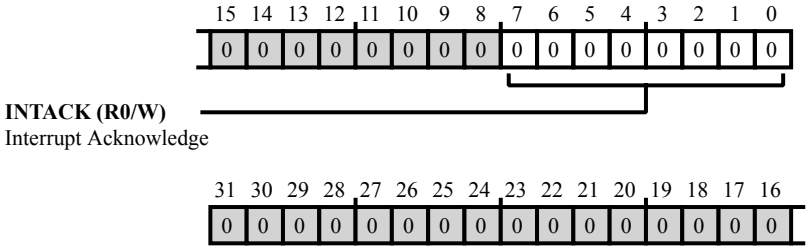


Figure 51-66: CSCTI_CTIINTACK Register Diagram

Table 51-73: CSCTI_CTIINTACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R0/W)	INTACK	Interrupt Acknowledge. The CSCTI_CTIINTACK.INTACK bit field acknowledges the corresponding event output. It is write only.

CTI Channel to Trigger 0 Enable Register

The CSCTI_CTIOUTEN0 register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (0xF). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

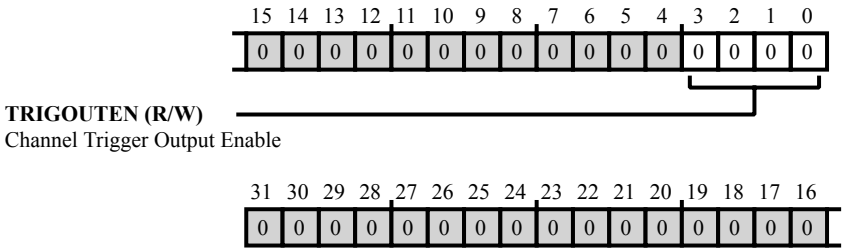


Figure 51-67: CSCTI_CTIOUTEN0 Register Diagram

Table 51-74: CSCTI_CTIOUTEN0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGOUTEN	Channel Trigger Output Enable. Channel to trigger output mapping. 0: The corresponding channel is ignored by the output trigger0. 1: When an event occurs on the channel corresponding to this bit, generate an event on event_out[0].

CTI Channel to Trigger 1 Enable Register

The CSCTI_CTIOUTEN1 register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (0xF). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

CTIOUTEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

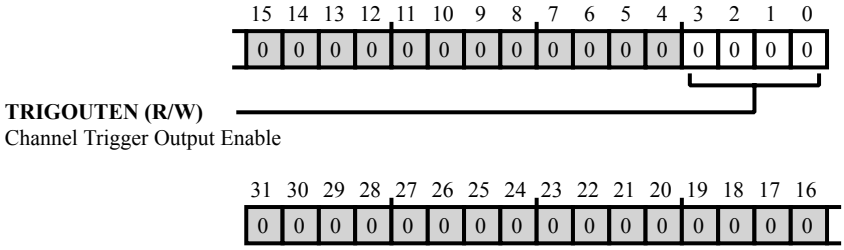


Figure 51-68: CSCTI_CTIOUTEN1 Register Diagram

Table 51-75: CSCTI_CTIOUTEN1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGOUTEN	Channel Trigger Output Enable. Channel to trigger output mapping. 0: The corresponding channel is ignored by the output trigger1. 1: When an event occurs on the channel corresponding to this bit, generate an event on event_out[1].

CTI Channel to Trigger 2 Enable Register

The CSCTI_CTIOUTEN2 register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (0xF). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

CTIOUTEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

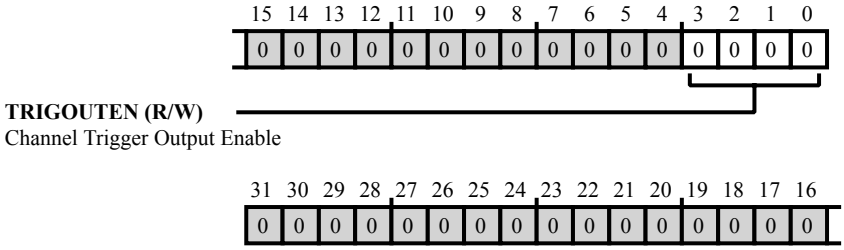


Figure 51-69: CSCTI_CTIOUTEN2 Register Diagram

Table 51-76: CSCTI_CTIOUTEN2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGOUTEN	Channel Trigger Output Enable. Channel to trigger output mapping. 0: The corresponding channel is ignored by the output trigger2. 1: When an event occurs on the channel corresponding to this bit, generate an event on event_out[2].

CTI Channel to Trigger 3 Enable Register

The CSCTI_CTIOUTEN3 register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (0xF). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

CTIOUTEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

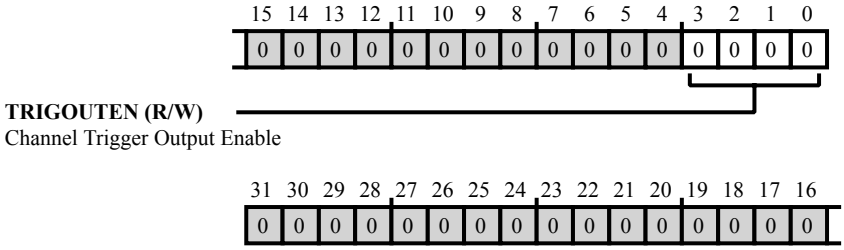


Figure 51-70: CSCTI_CTIOUTEN3 Register Diagram

Table 51-77: CSCTI_CTIOUTEN3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGOUTEN	Channel Trigger Output Enable. Channel to trigger output mapping. 0: The corresponding channel is ignored by the output trigger3. 1: When an event occurs on the channel corresponding to this bit, generate an event on event_out[3].

CTI Channel to Trigger 4 Enable Register

The CSCTI_CTIOUTEN4 register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (0xF). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

CTIOUTEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

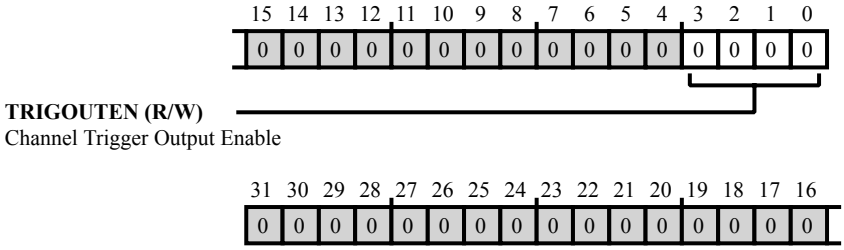


Figure 51-71: CSCTI_CTIOUTEN4 Register Diagram

Table 51-78: CSCTI_CTIOUTEN4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGOUTEN	Channel Trigger Output Enable. Channel to trigger output mapping. 0: The corresponding channel is ignored by the output trigger4. 1: When an event occurs on the channel corresponding to this bit, generate an event on event_out[4].

CTI Channel to Trigger 5 Enable Register

The CSCTI_CTIOUTEN5 register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (0xF). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

CTIOUTEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

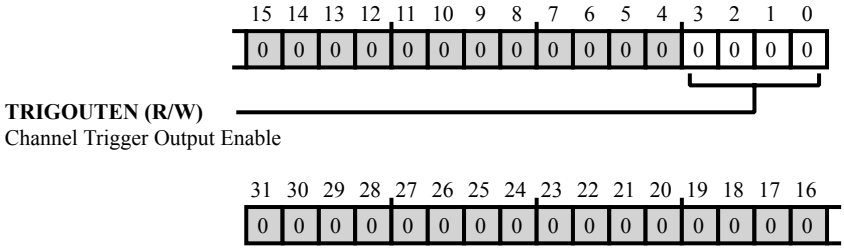


Figure 51-72: CSCTI_CTIOUTEN5 Register Diagram

Table 51-79: CSCTI_CTIOUTEN5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGOUTEN	Channel Trigger Output Enable. Channel to trigger output mapping. 0: The corresponding channel is ignored by the output trigger5. 1: When an event occurs on the channel corresponding to this bit, generate an event on event_out[5].

CTI Channel to Trigger 6 Enable Register

The CSCTI_CTIOUTEN6 register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (0xF). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

CTIOUTEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

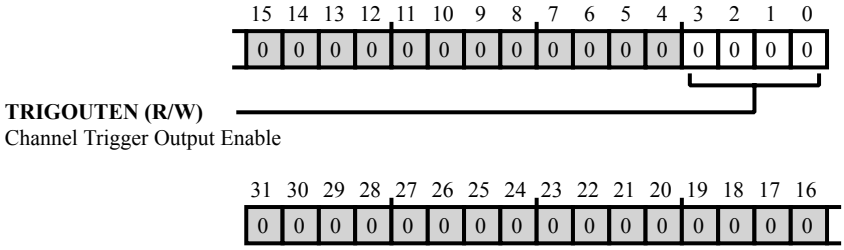


Figure 51-73: CSCTI_CTIOUTEN6 Register Diagram

Table 51-80: CSCTI_CTIOUTEN6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGOUTEN	Channel Trigger Output Enable. Channel to trigger output mapping. 0: The corresponding channel is ignored by the output trigger6. 1: When an event occurs on the channel corresponding to this bit, generate an event on event_out[6].

CTI Channel to Trigger 7 Enable Register

The CSCTI_CTIOUTEN7 register maps channels in the cross trigger system to trigger outputs. The CTIOUTEN registers are bit maps that allow any channel input to be mapped to the trigger output, including none (0x0) and all (0xF). There is one register per trigger output so it is possible to map any channel input to any combination of trigger outputs.

CTIOUTEN1-31 registers are IMPLEMENTATION-DEFINED. If they are not implemented, the locations are RO and return 0. If they are implemented, they are RW and return a reset value of 0.

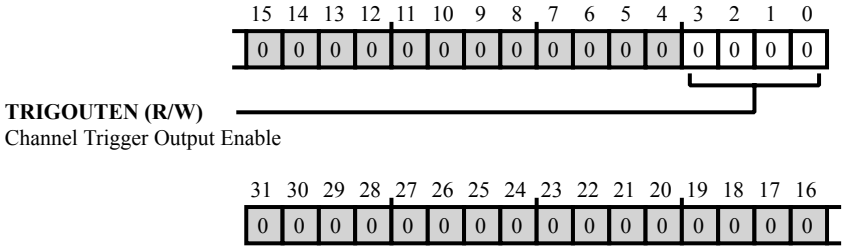


Figure 51-74: CSCTI_CTIOUTEN7 Register Diagram

Table 51-81: CSCTI_CTIOUTEN7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/W)	TRIGOUTEN	Channel Trigger Output Enable. Channel to trigger output mapping. 0: The corresponding channel is ignored by the output trigger7. 1: When an event occurs on the channel corresponding to this bit, generate an event on event_out[7].

CTI Trigger In Status Register

If the event_in input is driven by a source that generates single cycle pulses, the CSCTI_CTITRIGINSTATUS register is generally read as 0.

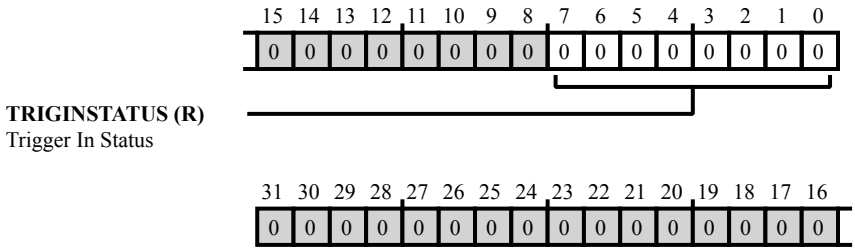


Figure 51-75: CSCTI_CTITRIGINSTATUS Register Diagram

Table 51-82: CSCTI_CTITRIGINSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	TRIGINSTATUS	Trigger In Status. Trigger input status. 0: One bit per trigger input. 0 means that the input is LOW. 1: One bit per trigger input. 1 means that the input is HIGH.

CTI Trigger Out Status Register

The CSCTI_CTITRIGOUTSTATUS register only has meaning if the trigger source drives static levels, rather than pulses.

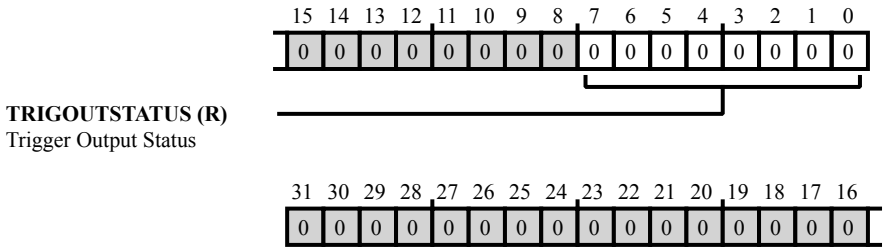


Figure 51-76: CSCTI_CTITRIGOUTSTATUS Register Diagram

Table 51-83: CSCTI_CTITRIGOUTSTATUS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	TRIGOUTSTATUS	Trigger Output Status. Trigger output status. 0: One bit per trigger output. 0 means that the output is LOW. 1: One bit per trigger output. 1 means that the output is HIGH.

Device ID

The CSCTI_DEVID register is implementation defined for each part number and designer. The register indicates the capabilities of the component.

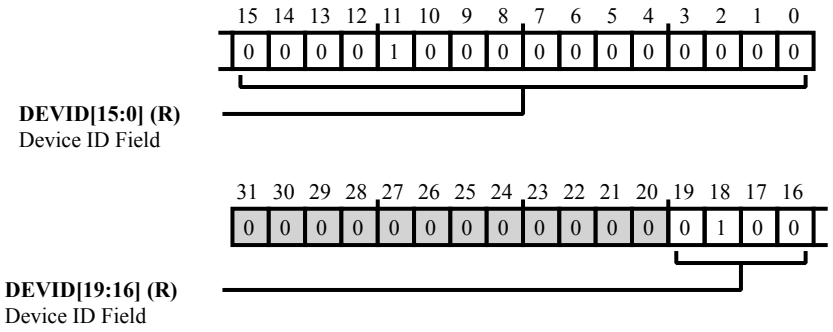


Figure 51-77: CSCTI_DEVID Register Diagram

Table 51-84: CSCTI_DEVID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
19:0 (R/NW)	DEVID	Device ID Field. Bits[19:16] Indicates the maximum number of triggers - the maximum of the two parameters, NUM_EVENT_SLAVES and NUM_EVENT_MASTERS. Bits[15:8] Indicates the maximum number of triggers - the maximum of the two parameters, NUM_EVENT_SLAVES and NUM_EVENT_MASTERS. Bits[7:5] Reserved bit or field with SBZP behavior Bits[4:0] Indicates the value of the EXT_MUX_NUM parameter, which determines if there is any external multiplexing on the trigger inputs and outputs. 0 indicates no multiplexing.

Device Type

A debugger can use the CSCTI_DEVTYPE register to get information about a component that has an unrecognized part number.

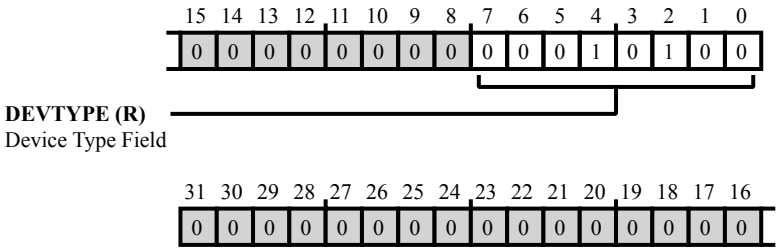


Figure 51-78: CSCTI_DEVTYPE Register Diagram

Table 51-85: CSCTI_DEVTYPE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	DEVTYPE	Device Type Field. Bits[7:4] Minor classification. Returns 0x1, indicating this component is a Trigger-Matrix. Bits[3:0] Major classification. Returns 0x4, indicating this component performs Debug Control.

Integration Test Channel Input

The CSCTI_ITCHIN register is used to view channel events. The integration test register includes a latch that is set when a pulse is received on a channel input. When read, a register bit reads as 1 if the channel has received a pulse since it was last read. The act of reading the register automatically clears the 1 to a 0. When performing integration testing it is therefore important to coordinate the setting of event latches and reading/clearing them.

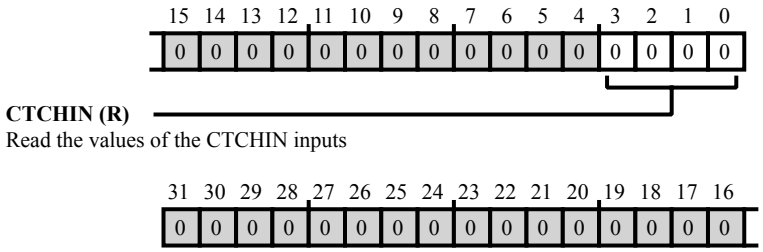


Figure 51-79: CSCTI_ITCHIN Register Diagram

Table 51-86: CSCTI_ITCHIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/NW)	CTCHIN	Read the values of the CTCHIN inputs. Reads the latched value of the channel inputs.

ITCHINACK

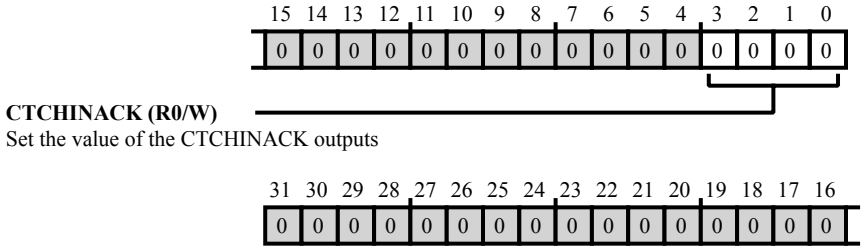


Figure 51-80: CSCTI_ITCHINACK Register Diagram

Table 51-87: CSCTI_ITCHINACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R0/W)	CTCHINACK	Set the value of the CTCHINACK outputs.

Integration Test Channel Output

The CSCTI_ITCHOUT register is used to generate channel events. Writing to the register creates a single pulse on the output. The CSCTI_ITCHOUT register is self-clearing.

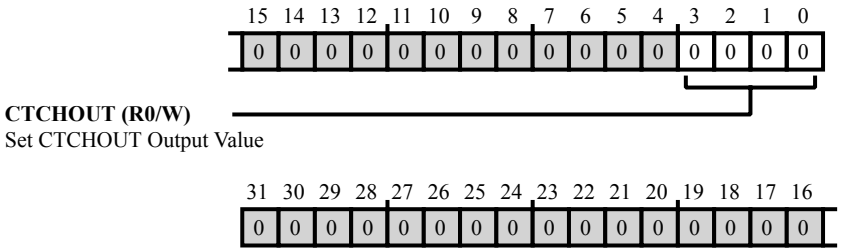


Figure 51-81: CSCTI_ITCHOUT Register Diagram

Table 51-88: CSCTI_ITCHOUT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R0/W)	CTCHOUT	Set CTCHOUT Output Value. Pulses the channel outputs. 0: No effect. 1: Pulse channel event for one clk cycle.

ITCHOUTACK

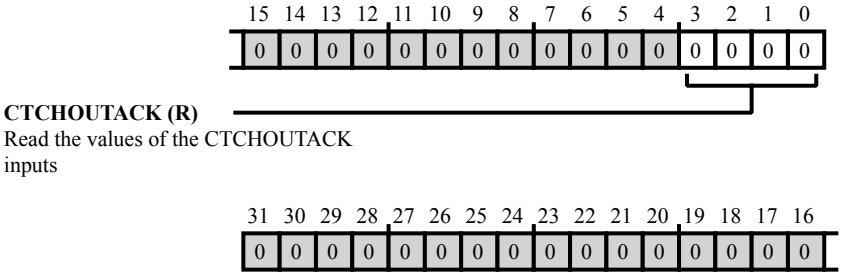


Figure 51-82: CSCTI_ITCHOUTACK Register Diagram

Table 51-89: CSCTI_ITCHOUTACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
3:0 (R/NW)	CTCHOUTACK	Read the values of the CTCHOUTACK inputs.

Integration Mode Control Register

The CSCTI_ITCTRL register is used to enable topology detection.

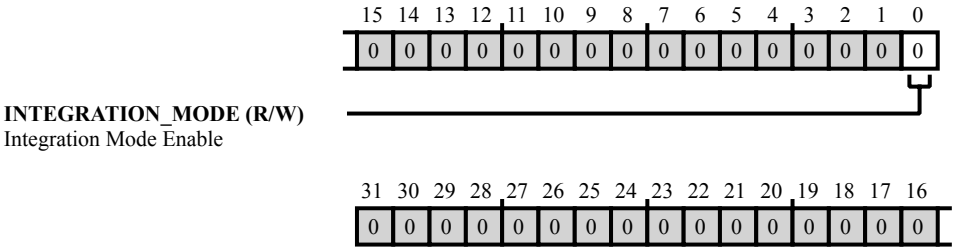


Figure 51-83: CSCTI_ITCTRL Register Diagram

Table 51-90: CSCTI_ITCTRL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
0 (R/W)	INTEGRATION_MODE	Integration Mode Enable. When set, the component enters integration mode, enabling topology detection or integration testing to be performed.

Integration Test Trigger Input

The CSCTI_ITTRIGIN register is used to view trigger events. The integration test register includes a latch that is set when a pulse is received on a trigger input. When read, a register bit reads as 1 if the trigger input has received a pulse since it was last read. The act of reading the register automatically clears the 1 to a 0. When performing integration testing it is therefore important to coordinate the setting of event latches and reading/clearing them.

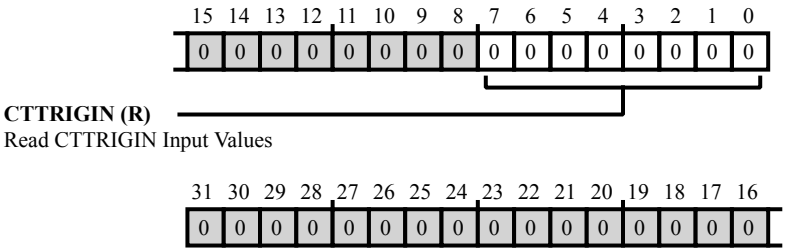


Figure 51-84: CSCTI_ITTRIGIN Register Diagram

Table 51-91: CSCTI_ITTRIGIN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	CTTRIGIN	Read CTTRIGIN Input Values. Reads the latched value of the trigger inputs.

ITTRIGINACK

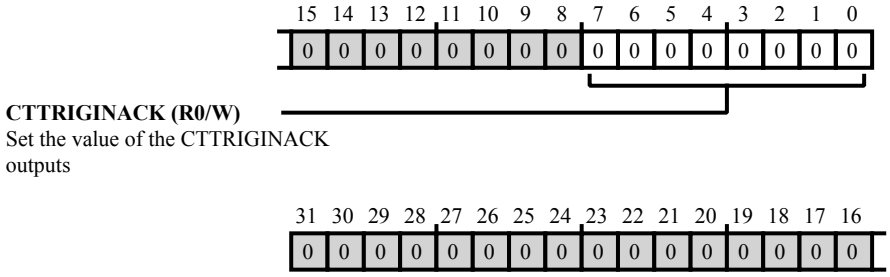


Figure 51-85: CSCTI_ITTRIGINACK Register Diagram

Table 51-92: CSCTI_ITTRIGINACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R0/W)	CTTRIGINACK	Set the value of the CTTRIGINACK outputs.

Integration Test Trigger Output

The CSCTI_ITTRIGOUT register is used to generate trigger events.

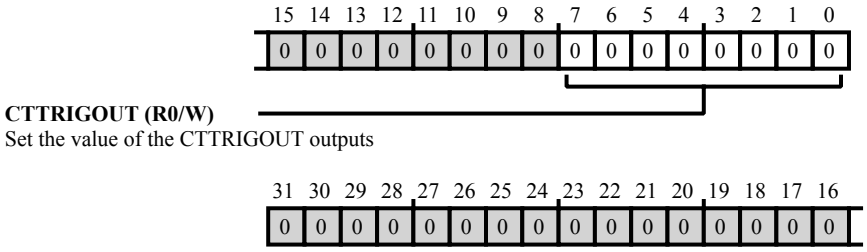


Figure 51-86: CSCTI_ITTRIGOUT Register Diagram

Table 51-93: CSCTI_ITTRIGOUT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R0/W)	CTTRIGOUT	Set the value of the CTTRIGOUT outputs. Set/clear trigger output signal. Reads return the value in the register if SW_HANDSHAKE=1, otherwise 0 is returned if SW_HANDSHAKE=0. Writes: 0: Clears the trigger output if SW_HANDSHAKE=1, no effect if SW_HANDSHAKE=0. 1: Sets the trigger output if SW_HANDSHAKE=1, pulses trigger output if SW_HANDSHAKE=0.

ITTRIGOUTACK

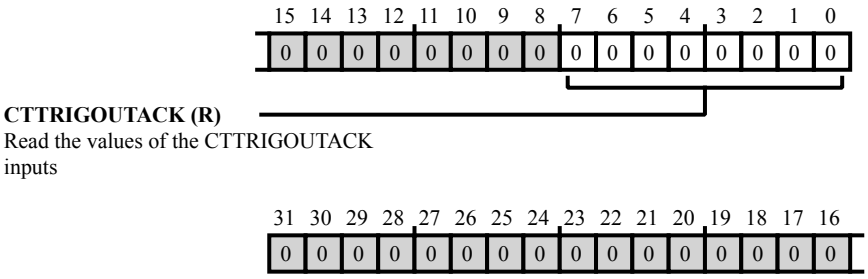


Figure 51-87: CSCTI_ITTRIGOUTACK Register Diagram

Table 51-94: CSCTI_ITTRIGOUTACK Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	CTTRIGOUTACK	Read the values of the CTTRIGOUTACK inputs.

Lock Access Register

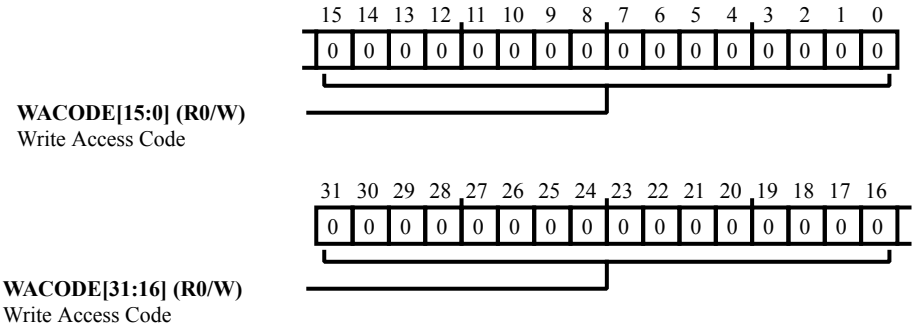


Figure 51-88: CSCTI_LAR Register Diagram

Table 51-95: CSCTI_LAR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R0/W)	WACODE	Write Access Code.

Lock Status Register

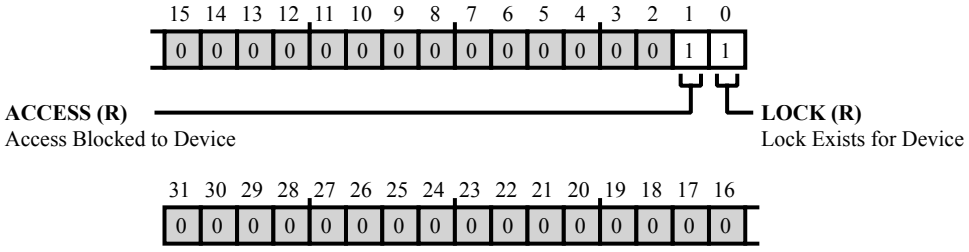


Figure 51-89: CSCTI_LSR Register Diagram

Table 51-96: CSCTI_LSR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/NW)	ACCESS	Access Blocked to Device.
0 (R/NW)	LOCK	Lock Exists for Device.

Peripheral ID0

The CSCTI_PERIPHID0 register is part of the set of peripheral identification registers.

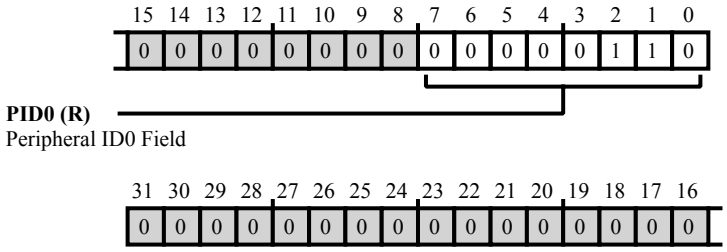


Figure 51-90: CSCTI_PERIPHID0 Register Diagram

Table 51-97: CSCTI_PERIPHID0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PID0	Peripheral ID0 Field. Part number, bits[7:0]. Taken together with PIDR1.PART_1 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral ID1

The CSCTI_PERIPHID1 register is part of the set of peripheral identification registers.

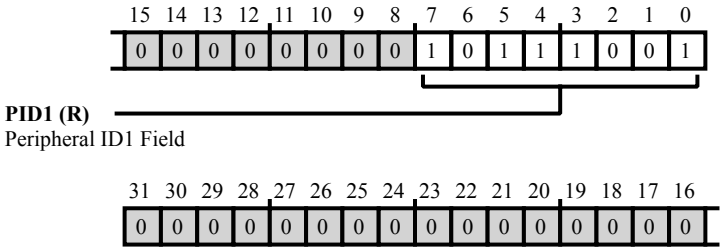


Figure 51-91: CSCTI_PERIPHID1 Register Diagram

Table 51-98: CSCTI_PERIPHID1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PID1	Peripheral ID1 Field. Bits[7:4] JEP106 identification code, bits[3:0]. Together, with PIDR4.DES_2 and PIDR2.DES_1, they indicate the designer of the component and not the implementer, except where the two are the same. Bits[3:0] Part number, bits[11:8]. Taken together with PIDR0.PART_0 it indicates the component. The Part Number is selected by the designer of the component.

Peripheral ID2

The CSCTI_PERIPHID2 register is part of the set of peripheral identification registers.

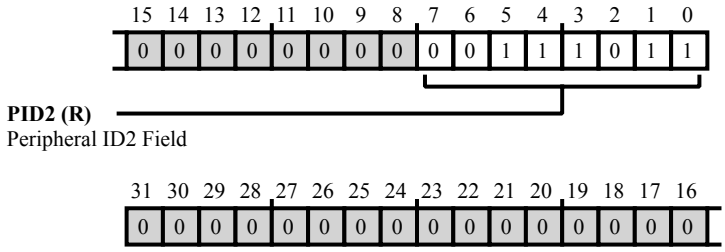


Figure 51-92: CSCTI_PERIPHID2 Register Diagram

Table 51-99: CSCTI_PERIPHID2 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PID2	Peripheral ID2 Field. Bits[7:4] Revision. It is an incremental value starting at 0x0 for the first design of a component. See the Component list in Chapter 1 for information on the RTL revision of the component. Bit[3] 1 - Always set. Indicates that a JEDEC assigned value is used. Bits[2:0] JEP106 identification code, bits[6:4]. Together, with PIDR4.DES_2 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral ID3

The CSCTI_PERIPHID3 register is part of the set of peripheral identification registers.

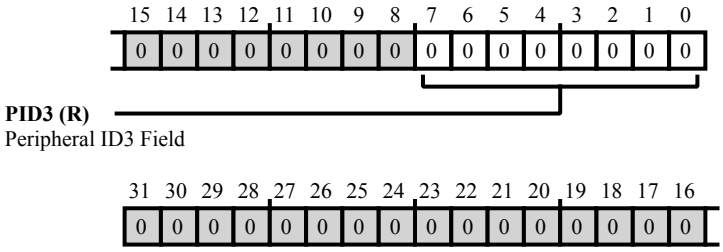


Figure 51-93: CSCTI_PERIPHID3 Register Diagram

Table 51-100: CSCTI_PERIPHID3 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PID3	Peripheral ID3 Field. Bits[7:4] This field indicates minor errata fixes specific to this design, for example metal fixes after implementation. In most cases this field is 0x0. Bits[3:0] Customer Modified. Where the component is reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

Peripheral ID4

The CSCTI_PERIPHID4 register is part of the set of peripheral identification registers.

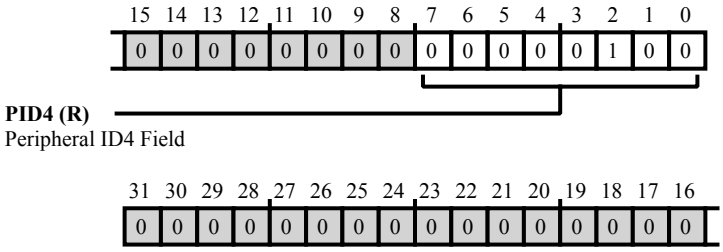


Figure 51-94: CSCTI_PERIPHID4 Register Diagram

Table 51-101: CSCTI_PERIPHID4 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PID4	Peripheral ID4 Field. Bits[7:4] Indicates the memory size that is used by this component. Returns 0 indicating that the component uses an UNKNOWN number of 4KB blocks. Using the SIZE field to indicate the size of the component is deprecated. Bits[3:0] JEP106 continuation code. Together, with PIDR2.DES_1 and PIDR1.DES_0, they indicate the designer of the component and not the implementer, except where the two are the same.

Peripheral ID5

The CSCTI_PERIPHID5 register is part of the set of peripheral identification registers.

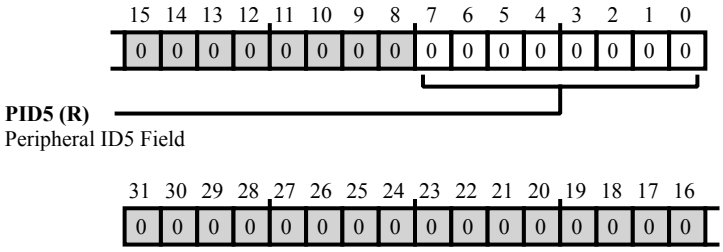


Figure 51-95: CSCTI_PERIPHID5 Register Diagram

Table 51-102: CSCTI_PERIPHID5 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PID5	Peripheral ID5 Field.

Peripheral ID6

The CSCTI_PERIPHID6 register is part of the set of peripheral identification registers.

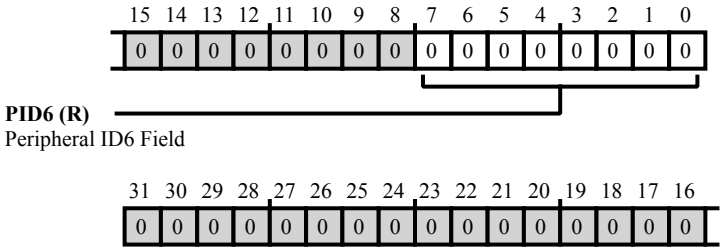


Figure 51-96: CSCTI_PERIPHID6 Register Diagram

Table 51-103: CSCTI_PERIPHID6 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PID6	Peripheral ID6 Field.

Peripheral ID7

The CSCTI_PERIPHID7 register is part of the set of peripheral identification registers.

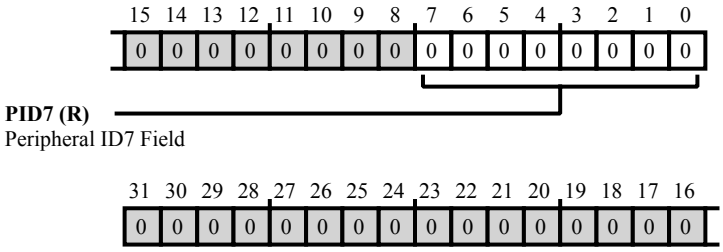


Figure 51-97: CSCTI_PERIPHID7 Register Diagram

Table 51-104: CSCTI_PERIPHID7 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/NW)	PID7	Peripheral ID7 Field.

Appendix A ADSP-SC59x Register List

This appendix lists Memory-Mapped Register address and register names. The modules are presented in alphabetical order.

Table A-1: ADSP-SC59x CSETF0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31109004	CSETF0_RSZ	CSETF0 RAM Size register	0x00000080
0x3110900C	CSETF0_STS	CSETF0 Status register	0x00000004
0x31109010	CSETF0_RRD	CSETF0 RAM Read Data register	0x00000000
0x31109014	CSETF0_RRP	CSETF0 RAM Read Pointer register	0x00000000
0x31109018	CSETF0_RWP	CSETF0 RAM Write Pointer register	0x00000000
0x3110901C	CSETF0_TRG	CSETF0 Trigger Counter register	0x00000000
0x31109020	CSETF0_CTL	CSETF0 Control Register	0x00000000
0x31109024	CSETF0_RWD	CSETF0 RAM Write Data register	0x00000000
0x31109028	CSETF0_MODE	CSETF0 Mode register	0x00000000
0x3110902C	CSETF0_LBUFLEVEL	CSETF0 Latched Buffer Fill Level	0x00000000
0x31109030	CSETF0_CBUFLEVEL	CSETF0 Current Buffer Fill Level	0x00000000
0x31109034	CSETF0_BUFWM	CSETF0 Buffer Level Water Mark	0x00000000
0x31109300	CSETF0_FFSR	CSETF0 Formatter and Flush Status Register	0x00000000
0x31109304	CSETF0_FFCR	CSETF0 Formatter and Flush Control Register	0x00000000
0x31109308	CSETF0_PSCR	CSETF0 Periodic Synchronization Counter Register	0x0000000A
0x31109ED0	CSETF0_ITATBMDATA0	CSETF0 Integration Test ATB Master Data 0 register	0x00000000
0x31109ED4	CSETF0_ITATBMCTR2	CSETF0 Integration Test ATB Master Control 2 register	0x00000000
0x31109ED8	CSETF0_ITATBMCTR1	CSETF0 Integration Test ATB Master Control 1 register	0x00000000
0x31109EDC	CSETF0_ITATBMCTR0	CSETF0 Integration Test ATB Master Control 0 register	0x00000000
0x31109EE0	CSETF0_ITEVTINTR	CSETF0 Integration Test Event and Interrupt Control Register	0x00000000
0x31109EE8	CSETF0_ITTRFLIN	CSETF0 Integration Test Trigger In and Flush In register	0x00000000
0x31109EEC	CSETF0_ITATBDATA0	CSETF0 Integration Test ATB Data 0 Register	0x00000000
0x31109EF0	CSETF0_ITATBCTR2	CSETF0 Integration Test ATB Control 2 Register	0x00000000
0x31109EF4	CSETF0_ITATBCTR1	CSETF0 Integration Test ATB Control 1 Register	0x00000000
0x31109EF8	CSETF0_ITATBCTR0	CSETF0 Integration Test ATB Control 0 Register	0x00000000
0x31109F00	CSETF0_ITCTRL	CSETF0 Integration Mode Control Register	0x00000000

Table A-1: ADSP-SC59x CSETF0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31109FA0	CSETF0_CLAIMSET	CSETF0 Claim Tag Set Register	0x0000000F
0x31109FA4	CSETF0_CLAIMCLR	CSETF0 Claim Tag Clear Register	0x00000000
0x31109FB8	CSETF0_AUTHSTATUS	CSETF0 Authentication Status Register	0x00000000
0x31109FC4	CSETF0_DEVID1	CSETF0 Device Configuration Register 1	0x00000001
0x31109FC8	CSETF0_DEVID	CSETF0 Device Configuration Register	0x00000380
0x31109FCC	CSETF0_DEVTYPE	CSETF0 Device Type Identifier Register	0x00000032
0x31109FD0	CSETF0_PIDR4	CSETF0 Peripheral Identification Register 4	0x00000004
0x31109FD4	CSETF0_PIDR5	CSETF0 Peripheral Identification Register 5	0x00000000
0x31109FD8	CSETF0_PIDR6	CSETF0 Peripheral Identification Register 6	0x00000000
0x31109FDC	CSETF0_PIDR7	CSETF0 Peripheral Identification Register 7	0x00000000
0x31109FE0	CSETF0_PIDR0	CSETF0 Peripheral Identification Register 0	0x000000EA
0x31109FE4	CSETF0_PIDR1	CSETF0 Peripheral Identification Register 1	0x000000B9
0x31109FE8	CSETF0_PIDR2	CSETF0 Peripheral Identification Register 2	0x0000004B
0x31109FEC	CSETF0_PIDR3	CSETF0 Peripheral Identification Register 3	0x00000000
0x31109FF0	CSETF0_CIDR0	CSETF0 Component Identification Register 0	0x0000000D
0x31109FF4	CSETF0_CIDR1	CSETF0 Component Identification Register 1	0x00000090
0x31109FF8	CSETF0_CIDR2	CSETF0 Component Identification Register 2	0x00000005
0x31109FFC	CSETF0_CIDR3	CSETF0 Component Identification Register 3	0x000000B1

Table A-2: ADSP-SC59x CSETR1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3110A004	CSETR1_RSZ	CSETR1 RAM Size register	0x00000000
0x3110A00C	CSETR1_STS	CSETR1 Status register	0x00000004
0x3110A010	CSETR1_RRD	CSETR1 RAM Read Data register	0x00000000
0x3110A014	CSETR1_RRP	CSETR1 RAM Read Pointer register	0x00000000
0x3110A018	CSETR1_RWP	CSETR1 RAM Write Pointer register	0x00000000
0x3110A01C	CSETR1_TRG	CSETR1 Trigger Counter register	0x00000000
0x3110A020	CSETR1_CTL	CSETR1 Control Register	0x00000000
0x3110A024	CSETR1_RWD	CSETR1 RAM Write Data register	0x00000000
0x3110A028	CSETR1_MODE	CSETR1 Mode register	0x00000000

Table A-2: ADSP-SC59x CSETR1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110A02C	CSETR1_LBUFLEVEL	CSETR1 Latched Buffer Fill Level	0x00000000
0x3110A030	CSETR1_CBUFLEVEL	CSETR1 Current Buffer Fill Level	0x00000000
0x3110A034	CSETR1_BUFWM	CSETR1 Buffer Level Water Mark	0x00000000
0x3110A038	CSETR1_RRPHI	CSETR1 RAM Read Pointer High register	0x00000000
0x3110A03C	CSETR1_RWPHI	CSETR1 RAM Write Pointer High register	0x00000000
0x3110A110	CSETR1_AXICTL	CSETR1 AXI Control Register	0x00000000
0x3110A118	CSETR1_DBALO	CSETR1 Data Buffer Address Low register	0x00000000
0x3110A11C	CSETR1_DBAHI	CSETR1 Data Buffer Address HIGH register	0x00000000
0x3110A120	CSETR1_RURP	CSETR1 RAM Update Read Pointer register	0x00000000
0x3110A300	CSETR1_FFSR	CSETR1 Formatter and Flush Status Register	0x00000000
0x3110A304	CSETR1_FFCR	CSETR1 Formatter and Flush Control Register	0x00000000
0x3110A308	CSETR1_PSCR	CSETR1 Periodic Synchronization Counter Register	0x0000000A
0x3110AEE0	CSETR1_ITEVTINTR	CSETR1 Integration Test Event and Interrupt Control Register	0x00000000
0x3110AEE8	CSETR1_ITTRFLIN	CSETR1 Integration Test Trigger In and Flush In register	0x00000000
0x3110AEEC	CSETR1_ITATBDATA0	CSETR1 Integration Test ATB Data 0 Register	0x00000000
0x3110AEF0	CSETR1_ITATBCTR2	CSETR1 Integration Test ATB Control 2 Register	0x00000000
0x3110AEF4	CSETR1_ITATBCTR1	CSETR1 Integration Test ATB Control 1 Register	0x00000000
0x3110AEF8	CSETR1_ITATBCTR0	CSETR1 Integration Test ATB Control 0 Register	0x00000000
0x3110AF00	CSETR1_ITCTRL	CSETR1 Integration Mode Control Register	0x00000000
0x3110AFA0	CSETR1_CLAIMSET	CSETR1 Claim Tag Set Register	0x0000000F
0x3110AFA4	CSETR1_CLAIMCLR	CSETR1 Claim Tag Clear Register	0x00000000
0x3110AFB8	CSETR1_AUTHSTATUS	CSETR1 Authentication Status Register	0x00000033
0x3110AFC4	CSETR1_DEVID1	CSETR1 Device Configuration Register 1	0x00000001
0x3110AFC8	CSETR1_DEVID	CSETR1 Device Configuration Register	0x03412A40
0x3110AFCC	CSETR1_DEVTYPE	CSETR1 Device Type Identifier Register	0x00000021
0x3110AFD0	CSETR1_PIDR4	CSETR1 Peripheral Identification Register 4	0x00000004
0x3110AFD4	CSETR1_PIDR5	CSETR1 Peripheral Identification Register 5	0x00000000
0x3110AFD8	CSETR1_PIDR6	CSETR1 Peripheral Identification Register 6	0x00000000
0x3110AFDC	CSETR1_PIDR7	CSETR1 Peripheral Identification Register 7	0x00000000
0x3110AFE0	CSETR1_PIDR0	CSETR1 Peripheral Identification Register 0	0x000000E8

Table A-2: ADSP-SC59x CSETR1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110AFE4	CSETR1_PIDR1	CSETR1 Peripheral Identification Register 1	0x000000B9
0x3110AFE8	CSETR1_PIDR2	CSETR1 Peripheral Identification Register 2	0x0000004B
0x3110AFEC	CSETR1_PIDR3	CSETR1 Peripheral Identification Register 3	0x00000000
0x3110AFF0	CSETR1_CIDR0	CSETR1 Component Identification Register 0	0x0000000D
0x3110AFF4	CSETR1_CIDR1	CSETR1 Component Identification Register 1	0x00000090
0x3110AFF8	CSETR1_CIDR2	CSETR1 Component Identification Register 2	0x00000005
0x3110AFFC	CSETR1_CIDR3	CSETR1 Component Identification Register 3	0x000000B1

Table A-3: ADSP-SC59x CSTF0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31108000	CSTF0_FUNNELCONTROL	CSTF0 Funnel Control register	0x00000300
0x31108004	CSTF0_PRIORITYCONTROL	CSTF0 Priority Control register	0x00000000
0x31108EEC	CSTF0_ITATBDATA0	CSTF0 Integration test data register	0x00000000
0x31108EF0	CSTF0_ITATBCTR3	CSTF0 Integration test control register 3	0x00000000
0x31108EF4	CSTF0_ITATBCTR2	CSTF0 Integration test control register 2	0x00000000
0x31108EF8	CSTF0_ITATBCTR1	CSTF0 Integration test control register 1	0x00000000
0x31108EFC	CSTF0_ITATBCTR0	CSTF0 Integration test control register 0	0x00000000
0x31108F00	CSTF0_ITCTRL	CSTF0 Integration Mode Control Register	0x00000000
0x31108FA0	CSTF0_CLAIMSET	CSTF0 Claim Tag Set Register	0x0000000F
0x31108FA4	CSTF0_CLAIMCLR	CSTF0 Claim Tag Clear Register	0x00000000
0x31108FA8	CSTF0_DEVAFF0	CSTF0 Device Affinity register 0	0x00000000
0x31108FAC	CSTF0_DEVAFF1	CSTF0 Device Affinity register 1	0x00000000
0x31108FB8	CSTF0_AUTHSTATUS	CSTF0 Authentication Status Register	0x00000000
0x31108FBC	CSTF0_DEVARCH	CSTF0 Device Architecture Register	0x00000000
0x31108FC0	CSTF0_DEVID2	CSTF0 Device Configuration Register 2	0x00000000
0x31108FC4	CSTF0_DEVID1	CSTF0 Device Configuration Register 1	0x00000000
0x31108FC8	CSTF0_DEVID	CSTF0 Device Configuration Register	0x00000034
0x31108FCC	CSTF0_DEVTYPE	CSTF0 Device Type Identifier Register	0x00000012
0x31108FD0	CSTF0_PIDR4	CSTF0 Peripheral Identification Register 4	0x00000004

Table A-3: ADSP-SC59x CSTF0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31108FD4	CSTF0_PIDR5	CSTF0 Peripheral Identification Register 5	0x00000000
0x31108FD8	CSTF0_PIDR6	CSTF0 Peripheral Identification Register 6	0x00000000
0x31108FDC	CSTF0_PIDR7	CSTF0 Peripheral Identification Register 7	0x00000000
0x31108FE0	CSTF0_PIDR0	CSTF0 Peripheral Identification Register 0	0x000000EB
0x31108FE4	CSTF0_PIDR1	CSTF0 Peripheral Identification Register 1	0x000000B9
0x31108FE8	CSTF0_PIDR2	CSTF0 Peripheral Identification Register 2	0x0000001B
0x31108FEC	CSTF0_PIDR3	CSTF0 Peripheral Identification Register 3	0x00000000
0x31108FF0	CSTF0_CIDR0	CSTF0 Component Identification Register 0	0x0000000D
0x31108FF4	CSTF0_CIDR1	CSTF0 Component Identification Register 1	0x00000090
0x31108FF8	CSTF0_CIDR2	CSTF0 Component Identification Register 2	0x00000005
0x31108FFC	CSTF0_CIDR3	CSTF0 Component Identification Register 3	0x000000B1

Table A-4: ADSP-SC59x A55CTI MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C000	A55CTI_CTICONTROL	A55CTI CTI Control register	0x00000000
0x3110C010	A55CTI_CTIINTACK	A55CTI CTI Interrupt Acknowledge register	0x00000000
0x3110C014	A55CTI_CTIAPPSET	A55CTI CTI Application Channel Set register	0x00000000
0x3110C018	A55CTI_CTIAPPCLEAR	A55CTI CTI Application Channel Clear register	0x00000000
0x3110C01C	A55CTI_CTIAPPPULSE	A55CTI CTI Application Channel Pulse register	0x00000000
0x3110C020	A55CTI_CTIINEN0	A55CTI CTI Trigger 0 to Channel Enable register	0x00000000
0x3110C024	A55CTI_CTIINEN1	A55CTI CTI Trigger 1 to Channel Enable register	0x00000000
0x3110C028	A55CTI_CTIINEN2	A55CTI CTI Trigger 2 to Channel Enable register	0x00000000
0x3110C02C	A55CTI_CTIINEN3	A55CTI CTI Trigger 3 to Channel Enable register	0x00000000
0x3110C030	A55CTI_CTIINEN4	A55CTI CTI Trigger 4 to Channel Enable register	0x00000000
0x3110C034	A55CTI_CTIINEN5	A55CTI CTI Trigger 5 to Channel Enable register	0x00000000
0x3110C038	A55CTI_CTIINEN6	A55CTI CTI Trigger 6 to Channel Enable register	0x00000000
0x3110C03C	A55CTI_CTIINEN7	A55CTI CTI Trigger 7 to Channel Enable register	0x00000000
0x3110C040	A55CTI_CTIINEN8	A55CTI CTI Trigger 8 to Channel Enable register	0x00000000
0x3110C044	A55CTI_CTIINEN9	A55CTI CTI Trigger 9 to Channel Enable register	0x00000000
0x3110C048	A55CTI_CTIINEN10	A55CTI CTI Trigger 10 to Channel Enable register	0x00000000

Table A-4: ADSP-SC59x A55CTI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C04C	A55CTI_CTIINEN11	A55CTI CTI Trigger 11 to Channel Enable register	0x00000000
0x3110C050	A55CTI_CTIINEN12	A55CTI CTI Trigger 12 to Channel Enable register	0x00000000
0x3110C054	A55CTI_CTIINEN13	A55CTI CTI Trigger 13 to Channel Enable register	0x00000000
0x3110C058	A55CTI_CTIINEN14	A55CTI CTI Trigger 14 to Channel Enable register	0x00000000
0x3110C05C	A55CTI_CTIINEN15	A55CTI CTI Trigger 15 to Channel Enable register	0x00000000
0x3110C060	A55CTI_CTIINEN16	A55CTI CTI Trigger 16 to Channel Enable register	0x00000000
0x3110C064	A55CTI_CTIINEN17	A55CTI CTI Trigger 17 to Channel Enable register	0x00000000
0x3110C068	A55CTI_CTIINEN18	A55CTI CTI Trigger 18 to Channel Enable register	0x00000000
0x3110C06C	A55CTI_CTIINEN19	A55CTI CTI Trigger 19 to Channel Enable register	0x00000000
0x3110C070	A55CTI_CTIINEN20	A55CTI CTI Trigger 20 to Channel Enable register	0x00000000
0x3110C074	A55CTI_CTIINEN21	A55CTI CTI Trigger 21 to Channel Enable register	0x00000000
0x3110C078	A55CTI_CTIINEN22	A55CTI CTI Trigger 22 to Channel Enable register	0x00000000
0x3110C07C	A55CTI_CTIINEN23	A55CTI CTI Trigger 23 to Channel Enable register	0x00000000
0x3110C080	A55CTI_CTIINEN24	A55CTI CTI Trigger 24 to Channel Enable register	0x00000000
0x3110C084	A55CTI_CTIINEN25	A55CTI CTI Trigger 25 to Channel Enable register	0x00000000
0x3110C088	A55CTI_CTIINEN26	A55CTI CTI Trigger 26 to Channel Enable register	0x00000000
0x3110C08C	A55CTI_CTIINEN27	A55CTI CTI Trigger 27 to Channel Enable register	0x00000000
0x3110C090	A55CTI_CTIINEN28	A55CTI CTI Trigger 28 to Channel Enable register	0x00000000
0x3110C094	A55CTI_CTIINEN29	A55CTI CTI Trigger 29 to Channel Enable register	0x00000000
0x3110C098	A55CTI_CTIINEN30	A55CTI CTI Trigger 30 to Channel Enable register	0x00000000
0x3110C09C	A55CTI_CTIINEN31	A55CTI CTI Trigger 31 to Channel Enable register	0x00000000
0x3110C0A0	A55CTI_CTIOUTEN0	A55CTI CTI Channel to Trigger 0 Enable register	0x00000000
0x3110C0A4	A55CTI_CTIOUTEN1	A55CTI CTI Channel to Trigger 1 Enable register	0x00000000
0x3110C0A8	A55CTI_CTIOUTEN2	A55CTI CTI Channel to Trigger 2 Enable register	0x00000000
0x3110C0AC	A55CTI_CTIOUTEN3	A55CTI CTI Channel to Trigger 3 Enable register	0x00000000
0x3110C0B0	A55CTI_CTIOUTEN4	A55CTI CTI Channel to Trigger 4 Enable register	0x00000000
0x3110C0B4	A55CTI_CTIOUTEN5	A55CTI CTI Channel to Trigger 5 Enable register	0x00000000
0x3110C0B8	A55CTI_CTIOUTEN6	A55CTI CTI Channel to Trigger 6 Enable register	0x00000000
0x3110C0BC	A55CTI_CTIOUTEN7	A55CTI CTI Channel to Trigger 7 Enable register	0x00000000
0x3110C0C0	A55CTI_CTIOUTEN8	A55CTI CTI Channel to Trigger 8 Enable register	0x00000000
0x3110C0C4	A55CTI_CTIOUTEN9	A55CTI CTI Channel to Trigger 9 Enable register	0x00000000

Table A-4: ADSP-SC59x A55CTI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C0C8	A55CTI_CTIOUTEN10	A55CTI CTI Channel to Trigger 10 Enable register	0x00000000
0x3110C0CC	A55CTI_CTIOUTEN11	A55CTI CTI Channel to Trigger 11 Enable register	0x00000000
0x3110C0D0	A55CTI_CTIOUTEN12	A55CTI CTI Channel to Trigger 12 Enable register	0x00000000
0x3110C0D4	A55CTI_CTIOUTEN13	A55CTI CTI Channel to Trigger 13 Enable register	0x00000000
0x3110C0D8	A55CTI_CTIOUTEN14	A55CTI CTI Channel to Trigger 14 Enable register	0x00000000
0x3110C0DC	A55CTI_CTIOUTEN15	A55CTI CTI Channel to Trigger 15 Enable register	0x00000000
0x3110C0E0	A55CTI_CTIOUTEN16	A55CTI CTI Channel to Trigger 16 Enable register	0x00000000
0x3110C0E4	A55CTI_CTIOUTEN17	A55CTI CTI Channel to Trigger 17 Enable register	0x00000000
0x3110C0E8	A55CTI_CTIOUTEN18	A55CTI CTI Channel to Trigger 18 Enable register	0x00000000
0x3110C0EC	A55CTI_CTIOUTEN19	A55CTI CTI Channel to Trigger 19 Enable register	0x00000000
0x3110C0F0	A55CTI_CTIOUTEN20	A55CTI CTI Channel to Trigger 20 Enable register	0x00000000
0x3110C0F4	A55CTI_CTIOUTEN21	A55CTI CTI Channel to Trigger 21 Enable register	0x00000000
0x3110C0F8	A55CTI_CTIOUTEN22	A55CTI CTI Channel to Trigger 22 Enable register	0x00000000
0x3110C0FC	A55CTI_CTIOUTEN23	A55CTI CTI Channel to Trigger 23 Enable register	0x00000000
0x3110C100	A55CTI_CTIOUTEN24	A55CTI CTI Channel to Trigger 24 Enable register	0x00000000
0x3110C104	A55CTI_CTIOUTEN25	A55CTI CTI Channel to Trigger 25 Enable register	0x00000000
0x3110C108	A55CTI_CTIOUTEN26	A55CTI CTI Channel to Trigger 26 Enable register	0x00000000
0x3110C10C	A55CTI_CTIOUTEN27	A55CTI CTI Channel to Trigger 27 Enable register	0x00000000
0x3110C110	A55CTI_CTIOUTEN28	A55CTI CTI Channel to Trigger 28 Enable register	0x00000000
0x3110C114	A55CTI_CTIOUTEN29	A55CTI CTI Channel to Trigger 29 Enable register	0x00000000
0x3110C118	A55CTI_CTIOUTEN30	A55CTI CTI Channel to Trigger 30 Enable register	0x00000000
0x3110C11C	A55CTI_CTIOUTEN31	A55CTI CTI Channel to Trigger 31 Enable register	0x00000000
0x3110C130	A55CTI_CTITRIGINSTATUS	A55CTI CTI Trigger Input Status register	0x00000000
0x3110C134	A55CTI_CTITRIGOUTSTATUS	A55CTI CTI Trigger Output Status register	0x00000000
0x3110C138	A55CTI_CTICHINSTATUS	A55CTI CTI Channel Input Status register	0x00000000
0x3110C13C	A55CTI_CTICHOUTSTATUS	A55CTI CTI Channel Output Status register	0x00000000
0x3110C140	A55CTI_CTIGATE	A55CTI Enable CTI Channel Gate register	0x0000000F
0x3110C144	A55CTI_ASICCTRL	A55CTI External Multiplexer Control register	0x00000000
0x3110CEE4	A55CTI_ITCHOUT	A55CTI Integration Test Channel Output register	0x00000000

Table A-4: ADSP-SC59x A55CTI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110CEE8	A55CTI_ITTRIGOUT	A55CTI Integration Test Trigger Output register	0x00000000
0x3110CEF4	A55CTI_ITCHIN	A55CTI Integration Test Channel Input register	0x00000000
0x3110CEF8	A55CTI_ITTRIGIN	A55CTI Integration Test Trigger Input register	0x00000000
0x3110CF00	A55CTI_ITCTRL	A55CTI Integration Mode Control Register	0x00000000
0x3110CFA0	A55CTI_CLAIMSET	A55CTI Claim Tag Set Register	0x0000000F
0x3110CFA4	A55CTI_CLAIMCLR	A55CTI Claim Tag Clear Register	0x00000000
0x3110CFA8	A55CTI_DEVAFF0	A55CTI Device Affinity register 0	0xFFFFFFFF
0x3110CFAC	A55CTI_DEVAFF1	A55CTI Device Affinity register 1	0xFFFFFFFF
0x3110CFB8	A55CTI_AUTHSTATUS	A55CTI Authentication Status Register	0x0000000A
0x3110CFBC	A55CTI_DEVARCH	A55CTI Device Architecture Register	0x47701A14
0x3110CFC8	A55CTI_DEVID	A55CTI Device Configuration Register	0x01040100
0x3110CFCC	A55CTI_DEVTYPE	A55CTI Device Type Identifier Register	0x00000014
0x3110CFD0	A55CTI_PIDR4	A55CTI Peripheral Identification Register 4	0x00000004
0x3110CFD4	A55CTI_PIDR5	A55CTI Peripheral Identification Register 5	0x00000000
0x3110CFD8	A55CTI_PIDR6	A55CTI Peripheral Identification Register 6	0x00000000
0x3110CFDC	A55CTI_PIDR7	A55CTI Peripheral Identification Register 7	0x00000000
0x3110CFE0	A55CTI_PIDR0	A55CTI Peripheral Identification Register 0	0x000000ED
0x3110CFE4	A55CTI_PIDR1	A55CTI Peripheral Identification Register 1	0x000000B9
0x3110CFE8	A55CTI_PIDR2	A55CTI Peripheral Identification Register 2	0x0000002B
0x3110CFEC	A55CTI_PIDR3	A55CTI Peripheral Identification Register 3	0x00000000
0x3110CFF0	A55CTI_CIDR0	A55CTI Component Identification Register 0	0x0000000D
0x3110CFF4	A55CTI_CIDR1	A55CTI Component Identification Register 1	0x00000090
0x3110CFF8	A55CTI_CIDR2	A55CTI Component Identification Register 2	0x00000005
0x3110CFFC	A55CTI_CIDR3	A55CTI Component Identification Register 3	0x000000B1
0x3110D000	SOC600CTI0_CTICONTROL	SOC600CTI0 CTI Control register	0x00000000
0x3110D010	SOC600CTI0_CTIINTACK	SOC600CTI0 CTI Interrupt Acknowledge register	0x00000000
0x3110D014	SOC600CTI0_CTIAPPSET	SOC600CTI0 CTI Application Channel Set register	0x00000000
0x3110D018	SOC600CTI0_CTIAPPCLEAR	SOC600CTI0 CTI Application Channel Clear register	0x00000000
0x3110D01C	SOC600CTI0_CTIAPPPULSE	SOC600CTI0 CTI Application Channel Pulse register	0x00000000

Table A-4: ADSP-SC59x A55CTI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110D020	SOC600CTI0_CTIINEN0	SOC600CTI0 CTI Trigger 0 to Channel Enable register	0x00000000
0x3110D024	SOC600CTI0_CTIINEN1	SOC600CTI0 CTI Trigger 1 to Channel Enable register	0x00000000
0x3110D028	SOC600CTI0_CTIINEN2	SOC600CTI0 CTI Trigger 2 to Channel Enable register	0x00000000
0x3110D02C	SOC600CTI0_CTIINEN3	SOC600CTI0 CTI Trigger 3 to Channel Enable register	0x00000000
0x3110D030	SOC600CTI0_CTIINEN4	SOC600CTI0 CTI Trigger 4 to Channel Enable register	0x00000000
0x3110D034	SOC600CTI0_CTIINEN5	SOC600CTI0 CTI Trigger 5 to Channel Enable register	0x00000000
0x3110D038	SOC600CTI0_CTIINEN6	SOC600CTI0 CTI Trigger 6 to Channel Enable register	0x00000000
0x3110D03C	SOC600CTI0_CTIINEN7	SOC600CTI0 CTI Trigger 7 to Channel Enable register	0x00000000
0x3110D040	SOC600CTI0_CTIINEN8	SOC600CTI0 CTI Trigger 8 to Channel Enable register	0x00000000
0x3110D044	SOC600CTI0_CTIINEN9	SOC600CTI0 CTI Trigger 9 to Channel Enable register	0x00000000
0x3110D048	SOC600CTI0_CTIINEN10	SOC600CTI0 CTI Trigger 10 to Channel Enable register	0x00000000
0x3110D04C	SOC600CTI0_CTIINEN11	SOC600CTI0 CTI Trigger 11 to Channel Enable register	0x00000000
0x3110D050	SOC600CTI0_CTIINEN12	SOC600CTI0 CTI Trigger 12 to Channel Enable register	0x00000000
0x3110D054	SOC600CTI0_CTIINEN13	SOC600CTI0 CTI Trigger 13 to Channel Enable register	0x00000000
0x3110D058	SOC600CTI0_CTIINEN14	SOC600CTI0 CTI Trigger 14 to Channel Enable register	0x00000000
0x3110D05C	SOC600CTI0_CTIINEN15	SOC600CTI0 CTI Trigger 15 to Channel Enable register	0x00000000
0x3110D060	SOC600CTI0_CTIINEN16	SOC600CTI0 CTI Trigger 16 to Channel Enable register	0x00000000
0x3110D064	SOC600CTI0_CTIINEN17	SOC600CTI0 CTI Trigger 17 to Channel Enable register	0x00000000
0x3110D068	SOC600CTI0_CTIINEN18	SOC600CTI0 CTI Trigger 18 to Channel Enable register	0x00000000
0x3110D06C	SOC600CTI0_CTIINEN19	SOC600CTI0 CTI Trigger 19 to Channel Enable register	0x00000000
0x3110D070	SOC600CTI0_CTIINEN20	SOC600CTI0 CTI Trigger 20 to Channel Enable register	0x00000000
0x3110D074	SOC600CTI0_CTIINEN21	SOC600CTI0 CTI Trigger 21 to Channel Enable register	0x00000000
0x3110D078	SOC600CTI0_CTIINEN22	SOC600CTI0 CTI Trigger 22 to Channel Enable register	0x00000000
0x3110D07C	SOC600CTI0_CTIINEN23	SOC600CTI0 CTI Trigger 23 to Channel Enable register	0x00000000
0x3110D080	SOC600CTI0_CTIINEN24	SOC600CTI0 CTI Trigger 24 to Channel Enable register	0x00000000
0x3110D084	SOC600CTI0_CTIINEN25	SOC600CTI0 CTI Trigger 25 to Channel Enable register	0x00000000
0x3110D088	SOC600CTI0_CTIINEN26	SOC600CTI0 CTI Trigger 26 to Channel Enable register	0x00000000
0x3110D08C	SOC600CTI0_CTIINEN27	SOC600CTI0 CTI Trigger 27 to Channel Enable register	0x00000000
0x3110D090	SOC600CTI0_CTIINEN28	SOC600CTI0 CTI Trigger 28 to Channel Enable register	0x00000000
0x3110D094	SOC600CTI0_CTIINEN29	SOC600CTI0 CTI Trigger 29 to Channel Enable register	0x00000000
0x3110D098	SOC600CTI0_CTIINEN30	SOC600CTI0 CTI Trigger 30 to Channel Enable register	0x00000000

Table A-4: ADSP-SC59x A55CTI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110D09C	SOC600CTI0_CTIINEN31	SOC600CTI0 CTI Trigger 31 to Channel Enable register	0x00000000
0x3110D0A0	SOC600CTI0_CTIOUTEN0	SOC600CTI0 CTI Channel to Trigger 0 Enable register	0x00000000
0x3110D0A4	SOC600CTI0_CTIOUTEN1	SOC600CTI0 CTI Channel to Trigger 1 Enable register	0x00000000
0x3110D0A8	SOC600CTI0_CTIOUTEN2	SOC600CTI0 CTI Channel to Trigger 2 Enable register	0x00000000
0x3110D0AC	SOC600CTI0_CTIOUTEN3	SOC600CTI0 CTI Channel to Trigger 3 Enable register	0x00000000
0x3110D0B0	SOC600CTI0_CTIOUTEN4	SOC600CTI0 CTI Channel to Trigger 4 Enable register	0x00000000
0x3110D0B4	SOC600CTI0_CTIOUTEN5	SOC600CTI0 CTI Channel to Trigger 5 Enable register	0x00000000
0x3110D0B8	SOC600CTI0_CTIOUTEN6	SOC600CTI0 CTI Channel to Trigger 6 Enable register	0x00000000
0x3110D0BC	SOC600CTI0_CTIOUTEN7	SOC600CTI0 CTI Channel to Trigger 7 Enable register	0x00000000
0x3110D0C0	SOC600CTI0_CTIOUTEN8	SOC600CTI0 CTI Channel to Trigger 8 Enable register	0x00000000
0x3110D0C4	SOC600CTI0_CTIOUTEN9	SOC600CTI0 CTI Channel to Trigger 9 Enable register	0x00000000
0x3110D0C8	SOC600CTI0_CTIOUTEN10	SOC600CTI0 CTI Channel to Trigger 10 Enable register	0x00000000
0x3110D0CC	SOC600CTI0_CTIOUTEN11	SOC600CTI0 CTI Channel to Trigger 11 Enable register	0x00000000
0x3110D0D0	SOC600CTI0_CTIOUTEN12	SOC600CTI0 CTI Channel to Trigger 12 Enable register	0x00000000
0x3110D0D4	SOC600CTI0_CTIOUTEN13	SOC600CTI0 CTI Channel to Trigger 13 Enable register	0x00000000
0x3110D0D8	SOC600CTI0_CTIOUTEN14	SOC600CTI0 CTI Channel to Trigger 14 Enable register	0x00000000
0x3110D0DC	SOC600CTI0_CTIOUTEN15	SOC600CTI0 CTI Channel to Trigger 15 Enable register	0x00000000
0x3110D0E0	SOC600CTI0_CTIOUTEN16	SOC600CTI0 CTI Channel to Trigger 16 Enable register	0x00000000
0x3110D0E4	SOC600CTI0_CTIOUTEN17	SOC600CTI0 CTI Channel to Trigger 17 Enable register	0x00000000
0x3110D0E8	SOC600CTI0_CTIOUTEN18	SOC600CTI0 CTI Channel to Trigger 18 Enable register	0x00000000
0x3110D0EC	SOC600CTI0_CTIOUTEN19	SOC600CTI0 CTI Channel to Trigger 19 Enable register	0x00000000
0x3110D0F0	SOC600CTI0_CTIOUTEN20	SOC600CTI0 CTI Channel to Trigger 20 Enable register	0x00000000
0x3110D0F4	SOC600CTI0_CTIOUTEN21	SOC600CTI0 CTI Channel to Trigger 21 Enable register	0x00000000

Table A-4: ADSP-SC59x A55CTI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110D0F8	SOC600CTI0_CTIOUTEN22	SOC600CTI0 CTI Channel to Trigger 22 Enable register	0x00000000
0x3110D0FC	SOC600CTI0_CTIOUTEN23	SOC600CTI0 CTI Channel to Trigger 23 Enable register	0x00000000
0x3110D100	SOC600CTI0_CTIOUTEN24	SOC600CTI0 CTI Channel to Trigger 24 Enable register	0x00000000
0x3110D104	SOC600CTI0_CTIOUTEN25	SOC600CTI0 CTI Channel to Trigger 25 Enable register	0x00000000
0x3110D108	SOC600CTI0_CTIOUTEN26	SOC600CTI0 CTI Channel to Trigger 26 Enable register	0x00000000
0x3110D10C	SOC600CTI0_CTIOUTEN27	SOC600CTI0 CTI Channel to Trigger 27 Enable register	0x00000000
0x3110D110	SOC600CTI0_CTIOUTEN28	SOC600CTI0 CTI Channel to Trigger 28 Enable register	0x00000000
0x3110D114	SOC600CTI0_CTIOUTEN29	SOC600CTI0 CTI Channel to Trigger 29 Enable register	0x00000000
0x3110D118	SOC600CTI0_CTIOUTEN30	SOC600CTI0 CTI Channel to Trigger 30 Enable register	0x00000000
0x3110D11C	SOC600CTI0_CTIOUTEN31	SOC600CTI0 CTI Channel to Trigger 31 Enable register	0x00000000
0x3110D130	SOC600CTI0_CTITRIGINSTATUS	SOC600CTI0 CTI Trigger Input Status register	0x00000000
0x3110D134	SOC600CTI0_CTITRIGOUTSTATUS	SOC600CTI0 CTI Trigger Output Status register	0x00000000
0x3110D138	SOC600CTI0_CTICHINSTATUS	SOC600CTI0 CTI Channel Input Status register	0x00000000
0x3110D13C	SOC600CTI0_CTICHOUTSTATUS	SOC600CTI0 CTI Channel Output Status register	0x00000000
0x3110D140	SOC600CTI0_CTIGATE	SOC600CTI0 Enable CTI Channel Gate register	0x0000000F
0x3110D144	SOC600CTI0_ASICCTRL	SOC600CTI0 External Multiplexer Control register	0x00000000
0x3110DEE4	SOC600CTI0_ITCHOUT	SOC600CTI0 Integration Test Channel Output register	0x00000000
0x3110DEE8	SOC600CTI0_ITTRIGOUT	SOC600CTI0 Integration Test Trigger Output register	0x00000000
0x3110DEF4	SOC600CTI0_ITCHIN	SOC600CTI0 Integration Test Channel Input register	0x00000000
0x3110DEF8	SOC600CTI0_ITTRIGIN	SOC600CTI0 Integration Test Trigger Input register	0x00000000
0x3110DF00	SOC600CTI0_ITCTRL	SOC600CTI0 Integration Mode Control Register	0x00000000

Table A-4: ADSP-SC59x A55CTI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110DFA0	SOC600CTI0_CLAIMSET	SOC600CTI0 Claim Tag Set Register	0x0000000F
0x3110DFA4	SOC600CTI0_CLAIMCLR	SOC600CTI0 Claim Tag Clear Register	0x00000000
0x3110DFA8	SOC600CTI0_DEVAFF0	SOC600CTI0 Device Affinity register 0	0xFFFFFFFF
0x3110DFAC	SOC600CTI0_DEVAFF1	SOC600CTI0 Device Affinity register 1	0xFFFFFFFF
0x3110DFB8	SOC600CTI0_AUTHSTATUS	SOC600CTI0 Authentication Status Register	0x0000000A
0x3110DFBC	SOC600CTI0_DEVARCH	SOC600CTI0 Device Architecture Register	0x47701A14
0x3110DFC8	SOC600CTI0_DEVID	SOC600CTI0 Device Configuration Register	0x01040100
0x3110DFCC	SOC600CTI0_DEVTYPE	SOC600CTI0 Device Type Identifier Register	0x00000014
0x3110DFD0	SOC600CTI0_PIDR4	SOC600CTI0 Peripheral Identification Register 4	0x00000004
0x3110DFD4	SOC600CTI0_PIDR5	SOC600CTI0 Peripheral Identification Register 5	0x00000000
0x3110DFD8	SOC600CTI0_PIDR6	SOC600CTI0 Peripheral Identification Register 6	0x00000000
0x3110DFDC	SOC600CTI0_PIDR7	SOC600CTI0 Peripheral Identification Register 7	0x00000000
0x3110DFE0	SOC600CTI0_PIDR0	SOC600CTI0 Peripheral Identification Register 0	0x000000ED
0x3110DFE4	SOC600CTI0_PIDR1	SOC600CTI0 Peripheral Identification Register 1	0x000000B9
0x3110DFE8	SOC600CTI0_PIDR2	SOC600CTI0 Peripheral Identification Register 2	0x0000002B
0x3110DFEC	SOC600CTI0_PIDR3	SOC600CTI0 Peripheral Identification Register 3	0x00000000
0x3110DFF0	SOC600CTI0_CIDR0	SOC600CTI0 Component Identification Register 0	0x0000000D
0x3110DFF4	SOC600CTI0_CIDR1	SOC600CTI0 Component Identification Register 1	0x00000090
0x3110DFF8	SOC600CTI0_CIDR2	SOC600CTI0 Component Identification Register 2	0x00000005
0x3110DFFC	SOC600CTI0_CIDR3	SOC600CTI0 Component Identification Register 3	0x000000B1

Table A-5: ADSP-SC59x SOC600CTI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C000	SOC600CTI1_CTICONTROL	SOC600CTI1 CTI Control register	0x00000000
0x3110C010	SOC600CTI1_CTIINTACK	SOC600CTI1 CTI Interrupt Acknowledge register	0x00000000
0x3110C014	SOC600CTI1_CTIAPPSET	SOC600CTI1 CTI Application Channel Set register	0x00000000
0x3110C018	SOC600CTI1_CTIAPPCLEAR	SOC600CTI1 CTI Application Channel Clear register	0x00000000
0x3110C01C	SOC600CTI1_CTIAPPPULSE	SOC600CTI1 CTI Application Channel Pulse register	0x00000000

Table A-5: ADSP-SC59x SOC600CTI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C020	SOC600CTI1_CTIINEN0	SOC600CTI1 CTI Trigger 0 to Channel Enable register	0x00000000
0x3110C024	SOC600CTI1_CTIINEN1	SOC600CTI1 CTI Trigger 1 to Channel Enable register	0x00000000
0x3110C028	SOC600CTI1_CTIINEN2	SOC600CTI1 CTI Trigger 2 to Channel Enable register	0x00000000
0x3110C02C	SOC600CTI1_CTIINEN3	SOC600CTI1 CTI Trigger 3 to Channel Enable register	0x00000000
0x3110C030	SOC600CTI1_CTIINEN4	SOC600CTI1 CTI Trigger 4 to Channel Enable register	0x00000000
0x3110C034	SOC600CTI1_CTIINEN5	SOC600CTI1 CTI Trigger 5 to Channel Enable register	0x00000000
0x3110C038	SOC600CTI1_CTIINEN6	SOC600CTI1 CTI Trigger 6 to Channel Enable register	0x00000000
0x3110C03C	SOC600CTI1_CTIINEN7	SOC600CTI1 CTI Trigger 7 to Channel Enable register	0x00000000
0x3110C040	SOC600CTI1_CTIINEN8	SOC600CTI1 CTI Trigger 8 to Channel Enable register	0x00000000
0x3110C044	SOC600CTI1_CTIINEN9	SOC600CTI1 CTI Trigger 9 to Channel Enable register	0x00000000
0x3110C048	SOC600CTI1_CTIINEN10	SOC600CTI1 CTI Trigger 10 to Channel Enable register	0x00000000
0x3110C04C	SOC600CTI1_CTIINEN11	SOC600CTI1 CTI Trigger 11 to Channel Enable register	0x00000000
0x3110C050	SOC600CTI1_CTIINEN12	SOC600CTI1 CTI Trigger 12 to Channel Enable register	0x00000000
0x3110C054	SOC600CTI1_CTIINEN13	SOC600CTI1 CTI Trigger 13 to Channel Enable register	0x00000000
0x3110C058	SOC600CTI1_CTIINEN14	SOC600CTI1 CTI Trigger 14 to Channel Enable register	0x00000000
0x3110C05C	SOC600CTI1_CTIINEN15	SOC600CTI1 CTI Trigger 15 to Channel Enable register	0x00000000
0x3110C060	SOC600CTI1_CTIINEN16	SOC600CTI1 CTI Trigger 16 to Channel Enable register	0x00000000
0x3110C064	SOC600CTI1_CTIINEN17	SOC600CTI1 CTI Trigger 17 to Channel Enable register	0x00000000
0x3110C068	SOC600CTI1_CTIINEN18	SOC600CTI1 CTI Trigger 18 to Channel Enable register	0x00000000
0x3110C06C	SOC600CTI1_CTIINEN19	SOC600CTI1 CTI Trigger 19 to Channel Enable register	0x00000000
0x3110C070	SOC600CTI1_CTIINEN20	SOC600CTI1 CTI Trigger 20 to Channel Enable register	0x00000000
0x3110C074	SOC600CTI1_CTIINEN21	SOC600CTI1 CTI Trigger 21 to Channel Enable register	0x00000000
0x3110C078	SOC600CTI1_CTIINEN22	SOC600CTI1 CTI Trigger 22 to Channel Enable register	0x00000000
0x3110C07C	SOC600CTI1_CTIINEN23	SOC600CTI1 CTI Trigger 23 to Channel Enable register	0x00000000
0x3110C080	SOC600CTI1_CTIINEN24	SOC600CTI1 CTI Trigger 24 to Channel Enable register	0x00000000
0x3110C084	SOC600CTI1_CTIINEN25	SOC600CTI1 CTI Trigger 25 to Channel Enable register	0x00000000
0x3110C088	SOC600CTI1_CTIINEN26	SOC600CTI1 CTI Trigger 26 to Channel Enable register	0x00000000
0x3110C08C	SOC600CTI1_CTIINEN27	SOC600CTI1 CTI Trigger 27 to Channel Enable register	0x00000000
0x3110C090	SOC600CTI1_CTIINEN28	SOC600CTI1 CTI Trigger 28 to Channel Enable register	0x00000000
0x3110C094	SOC600CTI1_CTIINEN29	SOC600CTI1 CTI Trigger 29 to Channel Enable register	0x00000000
0x3110C098	SOC600CTI1_CTIINEN30	SOC600CTI1 CTI Trigger 30 to Channel Enable register	0x00000000

Table A-5: ADSP-SC59x SOC600CTI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C09C	SOC600CTI1_CTIINEN31	SOC600CTI1 CTI Trigger 31 to Channel Enable register	0x00000000
0x3110C0A0	SOC600CTI1_CTIOUTEN0	SOC600CTI1 CTI Channel to Trigger 0 Enable register	0x00000000
0x3110C0A4	SOC600CTI1_CTIOUTEN1	SOC600CTI1 CTI Channel to Trigger 1 Enable register	0x00000000
0x3110C0A8	SOC600CTI1_CTIOUTEN2	SOC600CTI1 CTI Channel to Trigger 2 Enable register	0x00000000
0x3110C0AC	SOC600CTI1_CTIOUTEN3	SOC600CTI1 CTI Channel to Trigger 3 Enable register	0x00000000
0x3110C0B0	SOC600CTI1_CTIOUTEN4	SOC600CTI1 CTI Channel to Trigger 4 Enable register	0x00000000
0x3110C0B4	SOC600CTI1_CTIOUTEN5	SOC600CTI1 CTI Channel to Trigger 5 Enable register	0x00000000
0x3110C0B8	SOC600CTI1_CTIOUTEN6	SOC600CTI1 CTI Channel to Trigger 6 Enable register	0x00000000
0x3110C0BC	SOC600CTI1_CTIOUTEN7	SOC600CTI1 CTI Channel to Trigger 7 Enable register	0x00000000
0x3110C0C0	SOC600CTI1_CTIOUTEN8	SOC600CTI1 CTI Channel to Trigger 8 Enable register	0x00000000
0x3110C0C4	SOC600CTI1_CTIOUTEN9	SOC600CTI1 CTI Channel to Trigger 9 Enable register	0x00000000
0x3110C0C8	SOC600CTI1_CTIOUTEN10	SOC600CTI1 CTI Channel to Trigger 10 Enable register	0x00000000
0x3110C0CC	SOC600CTI1_CTIOUTEN11	SOC600CTI1 CTI Channel to Trigger 11 Enable register	0x00000000
0x3110C0D0	SOC600CTI1_CTIOUTEN12	SOC600CTI1 CTI Channel to Trigger 12 Enable register	0x00000000
0x3110C0D4	SOC600CTI1_CTIOUTEN13	SOC600CTI1 CTI Channel to Trigger 13 Enable register	0x00000000
0x3110C0D8	SOC600CTI1_CTIOUTEN14	SOC600CTI1 CTI Channel to Trigger 14 Enable register	0x00000000
0x3110C0DC	SOC600CTI1_CTIOUTEN15	SOC600CTI1 CTI Channel to Trigger 15 Enable register	0x00000000
0x3110C0E0	SOC600CTI1_CTIOUTEN16	SOC600CTI1 CTI Channel to Trigger 16 Enable register	0x00000000
0x3110C0E4	SOC600CTI1_CTIOUTEN17	SOC600CTI1 CTI Channel to Trigger 17 Enable register	0x00000000
0x3110C0E8	SOC600CTI1_CTIOUTEN18	SOC600CTI1 CTI Channel to Trigger 18 Enable register	0x00000000
0x3110C0EC	SOC600CTI1_CTIOUTEN19	SOC600CTI1 CTI Channel to Trigger 19 Enable register	0x00000000
0x3110C0F0	SOC600CTI1_CTIOUTEN20	SOC600CTI1 CTI Channel to Trigger 20 Enable register	0x00000000
0x3110C0F4	SOC600CTI1_CTIOUTEN21	SOC600CTI1 CTI Channel to Trigger 21 Enable register	0x00000000

Table A-5: ADSP-SC59x SOC600CTI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110C0F8	SOC600CTI1_CTIOUTEN22	SOC600CTI1 CTI Channel to Trigger 22 Enable register	0x00000000
0x3110C0FC	SOC600CTI1_CTIOUTEN23	SOC600CTI1 CTI Channel to Trigger 23 Enable register	0x00000000
0x3110C100	SOC600CTI1_CTIOUTEN24	SOC600CTI1 CTI Channel to Trigger 24 Enable register	0x00000000
0x3110C104	SOC600CTI1_CTIOUTEN25	SOC600CTI1 CTI Channel to Trigger 25 Enable register	0x00000000
0x3110C108	SOC600CTI1_CTIOUTEN26	SOC600CTI1 CTI Channel to Trigger 26 Enable register	0x00000000
0x3110C10C	SOC600CTI1_CTIOUTEN27	SOC600CTI1 CTI Channel to Trigger 27 Enable register	0x00000000
0x3110C110	SOC600CTI1_CTIOUTEN28	SOC600CTI1 CTI Channel to Trigger 28 Enable register	0x00000000
0x3110C114	SOC600CTI1_CTIOUTEN29	SOC600CTI1 CTI Channel to Trigger 29 Enable register	0x00000000
0x3110C118	SOC600CTI1_CTIOUTEN30	SOC600CTI1 CTI Channel to Trigger 30 Enable register	0x00000000
0x3110C11C	SOC600CTI1_CTIOUTEN31	SOC600CTI1 CTI Channel to Trigger 31 Enable register	0x00000000
0x3110C130	SOC600CTI1_CTITRIGINSTATUS	SOC600CTI1 CTI Trigger Input Status register	0x00000000
0x3110C134	SOC600CTI1_CTITRIGOUTSTATUS	SOC600CTI1 CTI Trigger Output Status register	0x00000000
0x3110C138	SOC600CTI1_CTICHINSTATUS	SOC600CTI1 CTI Channel Input Status register	0x00000000
0x3110C13C	SOC600CTI1_CTICHOUTSTATUS	SOC600CTI1 CTI Channel Output Status register	0x00000000
0x3110C140	SOC600CTI1_CTIGATE	SOC600CTI1 Enable CTI Channel Gate register	0x0000000F
0x3110C144	SOC600CTI1_ASICCTRL	SOC600CTI1 External Multiplexer Control register	0x00000000
0x3110CEE4	SOC600CTI1_ITCHOUT	SOC600CTI1 Integration Test Channel Output register	0x00000000
0x3110CEE8	SOC600CTI1_ITTRIGOUT	SOC600CTI1 Integration Test Trigger Output register	0x00000000
0x3110CEF4	SOC600CTI1_ITCHIN	SOC600CTI1 Integration Test Channel Input register	0x00000000
0x3110CEF8	SOC600CTI1_ITTRIGIN	SOC600CTI1 Integration Test Trigger Input register	0x00000000
0x3110CF00	SOC600CTI1_ITCTRL	SOC600CTI1 Integration Mode Control Register	0x00000000

Table A-5: ADSP-SC59x SOC600CTI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110CFA0	SOC600CTI1_CLAIMSET	SOC600CTI1 Claim Tag Set Register	0x0000000F
0x3110CFA4	SOC600CTI1_CLAIMCLR	SOC600CTI1 Claim Tag Clear Register	0x00000000
0x3110CFA8	SOC600CTI1_DEVAFF0	SOC600CTI1 Device Affinity register 0	0xFFFFFFFF
0x3110CFAC	SOC600CTI1_DEVAFF1	SOC600CTI1 Device Affinity register 1	0xFFFFFFFF
0x3110CFB8	SOC600CTI1_AUTHSTATUS	SOC600CTI1 Authentication Status Register	0x0000000A
0x3110CFBC	SOC600CTI1_DEVARCH	SOC600CTI1 Device Architecture Register	0x47701A14
0x3110CFC8	SOC600CTI1_DEVID	SOC600CTI1 Device Configuration Register	0x01040100
0x3110CFCC	SOC600CTI1_DEVTYPE	SOC600CTI1 Device Type Identifier Register	0x00000014
0x3110CFD0	SOC600CTI1_PIDR4	SOC600CTI1 Peripheral Identification Register 4	0x00000004
0x3110CFD4	SOC600CTI1_PIDR5	SOC600CTI1 Peripheral Identification Register 5	0x00000000
0x3110CFD8	SOC600CTI1_PIDR6	SOC600CTI1 Peripheral Identification Register 6	0x00000000
0x3110CFDC	SOC600CTI1_PIDR7	SOC600CTI1 Peripheral Identification Register 7	0x00000000
0x3110CFE0	SOC600CTI1_PIDR0	SOC600CTI1 Peripheral Identification Register 0	0x000000ED
0x3110CFE4	SOC600CTI1_PIDR1	SOC600CTI1 Peripheral Identification Register 1	0x000000B9
0x3110CFE8	SOC600CTI1_PIDR2	SOC600CTI1 Peripheral Identification Register 2	0x0000002B
0x3110CFEC	SOC600CTI1_PIDR3	SOC600CTI1 Peripheral Identification Register 3	0x00000000
0x3110CFF0	SOC600CTI1_CIDR0	SOC600CTI1 Component Identification Register 0	0x0000000D
0x3110CFF4	SOC600CTI1_CIDR1	SOC600CTI1 Component Identification Register 1	0x00000090
0x3110CFF8	SOC600CTI1_CIDR2	SOC600CTI1 Component Identification Register 2	0x00000005
0x3110CFFC	SOC600CTI1_CIDR3	SOC600CTI1 Component Identification Register 3	0x000000B1

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31100000	DAPROM0_ROMENTRY0	DAPROM0 ROM Entries register 0	0x00001003
0x31100004	DAPROM0_ROMENTRY1	DAPROM0 ROM Entries register 1	0x00002003
0x31100008	DAPROM0_ROMENTRY2	DAPROM0 ROM Entries register 2	0x00003003
0x3110000C	DAPROM0_ROMENTRY3	DAPROM0 ROM Entries register 3	0x00004003
0x31100010	DAPROM0_ROMENTRY4	DAPROM0 ROM Entries register 4	0x00005003
0x31100014	DAPROM0_ROMENTRY5	DAPROM0 ROM Entries register 5	0x00006003
0x31100018	DAPROM0_ROMENTRY6	DAPROM0 ROM Entries register 6	0x00007003

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110001C	DAPROM0_ROMENTRY7	DAPROM0 ROM Entries register 7	0x00008003
0x31100020	DAPROM0_ROMENTRY8	DAPROM0 ROM Entries register 8	0x00009003
0x31100024	DAPROM0_ROMENTRY9	DAPROM0 ROM Entries register 9	0x0000A003
0x31100028	DAPROM0_ROMENTRY10	DAPROM0 ROM Entries register 10	0x0000B003
0x3110002C	DAPROM0_ROMENTRY11	DAPROM0 ROM Entries register 11	0x0000C003
0x31100030	DAPROM0_ROMENTRY12	DAPROM0 ROM Entries register 12	0x0000D003
0x31100034	DAPROM0_ROMENTRY13	DAPROM0 ROM Entries register 13	0x0000E003
0x31100038	DAPROM0_ROMENTRY14	DAPROM0 ROM Entries register 14	0x00010003
0x3110003C	DAPROM0_ROMENTRY15	DAPROM0 ROM Entries register 15	0x00000000
0x31100040	DAPROM0_ROMENTRY16	DAPROM0 ROM Entries register 16	0x00000000
0x31100044	DAPROM0_ROMENTRY17	DAPROM0 ROM Entries register 17	0x00000000
0x31100048	DAPROM0_ROMENTRY18	DAPROM0 ROM Entries register 18	0x00000000
0x3110004C	DAPROM0_ROMENTRY19	DAPROM0 ROM Entries register 19	0x00000000
0x31100050	DAPROM0_ROMENTRY20	DAPROM0 ROM Entries register 20	0x00000000
0x31100054	DAPROM0_ROMENTRY21	DAPROM0 ROM Entries register 21	0x00000000
0x31100058	DAPROM0_ROMENTRY22	DAPROM0 ROM Entries register 22	0x00000000
0x3110005C	DAPROM0_ROMENTRY23	DAPROM0 ROM Entries register 23	0x00000000
0x31100060	DAPROM0_ROMENTRY24	DAPROM0 ROM Entries register 24	0x00000000
0x31100064	DAPROM0_ROMENTRY25	DAPROM0 ROM Entries register 25	0x00000000
0x31100068	DAPROM0_ROMENTRY26	DAPROM0 ROM Entries register 26	0x00000000
0x3110006C	DAPROM0_ROMENTRY27	DAPROM0 ROM Entries register 27	0x00000000
0x31100070	DAPROM0_ROMENTRY28	DAPROM0 ROM Entries register 28	0x00000000
0x31100074	DAPROM0_ROMENTRY29	DAPROM0 ROM Entries register 29	0x00000000
0x31100078	DAPROM0_ROMENTRY30	DAPROM0 ROM Entries register 30	0x00000000
0x3110007C	DAPROM0_ROMENTRY31	DAPROM0 ROM Entries register 31	0x00000000
0x31100080	DAPROM0_ROMENTRY32	DAPROM0 ROM Entries register 32	0x00000000
0x31100084	DAPROM0_ROMENTRY33	DAPROM0 ROM Entries register 33	0x00000000
0x31100088	DAPROM0_ROMENTRY34	DAPROM0 ROM Entries register 34	0x00000000
0x3110008C	DAPROM0_ROMENTRY35	DAPROM0 ROM Entries register 35	0x00000000
0x31100090	DAPROM0_ROMENTRY36	DAPROM0 ROM Entries register 36	0x00000000
0x31100094	DAPROM0_ROMENTRY37	DAPROM0 ROM Entries register 37	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100098	DAPROM0_ROMENTRY38	DAPROM0 ROM Entries register 38	0x00000000
0x3110009C	DAPROM0_ROMENTRY39	DAPROM0 ROM Entries register 39	0x00000000
0x311000A0	DAPROM0_ROMENTRY40	DAPROM0 ROM Entries register 40	0x00000000
0x311000A4	DAPROM0_ROMENTRY41	DAPROM0 ROM Entries register 41	0x00000000
0x311000A8	DAPROM0_ROMENTRY42	DAPROM0 ROM Entries register 42	0x00000000
0x311000AC	DAPROM0_ROMENTRY43	DAPROM0 ROM Entries register 43	0x00000000
0x311000B0	DAPROM0_ROMENTRY44	DAPROM0 ROM Entries register 44	0x00000000
0x311000B4	DAPROM0_ROMENTRY45	DAPROM0 ROM Entries register 45	0x00000000
0x311000B8	DAPROM0_ROMENTRY46	DAPROM0 ROM Entries register 46	0x00000000
0x311000BC	DAPROM0_ROMENTRY47	DAPROM0 ROM Entries register 47	0x00000000
0x311000C0	DAPROM0_ROMENTRY48	DAPROM0 ROM Entries register 48	0x00000000
0x311000C4	DAPROM0_ROMENTRY49	DAPROM0 ROM Entries register 49	0x00000000
0x311000C8	DAPROM0_ROMENTRY50	DAPROM0 ROM Entries register 50	0x00000000
0x311000CC	DAPROM0_ROMENTRY51	DAPROM0 ROM Entries register 51	0x00000000
0x311000D0	DAPROM0_ROMENTRY52	DAPROM0 ROM Entries register 52	0x00000000
0x311000D4	DAPROM0_ROMENTRY53	DAPROM0 ROM Entries register 53	0x00000000
0x311000D8	DAPROM0_ROMENTRY54	DAPROM0 ROM Entries register 54	0x00000000
0x311000DC	DAPROM0_ROMENTRY55	DAPROM0 ROM Entries register 55	0x00000000
0x311000E0	DAPROM0_ROMENTRY56	DAPROM0 ROM Entries register 56	0x00000000
0x311000E4	DAPROM0_ROMENTRY57	DAPROM0 ROM Entries register 57	0x00000000
0x311000E8	DAPROM0_ROMENTRY58	DAPROM0 ROM Entries register 58	0x00000000
0x311000EC	DAPROM0_ROMENTRY59	DAPROM0 ROM Entries register 59	0x00000000
0x311000F0	DAPROM0_ROMENTRY60	DAPROM0 ROM Entries register 60	0x00000000
0x311000F4	DAPROM0_ROMENTRY61	DAPROM0 ROM Entries register 61	0x00000000
0x311000F8	DAPROM0_ROMENTRY62	DAPROM0 ROM Entries register 62	0x00000000
0x311000FC	DAPROM0_ROMENTRY63	DAPROM0 ROM Entries register 63	0x00000000
0x31100100	DAPROM0_ROMENTRY64	DAPROM0 ROM Entries register 64	0x00000000
0x31100104	DAPROM0_ROMENTRY65	DAPROM0 ROM Entries register 65	0x00000000
0x31100108	DAPROM0_ROMENTRY66	DAPROM0 ROM Entries register 66	0x00000000
0x3110010C	DAPROM0_ROMENTRY67	DAPROM0 ROM Entries register 67	0x00000000
0x31100110	DAPROM0_ROMENTRY68	DAPROM0 ROM Entries register 68	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100114	DAPROM0_ROMENTRY69	DAPROM0 ROM Entries register 69	0x00000000
0x31100118	DAPROM0_ROMENTRY70	DAPROM0 ROM Entries register 70	0x00000000
0x3110011C	DAPROM0_ROMENTRY71	DAPROM0 ROM Entries register 71	0x00000000
0x31100120	DAPROM0_ROMENTRY72	DAPROM0 ROM Entries register 72	0x00000000
0x31100124	DAPROM0_ROMENTRY73	DAPROM0 ROM Entries register 73	0x00000000
0x31100128	DAPROM0_ROMENTRY74	DAPROM0 ROM Entries register 74	0x00000000
0x3110012C	DAPROM0_ROMENTRY75	DAPROM0 ROM Entries register 75	0x00000000
0x31100130	DAPROM0_ROMENTRY76	DAPROM0 ROM Entries register 76	0x00000000
0x31100134	DAPROM0_ROMENTRY77	DAPROM0 ROM Entries register 77	0x00000000
0x31100138	DAPROM0_ROMENTRY78	DAPROM0 ROM Entries register 78	0x00000000
0x3110013C	DAPROM0_ROMENTRY79	DAPROM0 ROM Entries register 79	0x00000000
0x31100140	DAPROM0_ROMENTRY80	DAPROM0 ROM Entries register 80	0x00000000
0x31100144	DAPROM0_ROMENTRY81	DAPROM0 ROM Entries register 81	0x00000000
0x31100148	DAPROM0_ROMENTRY82	DAPROM0 ROM Entries register 82	0x00000000
0x3110014C	DAPROM0_ROMENTRY83	DAPROM0 ROM Entries register 83	0x00000000
0x31100150	DAPROM0_ROMENTRY84	DAPROM0 ROM Entries register 84	0x00000000
0x31100154	DAPROM0_ROMENTRY85	DAPROM0 ROM Entries register 85	0x00000000
0x31100158	DAPROM0_ROMENTRY86	DAPROM0 ROM Entries register 86	0x00000000
0x3110015C	DAPROM0_ROMENTRY87	DAPROM0 ROM Entries register 87	0x00000000
0x31100160	DAPROM0_ROMENTRY88	DAPROM0 ROM Entries register 88	0x00000000
0x31100164	DAPROM0_ROMENTRY89	DAPROM0 ROM Entries register 89	0x00000000
0x31100168	DAPROM0_ROMENTRY90	DAPROM0 ROM Entries register 90	0x00000000
0x3110016C	DAPROM0_ROMENTRY91	DAPROM0 ROM Entries register 91	0x00000000
0x31100170	DAPROM0_ROMENTRY92	DAPROM0 ROM Entries register 92	0x00000000
0x31100174	DAPROM0_ROMENTRY93	DAPROM0 ROM Entries register 93	0x00000000
0x31100178	DAPROM0_ROMENTRY94	DAPROM0 ROM Entries register 94	0x00000000
0x3110017C	DAPROM0_ROMENTRY95	DAPROM0 ROM Entries register 95	0x00000000
0x31100180	DAPROM0_ROMENTRY96	DAPROM0 ROM Entries register 96	0x00000000
0x31100184	DAPROM0_ROMENTRY97	DAPROM0 ROM Entries register 97	0x00000000
0x31100188	DAPROM0_ROMENTRY98	DAPROM0 ROM Entries register 98	0x00000000
0x3110018C	DAPROM0_ROMENTRY99	DAPROM0 ROM Entries register 99	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100190	DAPROM0_ROMENTRY100	DAPROM0 ROM Entries register 100	0x00000000
0x31100194	DAPROM0_ROMENTRY101	DAPROM0 ROM Entries register 101	0x00000000
0x31100198	DAPROM0_ROMENTRY102	DAPROM0 ROM Entries register 102	0x00000000
0x3110019C	DAPROM0_ROMENTRY103	DAPROM0 ROM Entries register 103	0x00000000
0x311001A0	DAPROM0_ROMENTRY104	DAPROM0 ROM Entries register 104	0x00000000
0x311001A4	DAPROM0_ROMENTRY105	DAPROM0 ROM Entries register 105	0x00000000
0x311001A8	DAPROM0_ROMENTRY106	DAPROM0 ROM Entries register 106	0x00000000
0x311001AC	DAPROM0_ROMENTRY107	DAPROM0 ROM Entries register 107	0x00000000
0x311001B0	DAPROM0_ROMENTRY108	DAPROM0 ROM Entries register 108	0x00000000
0x311001B4	DAPROM0_ROMENTRY109	DAPROM0 ROM Entries register 109	0x00000000
0x311001B8	DAPROM0_ROMENTRY110	DAPROM0 ROM Entries register 110	0x00000000
0x311001BC	DAPROM0_ROMENTRY111	DAPROM0 ROM Entries register 111	0x00000000
0x311001C0	DAPROM0_ROMENTRY112	DAPROM0 ROM Entries register 112	0x00000000
0x311001C4	DAPROM0_ROMENTRY113	DAPROM0 ROM Entries register 113	0x00000000
0x311001C8	DAPROM0_ROMENTRY114	DAPROM0 ROM Entries register 114	0x00000000
0x311001CC	DAPROM0_ROMENTRY115	DAPROM0 ROM Entries register 115	0x00000000
0x311001D0	DAPROM0_ROMENTRY116	DAPROM0 ROM Entries register 116	0x00000000
0x311001D4	DAPROM0_ROMENTRY117	DAPROM0 ROM Entries register 117	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311001D8	DAPROM0_ROMENTRY118	DAPROM0 ROM Entries register 118	0x00000000
0x311001DC	DAPROM0_ROMENTRY119	DAPROM0 ROM Entries register 119	0x00000000
0x311001E0	DAPROM0_ROMENTRY120	DAPROM0 ROM Entries register 120	0x00000000
0x311001E4	DAPROM0_ROMENTRY121	DAPROM0 ROM Entries register 121	0x00000000
0x311001E8	DAPROM0_ROMENTRY122	DAPROM0 ROM Entries register 122	0x00000000
0x311001EC	DAPROM0_ROMENTRY123	DAPROM0 ROM Entries register 123	0x00000000
0x311001F0	DAPROM0_ROMENTRY124	DAPROM0 ROM Entries register 124	0x00000000
0x311001F4	DAPROM0_ROMENTRY125	DAPROM0 ROM Entries register 125	0x00000000
0x311001F8	DAPROM0_ROMENTRY126	DAPROM0 ROM Entries register 126	0x00000000
0x311001FC	DAPROM0_ROMENTRY127	DAPROM0 ROM Entries register 127	0x00000000
0x31100200	DAPROM0_ROMENTRY128	DAPROM0 ROM Entries register 128	0x00000000
0x31100204	DAPROM0_ROMENTRY129	DAPROM0 ROM Entries register 129	0x00000000
0x31100208	DAPROM0_ROMENTRY130	DAPROM0 ROM Entries register 130	0x00000000
0x3110020C	DAPROM0_ROMENTRY131	DAPROM0 ROM Entries register 131	0x00000000
0x31100210	DAPROM0_ROMENTRY132	DAPROM0 ROM Entries register 132	0x00000000
0x31100214	DAPROM0_ROMENTRY133	DAPROM0 ROM Entries register 133	0x00000000
0x31100218	DAPROM0_ROMENTRY134	DAPROM0 ROM Entries register 134	0x00000000
0x3110021C	DAPROM0_ROMENTRY135	DAPROM0 ROM Entries register 135	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100220	DAPROM0_ROMENTRY136	DAPROM0 ROM Entries register 136	0x00000000
0x31100224	DAPROM0_ROMENTRY137	DAPROM0 ROM Entries register 137	0x00000000
0x31100228	DAPROM0_ROMENTRY138	DAPROM0 ROM Entries register 138	0x00000000
0x3110022C	DAPROM0_ROMENTRY139	DAPROM0 ROM Entries register 139	0x00000000
0x31100230	DAPROM0_ROMENTRY140	DAPROM0 ROM Entries register 140	0x00000000
0x31100234	DAPROM0_ROMENTRY141	DAPROM0 ROM Entries register 141	0x00000000
0x31100238	DAPROM0_ROMENTRY142	DAPROM0 ROM Entries register 142	0x00000000
0x3110023C	DAPROM0_ROMENTRY143	DAPROM0 ROM Entries register 143	0x00000000
0x31100240	DAPROM0_ROMENTRY144	DAPROM0 ROM Entries register 144	0x00000000
0x31100244	DAPROM0_ROMENTRY145	DAPROM0 ROM Entries register 145	0x00000000
0x31100248	DAPROM0_ROMENTRY146	DAPROM0 ROM Entries register 146	0x00000000
0x3110024C	DAPROM0_ROMENTRY147	DAPROM0 ROM Entries register 147	0x00000000
0x31100250	DAPROM0_ROMENTRY148	DAPROM0 ROM Entries register 148	0x00000000
0x31100254	DAPROM0_ROMENTRY149	DAPROM0 ROM Entries register 149	0x00000000
0x31100258	DAPROM0_ROMENTRY150	DAPROM0 ROM Entries register 150	0x00000000
0x3110025C	DAPROM0_ROMENTRY151	DAPROM0 ROM Entries register 151	0x00000000
0x31100260	DAPROM0_ROMENTRY152	DAPROM0 ROM Entries register 152	0x00000000
0x31100264	DAPROM0_ROMENTRY153	DAPROM0 ROM Entries register 153	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100268	DAPROM0_ROMENTRY154	DAPROM0 ROM Entries register 154	0x00000000
0x3110026C	DAPROM0_ROMENTRY155	DAPROM0 ROM Entries register 155	0x00000000
0x31100270	DAPROM0_ROMENTRY156	DAPROM0 ROM Entries register 156	0x00000000
0x31100274	DAPROM0_ROMENTRY157	DAPROM0 ROM Entries register 157	0x00000000
0x31100278	DAPROM0_ROMENTRY158	DAPROM0 ROM Entries register 158	0x00000000
0x3110027C	DAPROM0_ROMENTRY159	DAPROM0 ROM Entries register 159	0x00000000
0x31100280	DAPROM0_ROMENTRY160	DAPROM0 ROM Entries register 160	0x00000000
0x31100284	DAPROM0_ROMENTRY161	DAPROM0 ROM Entries register 161	0x00000000
0x31100288	DAPROM0_ROMENTRY162	DAPROM0 ROM Entries register 162	0x00000000
0x3110028C	DAPROM0_ROMENTRY163	DAPROM0 ROM Entries register 163	0x00000000
0x31100290	DAPROM0_ROMENTRY164	DAPROM0 ROM Entries register 164	0x00000000
0x31100294	DAPROM0_ROMENTRY165	DAPROM0 ROM Entries register 165	0x00000000
0x31100298	DAPROM0_ROMENTRY166	DAPROM0 ROM Entries register 166	0x00000000
0x3110029C	DAPROM0_ROMENTRY167	DAPROM0 ROM Entries register 167	0x00000000
0x311002A0	DAPROM0_ROMENTRY168	DAPROM0 ROM Entries register 168	0x00000000
0x311002A4	DAPROM0_ROMENTRY169	DAPROM0 ROM Entries register 169	0x00000000
0x311002A8	DAPROM0_ROMENTRY170	DAPROM0 ROM Entries register 170	0x00000000
0x311002AC	DAPROM0_ROMENTRY171	DAPROM0 ROM Entries register 171	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311002B0	DAPROM0_ROMENTRY172	DAPROM0 ROM Entries register 172	0x00000000
0x311002B4	DAPROM0_ROMENTRY173	DAPROM0 ROM Entries register 173	0x00000000
0x311002B8	DAPROM0_ROMENTRY174	DAPROM0 ROM Entries register 174	0x00000000
0x311002BC	DAPROM0_ROMENTRY175	DAPROM0 ROM Entries register 175	0x00000000
0x311002C0	DAPROM0_ROMENTRY176	DAPROM0 ROM Entries register 176	0x00000000
0x311002C4	DAPROM0_ROMENTRY177	DAPROM0 ROM Entries register 177	0x00000000
0x311002C8	DAPROM0_ROMENTRY178	DAPROM0 ROM Entries register 178	0x00000000
0x311002CC	DAPROM0_ROMENTRY179	DAPROM0 ROM Entries register 179	0x00000000
0x311002D0	DAPROM0_ROMENTRY180	DAPROM0 ROM Entries register 180	0x00000000
0x311002D4	DAPROM0_ROMENTRY181	DAPROM0 ROM Entries register 181	0x00000000
0x311002D8	DAPROM0_ROMENTRY182	DAPROM0 ROM Entries register 182	0x00000000
0x311002DC	DAPROM0_ROMENTRY183	DAPROM0 ROM Entries register 183	0x00000000
0x311002E0	DAPROM0_ROMENTRY184	DAPROM0 ROM Entries register 184	0x00000000
0x311002E4	DAPROM0_ROMENTRY185	DAPROM0 ROM Entries register 185	0x00000000
0x311002E8	DAPROM0_ROMENTRY186	DAPROM0 ROM Entries register 186	0x00000000
0x311002EC	DAPROM0_ROMENTRY187	DAPROM0 ROM Entries register 187	0x00000000
0x311002F0	DAPROM0_ROMENTRY188	DAPROM0 ROM Entries register 188	0x00000000
0x311002F4	DAPROM0_ROMENTRY189	DAPROM0 ROM Entries register 189	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311002F8	DAPROM0_ROMENTRY190	DAPROM0 ROM Entries register 190	0x00000000
0x311002FC	DAPROM0_ROMENTRY191	DAPROM0 ROM Entries register 191	0x00000000
0x31100300	DAPROM0_ROMENTRY192	DAPROM0 ROM Entries register 192	0x00000000
0x31100304	DAPROM0_ROMENTRY193	DAPROM0 ROM Entries register 193	0x00000000
0x31100308	DAPROM0_ROMENTRY194	DAPROM0 ROM Entries register 194	0x00000000
0x3110030C	DAPROM0_ROMENTRY195	DAPROM0 ROM Entries register 195	0x00000000
0x31100310	DAPROM0_ROMENTRY196	DAPROM0 ROM Entries register 196	0x00000000
0x31100314	DAPROM0_ROMENTRY197	DAPROM0 ROM Entries register 197	0x00000000
0x31100318	DAPROM0_ROMENTRY198	DAPROM0 ROM Entries register 198	0x00000000
0x3110031C	DAPROM0_ROMENTRY199	DAPROM0 ROM Entries register 199	0x00000000
0x31100320	DAPROM0_ROMENTRY200	DAPROM0 ROM Entries register 200	0x00000000
0x31100324	DAPROM0_ROMENTRY201	DAPROM0 ROM Entries register 201	0x00000000
0x31100328	DAPROM0_ROMENTRY202	DAPROM0 ROM Entries register 202	0x00000000
0x3110032C	DAPROM0_ROMENTRY203	DAPROM0 ROM Entries register 203	0x00000000
0x31100330	DAPROM0_ROMENTRY204	DAPROM0 ROM Entries register 204	0x00000000
0x31100334	DAPROM0_ROMENTRY205	DAPROM0 ROM Entries register 205	0x00000000
0x31100338	DAPROM0_ROMENTRY206	DAPROM0 ROM Entries register 206	0x00000000
0x3110033C	DAPROM0_ROMENTRY207	DAPROM0 ROM Entries register 207	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100340	DAPROM0_ROMENTRY208	DAPROM0 ROM Entries register 208	0x00000000
0x31100344	DAPROM0_ROMENTRY209	DAPROM0 ROM Entries register 209	0x00000000
0x31100348	DAPROM0_ROMENTRY210	DAPROM0 ROM Entries register 210	0x00000000
0x3110034C	DAPROM0_ROMENTRY211	DAPROM0 ROM Entries register 211	0x00000000
0x31100350	DAPROM0_ROMENTRY212	DAPROM0 ROM Entries register 212	0x00000000
0x31100354	DAPROM0_ROMENTRY213	DAPROM0 ROM Entries register 213	0x00000000
0x31100358	DAPROM0_ROMENTRY214	DAPROM0 ROM Entries register 214	0x00000000
0x3110035C	DAPROM0_ROMENTRY215	DAPROM0 ROM Entries register 215	0x00000000
0x31100360	DAPROM0_ROMENTRY216	DAPROM0 ROM Entries register 216	0x00000000
0x31100364	DAPROM0_ROMENTRY217	DAPROM0 ROM Entries register 217	0x00000000
0x31100368	DAPROM0_ROMENTRY218	DAPROM0 ROM Entries register 218	0x00000000
0x3110036C	DAPROM0_ROMENTRY219	DAPROM0 ROM Entries register 219	0x00000000
0x31100370	DAPROM0_ROMENTRY220	DAPROM0 ROM Entries register 220	0x00000000
0x31100374	DAPROM0_ROMENTRY221	DAPROM0 ROM Entries register 221	0x00000000
0x31100378	DAPROM0_ROMENTRY222	DAPROM0 ROM Entries register 222	0x00000000
0x3110037C	DAPROM0_ROMENTRY223	DAPROM0 ROM Entries register 223	0x00000000
0x31100380	DAPROM0_ROMENTRY224	DAPROM0 ROM Entries register 224	0x00000000
0x31100384	DAPROM0_ROMENTRY225	DAPROM0 ROM Entries register 225	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100388	DAPROM0_ROMENTRY226	DAPROM0 ROM Entries register 226	0x00000000
0x3110038C	DAPROM0_ROMENTRY227	DAPROM0 ROM Entries register 227	0x00000000
0x31100390	DAPROM0_ROMENTRY228	DAPROM0 ROM Entries register 228	0x00000000
0x31100394	DAPROM0_ROMENTRY229	DAPROM0 ROM Entries register 229	0x00000000
0x31100398	DAPROM0_ROMENTRY230	DAPROM0 ROM Entries register 230	0x00000000
0x3110039C	DAPROM0_ROMENTRY231	DAPROM0 ROM Entries register 231	0x00000000
0x311003A0	DAPROM0_ROMENTRY232	DAPROM0 ROM Entries register 232	0x00000000
0x311003A4	DAPROM0_ROMENTRY233	DAPROM0 ROM Entries register 233	0x00000000
0x311003A8	DAPROM0_ROMENTRY234	DAPROM0 ROM Entries register 234	0x00000000
0x311003AC	DAPROM0_ROMENTRY235	DAPROM0 ROM Entries register 235	0x00000000
0x311003B0	DAPROM0_ROMENTRY236	DAPROM0 ROM Entries register 236	0x00000000
0x311003B4	DAPROM0_ROMENTRY237	DAPROM0 ROM Entries register 237	0x00000000
0x311003B8	DAPROM0_ROMENTRY238	DAPROM0 ROM Entries register 238	0x00000000
0x311003BC	DAPROM0_ROMENTRY239	DAPROM0 ROM Entries register 239	0x00000000
0x311003C0	DAPROM0_ROMENTRY240	DAPROM0 ROM Entries register 240	0x00000000
0x311003C4	DAPROM0_ROMENTRY241	DAPROM0 ROM Entries register 241	0x00000000
0x311003C8	DAPROM0_ROMENTRY242	DAPROM0 ROM Entries register 242	0x00000000
0x311003CC	DAPROM0_ROMENTRY243	DAPROM0 ROM Entries register 243	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311003D0	DAPROM0_ROMENTRY244	DAPROM0 ROM Entries register 244	0x00000000
0x311003D4	DAPROM0_ROMENTRY245	DAPROM0 ROM Entries register 245	0x00000000
0x311003D8	DAPROM0_ROMENTRY246	DAPROM0 ROM Entries register 246	0x00000000
0x311003DC	DAPROM0_ROMENTRY247	DAPROM0 ROM Entries register 247	0x00000000
0x311003E0	DAPROM0_ROMENTRY248	DAPROM0 ROM Entries register 248	0x00000000
0x311003E4	DAPROM0_ROMENTRY249	DAPROM0 ROM Entries register 249	0x00000000
0x311003E8	DAPROM0_ROMENTRY250	DAPROM0 ROM Entries register 250	0x00000000
0x311003EC	DAPROM0_ROMENTRY251	DAPROM0 ROM Entries register 251	0x00000000
0x311003F0	DAPROM0_ROMENTRY252	DAPROM0 ROM Entries register 252	0x00000000
0x311003F4	DAPROM0_ROMENTRY253	DAPROM0 ROM Entries register 253	0x00000000
0x311003F8	DAPROM0_ROMENTRY254	DAPROM0 ROM Entries register 254	0x00000000
0x311003FC	DAPROM0_ROMENTRY255	DAPROM0 ROM Entries register 255	0x00000000
0x31100400	DAPROM0_ROMENTRY256	DAPROM0 ROM Entries register 256	0x00000000
0x31100404	DAPROM0_ROMENTRY257	DAPROM0 ROM Entries register 257	0x00000000
0x31100408	DAPROM0_ROMENTRY258	DAPROM0 ROM Entries register 258	0x00000000
0x3110040C	DAPROM0_ROMENTRY259	DAPROM0 ROM Entries register 259	0x00000000
0x31100410	DAPROM0_ROMENTRY260	DAPROM0 ROM Entries register 260	0x00000000
0x31100414	DAPROM0_ROMENTRY261	DAPROM0 ROM Entries register 261	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100418	DAPROM0_ROMENTRY262	DAPROM0 ROM Entries register 262	0x00000000
0x3110041C	DAPROM0_ROMENTRY263	DAPROM0 ROM Entries register 263	0x00000000
0x31100420	DAPROM0_ROMENTRY264	DAPROM0 ROM Entries register 264	0x00000000
0x31100424	DAPROM0_ROMENTRY265	DAPROM0 ROM Entries register 265	0x00000000
0x31100428	DAPROM0_ROMENTRY266	DAPROM0 ROM Entries register 266	0x00000000
0x3110042C	DAPROM0_ROMENTRY267	DAPROM0 ROM Entries register 267	0x00000000
0x31100430	DAPROM0_ROMENTRY268	DAPROM0 ROM Entries register 268	0x00000000
0x31100434	DAPROM0_ROMENTRY269	DAPROM0 ROM Entries register 269	0x00000000
0x31100438	DAPROM0_ROMENTRY270	DAPROM0 ROM Entries register 270	0x00000000
0x3110043C	DAPROM0_ROMENTRY271	DAPROM0 ROM Entries register 271	0x00000000
0x31100440	DAPROM0_ROMENTRY272	DAPROM0 ROM Entries register 272	0x00000000
0x31100444	DAPROM0_ROMENTRY273	DAPROM0 ROM Entries register 273	0x00000000
0x31100448	DAPROM0_ROMENTRY274	DAPROM0 ROM Entries register 274	0x00000000
0x3110044C	DAPROM0_ROMENTRY275	DAPROM0 ROM Entries register 275	0x00000000
0x31100450	DAPROM0_ROMENTRY276	DAPROM0 ROM Entries register 276	0x00000000
0x31100454	DAPROM0_ROMENTRY277	DAPROM0 ROM Entries register 277	0x00000000
0x31100458	DAPROM0_ROMENTRY278	DAPROM0 ROM Entries register 278	0x00000000
0x3110045C	DAPROM0_ROMENTRY279	DAPROM0 ROM Entries register 279	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100460	DAPROM0_ROMENTRY280	DAPROM0 ROM Entries register 280	0x00000000
0x31100464	DAPROM0_ROMENTRY281	DAPROM0 ROM Entries register 281	0x00000000
0x31100468	DAPROM0_ROMENTRY282	DAPROM0 ROM Entries register 282	0x00000000
0x3110046C	DAPROM0_ROMENTRY283	DAPROM0 ROM Entries register 283	0x00000000
0x31100470	DAPROM0_ROMENTRY284	DAPROM0 ROM Entries register 284	0x00000000
0x31100474	DAPROM0_ROMENTRY285	DAPROM0 ROM Entries register 285	0x00000000
0x31100478	DAPROM0_ROMENTRY286	DAPROM0 ROM Entries register 286	0x00000000
0x3110047C	DAPROM0_ROMENTRY287	DAPROM0 ROM Entries register 287	0x00000000
0x31100480	DAPROM0_ROMENTRY288	DAPROM0 ROM Entries register 288	0x00000000
0x31100484	DAPROM0_ROMENTRY289	DAPROM0 ROM Entries register 289	0x00000000
0x31100488	DAPROM0_ROMENTRY290	DAPROM0 ROM Entries register 290	0x00000000
0x3110048C	DAPROM0_ROMENTRY291	DAPROM0 ROM Entries register 291	0x00000000
0x31100490	DAPROM0_ROMENTRY292	DAPROM0 ROM Entries register 292	0x00000000
0x31100494	DAPROM0_ROMENTRY293	DAPROM0 ROM Entries register 293	0x00000000
0x31100498	DAPROM0_ROMENTRY294	DAPROM0 ROM Entries register 294	0x00000000
0x3110049C	DAPROM0_ROMENTRY295	DAPROM0 ROM Entries register 295	0x00000000
0x311004A0	DAPROM0_ROMENTRY296	DAPROM0 ROM Entries register 296	0x00000000
0x311004A4	DAPROM0_ROMENTRY297	DAPROM0 ROM Entries register 297	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311004A8	DAPROM0_ROMENTRY298	DAPROM0 ROM Entries register 298	0x00000000
0x311004AC	DAPROM0_ROMENTRY299	DAPROM0 ROM Entries register 299	0x00000000
0x311004B0	DAPROM0_ROMENTRY300	DAPROM0 ROM Entries register 300	0x00000000
0x311004B4	DAPROM0_ROMENTRY301	DAPROM0 ROM Entries register 301	0x00000000
0x311004B8	DAPROM0_ROMENTRY302	DAPROM0 ROM Entries register 302	0x00000000
0x311004BC	DAPROM0_ROMENTRY303	DAPROM0 ROM Entries register 303	0x00000000
0x311004C0	DAPROM0_ROMENTRY304	DAPROM0 ROM Entries register 304	0x00000000
0x311004C4	DAPROM0_ROMENTRY305	DAPROM0 ROM Entries register 305	0x00000000
0x311004C8	DAPROM0_ROMENTRY306	DAPROM0 ROM Entries register 306	0x00000000
0x311004CC	DAPROM0_ROMENTRY307	DAPROM0 ROM Entries register 307	0x00000000
0x311004D0	DAPROM0_ROMENTRY308	DAPROM0 ROM Entries register 308	0x00000000
0x311004D4	DAPROM0_ROMENTRY309	DAPROM0 ROM Entries register 309	0x00000000
0x311004D8	DAPROM0_ROMENTRY310	DAPROM0 ROM Entries register 310	0x00000000
0x311004DC	DAPROM0_ROMENTRY311	DAPROM0 ROM Entries register 311	0x00000000
0x311004E0	DAPROM0_ROMENTRY312	DAPROM0 ROM Entries register 312	0x00000000
0x311004E4	DAPROM0_ROMENTRY313	DAPROM0 ROM Entries register 313	0x00000000
0x311004E8	DAPROM0_ROMENTRY314	DAPROM0 ROM Entries register 314	0x00000000
0x311004EC	DAPROM0_ROMENTRY315	DAPROM0 ROM Entries register 315	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311004F0	DAPROM0_ROMENTRY316	DAPROM0 ROM Entries register 316	0x00000000
0x311004F4	DAPROM0_ROMENTRY317	DAPROM0 ROM Entries register 317	0x00000000
0x311004F8	DAPROM0_ROMENTRY318	DAPROM0 ROM Entries register 318	0x00000000
0x311004FC	DAPROM0_ROMENTRY319	DAPROM0 ROM Entries register 319	0x00000000
0x31100500	DAPROM0_ROMENTRY320	DAPROM0 ROM Entries register 320	0x00000000
0x31100504	DAPROM0_ROMENTRY321	DAPROM0 ROM Entries register 321	0x00000000
0x31100508	DAPROM0_ROMENTRY322	DAPROM0 ROM Entries register 322	0x00000000
0x3110050C	DAPROM0_ROMENTRY323	DAPROM0 ROM Entries register 323	0x00000000
0x31100510	DAPROM0_ROMENTRY324	DAPROM0 ROM Entries register 324	0x00000000
0x31100514	DAPROM0_ROMENTRY325	DAPROM0 ROM Entries register 325	0x00000000
0x31100518	DAPROM0_ROMENTRY326	DAPROM0 ROM Entries register 326	0x00000000
0x3110051C	DAPROM0_ROMENTRY327	DAPROM0 ROM Entries register 327	0x00000000
0x31100520	DAPROM0_ROMENTRY328	DAPROM0 ROM Entries register 328	0x00000000
0x31100524	DAPROM0_ROMENTRY329	DAPROM0 ROM Entries register 329	0x00000000
0x31100528	DAPROM0_ROMENTRY330	DAPROM0 ROM Entries register 330	0x00000000
0x3110052C	DAPROM0_ROMENTRY331	DAPROM0 ROM Entries register 331	0x00000000
0x31100530	DAPROM0_ROMENTRY332	DAPROM0 ROM Entries register 332	0x00000000
0x31100534	DAPROM0_ROMENTRY333	DAPROM0 ROM Entries register 333	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100538	DAPROM0_ROMENTRY334	DAPROM0 ROM Entries register 334	0x00000000
0x3110053C	DAPROM0_ROMENTRY335	DAPROM0 ROM Entries register 335	0x00000000
0x31100540	DAPROM0_ROMENTRY336	DAPROM0 ROM Entries register 336	0x00000000
0x31100544	DAPROM0_ROMENTRY337	DAPROM0 ROM Entries register 337	0x00000000
0x31100548	DAPROM0_ROMENTRY338	DAPROM0 ROM Entries register 338	0x00000000
0x3110054C	DAPROM0_ROMENTRY339	DAPROM0 ROM Entries register 339	0x00000000
0x31100550	DAPROM0_ROMENTRY340	DAPROM0 ROM Entries register 340	0x00000000
0x31100554	DAPROM0_ROMENTRY341	DAPROM0 ROM Entries register 341	0x00000000
0x31100558	DAPROM0_ROMENTRY342	DAPROM0 ROM Entries register 342	0x00000000
0x3110055C	DAPROM0_ROMENTRY343	DAPROM0 ROM Entries register 343	0x00000000
0x31100560	DAPROM0_ROMENTRY344	DAPROM0 ROM Entries register 344	0x00000000
0x31100564	DAPROM0_ROMENTRY345	DAPROM0 ROM Entries register 345	0x00000000
0x31100568	DAPROM0_ROMENTRY346	DAPROM0 ROM Entries register 346	0x00000000
0x3110056C	DAPROM0_ROMENTRY347	DAPROM0 ROM Entries register 347	0x00000000
0x31100570	DAPROM0_ROMENTRY348	DAPROM0 ROM Entries register 348	0x00000000
0x31100574	DAPROM0_ROMENTRY349	DAPROM0 ROM Entries register 349	0x00000000
0x31100578	DAPROM0_ROMENTRY350	DAPROM0 ROM Entries register 350	0x00000000
0x3110057C	DAPROM0_ROMENTRY351	DAPROM0 ROM Entries register 351	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100580	DAPROM0_ROMENTRY352	DAPROM0 ROM Entries register 352	0x00000000
0x31100584	DAPROM0_ROMENTRY353	DAPROM0 ROM Entries register 353	0x00000000
0x31100588	DAPROM0_ROMENTRY354	DAPROM0 ROM Entries register 354	0x00000000
0x3110058C	DAPROM0_ROMENTRY355	DAPROM0 ROM Entries register 355	0x00000000
0x31100590	DAPROM0_ROMENTRY356	DAPROM0 ROM Entries register 356	0x00000000
0x31100594	DAPROM0_ROMENTRY357	DAPROM0 ROM Entries register 357	0x00000000
0x31100598	DAPROM0_ROMENTRY358	DAPROM0 ROM Entries register 358	0x00000000
0x3110059C	DAPROM0_ROMENTRY359	DAPROM0 ROM Entries register 359	0x00000000
0x311005A0	DAPROM0_ROMENTRY360	DAPROM0 ROM Entries register 360	0x00000000
0x311005A4	DAPROM0_ROMENTRY361	DAPROM0 ROM Entries register 361	0x00000000
0x311005A8	DAPROM0_ROMENTRY362	DAPROM0 ROM Entries register 362	0x00000000
0x311005AC	DAPROM0_ROMENTRY363	DAPROM0 ROM Entries register 363	0x00000000
0x311005B0	DAPROM0_ROMENTRY364	DAPROM0 ROM Entries register 364	0x00000000
0x311005B4	DAPROM0_ROMENTRY365	DAPROM0 ROM Entries register 365	0x00000000
0x311005B8	DAPROM0_ROMENTRY366	DAPROM0 ROM Entries register 366	0x00000000
0x311005BC	DAPROM0_ROMENTRY367	DAPROM0 ROM Entries register 367	0x00000000
0x311005C0	DAPROM0_ROMENTRY368	DAPROM0 ROM Entries register 368	0x00000000
0x311005C4	DAPROM0_ROMENTRY369	DAPROM0 ROM Entries register 369	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311005C8	DAPROM0_ROMENTRY370	DAPROM0 ROM Entries register 370	0x00000000
0x311005CC	DAPROM0_ROMENTRY371	DAPROM0 ROM Entries register 371	0x00000000
0x311005D0	DAPROM0_ROMENTRY372	DAPROM0 ROM Entries register 372	0x00000000
0x311005D4	DAPROM0_ROMENTRY373	DAPROM0 ROM Entries register 373	0x00000000
0x311005D8	DAPROM0_ROMENTRY374	DAPROM0 ROM Entries register 374	0x00000000
0x311005DC	DAPROM0_ROMENTRY375	DAPROM0 ROM Entries register 375	0x00000000
0x311005E0	DAPROM0_ROMENTRY376	DAPROM0 ROM Entries register 376	0x00000000
0x311005E4	DAPROM0_ROMENTRY377	DAPROM0 ROM Entries register 377	0x00000000
0x311005E8	DAPROM0_ROMENTRY378	DAPROM0 ROM Entries register 378	0x00000000
0x311005EC	DAPROM0_ROMENTRY379	DAPROM0 ROM Entries register 379	0x00000000
0x311005F0	DAPROM0_ROMENTRY380	DAPROM0 ROM Entries register 380	0x00000000
0x311005F4	DAPROM0_ROMENTRY381	DAPROM0 ROM Entries register 381	0x00000000
0x311005F8	DAPROM0_ROMENTRY382	DAPROM0 ROM Entries register 382	0x00000000
0x311005FC	DAPROM0_ROMENTRY383	DAPROM0 ROM Entries register 383	0x00000000
0x31100600	DAPROM0_ROMENTRY384	DAPROM0 ROM Entries register 384	0x00000000
0x31100604	DAPROM0_ROMENTRY385	DAPROM0 ROM Entries register 385	0x00000000
0x31100608	DAPROM0_ROMENTRY386	DAPROM0 ROM Entries register 386	0x00000000
0x3110060C	DAPROM0_ROMENTRY387	DAPROM0 ROM Entries register 387	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100610	DAPROM0_ROMENTRY388	DAPROM0 ROM Entries register 388	0x00000000
0x31100614	DAPROM0_ROMENTRY389	DAPROM0 ROM Entries register 389	0x00000000
0x31100618	DAPROM0_ROMENTRY390	DAPROM0 ROM Entries register 390	0x00000000
0x3110061C	DAPROM0_ROMENTRY391	DAPROM0 ROM Entries register 391	0x00000000
0x31100620	DAPROM0_ROMENTRY392	DAPROM0 ROM Entries register 392	0x00000000
0x31100624	DAPROM0_ROMENTRY393	DAPROM0 ROM Entries register 393	0x00000000
0x31100628	DAPROM0_ROMENTRY394	DAPROM0 ROM Entries register 394	0x00000000
0x3110062C	DAPROM0_ROMENTRY395	DAPROM0 ROM Entries register 395	0x00000000
0x31100630	DAPROM0_ROMENTRY396	DAPROM0 ROM Entries register 396	0x00000000
0x31100634	DAPROM0_ROMENTRY397	DAPROM0 ROM Entries register 397	0x00000000
0x31100638	DAPROM0_ROMENTRY398	DAPROM0 ROM Entries register 398	0x00000000
0x3110063C	DAPROM0_ROMENTRY399	DAPROM0 ROM Entries register 399	0x00000000
0x31100640	DAPROM0_ROMENTRY400	DAPROM0 ROM Entries register 400	0x00000000
0x31100644	DAPROM0_ROMENTRY401	DAPROM0 ROM Entries register 401	0x00000000
0x31100648	DAPROM0_ROMENTRY402	DAPROM0 ROM Entries register 402	0x00000000
0x3110064C	DAPROM0_ROMENTRY403	DAPROM0 ROM Entries register 403	0x00000000
0x31100650	DAPROM0_ROMENTRY404	DAPROM0 ROM Entries register 404	0x00000000
0x31100654	DAPROM0_ROMENTRY405	DAPROM0 ROM Entries register 405	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100658	DAPROM0_ROMENTRY406	DAPROM0 ROM Entries register 406	0x00000000
0x3110065C	DAPROM0_ROMENTRY407	DAPROM0 ROM Entries register 407	0x00000000
0x31100660	DAPROM0_ROMENTRY408	DAPROM0 ROM Entries register 408	0x00000000
0x31100664	DAPROM0_ROMENTRY409	DAPROM0 ROM Entries register 409	0x00000000
0x31100668	DAPROM0_ROMENTRY410	DAPROM0 ROM Entries register 410	0x00000000
0x3110066C	DAPROM0_ROMENTRY411	DAPROM0 ROM Entries register 411	0x00000000
0x31100670	DAPROM0_ROMENTRY412	DAPROM0 ROM Entries register 412	0x00000000
0x31100674	DAPROM0_ROMENTRY413	DAPROM0 ROM Entries register 413	0x00000000
0x31100678	DAPROM0_ROMENTRY414	DAPROM0 ROM Entries register 414	0x00000000
0x3110067C	DAPROM0_ROMENTRY415	DAPROM0 ROM Entries register 415	0x00000000
0x31100680	DAPROM0_ROMENTRY416	DAPROM0 ROM Entries register 416	0x00000000
0x31100684	DAPROM0_ROMENTRY417	DAPROM0 ROM Entries register 417	0x00000000
0x31100688	DAPROM0_ROMENTRY418	DAPROM0 ROM Entries register 418	0x00000000
0x3110068C	DAPROM0_ROMENTRY419	DAPROM0 ROM Entries register 419	0x00000000
0x31100690	DAPROM0_ROMENTRY420	DAPROM0 ROM Entries register 420	0x00000000
0x31100694	DAPROM0_ROMENTRY421	DAPROM0 ROM Entries register 421	0x00000000
0x31100698	DAPROM0_ROMENTRY422	DAPROM0 ROM Entries register 422	0x00000000
0x3110069C	DAPROM0_ROMENTRY423	DAPROM0 ROM Entries register 423	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311006A0	DAPROM0_ROMENTRY424	DAPROM0 ROM Entries register 424	0x00000000
0x311006A4	DAPROM0_ROMENTRY425	DAPROM0 ROM Entries register 425	0x00000000
0x311006A8	DAPROM0_ROMENTRY426	DAPROM0 ROM Entries register 426	0x00000000
0x311006AC	DAPROM0_ROMENTRY427	DAPROM0 ROM Entries register 427	0x00000000
0x311006B0	DAPROM0_ROMENTRY428	DAPROM0 ROM Entries register 428	0x00000000
0x311006B4	DAPROM0_ROMENTRY429	DAPROM0 ROM Entries register 429	0x00000000
0x311006B8	DAPROM0_ROMENTRY430	DAPROM0 ROM Entries register 430	0x00000000
0x311006BC	DAPROM0_ROMENTRY431	DAPROM0 ROM Entries register 431	0x00000000
0x311006C0	DAPROM0_ROMENTRY432	DAPROM0 ROM Entries register 432	0x00000000
0x311006C4	DAPROM0_ROMENTRY433	DAPROM0 ROM Entries register 433	0x00000000
0x311006C8	DAPROM0_ROMENTRY434	DAPROM0 ROM Entries register 434	0x00000000
0x311006CC	DAPROM0_ROMENTRY435	DAPROM0 ROM Entries register 435	0x00000000
0x311006D0	DAPROM0_ROMENTRY436	DAPROM0 ROM Entries register 436	0x00000000
0x311006D4	DAPROM0_ROMENTRY437	DAPROM0 ROM Entries register 437	0x00000000
0x311006D8	DAPROM0_ROMENTRY438	DAPROM0 ROM Entries register 438	0x00000000
0x311006DC	DAPROM0_ROMENTRY439	DAPROM0 ROM Entries register 439	0x00000000
0x311006E0	DAPROM0_ROMENTRY440	DAPROM0 ROM Entries register 440	0x00000000
0x311006E4	DAPROM0_ROMENTRY441	DAPROM0 ROM Entries register 441	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311006E8	DAPROM0_ROMENTRY442	DAPROM0 ROM Entries register 442	0x00000000
0x311006EC	DAPROM0_ROMENTRY443	DAPROM0 ROM Entries register 443	0x00000000
0x311006F0	DAPROM0_ROMENTRY444	DAPROM0 ROM Entries register 444	0x00000000
0x311006F4	DAPROM0_ROMENTRY445	DAPROM0 ROM Entries register 445	0x00000000
0x311006F8	DAPROM0_ROMENTRY446	DAPROM0 ROM Entries register 446	0x00000000
0x311006FC	DAPROM0_ROMENTRY447	DAPROM0 ROM Entries register 447	0x00000000
0x31100700	DAPROM0_ROMENTRY448	DAPROM0 ROM Entries register 448	0x00000000
0x31100704	DAPROM0_ROMENTRY449	DAPROM0 ROM Entries register 449	0x00000000
0x31100708	DAPROM0_ROMENTRY450	DAPROM0 ROM Entries register 450	0x00000000
0x3110070C	DAPROM0_ROMENTRY451	DAPROM0 ROM Entries register 451	0x00000000
0x31100710	DAPROM0_ROMENTRY452	DAPROM0 ROM Entries register 452	0x00000000
0x31100714	DAPROM0_ROMENTRY453	DAPROM0 ROM Entries register 453	0x00000000
0x31100718	DAPROM0_ROMENTRY454	DAPROM0 ROM Entries register 454	0x00000000
0x3110071C	DAPROM0_ROMENTRY455	DAPROM0 ROM Entries register 455	0x00000000
0x31100720	DAPROM0_ROMENTRY456	DAPROM0 ROM Entries register 456	0x00000000
0x31100724	DAPROM0_ROMENTRY457	DAPROM0 ROM Entries register 457	0x00000000
0x31100728	DAPROM0_ROMENTRY458	DAPROM0 ROM Entries register 458	0x00000000
0x3110072C	DAPROM0_ROMENTRY459	DAPROM0 ROM Entries register 459	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100730	DAPROM0_ROMENTRY460	DAPROM0 ROM Entries register 460	0x00000000
0x31100734	DAPROM0_ROMENTRY461	DAPROM0 ROM Entries register 461	0x00000000
0x31100738	DAPROM0_ROMENTRY462	DAPROM0 ROM Entries register 462	0x00000000
0x3110073C	DAPROM0_ROMENTRY463	DAPROM0 ROM Entries register 463	0x00000000
0x31100740	DAPROM0_ROMENTRY464	DAPROM0 ROM Entries register 464	0x00000000
0x31100744	DAPROM0_ROMENTRY465	DAPROM0 ROM Entries register 465	0x00000000
0x31100748	DAPROM0_ROMENTRY466	DAPROM0 ROM Entries register 466	0x00000000
0x3110074C	DAPROM0_ROMENTRY467	DAPROM0 ROM Entries register 467	0x00000000
0x31100750	DAPROM0_ROMENTRY468	DAPROM0 ROM Entries register 468	0x00000000
0x31100754	DAPROM0_ROMENTRY469	DAPROM0 ROM Entries register 469	0x00000000
0x31100758	DAPROM0_ROMENTRY470	DAPROM0 ROM Entries register 470	0x00000000
0x3110075C	DAPROM0_ROMENTRY471	DAPROM0 ROM Entries register 471	0x00000000
0x31100760	DAPROM0_ROMENTRY472	DAPROM0 ROM Entries register 472	0x00000000
0x31100764	DAPROM0_ROMENTRY473	DAPROM0 ROM Entries register 473	0x00000000
0x31100768	DAPROM0_ROMENTRY474	DAPROM0 ROM Entries register 474	0x00000000
0x3110076C	DAPROM0_ROMENTRY475	DAPROM0 ROM Entries register 475	0x00000000
0x31100770	DAPROM0_ROMENTRY476	DAPROM0 ROM Entries register 476	0x00000000
0x31100774	DAPROM0_ROMENTRY477	DAPROM0 ROM Entries register 477	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100778	DAPROM0_ROMENTRY478	DAPROM0 ROM Entries register 478	0x00000000
0x3110077C	DAPROM0_ROMENTRY479	DAPROM0 ROM Entries register 479	0x00000000
0x31100780	DAPROM0_ROMENTRY480	DAPROM0 ROM Entries register 480	0x00000000
0x31100784	DAPROM0_ROMENTRY481	DAPROM0 ROM Entries register 481	0x00000000
0x31100788	DAPROM0_ROMENTRY482	DAPROM0 ROM Entries register 482	0x00000000
0x3110078C	DAPROM0_ROMENTRY483	DAPROM0 ROM Entries register 483	0x00000000
0x31100790	DAPROM0_ROMENTRY484	DAPROM0 ROM Entries register 484	0x00000000
0x31100794	DAPROM0_ROMENTRY485	DAPROM0 ROM Entries register 485	0x00000000
0x31100798	DAPROM0_ROMENTRY486	DAPROM0 ROM Entries register 486	0x00000000
0x3110079C	DAPROM0_ROMENTRY487	DAPROM0 ROM Entries register 487	0x00000000
0x311007A0	DAPROM0_ROMENTRY488	DAPROM0 ROM Entries register 488	0x00000000
0x311007A4	DAPROM0_ROMENTRY489	DAPROM0 ROM Entries register 489	0x00000000
0x311007A8	DAPROM0_ROMENTRY490	DAPROM0 ROM Entries register 490	0x00000000
0x311007AC	DAPROM0_ROMENTRY491	DAPROM0 ROM Entries register 491	0x00000000
0x311007B0	DAPROM0_ROMENTRY492	DAPROM0 ROM Entries register 492	0x00000000
0x311007B4	DAPROM0_ROMENTRY493	DAPROM0 ROM Entries register 493	0x00000000
0x311007B8	DAPROM0_ROMENTRY494	DAPROM0 ROM Entries register 494	0x00000000
0x311007BC	DAPROM0_ROMENTRY495	DAPROM0 ROM Entries register 495	0x00000000

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311007C0	DAPROM0_ROMENTRY496	DAPROM0 ROM Entries register 496	0x00000000
0x311007C4	DAPROM0_ROMENTRY497	DAPROM0 ROM Entries register 497	0x00000000
0x311007C8	DAPROM0_ROMENTRY498	DAPROM0 ROM Entries register 498	0x00000000
0x311007CC	DAPROM0_ROMENTRY499	DAPROM0 ROM Entries register 499	0x00000000
0x311007D0	DAPROM0_ROMENTRY500	DAPROM0 ROM Entries register 500	0x00000000
0x311007D4	DAPROM0_ROMENTRY501	DAPROM0 ROM Entries register 501	0x00000000
0x311007D8	DAPROM0_ROMENTRY502	DAPROM0 ROM Entries register 502	0x00000000
0x311007DC	DAPROM0_ROMENTRY503	DAPROM0 ROM Entries register 503	0x00000000
0x311007E0	DAPROM0_ROMENTRY504	DAPROM0 ROM Entries register 504	0x00000000
0x311007E4	DAPROM0_ROMENTRY505	DAPROM0 ROM Entries register 505	0x00000000
0x311007E8	DAPROM0_ROMENTRY506	DAPROM0 ROM Entries register 506	0x00000000
0x311007EC	DAPROM0_ROMENTRY507	DAPROM0 ROM Entries register 507	0x00000000
0x311007F0	DAPROM0_ROMENTRY508	DAPROM0 ROM Entries register 508	0x00000000
0x311007F4	DAPROM0_ROMENTRY509	DAPROM0 ROM Entries register 509	0x00000000
0x311007F8	DAPROM0_ROMENTRY510	DAPROM0 ROM Entries register 510	0x00000000
0x311007FC	DAPROM0_ROMENTRY511	DAPROM0 ROM Entries register 511	0x00000000
0x31100800	DAPROM0_ROMENTRY512	DAPROM0 ROM Entries register 512	0x00000000
0x31100FB8	DAPROM0_AUTHSTATUS	DAPROM0 Authentication Status Register	0x00000000
0x31100FBC	DAPROM0_DEVARCH	DAPROM0 Device Architecture Register	0x47700AF7

Table A-6: ADSP-SC59x DAPROM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31100FC8	DAPROM0_DEVID	DAPROM0 Device Configuration Register	0x00000000
0x31100FD0	DAPROM0_PIDR4	DAPROM0 Peripheral Identification Register 4	0x00000000
0x31100FD4	DAPROM0_PIDR5	DAPROM0 Peripheral Identification Register 5	0x00000000
0x31100FD8	DAPROM0_PIDR6	DAPROM0 Peripheral Identification Register 6	0x00000000
0x31100FDC	DAPROM0_PIDR7	DAPROM0 Peripheral Identification Register 7	0x00000000
0x31100FE0	DAPROM0_PIDR0	DAPROM0 Peripheral Identification Register 0	0x0000002E
0x31100FE4	DAPROM0_PIDR1	DAPROM0 Peripheral Identification Register 1	0x00000058
0x31100FE8	DAPROM0_PIDR2	DAPROM0 Peripheral Identification Register 2	0x0000000E
0x31100FEC	DAPROM0_PIDR3	DAPROM0 Peripheral Identification Register 3	0x00000020
0x31100FF0	DAPROM0_CIDR0	DAPROM0 Component Identification Register 0	0x0000000D
0x31100FF4	DAPROM0_CIDR1	DAPROM0 Component Identification Register 1	0x00000090
0x31100FF8	DAPROM0_CIDR2	DAPROM0 Component Identification Register 2	0x00000005
0x31100FFC	DAPROM0_CIDR3	DAPROM0 Component Identification Register 3	0x000000B1

Table A-7: ADSP-SC59x TRACE0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3110B000	TRACE0_SSPSR	TRACE0 Supported Port Size Register	0xFFFFFFFF
0x3110B004	TRACE0_CSPSR	TRACE0 Current Port Size Register	0x00000001
0x3110B100	TRACE0_STMR	TRACE0 Supported Trigger Modes Register	0x0000011F
0x3110B104	TRACE0_TCVR	TRACE0 Trigger Counter Value Register	0x00000000
0x3110B108	TRACE0_TCMR	TRACE0 Trigger Counter Multiplier Register	0x00000000
0x3110B200	TRACE0_STPMR	TRACE0 Supported Test Patterns/Modes Register	0x0003000F
0x3110B204	TRACE0_CTPMR	TRACE0 Current Test Patterns/Modes Register	0x00000000
0x3110B208	TRACE0_TPRCR	TRACE0 Test Pattern Repeat Counter Register	0x00000000
0x3110B300	TRACE0_FFSR	TRACE0 Formatter and Flush Status Register	0x00000006
0x3110B304	TRACE0_FFCR	TRACE0 Formatter and Flush Control Register	0x00001000
0x3110B308	TRACE0_FSCR	TRACE0 Formatter Synchronization Count Register	0x00000040
0x3110B400	TRACE0_EXTCTLIN	TRACE0 External Control Port In Register	0x00000000
0x3110B404	TRACE0_EXTCTLOUT	TRACE0 External Control Port Out Register	0x00000000
0x3110BEE8	TRACE0_ITTRFLIN	TRACE0 Integration Test Trigger In and Flush In Register	0x00000000

Table A-7: ADSP-SC59x TRACE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110BEEC	TRACE0_ITATBDATA0	TRACE0 Integration Test ATB Data Register 0	0x00000000
0x3110BEF0	TRACE0_ITATBCTR2	TRACE0 Integration Test ATB Control Register 2	0x00000000
0x3110BEF4	TRACE0_ITATBCTR1	TRACE0 Integration Test ATB Control Register 1	0x00000000
0x3110BEF8	TRACE0_ITATBCTR0	TRACE0 Integration Test ATB Control Register 0	0x00000000
0x3110BEFC	TRACE0_ITOUTCTR	TRACE0 Integration Test Output Control Register	0x00000000
0x3110BF00	TRACE0_ITCTRL	TRACE0 Integration Mode Control Register	0x00000000
0x3110BFA0	TRACE0_CLAIMSET	TRACE0 Claim Tag Set Register	0x0000000F
0x3110BFA4	TRACE0_CLAIMCLR	TRACE0 Claim Tag Clear Register	0x00000000
0x3110BFB8	TRACE0_AUTHSTATUS	TRACE0 Authentication Status Register	0x00000000
0x3110BFBC	TRACE0_DEVARCH	TRACE0 Device Architecture Register	0x00000000
0x3110BFC8	TRACE0_DEVID	TRACE0 Device Configuration Register	0x00000020
0x3110BFCC	TRACE0_DEVTYPE	TRACE0 Device Type Identifier Register	0x00000011
0x3110BFD0	TRACE0_PIDR4	TRACE0 Peripheral Identification Register 4	0x00000004
0x3110BFD4	TRACE0_PIDR5	TRACE0 Peripheral Identification Register 5	0x00000000
0x3110BFD8	TRACE0_PIDR6	TRACE0 Peripheral Identification Register 6	0x00000000
0x3110BFDC	TRACE0_PIDR7	TRACE0 Peripheral Identification Register 7	0x00000000
0x3110BFE0	TRACE0_PIDR0	TRACE0 Peripheral Identification Register 0	0x000000E7
0x3110BFE4	TRACE0_PIDR1	TRACE0 Peripheral Identification Register 1	0x000000B9
0x3110BFE8	TRACE0_PIDR2	TRACE0 Peripheral Identification Register 2	0x0000001B
0x3110BFEC	TRACE0_PIDR3	TRACE0 Peripheral Identification Register 3	0x00000000
0x3110BFF0	TRACE0_CIDR0	TRACE0 Component Identification Register 0	0x0000000D
0x3110BFF4	TRACE0_CIDR1	TRACE0 Component Identification Register 1	0x00000090
0x3110BFF8	TRACE0_CIDR2	TRACE0 Component Identification Register 2	0x00000005
0x3110BFFC	TRACE0_CIDR3	TRACE0 Component Identification Register 3	0x000000B1

Table A-8: ADSP-SC59x TSGENWR0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AE000	TSGENWR0_CNTCR	TSGENWR0 Counter Control Register	0x00000000
0x310AE004	TSGENWR0_CNTSR	TSGENWR0 Counter Status Register	0x00000000
0x310AE008	TSGENWR0_CNTCVL	TSGENWR0 Current value of Counter[31:0]	0x00000000

Table A-8: ADSP-SC59x TSGENWR0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310AE00C	TSGENWR0_CNTCVU	TSGENWR0 Current value of Counter[63:32]	0x00000000
0x310AE020	TSGENWR0_CNTFID0	TSGENWR0 Base Frequency ID register	0x00000000
0x310AEEF8	TSGENWR0_ITSTAT	TSGENWR0 Integration Test Status Register	0x00000000
0x310AEF00	TSGENWR0_ITCTRL	TSGENWR0 Integration Mode Control Register	0x00000000
0x310AEFD0	TSGENWR0_PIDR4	TSGENWR0 Peripheral Identification Register 4	0x00000004
0x310AEFD4	TSGENWR0_PIDR5	TSGENWR0 Peripheral Identification Register 5	0x00000000
0x310AEFD8	TSGENWR0_PIDR6	TSGENWR0 Peripheral Identification Register 6	0x00000000
0x310AEFDC	TSGENWR0_PIDR7	TSGENWR0 Peripheral Identification Register 7	0x00000000
0x310AEFE0	TSGENWR0_PIDR0	TSGENWR0 Peripheral Identification Register 0	0x00000093
0x310AEFE4	TSGENWR0_PIDR1	TSGENWR0 Peripheral Identification Register 1	0x000000B1
0x310AEFE8	TSGENWR0_PIDR2	TSGENWR0 Peripheral Identification Register 2	0x0000000B
0x310AEFEC	TSGENWR0_PIDR3	TSGENWR0 Peripheral Identification Register 3	0x00000000
0x310AEFF0	TSGENWR0_CIDR0	TSGENWR0 Component Identification Register 0	0x0000000D
0x310AEFF4	TSGENWR0_CIDR1	TSGENWR0 Component Identification Register 1	0x000000F0
0x310AEFF8	TSGENWR0_CIDR2	TSGENWR0 Component Identification Register 2	0x00000005
0x310AEFFC	TSGENWR0_CIDR3	TSGENWR0 Component Identification Register 3	0x000000B1
0x3110E000	CSTSGENWR0_CNTCR	CSTSGENWR0 Counter Control Register	0x00000000
0x3110E004	CSTSGENWR0_CNTSR	CSTSGENWR0 Counter Status Register	0x00000000
0x3110E008	CSTSGENWR0_CNTCVL	CSTSGENWR0 Current value of Counter[31:0]	0x00000000
0x3110E00C	CSTSGENWR0_CNTCVU	CSTSGENWR0 Current value of Counter[63:32]	0x00000000
0x3110E020	CSTSGENWR0_CNTFID0	CSTSGENWR0 Base Frequency ID register	0x00000000
0x3110EEF8	CSTSGENWR0_ITSTAT	CSTSGENWR0 Integration Test Status Register	0x00000000
0x3110EF00	CSTSGENWR0_ITCTRL	CSTSGENWR0 Integration Mode Control Register	0x00000000
0x3110EFD0	CSTSGENWR0_PIDR4	CSTSGENWR0 Peripheral Identification Register 4	0x00000004
0x3110EFD4	CSTSGENWR0_PIDR5	CSTSGENWR0 Peripheral Identification Register 5	0x00000000
0x3110EFD8	CSTSGENWR0_PIDR6	CSTSGENWR0 Peripheral Identification Register 6	0x00000000
0x3110EFDC	CSTSGENWR0_PIDR7	CSTSGENWR0 Peripheral Identification Register 7	0x00000000
0x3110EFE0	CSTSGENWR0_PIDR0	CSTSGENWR0 Peripheral Identification Register 0	0x00000093
0x3110EFE4	CSTSGENWR0_PIDR1	CSTSGENWR0 Peripheral Identification Register 1	0x000000B1
0x3110EFE8	CSTSGENWR0_PIDR2	CSTSGENWR0 Peripheral Identification Register 2	0x0000000B
0x3110EFEC	CSTSGENWR0_PIDR3	CSTSGENWR0 Peripheral Identification Register 3	0x00000000

Table A-8: ADSP-SC59x TSGENWR0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3110EFF0	CSTSGENWR0_CIDR0	CSTSGENWR0 Component Identification Register 0	0x0000000D
0x3110EFF4	CSTSGENWR0_CIDR1	CSTSGENWR0 Component Identification Register 1	0x000000F0
0x3110EFF8	CSTSGENWR0_CIDR2	CSTSGENWR0 Component Identification Register 2	0x00000005
0x3110EFFC	CSTSGENWR0_CIDR3	CSTSGENWR0 Component Identification Register 3	0x000000B1

Table A-9: ADSP-SC59x TSGENRD0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AF000	TSGENRD0_CNTCVLREAD	TSGENRD0 Current value of Counter[31:0]	0x00000000
0x310AF004	TSGENRD0_CNTCVUREAD	TSGENRD0 Current value of Counter[63:32]	0x00000000
0x310AFFD0	TSGENRD0_PIDR4	TSGENRD0 Peripheral Identification Register 4	0x00000004
0x310AFFD4	TSGENRD0_PIDR5	TSGENRD0 Peripheral Identification Register 5	0x00000000
0x310AFFD8	TSGENRD0_PIDR6	TSGENRD0 Peripheral Identification Register 6	0x00000000
0x310AFFDC	TSGENRD0_PIDR7	TSGENRD0 Peripheral Identification Register 7	0x00000000
0x310AFFE0	TSGENRD0_PIDR0	TSGENRD0 Peripheral Identification Register 0	0x00000093
0x310AFFE4	TSGENRD0_PIDR1	TSGENRD0 Peripheral Identification Register 1	0x000000B1
0x310AFFE8	TSGENRD0_PIDR2	TSGENRD0 Peripheral Identification Register 2	0x0000000B
0x310AFFEC	TSGENRD0_PIDR3	TSGENRD0 Peripheral Identification Register 3	0x00000000
0x310AFFF0	TSGENRD0_CIDR0	TSGENRD0 Component Identification Register 0	0x0000000D
0x310AFFF4	TSGENRD0_CIDR1	TSGENRD0 Component Identification Register 1	0x000000F0
0x310AFFF8	TSGENRD0_CIDR2	TSGENRD0 Component Identification Register 2	0x00000005
0x310AFFFC	TSGENRD0_CIDR3	TSGENRD0 Component Identification Register 3	0x000000B1

Table A-10: ADSP-SC59x DMC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31071000	DMC0_DDR_LANE0_CTL0	DMC0 Data Lane 0 Control Register 0	0x00000000
0x31071004	DMC0_DDR_LANE0_CTL1	DMC0 Data Lane 0 Control Register 1	0x00000000
0x3107100C	DMC0_DDR_LANE1_CTL0	DMC0 Data Lane 1 Control Register 0	0x00000000

Table A-10: ADSP-SC59x DMC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31071010	DMC0_DDR_LANE1_CTL 1	DMC0 Data Lane 1 Control Register 1	0x00000000
0x31071018	DMC0_DDR_ROOT_CTL	DMC0 DDR ROOT Module Control Register	0x00000800
0x31071034	DMC0_DDR_ZQ_CTL0	DMC0 DDR Calibration Control Register 0	0xE0000000
0x31071038	DMC0_DDR_ZQ_CTL1	DMC0 DDR Calibration Control Register 1	0x00000000
0x3107103C	DMC0_DDR_ZQ_CTL2	DMC0 DDR Calibration Control Register 2	0x00000000
0x31071068	DMC0_DDR_CA_CTL	DMC0 DDR CA Lane Control Register	0x00000000
0x31071074	DMC0_DDR_SCRATCH_2	DMC0 Scratch Register 2	0x00000000
0x31071078	DMC0_DDR_SCRATCH_3	DMC0 Scratch Register 3	0x00000000
0x3107107C	DMC0_DDR_SCRATCH_4	DMC0 Scratch Register 4	0x00000000
0x31071080	DMC0_DDR_SCRATCH_5	DMC0 Scratch Register 5	0x00000000
0x31071084	DMC0_DDR_SCRATCH_6	DMC0 Scratch Register 6	0x00000000
0x31071088	DMC0_DDR_SCRATCH_7	DMC0 Scratch Register 7	0x00000000

Table A-11: ADSP-SC59x ASRC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9240	ASRC0_CTL01	ASRC0 Control Register for ASRC 0 and 1	0x00000000
0x310C9244	ASRC0_CTL23	ASRC0 Control Register for ASRC 2 and 3	0x00000000
0x310C9248	ASRC0_MUTE	ASRC0 Mute Register	0x00000000
0x310C9260	ASRC0_RAT01	ASRC0 Ratio Register for ASRC 0 and 1	0x80008000
0x310C9264	ASRC0_RAT23	ASRC0 Ratio Register for ASRC 2 and 3	0x80008000

Table A-12: ADSP-SC59x ASRC1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA240	ASRC1_CTL01	ASRC1 Control Register for ASRC 0 and 1	0x00000000
0x310CA244	ASRC1_CTL23	ASRC1 Control Register for ASRC 2 and 3	0x00000000
0x310CA248	ASRC1_MUTE	ASRC1 Mute Register	0x00000000
0x310CA260	ASRC1_RAT01	ASRC1 Ratio Register for ASRC 0 and 1	0x80008000
0x310CA264	ASRC1_RAT23	ASRC1 Ratio Register for ASRC 2 and 3	0x80008000

Table A-13: ADSP-SC59x CANFD0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31046000	CANFD0_CFG	CANFD0 Module Configuration Register	0x5980000F
0x31046004	CANFD0_CTL1	CANFD0 Control 1 Register	0x00000000
0x31046008	CANFD0_TMR	CANFD0 Free Running Timer Register	0x00000000
0x31046010	CANFD0_RX_MB_GMSK	CANFD0 Receive Mailbox Global Mask Register	0x00000000
0x31046014	CANFD0_RX_14_MSK	CANFD0 Receive Mailbox14 Mask Register	0x00000000
0x31046018	CANFD0_RX_15_MSK	CANFD0 Receive Mailbox15 Mask Register	0x00000000
0x3104601C	CANFD0_ECR	CANFD0 Error Counter Register	0x00000000
0x31046020	CANFD0_ESR1	CANFD0 Error and Status 1 Register	0x00000000
0x31046024	CANFD0_IMSK2	CANFD0 Mailbox Interrupt Mask 2 Register	0x00000000
0x31046028	CANFD0_IMSK1	CANFD0 Mailbox Interrupt Mask 1 Register	0x00000000
0x3104602C	CANFD0_IFLG2	CANFD0 Mailbox Interrupt Flag 2 Register	0x00000000
0x31046030	CANFD0_IFLG1	CANFD0 Mailbox Interrupt Flag 1 Register	0x00000000
0x31046034	CANFD0_CTL2	CANFD0 Control 2 Register	0x00800000
0x31046038	CANFD0_ESR2	CANFD0 Error and Status 2 Register	0x00000000
0x31046044	CANFD0_CRC	CANFD0 CRC Register	0x00000000
0x31046048	CANFD0_RX_FIFO_GMSK	CANFD0 Receive FIFO Global Mask Register	0x00000000
0x3104604C	CANFD0_RX_FIFO	CANFD0 Receive FIFO Information Register	0x00000000
0x31046050	CANFD0_TIMING	CANFD0 Can Bit Timing Register	0x00000000
0x31046880	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046884	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046888	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104688C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046890	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046894	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046898	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104689C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468A0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468A4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468A8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468AC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468B0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000

Table A-13: ADSP-SC59x CANFD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310468B4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468B8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468BC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468C0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468C4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468C8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468CC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468D0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468D4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468D8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468DC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468E0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468E4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468E8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468EC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468F0	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468F4	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468F8	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x310468FC	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046900	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046904	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046908	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104690C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046910	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046914	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046918	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104691C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046920	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046924	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046928	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104692C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000

Table A-13: ADSP-SC59x CANFD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31046930	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046934	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046938	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104693C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046940	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046944	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046948	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104694C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046950	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046954	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046958	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104695C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046960	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046964	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046968	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104696C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046970	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046974	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046978	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x3104697C	CANFD0_RX_IMSK[n]	CANFD0 Receive Individual Mask Register	0x00000000
0x31046AE0	CANFD0_MEC	CANFD0 Memory Error Control Register	0x800C0080
0x31046AE4	CANFD0_ERR_IADDR	CANFD0 Error Injection Address Register	0x00000000
0x31046AE8	CANFD0_ERR_IDP	CANFD0 Error Injection Data Pattern Register	0x00000000
0x31046AEC	CANFD0_ERR_IPP	CANFD0 Error Injection Parity Pattern Register	0x00000000
0x31046AF0	CANFD0_ERR_RADDR	CANFD0 Error Report Address Register	0x00000000
0x31046AF4	CANFD0_ERR_RDAT	CANFD0 Error Report Data Register	0x00000000
0x31046AF8	CANFD0_ERR_RSYN	CANFD0 Error Report Syndrome Register	0x00000000
0x31046AFC	CANFD0_ERR_STAT	CANFD0 Error Status Register	0x00000000
0x31046B00	CANFD0_PN_CTL1	CANFD0 Pretended Networking Control1 Register	0x00000100
0x31046B04	CANFD0_PN_CTL2	CANFD0 Pretended Networking Control2 Register	0x00000000
0x31046B08	CANFD0_WUM	CANFD0 Pretended Networking Wakeup Match Register	0x00000000

Table A-13: ADSP-SC59x CANFD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31046B0C	CANFD0_FLTR_ID1	CANFD0 Pretended Networking ID Filter1 Register	0x00000000
0x31046B10	CANFD0_FLTR_DLC	CANFD0 Pretended Networking DLC Filter Register	0x00000008
0x31046B14	CANFD0_FLTR_DATA1_LO	CANFD0 Pretended Networking Payload Low Filter1 Register	0x00000000
0x31046B18	CANFD0_FLTR_DATA1_HI	CANFD0 Pretended Networking Payload Low Filter2 Register	0x00000000
0x31046B1C	CANFD0_FLTR_ID2_IDMSK	CANFD0 Pretended Networking ID Filter2 / IDMask Register	0x00000000
0x31046B20	CANFD0_FLTR_DATA2_DMSK_LO	CANFD0 Pretended Networking Payload Low Filter2 / Payload Low Mask Register	0x00000000
0x31046B24	CANFD0_FLTR_DATA2_DMSK_HI	CANFD0 Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register	0x00000000
0x31046B40	CANFD0_WMB[n]_STAT	CANFD0 Wakeup Message Buffer Control/Status Register	0x00000000
0x31046B44	CANFD0_WMB[n]_ID	CANFD0 Wakeup Message ID Buffer Register	0x00000000
0x31046B48	CANFD0_WMB[n]_DATA_LO	CANFD0 Wakeup Message Buffer Data 0-3 Register	0x00000000
0x31046B4C	CANFD0_WMB[n]_DATA_HI	CANFD0 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31046B50	CANFD0_WMB[n]_STAT	CANFD0 Wakeup Message Buffer Control/Status Register	0x00000000
0x31046B54	CANFD0_WMB[n]_ID	CANFD0 Wakeup Message ID Buffer Register	0x00000000
0x31046B58	CANFD0_WMB[n]_DATA_LO	CANFD0 Wakeup Message Buffer Data 0-3 Register	0x00000000
0x31046B5C	CANFD0_WMB[n]_DATA_HI	CANFD0 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31046B60	CANFD0_WMB[n]_STAT	CANFD0 Wakeup Message Buffer Control/Status Register	0x00000000
0x31046B64	CANFD0_WMB[n]_ID	CANFD0 Wakeup Message ID Buffer Register	0x00000000
0x31046B68	CANFD0_WMB[n]_DATA_LO	CANFD0 Wakeup Message Buffer Data 0-3 Register	0x00000000
0x31046B6C	CANFD0_WMB[n]_DATA_HI	CANFD0 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31046B70	CANFD0_WMB[n]_STAT	CANFD0 Wakeup Message Buffer Control/Status Register	0x00000000
0x31046B74	CANFD0_WMB[n]_ID	CANFD0 Wakeup Message ID Buffer Register	0x00000000
0x31046B78	CANFD0_WMB[n]_DATA_LO	CANFD0 Wakeup Message Buffer Data 0-3 Register	0x00000000

Table A-13: ADSP-SC59x CANFD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31046B7C	CANFD0_WMB[n]_DATA_HI	CANFD0 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31046C00	CANFD0_FD_CTL	CANFD0 CANFD Control Register	0x80000100
0x31046C04	CANFD0_FD_TIMING	CANFD0 CANFD Bit Timing Register	0x00000000
0x31046C08	CANFD0_FD_CRC	CANFD0 CANFD CRC Register	0x00000000

Table A-14: ADSP-SC59x CANFD1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31047000	CANFD1_CFG	CANFD1 Module Configuration Register	0x5980000F
0x31047004	CANFD1_CTL1	CANFD1 Control 1 Register	0x00000000
0x31047008	CANFD1_TMR	CANFD1 Free Running Timer Register	0x00000000
0x31047010	CANFD1_RX_MB_GMSK	CANFD1 Receive Mailbox Global Mask Register	0x00000000
0x31047014	CANFD1_RX_14_MSK	CANFD1 Receive Mailbox14 Mask Register	0x00000000
0x31047018	CANFD1_RX_15_MSK	CANFD1 Receive Mailbox15 Mask Register	0x00000000
0x3104701C	CANFD1_ECR	CANFD1 Error Counter Register	0x00000000
0x31047020	CANFD1_ESR1	CANFD1 Error and Status 1 Register	0x00000000
0x31047024	CANFD1_IMSK2	CANFD1 Mailbox Interrupt Mask 2 Register	0x00000000
0x31047028	CANFD1_IMSK1	CANFD1 Mailbox Interrupt Mask 1 Register	0x00000000
0x3104702C	CANFD1_IFLG2	CANFD1 Mailbox Interrupt Flag 2 Register	0x00000000
0x31047030	CANFD1_IFLG1	CANFD1 Mailbox Interrupt Flag 1 Register	0x00000000
0x31047034	CANFD1_CTL2	CANFD1 Control 2 Register	0x00800000
0x31047038	CANFD1_ESR2	CANFD1 Error and Status 2 Register	0x00000000
0x31047044	CANFD1_CRC	CANFD1 CRC Register	0x00000000
0x31047048	CANFD1_RX_FIFO_GMSK	CANFD1 Receive FIFO Global Mask Register	0x00000000
0x3104704C	CANFD1_RX_FIFO	CANFD1 Receive FIFO Information Register	0x00000000
0x31047050	CANFD1_TIMING	CANFD1 Can Bit Timing Register	0x00000000
0x31047880	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047884	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047888	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104788C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047890	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000

Table A-14: ADSP-SC59x CANFD1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31047894	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047898	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104789C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478A0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478A4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478A8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478AC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478B0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478B4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478B8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478BC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478C0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478C4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478C8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478CC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478D0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478D4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478D8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478DC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478E0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478E4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478E8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478EC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478F0	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478F4	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478F8	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x310478FC	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047900	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047904	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047908	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104790C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000

Table A-14: ADSP-SC59x CANFD1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31047910	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047914	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047918	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104791C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047920	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047924	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047928	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104792C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047930	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047934	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047938	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104793C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047940	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047944	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047948	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104794C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047950	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047954	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047958	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104795C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047960	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047964	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047968	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104796C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047970	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047974	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047978	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x3104797C	CANFD1_RX_IMSK[n]	CANFD1 Receive Individual Mask Register	0x00000000
0x31047AE0	CANFD1_MEC	CANFD1 Memory Error Control Register	0x800C0080
0x31047AE4	CANFD1_ERR_IADDR	CANFD1 Error Injection Address Register	0x00000000
0x31047AE8	CANFD1_ERR_IDP	CANFD1 Error Injection Data Pattern Register	0x00000000

Table A-14: ADSP-SC59x CANFD1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31047AEC	CANFD1_ERR_IPP	CANFD1 Error Injection Parity Pattern Register	0x00000000
0x31047AF0	CANFD1_ERR_RADDR	CANFD1 Error Report Address Register	0x00000000
0x31047AF4	CANFD1_ERR_RDAT	CANFD1 Error Report Data Register	0x00000000
0x31047AF8	CANFD1_ERR_RSYN	CANFD1 Error Report Syndrome Register	0x00000000
0x31047AFC	CANFD1_ERR_STAT	CANFD1 Error Status Register	0x00000000
0x31047B00	CANFD1_PN_CTL1	CANFD1 Pretended Networking Control1 Register	0x00000100
0x31047B04	CANFD1_PN_CTL2	CANFD1 Pretended Networking Control2 Register	0x00000000
0x31047B08	CANFD1_WUM	CANFD1 Pretended Networking Wakeup Match Register	0x00000000
0x31047B0C	CANFD1_FLTR_ID1	CANFD1 Pretended Networking ID Filter1 Register	0x00000000
0x31047B10	CANFD1_FLTR_DLC	CANFD1 Pretended Networking DLC Filter Register	0x00000008
0x31047B14	CANFD1_FLTR_DATA1_LO	CANFD1 Pretended Networking Payload Low Filter1 Register	0x00000000
0x31047B18	CANFD1_FLTR_DATA1_HI	CANFD1 Pretended Networking Payload Low Filter2 Register	0x00000000
0x31047B1C	CANFD1_FLTR_ID2_IDMSK	CANFD1 Pretended Networking ID Filter2 / IDMask Register	0x00000000
0x31047B20	CANFD1_FLTR_DATA2_DMSK_LO	CANFD1 Pretended Networking Payload Low Filter2 / Payload Low Mask Register	0x00000000
0x31047B24	CANFD1_FLTR_DATA2_DMSK_HI	CANFD1 Pretended Networking Payload High Filter2 High Order Bits / Payload High Mask Register	0x00000000
0x31047B40	CANFD1_WMB[n]_STAT	CANFD1 Wakeup Message Buffer Control/Status Register	0x00000000
0x31047B44	CANFD1_WMB[n]_ID	CANFD1 Wakeup Message ID Buffer Register	0x00000000
0x31047B48	CANFD1_WMB[n]_DATA_LO	CANFD1 Wakeup Message Buffer Data 0-3 Register	0x00000000
0x31047B4C	CANFD1_WMB[n]_DATA_HI	CANFD1 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31047B50	CANFD1_WMB[n]_STAT	CANFD1 Wakeup Message Buffer Control/Status Register	0x00000000
0x31047B54	CANFD1_WMB[n]_ID	CANFD1 Wakeup Message ID Buffer Register	0x00000000
0x31047B58	CANFD1_WMB[n]_DATA_LO	CANFD1 Wakeup Message Buffer Data 0-3 Register	0x00000000
0x31047B5C	CANFD1_WMB[n]_DATA_HI	CANFD1 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31047B60	CANFD1_WMB[n]_STAT	CANFD1 Wakeup Message Buffer Control/Status Register	0x00000000
0x31047B64	CANFD1_WMB[n]_ID	CANFD1 Wakeup Message ID Buffer Register	0x00000000

Table A-14: ADSP-SC59x CANFD1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31047B68	CANFD1_WMB[n]_DATA_LO	CANFD1 Wakeup Message Buffer Data 0-3 Register	0x00000000
0x31047B6C	CANFD1_WMB[n]_DATA_HI	CANFD1 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31047B70	CANFD1_WMB[n]_STAT	CANFD1 Wakeup Message Buffer Control/Status Register	0x00000000
0x31047B74	CANFD1_WMB[n]_ID	CANFD1 Wakeup Message ID Buffer Register	0x00000000
0x31047B78	CANFD1_WMB[n]_DATA_LO	CANFD1 Wakeup Message Buffer Data 0-3 Register	0x00000000
0x31047B7C	CANFD1_WMB[n]_DATA_HI	CANFD1 Wakeup Message Buffer Data 4-7 Register	0x00000000
0x31047C00	CANFD1_FD_CTL	CANFD1 CANFD Control Register	0x80000100
0x31047C04	CANFD1_FD_TIMING	CANFD1 CANFD Bit Timing Register	0x00000000
0x31047C08	CANFD1_FD_CRC	CANFD1 CANFD CRC Register	0x00000000

Table A-15: ADSP-SC59x CDU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108F000	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F004	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F008	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F00C	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F010	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F014	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F018	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F01C	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F020	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F024	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F028	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F02C	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F030	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F034	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F038	CDU0_CFG[n]	CDU0 CDU Configuration	0x00000001
0x3108F040	CDU0_STAT	CDU0 CDU Status	0x00000000

Table A-15: ADSP-SC59x CDU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108F044	CDU0_CLKINSEL	CDU0 CLKIN Select	0x00000000
0x3108F048	CDU0_REVID	CDU0 CDU Revision ID	0x00000011

Table A-16: ADSP-SC59x CGU MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108D000	CGU_CTL	CGU Control Register	0x00003C00
0x3108D000	CGU0_CTL	CGU0 Control Register	0x00003C00
0x3108D004	CGU_PLLCTL	CGU PLL Control Register	0x00000000
0x3108D004	CGU0_PLLCTL	CGU0 PLL Control Register	0x00000000
0x3108D008	CGU_STAT	CGU Status Register	0x0000000F
0x3108D008	CGU0_STAT	CGU0 Status Register	0x0000000F
0x3108D00C	CGU_DIV	CGU Clocks Divisor Register	0x0A034482
0x3108D00C	CGU0_DIV	CGU0 Clocks Divisor Register	0x0A034482
0x3108D010	CGU_CLKOUTSEL	CGU CLKOUT Select Register	0x00000000
0x3108D010	CGU0_CLKOUTSEL	CGU0 CLKOUT Select Register	0x00000000
0x3108D014	CGU_OSCWDCTL	CGU Oscillator Watchdog Register	0x00007F00
0x3108D014	CGU0_OSCWDCTL	CGU0 Oscillator Watchdog Register	0x00007F00
0x3108D018	CGU_TSCTL	CGU Time Stamp Control Register	0x00000000
0x3108D018	CGU0_TSCTL	CGU0 Time Stamp Control Register	0x00000000
0x3108D01C	CGU_TSVALUE0	CGU Time Stamp Counter Initial 32 LSB Value Register	0x00000000
0x3108D01C	CGU0_TSVALUE0	CGU0 Time Stamp Counter Initial 32 LSB Value Register	0x00000000
0x3108D020	CGU_TSVALUE1	CGU Time Stamp Counter Initial MSB Value Register	0x00000000
0x3108D020	CGU0_TSVALUE1	CGU0 Time Stamp Counter Initial MSB Value Register	0x00000000
0x3108D024	CGU_TSCOUNT0	CGU Time Stamp Counter 32 LSB Register	0x00000000
0x3108D024	CGU0_TSCOUNT0	CGU0 Time Stamp Counter 32 LSB Register	0x00000000
0x3108D028	CGU_TSCOUNT1	CGU Time Stamp Counter 32 MSB Register	0x00000000
0x3108D028	CGU0_TSCOUNT1	CGU0 Time Stamp Counter 32 MSB Register	0x00000000
0x3108D02C	CGU_CCBF_DIS	CGU Core Clock Buffer Disable Register	0x00000000
0x3108D02C	CGU0_CCBF_DIS	CGU0 Core Clock Buffer Disable Register	0x00000000
0x3108D030	CGU_CCBF_STAT	CGU Core Clock Buffer Status Register	0x00000000

Table A-16: ADSP-SC59x CGU MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108D030	CGU0_CCBF_STAT	CGU0 Core Clock Buffer Status Register	0x00000000
0x3108D038	CGU_SCBF_DIS	CGU System Clock Buffer Disable Register	0x00000000
0x3108D038	CGU0_SCBF_DIS	CGU0 System Clock Buffer Disable Register	0x00000000
0x3108D03C	CGU_SCBF_STAT	CGU System Clock Buffer Status Register	0x00000000
0x3108D03C	CGU0_SCBF_STAT	CGU0 System Clock Buffer Status Register	0x00000000
0x3108D040	CGU_DIVEX	CGU DIV Register Extension	0x00200030
0x3108D040	CGU0_DIVEX	CGU0 DIV Register Extension	0x00200030
0x3108D048	CGU_REVID	CGU Revision ID Register	0x00000030
0x3108D048	CGU0_REVID	CGU0 Revision ID Register	0x00000030

Table A-17: ADSP-SC59x CGU1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108E000	CGU1_CTL	CGU1 Control Register	0x00003C00
0x3108E004	CGU1_PLLCTL	CGU1 PLL Control Register	0x00000000
0x3108E008	CGU1_STAT	CGU1 Status Register	0x0000000F
0x3108E00C	CGU1_DIV	CGU1 Clocks Divisor Register	0x0A034482
0x3108E014	CGU1_OSCWDCTL	CGU1 Oscillator Watchdog Register	0x00007F00
0x3108E02C	CGU1_CCBF_DIS	CGU1 Core Clock Buffer Disable Register	0x00000000
0x3108E030	CGU1_CCBF_STAT	CGU1 Core Clock Buffer Status Register	0x00000000
0x3108E038	CGU1_SCBF_DIS	CGU1 System Clock Buffer Disable Register	0x00000000
0x3108E03C	CGU1_SCBF_STAT	CGU1 System Clock Buffer Status Register	0x00000000
0x3108E040	CGU1_DIVEX	CGU1 DIV Register Extension	0x00200030
0x3108E048	CGU1_REVID	CGU1 Revision ID Register	0x00000030

Table A-18: ADSP-SC59x CNT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3100B000	CNT0_CFG	CNT0 Configuration Register	0x00000000
0x3100B004	CNT0_IMSK	CNT0 Interrupt Mask Register	0x00000000
0x3100B008	CNT0_STAT	CNT0 Status Register	0x00000000
0x3100B00C	CNT0_CMD	CNT0 Command Register	0x00000000

Table A-18: ADSP-SC59x CNT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3100B010	CNT0_DEBNCE	CNT0 Debounce Register	0x00000000
0x3100B014	CNT0_CNTR	CNT0 Counter Register	0x00000000
0x3100B018	CNT0_MAX	CNT0 Maximum Count Register	0x00000000
0x3100B01C	CNT0_MIN	CNT0 Minimum Count Register	0x00000000

Table A-19: ADSP-SC59x CRC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A5000	CRC0_CTL	CRC0 Control Register	0x00000000
0x310A5004	CRC0_DCNT	CRC0 Data Word Count Register	0x00000000
0x310A5008	CRC0_DCNTRLD	CRC0 Data Word Count Reload Register	0x00000000
0x310A5014	CRC0_COMP	CRC0 Data Compare Register	0x00000000
0x310A5018	CRC0_FILLVAL	CRC0 Fill Value Register	0x00000000
0x310A501C	CRC0_DFIFO	CRC0 Data FIFO Register	0x00000000
0x310A5020	CRC0_INEN	CRC0 Interrupt Enable Register	0x00000000
0x310A5024	CRC0_INEN_SET	CRC0 Interrupt Enable Set Register	0x00000000
0x310A5028	CRC0_INEN_CLR	CRC0 Interrupt Enable Clear Register	0x00000000
0x310A502C	CRC0_POLY	CRC0 Polynomial Register	0x00000000
0x310A5040	CRC0_STAT	CRC0 Status Register	0x00000000
0x310A5044	CRC0_DCNTCAP	CRC0 Data Count Capture Register	0x00000000
0x310A504C	CRC0_RESULT_FIN	CRC0 CRC Final Result Register	0x00000000
0x310A5050	CRC0_RESULT_CUR	CRC0 CRC Current Result Register	0x00000000

Table A-20: ADSP-SC59x CRC1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A6000	CRC1_CTL	CRC1 Control Register	0x00000000
0x310A6004	CRC1_DCNT	CRC1 Data Word Count Register	0x00000000
0x310A6008	CRC1_DCNTRLD	CRC1 Data Word Count Reload Register	0x00000000
0x310A6014	CRC1_COMP	CRC1 Data Compare Register	0x00000000
0x310A6018	CRC1_FILLVAL	CRC1 Fill Value Register	0x00000000
0x310A601C	CRC1_DFIFO	CRC1 Data FIFO Register	0x00000000

Table A-20: ADSP-SC59x CRC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A6020	CRC1_INEN	CRC1 Interrupt Enable Register	0x00000000
0x310A6024	CRC1_INEN_SET	CRC1 Interrupt Enable Set Register	0x00000000
0x310A6028	CRC1_INEN_CLR	CRC1 Interrupt Enable Clear Register	0x00000000
0x310A602C	CRC1_POLY	CRC1 Polynomial Register	0x00000000
0x310A6040	CRC1_STAT	CRC1 Status Register	0x00000000
0x310A6044	CRC1_DCNTCAP	CRC1 Data Count Capture Register	0x00000000
0x310A604C	CRC1_RESULT_FIN	CRC1 CRC Final Result Register	0x00000000
0x310A6050	CRC1_RESULT_CUR	CRC1 CRC Current Result Register	0x00000000

Table A-21: ADSP-SC59x CRC2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AA000	CRC2_CTL	CRC2 Control Register	0x00000000
0x310AA004	CRC2_DCNT	CRC2 Data Word Count Register	0x00000000
0x310AA008	CRC2_DCNTRLD	CRC2 Data Word Count Reload Register	0x00000000
0x310AA014	CRC2_COMP	CRC2 Data Compare Register	0x00000000
0x310AA018	CRC2_FILLVAL	CRC2 Fill Value Register	0x00000000
0x310AA01C	CRC2_DFIFO	CRC2 Data FIFO Register	0x00000000
0x310AA020	CRC2_INEN	CRC2 Interrupt Enable Register	0x00000000
0x310AA024	CRC2_INEN_SET	CRC2 Interrupt Enable Set Register	0x00000000
0x310AA028	CRC2_INEN_CLR	CRC2 Interrupt Enable Clear Register	0x00000000
0x310AA02C	CRC2_POLY	CRC2 Polynomial Register	0x00000000
0x310AA040	CRC2_STAT	CRC2 Status Register	0x00000000
0x310AA044	CRC2_DCNTCAP	CRC2 Data Count Capture Register	0x00000000
0x310AA04C	CRC2_RESULT_FIN	CRC2 CRC Final Result Register	0x00000000
0x310AA050	CRC2_RESULT_CUR	CRC2 CRC Current Result Register	0x00000000

Table A-22: ADSP-SC59x CRC3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AB000	CRC3_CTL	CRC3 Control Register	0x00000000
0x310AB004	CRC3_DCNT	CRC3 Data Word Count Register	0x00000000

Table A-22: ADSP-SC59x CRC3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310AB008	CRC3_DCNTRLD	CRC3 Data Word Count Reload Register	0x00000000
0x310AB014	CRC3_COMP	CRC3 Data Compare Register	0x00000000
0x310AB018	CRC3_FILLVAL	CRC3 Fill Value Register	0x00000000
0x310AB01C	CRC3_DFIFO	CRC3 Data FIFO Register	0x00000000
0x310AB020	CRC3_INEN	CRC3 Interrupt Enable Register	0x00000000
0x310AB024	CRC3_INEN_SET	CRC3 Interrupt Enable Set Register	0x00000000
0x310AB028	CRC3_INEN_CLR	CRC3 Interrupt Enable Clear Register	0x00000000
0x310AB02C	CRC3_POLY	CRC3 Polynomial Register	0x00000000
0x310AB040	CRC3_STAT	CRC3 Status Register	0x00000000
0x310AB044	CRC3_DCNTCAP	CRC3 Data Count Capture Register	0x00000000
0x310AB04C	CRC3_RESULT_FIN	CRC3 CRC Final Result Register	0x00000000
0x310AB050	CRC3_RESULT_CUR	CRC3 CRC Current Result Register	0x00000000

Table A-23: ADSP-SC59x CSCTI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31102000	CSCTI0_CTICONTROL	CSCTI0 CTI Control Register	0x00000000
0x31102000	CSCTI_CTICONTROL	CSCTI CTI Control Register	0x00000000
0x31102010	CSCTI0_CTIINTACK	CSCTI0 CTI Interrupt Acknowledge Register	0x00000000
0x31102010	CSCTI_CTIINTACK	CSCTI CTI Interrupt Acknowledge Register	0x00000000
0x31102014	CSCTI0_CTIAPPSET	CSCTI0 CTI Application Trigger Set Register	0x00000000
0x31102014	CSCTI_CTIAPPSET	CSCTI CTI Application Trigger Set Register	0x00000000
0x31102018	CSCTI0_CTIAPPCLEAR	CSCTI0 CTI Application Trigger Clear Register	0x00000000
0x31102018	CSCTI_CTIAPPCLEAR	CSCTI CTI Application Trigger Clear Register	0x00000000
0x3110201C	CSCTI0_CTIAPPPULSE	CSCTI0 CTI Application Pulse Register	0x00000000
0x3110201C	CSCTI_CTIAPPPULSE	CSCTI CTI Application Pulse Register	0x00000000
0x31102020	CSCTI0_CTIINEN0	CSCTI0 CTI Trigger 0 to Channel Enable Register	0x00000000
0x31102020	CSCTI_CTIINEN0	CSCTI CTI Trigger 0 to Channel Enable Register	0x00000000
0x31102024	CSCTI0_CTIINEN1	CSCTI0 CTI Trigger 1 to Channel Enable Register	0x00000000
0x31102024	CSCTI_CTIINEN1	CSCTI CTI Trigger 1 to Channel Enable Register	0x00000000
0x31102028	CSCTI0_CTIINEN2	CSCTI0 CTI Trigger 2 to Channel Enable Register	0x00000000

Table A-23: ADSP-SC59x CSCTI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31102028	CSCTI_CTIINEN2	CSCTI CTI Trigger 2 to Channel Enable Register	0x00000000
0x3110202C	CSCTI0_CTIINEN3	CSCTI0 CTI Trigger 3 to Channel Enable Register	0x00000000
0x3110202C	CSCTI_CTIINEN3	CSCTI CTI Trigger 3 to Channel Enable Register	0x00000000
0x31102030	CSCTI0_CTIINEN4	CSCTI0 CTI Trigger 4 to Channel Enable Register	0x00000000
0x31102030	CSCTI_CTIINEN4	CSCTI CTI Trigger 4 to Channel Enable Register	0x00000000
0x31102034	CSCTI0_CTIINEN5	CSCTI0 CTI Trigger 5 to Channel Enable Register	0x00000000
0x31102034	CSCTI_CTIINEN5	CSCTI CTI Trigger 5 to Channel Enable Register	0x00000000
0x31102038	CSCTI0_CTIINEN6	CSCTI0 CTI Trigger 6 to Channel Enable Register	0x00000000
0x31102038	CSCTI_CTIINEN6	CSCTI CTI Trigger 6 to Channel Enable Register	0x00000000
0x3110203C	CSCTI0_CTIINEN7	CSCTI0 CTI Trigger 7 to Channel Enable Register	0x00000000
0x3110203C	CSCTI_CTIINEN7	CSCTI CTI Trigger 7 to Channel Enable Register	0x00000000
0x311020A0	CSCTI0_CTIOUTEN0	CSCTI0 CTI Channel to Trigger 0 Enable Register	0x00000000
0x311020A0	CSCTI_CTIOUTEN0	CSCTI CTI Channel to Trigger 0 Enable Register	0x00000000
0x311020A4	CSCTI0_CTIOUTEN1	CSCTI0 CTI Channel to Trigger 1 Enable Register	0x00000000
0x311020A4	CSCTI_CTIOUTEN1	CSCTI CTI Channel to Trigger 1 Enable Register	0x00000000
0x311020A8	CSCTI0_CTIOUTEN2	CSCTI0 CTI Channel to Trigger 2 Enable Register	0x00000000
0x311020A8	CSCTI_CTIOUTEN2	CSCTI CTI Channel to Trigger 2 Enable Register	0x00000000
0x311020AC	CSCTI0_CTIOUTEN3	CSCTI0 CTI Channel to Trigger 3 Enable Register	0x00000000
0x311020AC	CSCTI_CTIOUTEN3	CSCTI CTI Channel to Trigger 3 Enable Register	0x00000000
0x311020B0	CSCTI0_CTIOUTEN4	CSCTI0 CTI Channel to Trigger 4 Enable Register	0x00000000
0x311020B0	CSCTI_CTIOUTEN4	CSCTI CTI Channel to Trigger 4 Enable Register	0x00000000
0x311020B4	CSCTI0_CTIOUTEN5	CSCTI0 CTI Channel to Trigger 5 Enable Register	0x00000000
0x311020B4	CSCTI_CTIOUTEN5	CSCTI CTI Channel to Trigger 5 Enable Register	0x00000000
0x311020B8	CSCTI0_CTIOUTEN6	CSCTI0 CTI Channel to Trigger 6 Enable Register	0x00000000
0x311020B8	CSCTI_CTIOUTEN6	CSCTI CTI Channel to Trigger 6 Enable Register	0x00000000
0x311020BC	CSCTI0_CTIOUTEN7	CSCTI0 CTI Channel to Trigger 7 Enable Register	0x00000000
0x311020BC	CSCTI_CTIOUTEN7	CSCTI CTI Channel to Trigger 7 Enable Register	0x00000000
0x31102130	CSCTI0_CTITRIGINSTATUS	CSCTI0 CTI Trigger In Status Register	0x00000000
0x31102130	CSCTI_CTITRIGINSTATUS	CSCTI CTI Trigger In Status Register	0x00000000

Table A-23: ADSP-SC59x CSCTI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31102134	CSCTI0_CTITRIGOUTSTATUS	CSCTI0 CTI Trigger Out Status Register	0x00000000
0x31102134	CSCTI_CTITRIGOUTSTATUS	CSCTI CTI Trigger Out Status Register	0x00000000
0x31102138	CSCTI0_CTICHINSTATUS	CSCTI0 CTI Channel In Status Register	0x00000000
0x31102138	CSCTI_CTICHINSTATUS	CSCTI CTI Channel In Status Register	0x00000000
0x3110213C	CSCTI0_CTICHOUTSTATUS	CSCTI0 CTI Channel Out Status Register	0x00000000
0x3110213C	CSCTI_CTICHOUTSTATUS	CSCTI CTI Channel Out Status Register	0x00000000
0x31102140	CSCTI0_CTIGATE	CSCTI0 Enable CTI Channel Gate Register	0x0000000F
0x31102140	CSCTI_CTIGATE	CSCTI Enable CTI Channel Gate Register	0x0000000F
0x31102144	CSCTI0_ASICCTL	CSCTI0 External Multiplexor Control Register	0x00000000
0x31102144	CSCTI_ASICCTL	CSCTI External Multiplexor Control Register	0x00000000
0x31102EDC	CSCTI0_ITCHINACK	CSCTI0 ITCHINACK	0x00000000
0x31102EDC	CSCTI_ITCHINACK	CSCTI ITCHINACK	0x00000000
0x31102EE0	CSCTI0_ITTRIGINACK	CSCTI0 ITTRIGINACK	0x00000000
0x31102EE0	CSCTI_ITTRIGINACK	CSCTI ITTRIGINACK	0x00000000
0x31102EE4	CSCTI0_ITCHOUT	CSCTI0 Integration Test Channel Output	0x00000000
0x31102EE4	CSCTI_ITCHOUT	CSCTI Integration Test Channel Output	0x00000000
0x31102EE8	CSCTI0_ITTRIGOUT	CSCTI0 Integration Test Trigger Output	0x00000000
0x31102EE8	CSCTI_ITTRIGOUT	CSCTI Integration Test Trigger Output	0x00000000
0x31102EEC	CSCTI0_ITCHOUTACK	CSCTI0 ITCHOUTACK	0x00000000
0x31102EEC	CSCTI_ITCHOUTACK	CSCTI ITCHOUTACK	0x00000000
0x31102EF0	CSCTI0_ITTRIGOUTACK	CSCTI0 ITTRIGOUTACK	0x00000000
0x31102EF0	CSCTI_ITTRIGOUTACK	CSCTI ITTRIGOUTACK	0x00000000
0x31102EF4	CSCTI0_ITCHIN	CSCTI0 Integration Test Channel Input	0x00000000
0x31102EF4	CSCTI_ITCHIN	CSCTI Integration Test Channel Input	0x00000000
0x31102EF8	CSCTI0_ITTRIGIN	CSCTI0 Integration Test Trigger Input	0x00000000
0x31102EF8	CSCTI_ITTRIGIN	CSCTI Integration Test Trigger Input	0x00000000
0x31102F00	CSCTI0_ITCTRL	CSCTI0 Integration Mode Control Register	0x00000000
0x31102F00	CSCTI_ITCTRL	CSCTI Integration Mode Control Register	0x00000000

Table A-23: ADSP-SC59x CSCTI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31102FA0	CSCTI0_CLAIMSET	CSCTI0 Claim Tag Set Register	0x0000000F
0x31102FA0	CSCTI_CLAIMSET	CSCTI Claim Tag Set Register	0x0000000F
0x31102FA4	CSCTI0_CLAIMCLR	CSCTI0 Claim Tag Clear Register	0x00000000
0x31102FA4	CSCTI_CLAIMCLR	CSCTI Claim Tag Clear Register	0x00000000
0x31102FB0	CSCTI0_LAR	CSCTI0 Lock Access Register	0x00000000
0x31102FB0	CSCTI_LAR	CSCTI Lock Access Register	0x00000000
0x31102FB4	CSCTI0_LSR	CSCTI0 Lock Status Register	0x00000003
0x31102FB4	CSCTI_LSR	CSCTI Lock Status Register	0x00000003
0x31102FB8	CSCTI0_AUTHSTATUS	CSCTI0 Authentication Status	0x00000005
0x31102FB8	CSCTI_AUTHSTATUS	CSCTI Authentication Status	0x00000005
0x31102FC8	CSCTI0_DEVID	CSCTI0 Device ID	0x00040800
0x31102FC8	CSCTI_DEVID	CSCTI Device ID	0x00040800
0x31102FCC	CSCTI0_DEVTYPE	CSCTI0 Device Type	0x00000014
0x31102FCC	CSCTI_DEVTYPE	CSCTI Device Type	0x00000014
0x31102FD0	CSCTI0_PERIPHID4	CSCTI0 Peripheral ID4	0x00000004
0x31102FD0	CSCTI_PERIPHID4	CSCTI Peripheral ID4	0x00000004
0x31102FD4	CSCTI0_PERIPHID5	CSCTI0 Peripheral ID5	0x00000000
0x31102FD4	CSCTI_PERIPHID5	CSCTI Peripheral ID5	0x00000000
0x31102FD8	CSCTI0_PERIPHID6	CSCTI0 Peripheral ID6	0x00000000
0x31102FD8	CSCTI_PERIPHID6	CSCTI Peripheral ID6	0x00000000
0x31102FDC	CSCTI0_PERIPHID7	CSCTI0 Peripheral ID7	0x00000000
0x31102FDC	CSCTI_PERIPHID7	CSCTI Peripheral ID7	0x00000000
0x31102FE0	CSCTI0_PERIPHID0	CSCTI0 Peripheral ID0	0x00000006
0x31102FE0	CSCTI_PERIPHID0	CSCTI Peripheral ID0	0x00000006
0x31102FE4	CSCTI0_PERIPHID1	CSCTI0 Peripheral ID1	0x000000B9
0x31102FE4	CSCTI_PERIPHID1	CSCTI Peripheral ID1	0x000000B9
0x31102FE8	CSCTI0_PERIPHID2	CSCTI0 Peripheral ID2	0x0000003B
0x31102FE8	CSCTI_PERIPHID2	CSCTI Peripheral ID2	0x0000003B
0x31102FEC	CSCTI0_PERIPHID3	CSCTI0 Peripheral ID3	0x00000000
0x31102FEC	CSCTI_PERIPHID3	CSCTI Peripheral ID3	0x00000000
0x31102FF0	CSCTI0_COMPID0	CSCTI0 Component ID0	0x0000000D

Table A-23: ADSP-SC59x CSCTI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31102FF0	CSCTI_COMPID0	CSCTI Component ID0	0x0000000D
0x31102FF4	CSCTI0_COMPID1	CSCTI0 Component ID1	0x00000090
0x31102FF4	CSCTI_COMPID1	CSCTI Component ID1	0x00000090
0x31102FF8	CSCTI0_COMPID2	CSCTI0 Component ID2	0x00000005
0x31102FF8	CSCTI_COMPID2	CSCTI Component ID2	0x00000005
0x31102FFC	CSCTI0_COMPID3	CSCTI0 Component ID3	0x000000B1
0x31102FFC	CSCTI_COMPID3	CSCTI Component ID3	0x000000B1

Table A-24: ADSP-SC59x CSCTI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31106000	CSCTI1_CTICONTROL	CSCTI1 CTI Control Register	0x00000000
0x31106010	CSCTI1_CTIINTACK	CSCTI1 CTI Interrupt Acknowledge Register	0x00000000
0x31106014	CSCTI1_CTIAPPSET	CSCTI1 CTI Application Trigger Set Register	0x00000000
0x31106018	CSCTI1_CTIAPPCLEAR	CSCTI1 CTI Application Trigger Clear Register	0x00000000
0x3110601C	CSCTI1_CTIAPPPULSE	CSCTI1 CTI Application Pulse Register	0x00000000
0x31106020	CSCTI1_CTIINEN0	CSCTI1 CTI Trigger 0 to Channel Enable Register	0x00000000
0x31106024	CSCTI1_CTIINEN1	CSCTI1 CTI Trigger 1 to Channel Enable Register	0x00000000
0x31106028	CSCTI1_CTIINEN2	CSCTI1 CTI Trigger 2 to Channel Enable Register	0x00000000
0x3110602C	CSCTI1_CTIINEN3	CSCTI1 CTI Trigger 3 to Channel Enable Register	0x00000000
0x31106030	CSCTI1_CTIINEN4	CSCTI1 CTI Trigger 4 to Channel Enable Register	0x00000000
0x31106034	CSCTI1_CTIINEN5	CSCTI1 CTI Trigger 5 to Channel Enable Register	0x00000000
0x31106038	CSCTI1_CTIINEN6	CSCTI1 CTI Trigger 6 to Channel Enable Register	0x00000000
0x3110603C	CSCTI1_CTIINEN7	CSCTI1 CTI Trigger 7 to Channel Enable Register	0x00000000
0x311060A0	CSCTI1_CTIOUTEN0	CSCTI1 CTI Channel to Trigger 0 Enable Register	0x00000000
0x311060A4	CSCTI1_CTIOUTEN1	CSCTI1 CTI Channel to Trigger 1 Enable Register	0x00000000
0x311060A8	CSCTI1_CTIOUTEN2	CSCTI1 CTI Channel to Trigger 2 Enable Register	0x00000000
0x311060AC	CSCTI1_CTIOUTEN3	CSCTI1 CTI Channel to Trigger 3 Enable Register	0x00000000
0x311060B0	CSCTI1_CTIOUTEN4	CSCTI1 CTI Channel to Trigger 4 Enable Register	0x00000000
0x311060B4	CSCTI1_CTIOUTEN5	CSCTI1 CTI Channel to Trigger 5 Enable Register	0x00000000
0x311060B8	CSCTI1_CTIOUTEN6	CSCTI1 CTI Channel to Trigger 6 Enable Register	0x00000000

Table A-24: ADSP-SC59x CSCTI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311060BC	CSCTI1_CTIOUTEN7	CSCTI1 CTI Channel to Trigger 7 Enable Register	0x00000000
0x31106130	CSCTI1_CTITRIGINSTATUS	CSCTI1 CTI Trigger In Status Register	0x00000000
0x31106134	CSCTI1_CTITRIGOUTSTATUS	CSCTI1 CTI Trigger Out Status Register	0x00000000
0x31106138	CSCTI1_CTICHINSTATUS	CSCTI1 CTI Channel In Status Register	0x00000000
0x3110613C	CSCTI1_CTICHOUTSTATUS	CSCTI1 CTI Channel Out Status Register	0x00000000
0x31106140	CSCTI1_CTIGATE	CSCTI1 Enable CTI Channel Gate Register	0x0000000F
0x31106144	CSCTI1_ASICCTL	CSCTI1 External Multiplexor Control Register	0x00000000
0x31106EDC	CSCTI1_ITCHINACK	CSCTI1 ITCHINACK	0x00000000
0x31106EE0	CSCTI1_ITTRIGINACK	CSCTI1 ITTRIGINACK	0x00000000
0x31106EE4	CSCTI1_ITCHOUT	CSCTI1 Integration Test Channel Output	0x00000000
0x31106EE8	CSCTI1_ITTRIGOUT	CSCTI1 Integration Test Trigger Output	0x00000000
0x31106EEC	CSCTI1_ITCHOUTACK	CSCTI1 ITCHOUTACK	0x00000000
0x31106EF0	CSCTI1_ITTRIGOUTACK	CSCTI1 ITTRIGOUTACK	0x00000000
0x31106EF4	CSCTI1_ITCHIN	CSCTI1 Integration Test Channel Input	0x00000000
0x31106EF8	CSCTI1_ITTRIGIN	CSCTI1 Integration Test Trigger Input	0x00000000
0x31106F00	CSCTI1_ITCTRL	CSCTI1 Integration Mode Control Register	0x00000000
0x31106FA0	CSCTI1_CLAIMSET	CSCTI1 Claim Tag Set Register	0x0000000F
0x31106FA4	CSCTI1_CLAIMCLR	CSCTI1 Claim Tag Clear Register	0x00000000
0x31106FB0	CSCTI1_LAR	CSCTI1 Lock Access Register	0x00000000
0x31106FB4	CSCTI1_LSR	CSCTI1 Lock Status Register	0x00000003
0x31106FB8	CSCTI1_AUTHSTATUS	CSCTI1 Authentication Status	0x00000005
0x31106FC8	CSCTI1_DEVID	CSCTI1 Device ID	0x00040800
0x31106FCC	CSCTI1_DEVTYPE	CSCTI1 Device Type	0x00000014
0x31106FD0	CSCTI1_PERIPHID4	CSCTI1 Peripheral ID4	0x00000004
0x31106FD4	CSCTI1_PERIPHID5	CSCTI1 Peripheral ID5	0x00000000
0x31106FD8	CSCTI1_PERIPHID6	CSCTI1 Peripheral ID6	0x00000000
0x31106FDC	CSCTI1_PERIPHID7	CSCTI1 Peripheral ID7	0x00000000
0x31106FE0	CSCTI1_PERIPHID0	CSCTI1 Peripheral ID0	0x00000006
0x31106FE4	CSCTI1_PERIPHID1	CSCTI1 Peripheral ID1	0x000000B9

Table A-24: ADSP-SC59x CSCTI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31106FE8	CSCTI1_PERIPHID2	CSCTI1 Peripheral ID2	0x0000003B
0x31106FEC	CSCTI1_PERIPHID3	CSCTI1 Peripheral ID3	0x00000000
0x31106FF0	CSCTI1_COMPID0	CSCTI1 Component ID0	0x0000000D
0x31106FF4	CSCTI1_COMPID1	CSCTI1 Component ID1	0x00000090
0x31106FF8	CSCTI1_COMPID2	CSCTI1 Component ID2	0x00000005
0x31106FFC	CSCTI1_COMPID3	CSCTI1 Component ID3	0x000000B1

Table A-25: ADSP-SC59x CSPFT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31103000	CSPFT0_CTL	CSPFT0 Main Control Register	0x20000400
0x31103004	CSPFT0_HWFEAT	CSPFT0 Hardware Feature Register	0x01484002
0x31103010	CSPFT0_STAT	CSPFT0 Status Register	0x00000002
0x31103040	CSPFT0_ACVR[n]	CSPFT0 Address Comparator Value Register	0x00000000
0x31103044	CSPFT0_ACVR[n]	CSPFT0 Address Comparator Value Register	0x00000000
0x31103048	CSPFT0_ACVR[n]	CSPFT0 Address Comparator Value Register	0x00000000
0x3110304C	CSPFT0_ACVR[n]	CSPFT0 Address Comparator Value Register	0x00000000
0x31103080	CSPFT0_ACTR[n]	CSPFT0 Address Comparator Access Type Register	0x00000000
0x31103084	CSPFT0_ACTR[n]	CSPFT0 Address Comparator Access Type Register	0x00000000
0x31103088	CSPFT0_ACTR[n]	CSPFT0 Address Comparator Access Type Register	0x00000000
0x3110308C	CSPFT0_ACTR[n]	CSPFT0 Address Comparator Access Type Register	0x00000000
0x31103140	CSPFT0_CNTRLDVR[n]	CSPFT0 Counter Reload Value Register	0x00000000
0x31103144	CSPFT0_CNTRLDVR[n]	CSPFT0 Counter Reload Value Register	0x00000000
0x31103150	CSPFT0_CNTENR[n]	CSPFT0 Counter Enable Event Register	0x000077EF
0x31103154	CSPFT0_CNTENR[n]	CSPFT0 Counter Enable Event Register	0x000077EF
0x31103160	CSPFT0_CNTRLDEVR[n]	CSPFT0 Counter Reload Event Register	0x000077EF
0x31103164	CSPFT0_CNTRLDEVR[n]	CSPFT0 Counter Reload Event Register	0x000077EF
0x31103170	CSPFT0_CNTVR[n]	CSPFT0 Counter Value Register	0x00000000
0x31103174	CSPFT0_CNTVR[n]	CSPFT0 Counter Value Register	0x00000000
0x311031A0	CSPFT0_EXTOUTEVR[n]	CSPFT0 External Output Event Register	0x000077EF
0x311031A4	CSPFT0_EXTOUTEVR[n]	CSPFT0 External Output Event Register	0x000077EF

Table A-25: ADSP-SC59x CSPFT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311031A8	CSPFT0_EXTOUTEVR[n]	CSPFT0 External Output Event Register	0x000077EF
0x311031AC	CSPFT0_EXTOUTEVR[n]	CSPFT0 External Output Event Register	0x000077EF
0x311031B0	CSPFT0_CIDCVR[n]	CSPFT0 Context ID Comparator Value	0x00000000
0x311031BC	CSPFT0_CIDCMR	CSPFT0 Context ID Comparator Mask Register	0x00000000
0x311031E0	CSPFT0_SYNCFR	CSPFT0 Synchronization Frequency Register	0x00000000
0x311031E8	CSPFT0_CCER	CSPFT0 Configuration Code Extension Register	0x00800000
0x31103200	CSPFT0_TRACEIDR	CSPFT0 CoreSight Trace ID Register	0x00000000
0x31103FA0	CSPFT0_CLAIMSET	CSPFT0 Claim Tag Set Register	0x0000000F
0x31103FA4	CSPFT0_CLAIMCLR	CSPFT0 Claim Tag Clear Register	0x00000000
0x31103FB0	CSPFT0_LAR	CSPFT0 Lock Access Register	0x00000000
0x31103FB4	CSPFT0_LSR	CSPFT0 Lock Status Register	0x00000003
0x31103FB8	CSPFT0_AUTHSTATUS	CSPFT0 Authentication Status Register	0x00000080
0x31103FCC	CSPFT0_DEVTYPE	CSPFT0 Device Type Identifier Register	0x00000023
0x31103FD0	CSPFT0_PID4	CSPFT0 Peripheral ID4 Register	0x00000000
0x31103FE0	CSPFT0_PID0	CSPFT0 Peripheral ID0 Register	0x00000001
0x31103FE4	CSPFT0_PID1	CSPFT0 Peripheral ID1 Register	0x00000050
0x31103FE8	CSPFT0_PID2	CSPFT0 Peripheral ID2 Register	0x0000000E
0x31103FEC	CSPFT0_PID3	CSPFT0 Peripheral ID3 Register	0x00000000
0x31103FF0	CSPFT0_CID0	CSPFT0 Component ID0 Register	0x0000000D
0x31103FF4	CSPFT0_CID1	CSPFT0 Component ID1 Register	0x00000090
0x31103FF8	CSPFT0_CID2	CSPFT0 Component ID2 Register	0x00000005
0x31103FFC	CSPFT0_CID3	CSPFT0 Component ID3 Register	0x000000B1

Table A-26: ADSP-SC59x CSPFT1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31107000	CSPFT1_CTL	CSPFT1 Main Control Register	0x20000400
0x31107004	CSPFT1_HWFEAT	CSPFT1 Hardware Feature Register	0x01484002
0x31107010	CSPFT1_STAT	CSPFT1 Status Register	0x00000002
0x31107040	CSPFT1_ACVR[n]	CSPFT1 Address Comparator Value Register	0x00000000
0x31107044	CSPFT1_ACVR[n]	CSPFT1 Address Comparator Value Register	0x00000000

Table A-26: ADSP-SC59x CSPFT1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31107048	CSPFT1_ACVR[n]	CSPFT1 Address Comparator Value Register	0x00000000
0x3110704C	CSPFT1_ACVR[n]	CSPFT1 Address Comparator Value Register	0x00000000
0x31107080	CSPFT1_ACTR[n]	CSPFT1 Address Comparator Access Type Register	0x00000000
0x31107084	CSPFT1_ACTR[n]	CSPFT1 Address Comparator Access Type Register	0x00000000
0x31107088	CSPFT1_ACTR[n]	CSPFT1 Address Comparator Access Type Register	0x00000000
0x3110708C	CSPFT1_ACTR[n]	CSPFT1 Address Comparator Access Type Register	0x00000000
0x31107140	CSPFT1_CNTRLDVR[n]	CSPFT1 Counter Reload Value Register	0x00000000
0x31107144	CSPFT1_CNTRLDVR[n]	CSPFT1 Counter Reload Value Register	0x00000000
0x31107150	CSPFT1_CNTENR[n]	CSPFT1 Counter Enable Event Register	0x000077EF
0x31107154	CSPFT1_CNTENR[n]	CSPFT1 Counter Enable Event Register	0x000077EF
0x31107160	CSPFT1_CNTRLDEVR[n]	CSPFT1 Counter Reload Event Register	0x000077EF
0x31107164	CSPFT1_CNTRLDEVR[n]	CSPFT1 Counter Reload Event Register	0x000077EF
0x31107170	CSPFT1_CNTVR[n]	CSPFT1 Counter Value Register	0x00000000
0x31107174	CSPFT1_CNTVR[n]	CSPFT1 Counter Value Register	0x00000000
0x311071A0	CSPFT1_EXTOUTEVR[n]	CSPFT1 External Output Event Register	0x000077EF
0x311071A4	CSPFT1_EXTOUTEVR[n]	CSPFT1 External Output Event Register	0x000077EF
0x311071A8	CSPFT1_EXTOUTEVR[n]	CSPFT1 External Output Event Register	0x000077EF
0x311071AC	CSPFT1_EXTOUTEVR[n]	CSPFT1 External Output Event Register	0x000077EF
0x311071B0	CSPFT1_CIDCVR[n]	CSPFT1 Context ID Comparator Value	0x00000000
0x311071BC	CSPFT1_CIDCMR	CSPFT1 Context ID Comparator Mask Register	0x00000000
0x311071E0	CSPFT1_SYNCFR	CSPFT1 Synchronization Frequency Register	0x00000000
0x311071E8	CSPFT1_CCER	CSPFT1 Configuration Code Extension Register	0x00800000
0x31107200	CSPFT1_TRACEIDR	CSPFT1 CoreSight Trace ID Register	0x00000000
0x31107FA0	CSPFT1_CLAIMSET	CSPFT1 Claim Tag Set Register	0x0000000F
0x31107FA4	CSPFT1_CLAIMCLR	CSPFT1 Claim Tag Clear Register	0x00000000
0x31107FB0	CSPFT1_LAR	CSPFT1 Lock Access Register	0x00000000
0x31107FB4	CSPFT1_LSR	CSPFT1 Lock Status Register	0x00000003
0x31107FB8	CSPFT1_AUTHSTATUS	CSPFT1 Authentication Status Register	0x00000080
0x31107FCC	CSPFT1_DEVTYPE	CSPFT1 Device Type Identifier Register	0x00000023
0x31107FD0	CSPFT1_PID4	CSPFT1 Peripheral ID4 Register	0x00000000
0x31107FE0	CSPFT1_PID0	CSPFT1 Peripheral ID0 Register	0x00000001

Table A-26: ADSP-SC59x CSPFT1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31107FE4	CSPFT1_PID1	CSPFT1 Peripheral ID1 Register	0x00000050
0x31107FE8	CSPFT1_PID2	CSPFT1 Peripheral ID2 Register	0x0000000E
0x31107FEC	CSPFT1_PID3	CSPFT1 Peripheral ID3 Register	0x00000000
0x31107FF0	CSPFT1_CID0	CSPFT1 Component ID0 Register	0x0000000D
0x31107FF4	CSPFT1_CID1	CSPFT1 Component ID1 Register	0x00000090
0x31107FF8	CSPFT1_CID2	CSPFT1 Component ID2 Register	0x00000005
0x31107FFC	CSPFT1_CID3	CSPFT1 Component ID3 Register	0x000000B1

Table A-27: ADSP-SC59x STM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31104C04	STM0_STMDMASTARTR	STM0 DMA Transfer Start Register	0x00000000
0x31104C08	STM0_STMDMASTOPR	STM0 DMA Transfer Stop Register	0x00000000
0x31104C0C	STM0_STMDMASTATR	STM0 DMA Transfer Status Register	0x00000000
0x31104C10	STM0_STMDMACTLR	STM0 DMA Control Register	0x00000000
0x31104CFC	STM0_STMDMAIDR	STM0 DMA features ID Register	0x00000002
0x31104D00	STM0_STMHEER	STM0 Hardware Event Enable Register	0x00000000
0x31104D20	STM0_STMHETER	STM0 Hardware Event Trigger Enable Register	0x00000000
0x31104D60	STM0_STMHEBSR	STM0 Hardware Event Bank Select Register	0x00000000
0x31104D64	STM0_STMHEMCR	STM0 Hardware Event Main Control Register	0x00000000
0x31104D68	STM0_STMHEEXTMUXR	STM0 Hardware Event External Multiplex Control Register	0x00000000
0x31104DF4	STM0_STMHEMASTR	STM0 Hardware Event Master Number Register	0x00000080
0x31104DF8	STM0_STMHEFEAT1R	STM0 Hardware Event Features 1 Register	0x30200035
0x31104DFC	STM0_STMHEIDR	STM0 Hardware Event Features ID Register	0x00000011
0x31104E00	STM0_STMSPER	STM0 Stimulus Port Enable Register	0x00000000
0x31104E20	STM0_STMSPTER	STM0 Stimulus Port Trigger Enable Register	0x00000000
0x31104E60	STM0_STMSPSCR	STM0 Stimulus Port Select Configuration Register	0x00000000
0x31104E64	STM0_STMSPMSCR	STM0 Stimulus Port Master Select Configuration Register	0x00000000
0x31104E68	STM0_STMSPOVERRIDER	STM0 Stimulus Port Override Control Register	0x00000000
0x31104E6C	STM0_STMSPMOVERRIDER	STM0 Stimulus Port Master Override Control Register	0x00000000

Table A-27: ADSP-SC59x STM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31104E70	STM0_STMSPTRIGCSR	STM0 Stimulus Port Trigger Control and Status Register	0x00000000
0x31104E80	STM0_STMTCSR	STM0 Trace Control and Status Register	0x00000004
0x31104E84	STM0_STMTSSTIMR	STM0 Timestamp Stimulus Register	0x00000000
0x31104E8C	STM0_STMTSFREQR	STM0 Timestamp Frequency Register	0x00000000
0x31104E90	STM0_STMSYNCR	STM0 Synchronization Control Register	0x00000000
0x31104E94	STM0_STMAUXCR	STM0 Auxiliary Control Register	0x00000000
0x31104EA0	STM0_STMFEAT1R	STM0 STM Features 1 Register	0x006587D1
0x31104EA4	STM0_STMFEAT2R	STM0 STM Features 2 Register	0x000114F2
0x31104EA8	STM0_STMFEAT3R	STM0 STM Features 3 Register	0x0000007F
0x31104EE8	STM0_STMITTRIGGER	STM0 Integration Test for Cross-Trigger Outputs Register	0x00000000
0x31104EEC	STM0_STMITATBDATA0	STM0 Integration Mode ATB Data 0 Register	0x00000000
0x31104EF0	STM0_STMITATBCTR2	STM0 Integration Mode ATB Control 2 Register	0x00000000
0x31104EF4	STM0_STMITATBID	STM0 Integration Mode ATB Identification Register	0x00000000
0x31104EF8	STM0_STMITATBCTR0	STM0 Integration Mode ATB Control 0 Register	0x00000000
0x31104F00	STM0_STMITCTRL	STM0 Integration Mode Control Register	0x00000000
0x31104FA0	STM0_STMCLAIMSET	STM0 Claim Tag Set Register	0x0000000F
0x31104FA4	STM0_STMCLAIMCLR	STM0 Claim Tag Clear Register	0x00000000
0x31104FB0	STM0_STMLAR	STM0 Lock Access Register	0x00000000
0x31104FB4	STM0_STMLSR	STM0 Lock Status Register	0x00000003
0x31104FB8	STM0_STMAUTHSTATUS	STM0 Authentication Status Register	0x000000AA
0x31104FBC	STM0_STMDEVARCH	STM0 Device Architecture Register	0x8EE20A63
0x31104FC8	STM0_STMDEVID	STM0 Device Configuration Register	0x00010000
0x31104FCC	STM0_STMDEVTYPE	STM0 Device Type Identifier Register	0x00000063
0x31104FD0	STM0_STMPIDR4	STM0 Peripheral ID4 Register	0x00000004
0x31104FE0	STM0_STMPIDR0	STM0 Peripheral ID0 Register	0x00000063
0x31104FE4	STM0_STMPIDR1	STM0 Peripheral ID1 Register	0x000000B9
0x31104FE8	STM0_STMPIDR2	STM0 Peripheral ID2 Register	0x0000001B
0x31104FEC	STM0_STMPIDR3	STM0 Peripheral ID3 Register	0x00000000
0x31104FF0	STM0_STMCIDR0	STM0 Component ID0 Register	0x0000000D
0x31104FF4	STM0_STMCIDR1	STM0 Component ID1 Register	0x00000090
0x31104FF8	STM0_STMCIDR2	STM0 Component ID2 Register	0x00000005

Table A-27: ADSP-SC59x STM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31104FFC	STM0_STMCIDR3	STM0 Component ID3 Register	0x000000B1

Table A-28: ADSP-SC59x DAI MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9000	DAI_EXTD_CLK0	DAI Extended Clock Routing Control Register 0	0x00000000
0x310C9000	DAI0_EXTD_CLK0	DAI0 Extended Clock Routing Control Register 0	0x00000000
0x310C9004	DAI_EXTD_CLK1	DAI Extended Clock Routing Control Register 1	0x00000000
0x310C9004	DAI0_EXTD_CLK1	DAI0 Extended Clock Routing Control Register 1	0x00000000
0x310C9008	DAI_EXTD_CLK2	DAI Extended Clock Routing Control Register 2	0x00000000
0x310C9008	DAI0_EXTD_CLK2	DAI0 Extended Clock Routing Control Register 2	0x00000000
0x310C900C	DAI_EXTD_CLK3	DAI Extended Clock Routing Control Register 3	0x00000000
0x310C900C	DAI0_EXTD_CLK3	DAI0 Extended Clock Routing Control Register 3	0x00000000
0x310C9010	DAI_EXTD_CLK4	DAI Extended Clock Routing Control Register 4	0x00000000
0x310C9010	DAI0_EXTD_CLK4	DAI0 Extended Clock Routing Control Register 4	0x00000000
0x310C9014	DAI_EXTD_CLK5	DAI Extended Clock Routing Control Register 5	0x00000000
0x310C9014	DAI0_EXTD_CLK5	DAI0 Extended Clock Routing Control Register 5	0x00000000
0x310C9018	DAI_EXTD_DAT0	DAI Extended Serial Data Routing Control Register 0	0x00000000
0x310C9018	DAI0_EXTD_DAT0	DAI0 Extended Serial Data Routing Control Register 0	0x00000000
0x310C901C	DAI_EXTD_DAT1	DAI Extended Serial Data Routing Control Register 1	0x00000000
0x310C901C	DAI0_EXTD_DAT1	DAI0 Extended Serial Data Routing Control Register 1	0x00000000
0x310C9020	DAI_EXTD_DAT2	DAI Extended Serial Data Routing Control Register 2	0x00000000
0x310C9020	DAI0_EXTD_DAT2	DAI0 Extended Serial Data Routing Control Register 2	0x00000000
0x310C9024	DAI_EXTD_DAT3	DAI Extended Serial Data Routing Control Register 3	0x00000000
0x310C9024	DAI0_EXTD_DAT3	DAI0 Extended Serial Data Routing Control Register 3	0x00000000
0x310C9028	DAI_EXTD_DAT4	DAI Extended Serial Data Routing Control Register 4	0x00000000
0x310C9028	DAI0_EXTD_DAT4	DAI0 Extended Serial Data Routing Control Register 4	0x00000000
0x310C902C	DAI_EXTD_DAT5	DAI Extended Serial Data Routing Control Register 5	0x00000000
0x310C902C	DAI0_EXTD_DAT5	DAI0 Extended Serial Data Routing Control Register 5	0x00000000
0x310C9030	DAI_EXTD_DAT6	DAI Extended Serial Data Routing Control Register 6	0x00000000
0x310C9030	DAI0_EXTD_DAT6	DAI0 Extended Serial Data Routing Control Register 6	0x00000000

Table A-28: ADSP-SC59x DAI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9034	DAI_EXTD_FS0	DAI Extended Frame Sync Routing Control Register 0	0x00000000
0x310C9034	DAI0_EXTD_FS0	DAI0 Extended Frame Sync Routing Control Register 0	0x00000000
0x310C9038	DAI_EXTD_FS1	DAI Extended Frame Sync Routing Control Register 1	0x00000000
0x310C9038	DAI0_EXTD_FS1	DAI0 Extended Frame Sync Routing Control Register 1	0x00000000
0x310C903C	DAI_EXTD_FS2	DAI Extended Frame Sync Routing Control Register 2	0x00000000
0x310C903C	DAI0_EXTD_FS2	DAI0 Extended Frame Sync Routing Control Register 2	0x00000000
0x310C9044	DAI_EXTD_FS4	DAI Extended Frame Sync Routing Control Register 4	0x00000000
0x310C9044	DAI0_EXTD_FS4	DAI0 Extended Frame Sync Routing Control Register 4	0x00000000
0x310C9048	DAI_EXTD_PIN0	DAI Extended Pin Buffer Assignment Register 0	0x00000000
0x310C9048	DAI0_EXTD_PIN0	DAI0 Extended Pin Buffer Assignment Register 0	0x00000000
0x310C904C	DAI_EXTD_PIN1	DAI Extended Pin Buffer Assignment Register 1	0x00000000
0x310C904C	DAI0_EXTD_PIN1	DAI0 Extended Pin Buffer Assignment Register 1	0x00000000
0x310C9050	DAI_EXTD_PIN2	DAI Extended Pin Buffer Assignment Register 2	0x00000000
0x310C9050	DAI0_EXTD_PIN2	DAI0 Extended Pin Buffer Assignment Register 2	0x00000000
0x310C9054	DAI_EXTD_PIN3	DAI Extended Pin Buffer Assignment Register 3	0x00000000
0x310C9054	DAI0_EXTD_PIN3	DAI0 Extended Pin Buffer Assignment Register 3	0x00000000
0x310C9058	DAI_EXTD_PIN4	DAI Extended Pin Buffer Assignment Register 4	0x00000000
0x310C9058	DAI0_EXTD_PIN4	DAI0 Extended Pin Buffer Assignment Register 4	0x00000000
0x310C905C	DAI_EXTD_MISC0	DAI Extended Miscellaneous Control Register 0	0x00000000
0x310C905C	DAI0_EXTD_MISC0	DAI0 Extended Miscellaneous Control Register 0	0x00000000
0x310C9060	DAI_EXTD_MISC1	DAI Extended Miscellaneous Control Register 1	0x00000000
0x310C9060	DAI0_EXTD_MISC1	DAI0 Extended Miscellaneous Control Register 1	0x00000000
0x310C9064	DAI_EXTD_MISC2	DAI Extended Miscellaneous Control Register 2	0x00000000
0x310C9064	DAI0_EXTD_MISC2	DAI0 Extended Miscellaneous Control Register 2	0x00000000
0x310C9068	DAI_EXTD_PBEN0	DAI Extended Pin Buffer Enable Register 0	0x00000000
0x310C9068	DAI0_EXTD_PBEN0	DAI0 Extended Pin Buffer Enable Register 0	0x00000000
0x310C906C	DAI_EXTD_PBEN1	DAI Extended Pin Buffer Enable Register 1	0x00000000
0x310C906C	DAI0_EXTD_PBEN1	DAI0 Extended Pin Buffer Enable Register 1	0x00000000
0x310C9070	DAI_EXTD_PBEN2	DAI Extended Pin Buffer Enable Register 2	0x00000000
0x310C9070	DAI0_EXTD_PBEN2	DAI0 Extended Pin Buffer Enable Register 2	0x00000000
0x310C9074	DAI_EXTD_PBEN3	DAI Extended Pin Buffer Enable Register 3	0x00000000

Table A-28: ADSP-SC59x DAI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9074	DAI0_EXTD_PBEN3	DAI0 Extended Pin Buffer Enable Register 3	0x00000000
0x310C90C0	DAI_CLK0	DAI Clock Routing Control Register 0	0x252630C2
0x310C90C0	DAI0_CLK0	DAI0 Clock Routing Control Register 0	0x252630C2
0x310C90C4	DAI_CLK1	DAI Clock Routing Control Register 1	0x3DEF7BDE
0x310C90C4	DAI0_CLK1	DAI0 Clock Routing Control Register 1	0x3DEF7BDE
0x310C90C8	DAI_CLK2	DAI Clock Routing Control Register 2	0x01EF7BDE
0x310C90C8	DAI0_CLK2	DAI0 Clock Routing Control Register 2	0x01EF7BDE
0x310C90CC	DAI_CLK3	DAI Clock Routing Control Register 3	0x3C000000
0x310C90CC	DAI0_CLK3	DAI0 Clock Routing Control Register 3	0x3C000000
0x310C90D0	DAI_CLK4	DAI Clock Routing Control Register 4	0x3DEF03DE
0x310C90D0	DAI0_CLK4	DAI0 Clock Routing Control Register 4	0x3DEF03DE
0x310C90D4	DAI_CLK5	DAI Clock Routing Control Register 5	0x3DEF7BDE
0x310C90D4	DAI0_CLK5	DAI0 Clock Routing Control Register 5	0x3DEF7BDE
0x310C9100	DAI_DAT0	DAI Serial Data Routing Control Register 0	0x08144040
0x310C9100	DAI0_DAT0	DAI0 Serial Data Routing Control Register 0	0x08144040
0x310C9104	DAI_DAT1	DAI Serial Data Routing Control Register 1	0x0F38B289
0x310C9104	DAI0_DAT1	DAI0 Serial Data Routing Control Register 1	0x0F38B289
0x310C9108	DAI_DAT2	DAI Serial Data Routing Control Register 2	0x00000450
0x310C9108	DAI0_DAT2	DAI0 Serial Data Routing Control Register 2	0x00000450
0x310C910C	DAI_DAT3	DAI Serial Data Routing Control Register 3	0x00000000
0x310C910C	DAI0_DAT3	DAI0 Serial Data Routing Control Register 3	0x00000000
0x310C9110	DAI_DAT4	DAI Serial Data Routing Control Register 4	0x00000000
0x310C9110	DAI0_DAT4	DAI0 Serial Data Routing Control Register 4	0x00000000
0x310C9114	DAI_DAT5	DAI Serial Data Routing Control Register 5	0x00000000
0x310C9114	DAI0_DAT5	DAI0 Serial Data Routing Control Register 5	0x00000000
0x310C9118	DAI_DAT6	DAI Serial Data Routing Control Register 6	0x00FBEFBE
0x310C9118	DAI0_DAT6	DAI0 Serial Data Routing Control Register 6	0x00FBEFBE
0x310C9140	DAI_FS0	DAI Frame Sync Routing Control Register 0	0x2736B4E3
0x310C9140	DAI0_FS0	DAI0 Frame Sync Routing Control Register 0	0x2736B4E3
0x310C9144	DAI_FS1	DAI Frame Sync Routing Control Register 1	0x3DEF7BDE
0x310C9144	DAI0_FS1	DAI0 Frame Sync Routing Control Register 1	0x3DEF7BDE

Table A-28: ADSP-SC59x DAI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9148	DAI_FS2	DAI Frame Sync Routing Control Register 2	0x00007BDE
0x310C9148	DAI0_FS2	DAI0 Frame Sync Routing Control Register 2	0x00007BDE
0x310C9150	DAI_FS4	DAI Frame Sync Routing Control Register 4	0x000F7BDE
0x310C9150	DAI0_FS4	DAI0 Frame Sync Routing Control Register 4	0x000F7BDE
0x310C9180	DAI_PIN0	DAI Pin Buffer Assignment Register 0	0x04C80A94
0x310C9180	DAI0_PIN0	DAI0 Pin Buffer Assignment Register 0	0x04C80A94
0x310C9184	DAI_PIN1	DAI Pin Buffer Assignment Register 1	0x04E84B96
0x310C9184	DAI0_PIN1	DAI0 Pin Buffer Assignment Register 1	0x04E84B96
0x310C9188	DAI_PIN2	DAI Pin Buffer Assignment Register 2	0x03668C98
0x310C9188	DAI0_PIN2	DAI0 Pin Buffer Assignment Register 2	0x03668C98
0x310C918C	DAI_PIN3	DAI Pin Buffer Assignment Register 3	0x03A714A3
0x310C918C	DAI0_PIN3	DAI0 Pin Buffer Assignment Register 3	0x03A714A3
0x310C9190	DAI_PIN4	DAI Pin Buffer Assignment Register 4	0x05694F9E
0x310C9190	DAI0_PIN4	DAI0 Pin Buffer Assignment Register 4	0x05694F9E
0x310C91C0	DAI_MISC0	DAI Miscellaneous Control Register 0	0x3DEF7BDE
0x310C91C0	DAI0_MISC0	DAI0 Miscellaneous Control Register 0	0x3DEF7BDE
0x310C91C4	DAI_MISC1	DAI Miscellaneous Control Register 1	0x3DEF7BDE
0x310C91C4	DAI0_MISC1	DAI0 Miscellaneous Control Register 1	0x3DEF7BDE
0x310C91C8	DAI_MISC2	DAI Miscellaneous Control Register 1	0x000F7BDE
0x310C91C8	DAI0_MISC2	DAI0 Miscellaneous Control Register 1	0x000F7BDE
0x310C91E0	DAI_PBEN0	DAI Pin Buffer Enable Register 0	0x0E2482CA
0x310C91E0	DAI0_PBEN0	DAI0 Pin Buffer Enable Register 0	0x0E2482CA
0x310C91E4	DAI_PBEN1	DAI Pin Buffer Enable Register 1	0x1348D30F
0x310C91E4	DAI0_PBEN1	DAI0 Pin Buffer Enable Register 1	0x1348D30F
0x310C91E8	DAI_PBEN2	DAI Pin Buffer Enable Register 2	0x1A5545D6
0x310C91E8	DAI0_PBEN2	DAI0 Pin Buffer Enable Register 2	0x1A5545D6
0x310C91EC	DAI_PBEN3	DAI Pin Buffer Enable Register 3	0x1D71F79B
0x310C91EC	DAI0_PBEN3	DAI0 Pin Buffer Enable Register 3	0x1D71F79B
0x310C9200	DAI_IMSK_FE	DAI Falling-Edge Interrupt Mask Register	0x00000000
0x310C9200	DAI0_IMSK_FE	DAI0 Falling-Edge Interrupt Mask Register	0x00000000
0x310C9204	DAI_IMSK_RE	DAI Rising-Edge Interrupt Mask Register	0x00000000

Table A-28: ADSP-SC59x DAI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9204	DAI0_IMSK_RE	DAI0 Rising-Edge Interrupt Mask Register	0x00000000
0x310C9210	DAI_IMSK_PRI	DAI Core Interrupt Priority Assignment Register	0x00000000
0x310C9210	DAI0_IMSK_PRI	DAI0 Core Interrupt Priority Assignment Register	0x00000000
0x310C9220	DAI_IRPTL_H	DAI High Priority Interrupt Latch Register	0x00000000
0x310C9220	DAI0_IRPTL_H	DAI0 High Priority Interrupt Latch Register	0x00000000
0x310C9224	DAI_IRPTL_L	DAI Low Priority Interrupt Latch Register	0x00000000
0x310C9224	DAI0_IRPTL_L	DAI0 Low Priority Interrupt Latch Register	0x00000000
0x310C9230	DAI_IRPTL_HS	DAI Shadow High Priority Interrupt Latch Register	0x00000000
0x310C9230	DAI0_IRPTL_HS	DAI0 Shadow High Priority Interrupt Latch Register	0x00000000
0x310C9234	DAI_IRPTL_LS	DAI Shadow Low Priority Interrupt Latch Register	0x00000000
0x310C9234	DAI0_IRPTL_LS	DAI0 Shadow Low Priority Interrupt Latch Register	0x00000000
0x310C92E4	DAI_PIN_STAT	DAI Pin Status Register	0x00000000
0x310C92E4	DAI0_PIN_STAT	DAI0 Pin Status Register	0x00000000
0x310C92E8	DAI_GBL_SP_EN	DAI Global SPORT Enable Register	0x00000000
0x310C92E8	DAI0_GBL_SP_EN	DAI0 Global SPORT Enable Register	0x00000000
0x310C92EC	DAI_GBL_INT_EN	DAI Global SPORT Interrupt Grouping Register	0x00000000
0x310C92EC	DAI0_GBL_INT_EN	DAI0 Global SPORT Interrupt Grouping Register	0x00000000
0x310C92F0	DAI_GBL_PCG_EN	DAI Global PCG Enable Control Register	0x00000000
0x310C92F0	DAI0_GBL_PCG_EN	DAI0 Global PCG Enable Control Register	0x00000000

Table A-29: ADSP-SC59x DAI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA000	DAI1_EXTD_CLK0	DAI1 Extended Clock Routing Control Register 0	0x00000000
0x310CA004	DAI1_EXTD_CLK1	DAI1 Extended Clock Routing Control Register 1	0x00000000
0x310CA008	DAI1_EXTD_CLK2	DAI1 Extended Clock Routing Control Register 2	0x00000000
0x310CA00C	DAI1_EXTD_CLK3	DAI1 Extended Clock Routing Control Register 3	0x00000000
0x310CA010	DAI1_EXTD_CLK4	DAI1 Extended Clock Routing Control Register 4	0x00000000
0x310CA014	DAI1_EXTD_CLK5	DAI1 Extended Clock Routing Control Register 5	0x00000000
0x310CA018	DAI1_EXTD_DAT0	DAI1 Extended Serial Data Routing Control Register 0	0x00000000
0x310CA01C	DAI1_EXTD_DAT1	DAI1 Extended Serial Data Routing Control Register 1	0x00000000

Table A-29: ADSP-SC59x DAI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA020	DAI1_EXTD_DAT2	DAI1 Extended Serial Data Routing Control Register 2	0x00000000
0x310CA024	DAI1_EXTD_DAT3	DAI1 Extended Serial Data Routing Control Register 3	0x00000000
0x310CA028	DAI1_EXTD_DAT4	DAI1 Extended Serial Data Routing Control Register 4	0x00000000
0x310CA02C	DAI1_EXTD_DAT5	DAI1 Extended Serial Data Routing Control Register 5	0x00000000
0x310CA030	DAI1_EXTD_DAT6	DAI1 Extended Serial Data Routing Control Register 6	0x00000000
0x310CA034	DAI1_EXTD_FS0	DAI1 Extended Frame Sync Routing Control Register 0	0x00000000
0x310CA038	DAI1_EXTD_FS1	DAI1 Extended Frame Sync Routing Control Register 1	0x00000000
0x310CA03C	DAI1_EXTD_FS2	DAI1 Extended Frame Sync Routing Control Register 2	0x00000000
0x310CA044	DAI1_EXTD_FS4	DAI1 Extended Frame Sync Routing Control Register 4	0x00000000
0x310CA048	DAI1_EXTD_PIN0	DAI1 Extended Pin Buffer Assignment Register 0	0x00000000
0x310CA04C	DAI1_EXTD_PIN1	DAI1 Extended Pin Buffer Assignment Register 1	0x00000000
0x310CA050	DAI1_EXTD_PIN2	DAI1 Extended Pin Buffer Assignment Register 2	0x00000000
0x310CA054	DAI1_EXTD_PIN3	DAI1 Extended Pin Buffer Assignment Register 3	0x00000000
0x310CA058	DAI1_EXTD_PIN4	DAI1 Extended Pin Buffer Assignment Register 4	0x00000000
0x310CA05C	DAI1_EXTD_MISC0	DAI1 Extended Miscellaneous Control Register 0	0x00000000
0x310CA060	DAI1_EXTD_MISC1	DAI1 Extended Miscellaneous Control Register 1	0x00000000
0x310CA064	DAI1_EXTD_MISC2	DAI1 Extended Miscellaneous Control Register 2	0x00000000
0x310CA068	DAI1_EXTD_PBEN0	DAI1 Extended Pin Buffer Enable Register 0	0x00000000
0x310CA06C	DAI1_EXTD_PBEN1	DAI1 Extended Pin Buffer Enable Register 1	0x00000000
0x310CA070	DAI1_EXTD_PBEN2	DAI1 Extended Pin Buffer Enable Register 2	0x00000000
0x310CA074	DAI1_EXTD_PBEN3	DAI1 Extended Pin Buffer Enable Register 3	0x00000000
0x310CA0C0	DAI1_CLK0	DAI1 Clock Routing Control Register 0	0x252630C2
0x310CA0C4	DAI1_CLK1	DAI1 Clock Routing Control Register 1	0x3DEF7BDE
0x310CA0C8	DAI1_CLK2	DAI1 Clock Routing Control Register 2	0x01EF7BDE
0x310CA0CC	DAI1_CLK3	DAI1 Clock Routing Control Register 3	0x3C000000
0x310CA0D0	DAI1_CLK4	DAI1 Clock Routing Control Register 4	0x3DEF03DE
0x310CA0D4	DAI1_CLK5	DAI1 Clock Routing Control Register 5	0x3DEF7BDE
0x310CA100	DAI1_DAT0	DAI1 Serial Data Routing Control Register 0	0x08144040
0x310CA104	DAI1_DAT1	DAI1 Serial Data Routing Control Register 1	0x0F38B289
0x310CA108	DAI1_DAT2	DAI1 Serial Data Routing Control Register 2	0x00000450
0x310CA10C	DAI1_DAT3	DAI1 Serial Data Routing Control Register 3	0x00000000

Table A-29: ADSP-SC59x DAI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA110	DAI1_DAT4	DAI1 Serial Data Routing Control Register 4	0x00000000
0x310CA114	DAI1_DAT5	DAI1 Serial Data Routing Control Register 5	0x00000000
0x310CA118	DAI1_DAT6	DAI1 Serial Data Routing Control Register 6	0x00FBEFBE
0x310CA140	DAI1_FS0	DAI1 Frame Sync Routing Control Register 0	0x2736B4E3
0x310CA144	DAI1_FS1	DAI1 Frame Sync Routing Control Register 1	0x3DEF7BDE
0x310CA148	DAI1_FS2	DAI1 Frame Sync Routing Control Register 2	0x00007BDE
0x310CA150	DAI1_FS4	DAI1 Frame Sync Routing Control Register 4	0x000F7BDE
0x310CA180	DAI1_PIN0	DAI1 Pin Buffer Assignment Register 0	0x04C80A94
0x310CA184	DAI1_PIN1	DAI1 Pin Buffer Assignment Register 1	0x04E84B96
0x310CA188	DAI1_PIN2	DAI1 Pin Buffer Assignment Register 2	0x03668C98
0x310CA18C	DAI1_PIN3	DAI1 Pin Buffer Assignment Register 3	0x03A714A3
0x310CA190	DAI1_PIN4	DAI1 Pin Buffer Assignment Register 4	0x05694F9E
0x310CA1C0	DAI1_MISC0	DAI1 Miscellaneous Control Register 0	0x3DEF7BDE
0x310CA1C4	DAI1_MISC1	DAI1 Miscellaneous Control Register 1	0x3DEF7BDE
0x310CA1C8	DAI1_MISC2	DAI1 Miscellaneous Control Register 1	0x000F7BDE
0x310CA1E0	DAI1_PBEN0	DAI1 Pin Buffer Enable Register 0	0x0E2482CA
0x310CA1E4	DAI1_PBEN1	DAI1 Pin Buffer Enable Register 1	0x1348D30F
0x310CA1E8	DAI1_PBEN2	DAI1 Pin Buffer Enable Register 2	0x1A5545D6
0x310CA1EC	DAI1_PBEN3	DAI1 Pin Buffer Enable Register 3	0x1D71F79B
0x310CA200	DAI1_IMSK_FE	DAI1 Falling-Edge Interrupt Mask Register	0x00000000
0x310CA204	DAI1_IMSK_RE	DAI1 Rising-Edge Interrupt Mask Register	0x00000000
0x310CA210	DAI1_IMSK_PRI	DAI1 Core Interrupt Priority Assignment Register	0x00000000
0x310CA220	DAI1_IRPTL_H	DAI1 High Priority Interrupt Latch Register	0x00000000
0x310CA224	DAI1_IRPTL_L	DAI1 Low Priority Interrupt Latch Register	0x00000000
0x310CA230	DAI1_IRPTL_HS	DAI1 Shadow High Priority Interrupt Latch Register	0x00000000
0x310CA234	DAI1_IRPTL_LS	DAI1 Shadow Low Priority Interrupt Latch Register	0x00000000
0x310CA2E4	DAI1_PIN_STAT	DAI1 Pin Status Register	0x00000000
0x310CA2E8	DAI1_GBL_SP_EN	DAI1 Global SPORT Enable Register	0x00000000
0x310CA2EC	DAI1_GBL_INT_EN	DAI1 Global SPORT Interrupt Grouping Register	0x00000000

Table A-30: ADSP-SC59x DDE MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFF080	DDE_DSCPTR_NXT	DDE Pointer to Next Initial Descriptor Register	0x00000000
0x30FFF084	DDE_ADDRSTART	DDE Start Address of Current Buffer Register	0x00000000
0x30FFF088	DDE_CFG	DDE Configuration Register	0x00000220
0x30FFF08C	DDE_XCNT	DDE Inner Loop Count Start Value Register	0x00000000
0x30FFF090	DDE_XMOD	DDE Inner Loop Address Increment Register	0x00000000
0x30FFF094	DDE_YCNT	DDE Outer Loop Count Start Value (2D only) Register	0x00000000
0x30FFF098	DDE_YMOD	DDE Outer Loop Address Increment (2D only) Register	0x00000000
0x30FFF0A4	DDE_DSCPTR_CUR	DDE Current Descriptor Pointer Register	0x00000000
0x30FFF0A8	DDE_DSCPTR_PRV	DDE Previous Initial Descriptor Pointer Register	0x00000000
0x30FFF0AC	DDE_ADDR_CUR	DDE Current Address Register	0x00000000
0x30FFF0B0	DDE_STAT	DDE Status Register	0x0000B000
0x30FFF0B4	DDE_XCNT_CUR	DDE Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x30FFF0B8	DDE_YCNT_CUR	DDE Current Row Count (2D only) Register	0x00000000
0x30FFF0C0	DDE_BWLCNT	DDE Bandwidth Limit Count Register	0x00000000
0x30FFF0C4	DDE_BWLCNT_CUR	DDE Bandwidth Limit Count Current Register	0x00000000
0x30FFF0C8	DDE_BWMCNT	DDE Bandwidth Monitor Count Register	0x00000000
0x30FFF0CC	DDE_BWMCNT_CUR	DDE Bandwidth Monitor Count Current Register	0x00000000
0x31022000	DMA0_DSCPTR_NXT	DMA0 Pointer to Next Initial Descriptor Register	0x00000000
0x31022004	DMA0_ADDRSTART	DMA0 Start Address of Current Buffer Register	0x00000000
0x31022008	DMA0_CFG	DMA0 Configuration Register	0x00000000
0x3102200C	DMA0_XCNT	DMA0 Inner Loop Count Start Value Register	0x00000000
0x31022010	DMA0_XMOD	DMA0 Inner Loop Address Increment Register	0x00000000
0x31022014	DMA0_YCNT	DMA0 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022018	DMA0_YMOD	DMA0 Outer Loop Address Increment (2D only) Register	0x00000000
0x31022024	DMA0_DSCPTR_CUR	DMA0 Current Descriptor Pointer Register	0x00000000
0x31022028	DMA0_DSCPTR_PRV	DMA0 Previous Initial Descriptor Pointer Register	0x00000000
0x3102202C	DMA0_ADDR_CUR	DMA0 Current Address Register	0x00000000
0x31022030	DMA0_STAT	DMA0 Status Register	0x00006000
0x31022034	DMA0_XCNT_CUR	DMA0 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000

Table A-30: ADSP-SC59x DDE MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31022038	DMA0_YCNT_CUR	DMA0 Current Row Count (2D only) Register	0x00000000
0x31022040	DMA0_BWLCNT	DMA0 Bandwidth Limit Count Register	0x00000000
0x31022044	DMA0_BWLCNT_CUR	DMA0 Bandwidth Limit Count Current Register	0x00000000
0x31022048	DMA0_BWMCNT	DMA0 Bandwidth Monitor Count Register	0x00000000
0x3102204C	DMA0_BWMCNT_CUR	DMA0 Bandwidth Monitor Count Current Register	0x00000000

Table A-31: ADSP-SC59x DMA1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022080	DMA1_DSCPTR_NXT	DMA1 Pointer to Next Initial Descriptor Register	0x00000000
0x31022084	DMA1_ADDRSTART	DMA1 Start Address of Current Buffer Register	0x00000000
0x31022088	DMA1_CFG	DMA1 Configuration Register	0x00000000
0x3102208C	DMA1_XCNT	DMA1 Inner Loop Count Start Value Register	0x00000000
0x31022090	DMA1_XMOD	DMA1 Inner Loop Address Increment Register	0x00000000
0x31022094	DMA1_YCNT	DMA1 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022098	DMA1_YMOD	DMA1 Outer Loop Address Increment (2D only) Register	0x00000000
0x310220A4	DMA1_DSCPTR_CUR	DMA1 Current Descriptor Pointer Register	0x00000000
0x310220A8	DMA1_DSCPTR_PRV	DMA1 Previous Initial Descriptor Pointer Register	0x00000000
0x310220AC	DMA1_ADDR_CUR	DMA1 Current Address Register	0x00000000
0x310220B0	DMA1_STAT	DMA1 Status Register	0x00006000
0x310220B4	DMA1_XCNT_CUR	DMA1 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310220B8	DMA1_YCNT_CUR	DMA1 Current Row Count (2D only) Register	0x00000000
0x310220C0	DMA1_BWLCNT	DMA1 Bandwidth Limit Count Register	0x00000000
0x310220C4	DMA1_BWLCNT_CUR	DMA1 Bandwidth Limit Count Current Register	0x00000000
0x310220C8	DMA1_BWMCNT	DMA1 Bandwidth Monitor Count Register	0x00000000
0x310220CC	DMA1_BWMCNT_CUR	DMA1 Bandwidth Monitor Count Current Register	0x00000000

Table A-32: ADSP-SC59x DMA10 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023000	DMA10_DSCPTR_NXT	DMA10 Pointer to Next Initial Descriptor Register	0x00000000

Table A-32: ADSP-SC59x DMA10 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31023004	DMA10_ADDRSTART	DMA10 Start Address of Current Buffer Register	0x00000000
0x31023008	DMA10_CFG	DMA10 Configuration Register	0x00000000
0x3102300C	DMA10_XCNT	DMA10 Inner Loop Count Start Value Register	0x00000000
0x31023010	DMA10_XMOD	DMA10 Inner Loop Address Increment Register	0x00000000
0x31023014	DMA10_YCNT	DMA10 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023018	DMA10_YMOD	DMA10 Outer Loop Address Increment (2D only) Register	0x00000000
0x31023024	DMA10_DSCPTR_CUR	DMA10 Current Descriptor Pointer Register	0x00000000
0x31023028	DMA10_DSCPTR_PRV	DMA10 Previous Initial Descriptor Pointer Register	0x00000000
0x3102302C	DMA10_ADDR_CUR	DMA10 Current Address Register	0x00000000
0x31023030	DMA10_STAT	DMA10 Status Register	0x00006000
0x31023034	DMA10_XCNT_CUR	DMA10 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31023038	DMA10_YCNT_CUR	DMA10 Current Row Count (2D only) Register	0x00000000
0x31023040	DMA10_BWLCNT	DMA10 Bandwidth Limit Count Register	0x00000000
0x31023044	DMA10_BWLCNT_CUR	DMA10 Bandwidth Limit Count Current Register	0x00000000
0x31023048	DMA10_BWMCNT	DMA10 Bandwidth Monitor Count Register	0x00000000
0x3102304C	DMA10_BWMCNT_CUR	DMA10 Bandwidth Monitor Count Current Register	0x00000000

Table A-33: ADSP-SC59x DMA11 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023080	DMA11_DSCPTR_NXT	DMA11 Pointer to Next Initial Descriptor Register	0x00000000
0x31023084	DMA11_ADDRSTART	DMA11 Start Address of Current Buffer Register	0x00000000
0x31023088	DMA11_CFG	DMA11 Configuration Register	0x00000000
0x3102308C	DMA11_XCNT	DMA11 Inner Loop Count Start Value Register	0x00000000
0x31023090	DMA11_XMOD	DMA11 Inner Loop Address Increment Register	0x00000000
0x31023094	DMA11_YCNT	DMA11 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023098	DMA11_YMOD	DMA11 Outer Loop Address Increment (2D only) Register	0x00000000
0x310230A4	DMA11_DSCPTR_CUR	DMA11 Current Descriptor Pointer Register	0x00000000

Table A-33: ADSP-SC59x DMA11 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310230A8	DMA11_DSCPTR_PRV	DMA11 Previous Initial Descriptor Pointer Register	0x00000000
0x310230AC	DMA11_ADDR_CUR	DMA11 Current Address Register	0x00000000
0x310230B0	DMA11_STAT	DMA11 Status Register	0x00006000
0x310230B4	DMA11_XCNT_CUR	DMA11 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310230B8	DMA11_YCNT_CUR	DMA11 Current Row Count (2D only) Register	0x00000000
0x310230C0	DMA11_BWLCNT	DMA11 Bandwidth Limit Count Register	0x00000000
0x310230C4	DMA11_BWLCNT_CUR	DMA11 Bandwidth Limit Count Current Register	0x00000000
0x310230C8	DMA11_BWMCNT	DMA11 Bandwidth Monitor Count Register	0x00000000
0x310230CC	DMA11_BWMCNT_CUR	DMA11 Bandwidth Monitor Count Current Register	0x00000000

Table A-34: ADSP-SC59x DMA12 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023100	DMA12_DSCPTR_NXT	DMA12 Pointer to Next Initial Descriptor Register	0x00000000
0x31023104	DMA12_ADDRSTART	DMA12 Start Address of Current Buffer Register	0x00000000
0x31023108	DMA12_CFG	DMA12 Configuration Register	0x00000000
0x3102310C	DMA12_XCNT	DMA12 Inner Loop Count Start Value Register	0x00000000
0x31023110	DMA12_XMOD	DMA12 Inner Loop Address Increment Register	0x00000000
0x31023114	DMA12_YCNT	DMA12 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023118	DMA12_YMOD	DMA12 Outer Loop Address Increment (2D only) Register	0x00000000
0x31023124	DMA12_DSCPTR_CUR	DMA12 Current Descriptor Pointer Register	0x00000000
0x31023128	DMA12_DSCPTR_PRV	DMA12 Previous Initial Descriptor Pointer Register	0x00000000
0x3102312C	DMA12_ADDR_CUR	DMA12 Current Address Register	0x00000000
0x31023130	DMA12_STAT	DMA12 Status Register	0x00006000
0x31023134	DMA12_XCNT_CUR	DMA12 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31023138	DMA12_YCNT_CUR	DMA12 Current Row Count (2D only) Register	0x00000000
0x31023140	DMA12_BWLCNT	DMA12 Bandwidth Limit Count Register	0x00000000
0x31023144	DMA12_BWLCNT_CUR	DMA12 Bandwidth Limit Count Current Register	0x00000000
0x31023148	DMA12_BWMCNT	DMA12 Bandwidth Monitor Count Register	0x00000000

Table A-34: ADSP-SC59x DMA12 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102314C	DMA12_BWMCNT_CUR	DMA12 Bandwidth Monitor Count Current Register	0x00000000

Table A-35: ADSP-SC59x DMA13 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023180	DMA13_DSCPTR_NXT	DMA13 Pointer to Next Initial Descriptor Register	0x00000000
0x31023184	DMA13_ADDRSTART	DMA13 Start Address of Current Buffer Register	0x00000000
0x31023188	DMA13_CFG	DMA13 Configuration Register	0x00000000
0x3102318C	DMA13_XCNT	DMA13 Inner Loop Count Start Value Register	0x00000000
0x31023190	DMA13_XMOD	DMA13 Inner Loop Address Increment Register	0x00000000
0x31023194	DMA13_YCNT	DMA13 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023198	DMA13_YMOD	DMA13 Outer Loop Address Increment (2D only) Register	0x00000000
0x310231A4	DMA13_DSCPTR_CUR	DMA13 Current Descriptor Pointer Register	0x00000000
0x310231A8	DMA13_DSCPTR_PRV	DMA13 Previous Initial Descriptor Pointer Register	0x00000000
0x310231AC	DMA13_ADDR_CUR	DMA13 Current Address Register	0x00000000
0x310231B0	DMA13_STAT	DMA13 Status Register	0x00006000
0x310231B4	DMA13_XCNT_CUR	DMA13 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310231B8	DMA13_YCNT_CUR	DMA13 Current Row Count (2D only) Register	0x00000000
0x310231C0	DMA13_BWLCNT	DMA13 Bandwidth Limit Count Register	0x00000000
0x310231C4	DMA13_BWLCNT_CUR	DMA13 Bandwidth Limit Count Current Register	0x00000000
0x310231C8	DMA13_BWMCNT	DMA13 Bandwidth Monitor Count Register	0x00000000
0x310231CC	DMA13_BWMCNT_CUR	DMA13 Bandwidth Monitor Count Current Register	0x00000000

Table A-36: ADSP-SC59x DMA14 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023200	DMA14_DSCPTR_NXT	DMA14 Pointer to Next Initial Descriptor Register	0x00000000
0x31023204	DMA14_ADDRSTART	DMA14 Start Address of Current Buffer Register	0x00000000
0x31023208	DMA14_CFG	DMA14 Configuration Register	0x00000000
0x3102320C	DMA14_XCNT	DMA14 Inner Loop Count Start Value Register	0x00000000

Table A-36: ADSP-SC59x DMA14 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31023210	DMA14_XMOD	DMA14 Inner Loop Address Increment Register	0x00000000
0x31023214	DMA14_YCNT	DMA14 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023218	DMA14_YMOD	DMA14 Outer Loop Address Increment (2D only) Register	0x00000000
0x31023224	DMA14_DSCPTR_CUR	DMA14 Current Descriptor Pointer Register	0x00000000
0x31023228	DMA14_DSCPTR_PRV	DMA14 Previous Initial Descriptor Pointer Register	0x00000000
0x3102322C	DMA14_ADDR_CUR	DMA14 Current Address Register	0x00000000
0x31023230	DMA14_STAT	DMA14 Status Register	0x00006000
0x31023234	DMA14_XCNT_CUR	DMA14 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31023238	DMA14_YCNT_CUR	DMA14 Current Row Count (2D only) Register	0x00000000
0x31023240	DMA14_BWLCNT	DMA14 Bandwidth Limit Count Register	0x00000000
0x31023244	DMA14_BWLCNT_CUR	DMA14 Bandwidth Limit Count Current Register	0x00000000
0x31023248	DMA14_BWMCNT	DMA14 Bandwidth Monitor Count Register	0x00000000
0x3102324C	DMA14_BWMCNT_CUR	DMA14 Bandwidth Monitor Count Current Register	0x00000000

Table A-37: ADSP-SC59x DMA15 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023280	DMA15_DSCPTR_NXT	DMA15 Pointer to Next Initial Descriptor Register	0x00000000
0x31023284	DMA15_ADDRSTART	DMA15 Start Address of Current Buffer Register	0x00000000
0x31023288	DMA15_CFG	DMA15 Configuration Register	0x00000000
0x3102328C	DMA15_XCNT	DMA15 Inner Loop Count Start Value Register	0x00000000
0x31023290	DMA15_XMOD	DMA15 Inner Loop Address Increment Register	0x00000000
0x31023294	DMA15_YCNT	DMA15 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023298	DMA15_YMOD	DMA15 Outer Loop Address Increment (2D only) Register	0x00000000
0x310232A4	DMA15_DSCPTR_CUR	DMA15 Current Descriptor Pointer Register	0x00000000
0x310232A8	DMA15_DSCPTR_PRV	DMA15 Previous Initial Descriptor Pointer Register	0x00000000
0x310232AC	DMA15_ADDR_CUR	DMA15 Current Address Register	0x00000000
0x310232B0	DMA15_STAT	DMA15 Status Register	0x00006000

Table A-37: ADSP-SC59x DMA15 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310232B4	DMA15_XCNT_CUR	DMA15 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310232B8	DMA15_YCNT_CUR	DMA15 Current Row Count (2D only) Register	0x00000000
0x310232C0	DMA15_BWLCNT	DMA15 Bandwidth Limit Count Register	0x00000000
0x310232C4	DMA15_BWLCNT_CUR	DMA15 Bandwidth Limit Count Current Register	0x00000000
0x310232C8	DMA15_BWMCNT	DMA15 Bandwidth Monitor Count Register	0x00000000
0x310232CC	DMA15_BWMCNT_CUR	DMA15 Bandwidth Monitor Count Current Register	0x00000000

Table A-38: ADSP-SC59x DMA16 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023300	DMA16_DSCPTR_NXT	DMA16 Pointer to Next Initial Descriptor Register	0x00000000
0x31023304	DMA16_ADDRSTART	DMA16 Start Address of Current Buffer Register	0x00000000
0x31023308	DMA16_CFG	DMA16 Configuration Register	0x00000000
0x3102330C	DMA16_XCNT	DMA16 Inner Loop Count Start Value Register	0x00000000
0x31023310	DMA16_XMOD	DMA16 Inner Loop Address Increment Register	0x00000000
0x31023314	DMA16_YCNT	DMA16 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023318	DMA16_YMOD	DMA16 Outer Loop Address Increment (2D only) Register	0x00000000
0x31023324	DMA16_DSCPTR_CUR	DMA16 Current Descriptor Pointer Register	0x00000000
0x31023328	DMA16_DSCPTR_PRV	DMA16 Previous Initial Descriptor Pointer Register	0x00000000
0x3102332C	DMA16_ADDR_CUR	DMA16 Current Address Register	0x00000000
0x31023330	DMA16_STAT	DMA16 Status Register	0x00006000
0x31023334	DMA16_XCNT_CUR	DMA16 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31023338	DMA16_YCNT_CUR	DMA16 Current Row Count (2D only) Register	0x00000000
0x31023340	DMA16_BWLCNT	DMA16 Bandwidth Limit Count Register	0x00000000
0x31023344	DMA16_BWLCNT_CUR	DMA16 Bandwidth Limit Count Current Register	0x00000000
0x31023348	DMA16_BWMCNT	DMA16 Bandwidth Monitor Count Register	0x00000000
0x3102334C	DMA16_BWMCNT_CUR	DMA16 Bandwidth Monitor Count Current Register	0x00000000

Table A-39: ADSP-SC59x DMA17 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31023380	DMA17_DSCPTR_NXT	DMA17 Pointer to Next Initial Descriptor Register	0x00000000
0x31023384	DMA17_ADDRSTART	DMA17 Start Address of Current Buffer Register	0x00000000
0x31023388	DMA17_CFG	DMA17 Configuration Register	0x00000000
0x3102338C	DMA17_XCNT	DMA17 Inner Loop Count Start Value Register	0x00000000
0x31023390	DMA17_XMOD	DMA17 Inner Loop Address Increment Register	0x00000000
0x31023394	DMA17_YCNT	DMA17 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31023398	DMA17_YMOD	DMA17 Outer Loop Address Increment (2D only) Register	0x00000000
0x310233A4	DMA17_DSCPTR_CUR	DMA17 Current Descriptor Pointer Register	0x00000000
0x310233A8	DMA17_DSCPTR_PRV	DMA17 Previous Initial Descriptor Pointer Register	0x00000000
0x310233AC	DMA17_ADDR_CUR	DMA17 Current Address Register	0x00000000
0x310233B0	DMA17_STAT	DMA17 Status Register	0x00006000
0x310233B4	DMA17_XCNT_CUR	DMA17 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310233B8	DMA17_YCNT_CUR	DMA17 Current Row Count (2D only) Register	0x00000000
0x310233C0	DMA17_BWLCNT	DMA17 Bandwidth Limit Count Register	0x00000000
0x310233C4	DMA17_BWLCNT_CUR	DMA17 Bandwidth Limit Count Current Register	0x00000000
0x310233C8	DMA17_BWMCNT	DMA17 Bandwidth Monitor Count Register	0x00000000
0x310233CC	DMA17_BWMCNT_CUR	DMA17 Bandwidth Monitor Count Current Register	0x00000000

Table A-40: ADSP-SC59x DMA18 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7100	DMA18_DSCPTR_NXT	DMA18 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7104	DMA18_ADDRSTART	DMA18 Start Address of Current Buffer Register	0x00000000
0x310A7108	DMA18_CFG	DMA18 Configuration Register	0x00000000
0x310A710C	DMA18_XCNT	DMA18 Inner Loop Count Start Value Register	0x00000000
0x310A7110	DMA18_XMOD	DMA18 Inner Loop Address Increment Register	0x00000000
0x310A7114	DMA18_YCNT	DMA18 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7118	DMA18_YMOD	DMA18 Outer Loop Address Increment (2D only) Register	0x00000000

Table A-40: ADSP-SC59x DMA18 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7124	DMA18_DSCPTR_CUR	DMA18 Current Descriptor Pointer Register	0x00000000
0x310A7128	DMA18_DSCPTR_PRV	DMA18 Previous Initial Descriptor Pointer Register	0x00000000
0x310A712C	DMA18_ADDR_CUR	DMA18 Current Address Register	0x00000000
0x310A7130	DMA18_STAT	DMA18 Status Register	0x00006000
0x310A7134	DMA18_XCNT_CUR	DMA18 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A7138	DMA18_YCNT_CUR	DMA18 Current Row Count (2D only) Register	0x00000000
0x310A7140	DMA18_BWLCNT	DMA18 Bandwidth Limit Count Register	0x00000000
0x310A7144	DMA18_BWLCNT_CUR	DMA18 Bandwidth Limit Count Current Register	0x00000000
0x310A7148	DMA18_BWMCNT	DMA18 Bandwidth Monitor Count Register	0x00000000
0x310A714C	DMA18_BWMCNT_CUR	DMA18 Bandwidth Monitor Count Current Register	0x00000000

Table A-41: ADSP-SC59x DMA19 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7180	DMA19_DSCPTR_NXT	DMA19 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7184	DMA19_ADDRSTART	DMA19 Start Address of Current Buffer Register	0x00000000
0x310A7188	DMA19_CFG	DMA19 Configuration Register	0x00000000
0x310A718C	DMA19_XCNT	DMA19 Inner Loop Count Start Value Register	0x00000000
0x310A7190	DMA19_XMOD	DMA19 Inner Loop Address Increment Register	0x00000000
0x310A7194	DMA19_YCNT	DMA19 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7198	DMA19_YMOD	DMA19 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A71A4	DMA19_DSCPTR_CUR	DMA19 Current Descriptor Pointer Register	0x00000000
0x310A71A8	DMA19_DSCPTR_PRV	DMA19 Previous Initial Descriptor Pointer Register	0x00000000
0x310A71AC	DMA19_ADDR_CUR	DMA19 Current Address Register	0x00000000
0x310A71B0	DMA19_STAT	DMA19 Status Register	0x00006000
0x310A71B4	DMA19_XCNT_CUR	DMA19 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A71B8	DMA19_YCNT_CUR	DMA19 Current Row Count (2D only) Register	0x00000000
0x310A71C0	DMA19_BWLCNT	DMA19 Bandwidth Limit Count Register	0x00000000
0x310A71C4	DMA19_BWLCNT_CUR	DMA19 Bandwidth Limit Count Current Register	0x00000000

Table A-41: ADSP-SC59x DMA19 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A71C8	DMA19_BWMCNT	DMA19 Bandwidth Monitor Count Register	0x00000000
0x310A71CC	DMA19_BWMCNT_CUR	DMA19 Bandwidth Monitor Count Current Register	0x00000000

Table A-42: ADSP-SC59x DMA2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022100	DMA2_DSCPTR_NXT	DMA2 Pointer to Next Initial Descriptor Register	0x00000000
0x31022104	DMA2_ADDRSTART	DMA2 Start Address of Current Buffer Register	0x00000000
0x31022108	DMA2_CFG	DMA2 Configuration Register	0x00000000
0x3102210C	DMA2_XCNT	DMA2 Inner Loop Count Start Value Register	0x00000000
0x31022110	DMA2_XMOD	DMA2 Inner Loop Address Increment Register	0x00000000
0x31022114	DMA2_YCNT	DMA2 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022118	DMA2_YMOD	DMA2 Outer Loop Address Increment (2D only) Register	0x00000000
0x31022124	DMA2_DSCPTR_CUR	DMA2 Current Descriptor Pointer Register	0x00000000
0x31022128	DMA2_DSCPTR_PRV	DMA2 Previous Initial Descriptor Pointer Register	0x00000000
0x3102212C	DMA2_ADDR_CUR	DMA2 Current Address Register	0x00000000
0x31022130	DMA2_STAT	DMA2 Status Register	0x00006000
0x31022134	DMA2_XCNT_CUR	DMA2 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31022138	DMA2_YCNT_CUR	DMA2 Current Row Count (2D only) Register	0x00000000
0x31022140	DMA2_BWLCNT	DMA2 Bandwidth Limit Count Register	0x00000000
0x31022144	DMA2_BWLCNT_CUR	DMA2 Bandwidth Limit Count Current Register	0x00000000
0x31022148	DMA2_BWMCNT	DMA2 Bandwidth Monitor Count Register	0x00000000
0x3102214C	DMA2_BWMCNT_CUR	DMA2 Bandwidth Monitor Count Current Register	0x00000000

Table A-43: ADSP-SC59x DMA20 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026080	DMA20_DSCPTR_NXT	DMA20 Pointer to Next Initial Descriptor Register	0x00000000
0x31026084	DMA20_ADDRSTART	DMA20 Start Address of Current Buffer Register	0x00000000
0x31026088	DMA20_CFG	DMA20 Configuration Register	0x00000000
0x3102608C	DMA20_XCNT	DMA20 Inner Loop Count Start Value Register	0x00000000

Table A-43: ADSP-SC59x DMA20 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31026090	DMA20_XMOD	DMA20 Inner Loop Address Increment Register	0x00000000
0x31026094	DMA20_YCNT	DMA20 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026098	DMA20_YMOD	DMA20 Outer Loop Address Increment (2D only) Register	0x00000000
0x310260A4	DMA20_DSCPTR_CUR	DMA20 Current Descriptor Pointer Register	0x00000000
0x310260A8	DMA20_DSCPTR_PRV	DMA20 Previous Initial Descriptor Pointer Register	0x00000000
0x310260AC	DMA20_ADDR_CUR	DMA20 Current Address Register	0x00000000
0x310260B0	DMA20_STAT	DMA20 Status Register	0x00006000
0x310260B4	DMA20_XCNT_CUR	DMA20 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310260B8	DMA20_YCNT_CUR	DMA20 Current Row Count (2D only) Register	0x00000000
0x310260C0	DMA20_BWLCNT	DMA20 Bandwidth Limit Count Register	0x00000000
0x310260C4	DMA20_BWLCNT_CUR	DMA20 Bandwidth Limit Count Current Register	0x00000000
0x310260C8	DMA20_BWMCNT	DMA20 Bandwidth Monitor Count Register	0x00000000
0x310260CC	DMA20_BWMCNT_CUR	DMA20 Bandwidth Monitor Count Current Register	0x00000000

Table A-44: ADSP-SC59x DMA21 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026000	DMA21_DSCPTR_NXT	DMA21 Pointer to Next Initial Descriptor Register	0x00000000
0x31026004	DMA21_ADDRSTART	DMA21 Start Address of Current Buffer Register	0x00000000
0x31026008	DMA21_CFG	DMA21 Configuration Register	0x00000000
0x3102600C	DMA21_XCNT	DMA21 Inner Loop Count Start Value Register	0x00000000
0x31026010	DMA21_XMOD	DMA21 Inner Loop Address Increment Register	0x00000000
0x31026014	DMA21_YCNT	DMA21 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026018	DMA21_YMOD	DMA21 Outer Loop Address Increment (2D only) Register	0x00000000
0x31026024	DMA21_DSCPTR_CUR	DMA21 Current Descriptor Pointer Register	0x00000000
0x31026028	DMA21_DSCPTR_PRV	DMA21 Previous Initial Descriptor Pointer Register	0x00000000
0x3102602C	DMA21_ADDR_CUR	DMA21 Current Address Register	0x00000000
0x31026030	DMA21_STAT	DMA21 Status Register	0x00006000

Table A-44: ADSP-SC59x DMA21 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31026034	DMA21_XCNT_CUR	DMA21 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31026038	DMA21_YCNT_CUR	DMA21 Current Row Count (2D only) Register	0x00000000
0x31026040	DMA21_BWLCNT	DMA21 Bandwidth Limit Count Register	0x00000000
0x31026044	DMA21_BWLCNT_CUR	DMA21 Bandwidth Limit Count Current Register	0x00000000
0x31026048	DMA21_BWMCNT	DMA21 Bandwidth Monitor Count Register	0x00000000
0x3102604C	DMA21_BWMCNT_CUR	DMA21 Bandwidth Monitor Count Current Register	0x00000000

Table A-45: ADSP-SC59x DMA22 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D000	DMA22_DSCPTR_NXT	DMA22 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D004	DMA22_ADDRSTART	DMA22 Start Address of Current Buffer Register	0x00000000
0x3102D008	DMA22_CFG	DMA22 Configuration Register	0x00000000
0x3102D00C	DMA22_XCNT	DMA22 Inner Loop Count Start Value Register	0x00000000
0x3102D010	DMA22_XMOD	DMA22 Inner Loop Address Increment Register	0x00000000
0x3102D014	DMA22_YCNT	DMA22 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D018	DMA22_YMOD	DMA22 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D024	DMA22_DSCPTR_CUR	DMA22 Current Descriptor Pointer Register	0x00000000
0x3102D028	DMA22_DSCPTR_PRV	DMA22 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D02C	DMA22_ADDR_CUR	DMA22 Current Address Register	0x00000000
0x3102D030	DMA22_STAT	DMA22 Status Register	0x00006000
0x3102D034	DMA22_XCNT_CUR	DMA22 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D038	DMA22_YCNT_CUR	DMA22 Current Row Count (2D only) Register	0x00000000
0x3102D040	DMA22_BWLCNT	DMA22 Bandwidth Limit Count Register	0x00000000
0x3102D044	DMA22_BWLCNT_CUR	DMA22 Bandwidth Limit Count Current Register	0x00000000
0x3102D048	DMA22_BWMCNT	DMA22 Bandwidth Monitor Count Register	0x00000000
0x3102D04C	DMA22_BWMCNT_CUR	DMA22 Bandwidth Monitor Count Current Register	0x00000000

Table A-46: ADSP-SC59x DMA23 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D080	DMA23_DSCPTR_NXT	DMA23 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D084	DMA23_ADDRSTART	DMA23 Start Address of Current Buffer Register	0x00000000
0x3102D088	DMA23_CFG	DMA23 Configuration Register	0x00000000
0x3102D08C	DMA23_XCNT	DMA23 Inner Loop Count Start Value Register	0x00000000
0x3102D090	DMA23_XMOD	DMA23 Inner Loop Address Increment Register	0x00000000
0x3102D094	DMA23_YCNT	DMA23 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D098	DMA23_YMOD	DMA23 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D0A4	DMA23_DSCPTR_CUR	DMA23 Current Descriptor Pointer Register	0x00000000
0x3102D0A8	DMA23_DSCPTR_PRV	DMA23 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D0AC	DMA23_ADDR_CUR	DMA23 Current Address Register	0x00000000
0x3102D0B0	DMA23_STAT	DMA23 Status Register	0x00006000
0x3102D0B4	DMA23_XCNT_CUR	DMA23 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D0B8	DMA23_YCNT_CUR	DMA23 Current Row Count (2D only) Register	0x00000000
0x3102D0C0	DMA23_BWLCNT	DMA23 Bandwidth Limit Count Register	0x00000000
0x3102D0C4	DMA23_BWLCNT_CUR	DMA23 Bandwidth Limit Count Current Register	0x00000000
0x3102D0C8	DMA23_BWMCNT	DMA23 Bandwidth Monitor Count Register	0x00000000
0x3102D0CC	DMA23_BWMCNT_CUR	DMA23 Bandwidth Monitor Count Current Register	0x00000000

Table A-47: ADSP-SC59x DMA24 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D100	DMA24_DSCPTR_NXT	DMA24 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D104	DMA24_ADDRSTART	DMA24 Start Address of Current Buffer Register	0x00000000
0x3102D108	DMA24_CFG	DMA24 Configuration Register	0x00000000
0x3102D10C	DMA24_XCNT	DMA24 Inner Loop Count Start Value Register	0x00000000
0x3102D110	DMA24_XMOD	DMA24 Inner Loop Address Increment Register	0x00000000
0x3102D114	DMA24_YCNT	DMA24 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D118	DMA24_YMOD	DMA24 Outer Loop Address Increment (2D only) Register	0x00000000

Table A-47: ADSP-SC59x DMA24 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D124	DMA24_DSCPTR_CUR	DMA24 Current Descriptor Pointer Register	0x00000000
0x3102D128	DMA24_DSCPTR_PRV	DMA24 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D12C	DMA24_ADDR_CUR	DMA24 Current Address Register	0x00000000
0x3102D130	DMA24_STAT	DMA24 Status Register	0x00006000
0x3102D134	DMA24_XCNT_CUR	DMA24 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D138	DMA24_YCNT_CUR	DMA24 Current Row Count (2D only) Register	0x00000000
0x3102D140	DMA24_BWLCNT	DMA24 Bandwidth Limit Count Register	0x00000000
0x3102D144	DMA24_BWLCNT_CUR	DMA24 Bandwidth Limit Count Current Register	0x00000000
0x3102D148	DMA24_BWMCNT	DMA24 Bandwidth Monitor Count Register	0x00000000
0x3102D14C	DMA24_BWMCNT_CUR	DMA24 Bandwidth Monitor Count Current Register	0x00000000

Table A-48: ADSP-SC59x DMA25 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D180	DMA25_DSCPTR_NXT	DMA25 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D184	DMA25_ADDRSTART	DMA25 Start Address of Current Buffer Register	0x00000000
0x3102D188	DMA25_CFG	DMA25 Configuration Register	0x00000000
0x3102D18C	DMA25_XCNT	DMA25 Inner Loop Count Start Value Register	0x00000000
0x3102D190	DMA25_XMOD	DMA25 Inner Loop Address Increment Register	0x00000000
0x3102D194	DMA25_YCNT	DMA25 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D198	DMA25_YMOD	DMA25 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D1A4	DMA25_DSCPTR_CUR	DMA25 Current Descriptor Pointer Register	0x00000000
0x3102D1A8	DMA25_DSCPTR_PRV	DMA25 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D1AC	DMA25_ADDR_CUR	DMA25 Current Address Register	0x00000000
0x3102D1B0	DMA25_STAT	DMA25 Status Register	0x00006000
0x3102D1B4	DMA25_XCNT_CUR	DMA25 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D1B8	DMA25_YCNT_CUR	DMA25 Current Row Count (2D only) Register	0x00000000
0x3102D1C0	DMA25_BWLCNT	DMA25 Bandwidth Limit Count Register	0x00000000
0x3102D1C4	DMA25_BWLCNT_CUR	DMA25 Bandwidth Limit Count Current Register	0x00000000

Table A-48: ADSP-SC59x DMA25 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D1C8	DMA25_BWMCNT	DMA25 Bandwidth Monitor Count Register	0x00000000
0x3102D1CC	DMA25_BWMCNT_CUR	DMA25 Bandwidth Monitor Count Current Register	0x00000000

Table A-49: ADSP-SC59x DMA26 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D200	DMA26_DSCPTR_NXT	DMA26 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D204	DMA26_ADDRSTART	DMA26 Start Address of Current Buffer Register	0x00000000
0x3102D208	DMA26_CFG	DMA26 Configuration Register	0x00000000
0x3102D20C	DMA26_XCNT	DMA26 Inner Loop Count Start Value Register	0x00000000
0x3102D210	DMA26_XMOD	DMA26 Inner Loop Address Increment Register	0x00000000
0x3102D214	DMA26_YCNT	DMA26 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D218	DMA26_YMOD	DMA26 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D224	DMA26_DSCPTR_CUR	DMA26 Current Descriptor Pointer Register	0x00000000
0x3102D228	DMA26_DSCPTR_PRV	DMA26 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D22C	DMA26_ADDR_CUR	DMA26 Current Address Register	0x00000000
0x3102D230	DMA26_STAT	DMA26 Status Register	0x00006000
0x3102D234	DMA26_XCNT_CUR	DMA26 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D238	DMA26_YCNT_CUR	DMA26 Current Row Count (2D only) Register	0x00000000
0x3102D240	DMA26_BWLCNT	DMA26 Bandwidth Limit Count Register	0x00000000
0x3102D244	DMA26_BWLCNT_CUR	DMA26 Bandwidth Limit Count Current Register	0x00000000
0x3102D248	DMA26_BWMCNT	DMA26 Bandwidth Monitor Count Register	0x00000000
0x3102D24C	DMA26_BWMCNT_CUR	DMA26 Bandwidth Monitor Count Current Register	0x00000000

Table A-50: ADSP-SC59x DMA27 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D280	DMA27_DSCPTR_NXT	DMA27 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D284	DMA27_ADDRSTART	DMA27 Start Address of Current Buffer Register	0x00000000
0x3102D288	DMA27_CFG	DMA27 Configuration Register	0x00000000

Table A-50: ADSP-SC59x DMA27 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D28C	DMA27_XCNT	DMA27 Inner Loop Count Start Value Register	0x00000000
0x3102D290	DMA27_XMOD	DMA27 Inner Loop Address Increment Register	0x00000000
0x3102D294	DMA27_YCNT	DMA27 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D298	DMA27_YMOD	DMA27 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D2A4	DMA27_DSCPTR_CUR	DMA27 Current Descriptor Pointer Register	0x00000000
0x3102D2A8	DMA27_DSCPTR_PRV	DMA27 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D2AC	DMA27_ADDR_CUR	DMA27 Current Address Register	0x00000000
0x3102D2B0	DMA27_STAT	DMA27 Status Register	0x00006000
0x3102D2B4	DMA27_XCNT_CUR	DMA27 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D2B8	DMA27_YCNT_CUR	DMA27 Current Row Count (2D only) Register	0x00000000
0x3102D2C0	DMA27_BWLCNT	DMA27 Bandwidth Limit Count Register	0x00000000
0x3102D2C4	DMA27_BWLCNT_CUR	DMA27 Bandwidth Limit Count Current Register	0x00000000
0x3102D2C8	DMA27_BWMCNT	DMA27 Bandwidth Monitor Count Register	0x00000000
0x3102D2CC	DMA27_BWMCNT_CUR	DMA27 Bandwidth Monitor Count Current Register	0x00000000

Table A-51: ADSP-SC59x DMA28 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026400	DMA28_DSCPTR_NXT	DMA28 Pointer to Next Initial Descriptor Register	0x00000000
0x31026404	DMA28_ADDRSTART	DMA28 Start Address of Current Buffer Register	0x00000000
0x31026408	DMA28_CFG	DMA28 Configuration Register	0x00000000
0x3102640C	DMA28_XCNT	DMA28 Inner Loop Count Start Value Register	0x00000000
0x31026410	DMA28_XMOD	DMA28 Inner Loop Address Increment Register	0x00000000
0x31026414	DMA28_YCNT	DMA28 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026418	DMA28_YMOD	DMA28 Outer Loop Address Increment (2D only) Register	0x00000000
0x31026424	DMA28_DSCPTR_CUR	DMA28 Current Descriptor Pointer Register	0x00000000
0x31026428	DMA28_DSCPTR_PRV	DMA28 Previous Initial Descriptor Pointer Register	0x00000000
0x3102642C	DMA28_ADDR_CUR	DMA28 Current Address Register	0x00000000

Table A-51: ADSP-SC59x DMA28 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31026430	DMA28_STAT	DMA28 Status Register	0x00006000
0x31026434	DMA28_XCNT_CUR	DMA28 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31026438	DMA28_YCNT_CUR	DMA28 Current Row Count (2D only) Register	0x00000000
0x31026440	DMA28_BWLCNT	DMA28 Bandwidth Limit Count Register	0x00000000
0x31026444	DMA28_BWLCNT_CUR	DMA28 Bandwidth Limit Count Current Register	0x00000000
0x31026448	DMA28_BWMCNT	DMA28 Bandwidth Monitor Count Register	0x00000000
0x3102644C	DMA28_BWMCNT_CUR	DMA28 Bandwidth Monitor Count Current Register	0x00000000

Table A-52: ADSP-SC59x DMA29 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026480	DMA29_DSCPTR_NXT	DMA29 Pointer to Next Initial Descriptor Register	0x00000000
0x31026484	DMA29_ADDRSTART	DMA29 Start Address of Current Buffer Register	0x00000000
0x31026488	DMA29_CFG	DMA29 Configuration Register	0x00000000
0x3102648C	DMA29_XCNT	DMA29 Inner Loop Count Start Value Register	0x00000000
0x31026490	DMA29_XMOD	DMA29 Inner Loop Address Increment Register	0x00000000
0x31026494	DMA29_YCNT	DMA29 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026498	DMA29_YMOD	DMA29 Outer Loop Address Increment (2D only) Register	0x00000000
0x310264A4	DMA29_DSCPTR_CUR	DMA29 Current Descriptor Pointer Register	0x00000000
0x310264A8	DMA29_DSCPTR_PRV	DMA29 Previous Initial Descriptor Pointer Register	0x00000000
0x310264AC	DMA29_ADDR_CUR	DMA29 Current Address Register	0x00000000
0x310264B0	DMA29_STAT	DMA29 Status Register	0x00006000
0x310264B4	DMA29_XCNT_CUR	DMA29 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310264B8	DMA29_YCNT_CUR	DMA29 Current Row Count (2D only) Register	0x00000000
0x310264C0	DMA29_BWLCNT	DMA29 Bandwidth Limit Count Register	0x00000000
0x310264C4	DMA29_BWLCNT_CUR	DMA29 Bandwidth Limit Count Current Register	0x00000000
0x310264C8	DMA29_BWMCNT	DMA29 Bandwidth Monitor Count Register	0x00000000
0x310264CC	DMA29_BWMCNT_CUR	DMA29 Bandwidth Monitor Count Current Register	0x00000000

Table A-53: ADSP-SC59x DMA3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022180	DMA3_DSCPTR_NXT	DMA3 Pointer to Next Initial Descriptor Register	0x00000000
0x31022184	DMA3_ADDRSTART	DMA3 Start Address of Current Buffer Register	0x00000000
0x31022188	DMA3_CFG	DMA3 Configuration Register	0x00000000
0x3102218C	DMA3_XCNT	DMA3 Inner Loop Count Start Value Register	0x00000000
0x31022190	DMA3_XMOD	DMA3 Inner Loop Address Increment Register	0x00000000
0x31022194	DMA3_YCNT	DMA3 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022198	DMA3_YMOD	DMA3 Outer Loop Address Increment (2D only) Register	0x00000000
0x310221A4	DMA3_DSCPTR_CUR	DMA3 Current Descriptor Pointer Register	0x00000000
0x310221A8	DMA3_DSCPTR_PRV	DMA3 Previous Initial Descriptor Pointer Register	0x00000000
0x310221AC	DMA3_ADDR_CUR	DMA3 Current Address Register	0x00000000
0x310221B0	DMA3_STAT	DMA3 Status Register	0x00006000
0x310221B4	DMA3_XCNT_CUR	DMA3 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310221B8	DMA3_YCNT_CUR	DMA3 Current Row Count (2D only) Register	0x00000000
0x310221C0	DMA3_BWLCNT	DMA3 Bandwidth Limit Count Register	0x00000000
0x310221C4	DMA3_BWLCNT_CUR	DMA3 Bandwidth Limit Count Current Register	0x00000000
0x310221C8	DMA3_BWMCNT	DMA3 Bandwidth Monitor Count Register	0x00000000
0x310221CC	DMA3_BWMCNT_CUR	DMA3 Bandwidth Monitor Count Current Register	0x00000000

Table A-54: ADSP-SC59x DMA30 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFF000	DMA30_DSCPTR_NXT	DMA30 Pointer to Next Initial Descriptor Register	0x00000000
0x30FFF004	DMA30_ADDRSTART	DMA30 Start Address of Current Buffer Register	0x00000000
0x30FFF008	DMA30_CFG	DMA30 Configuration Register	0x00000000
0x30FFF00C	DMA30_XCNT	DMA30 Inner Loop Count Start Value Register	0x00000000
0x30FFF010	DMA30_XMOD	DMA30 Inner Loop Address Increment Register	0x00000000
0x30FFF014	DMA30_YCNT	DMA30 Outer Loop Count Start Value (2D only) Register	0x00000000
0x30FFF018	DMA30_YMOD	DMA30 Outer Loop Address Increment (2D only) Register	0x00000000
0x30FFF024	DMA30_DSCPTR_CUR	DMA30 Current Descriptor Pointer Register	0x00000000

Table A-54: ADSP-SC59x DMA30 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFF028	DMA30_DSCPTR_PRV	DMA30 Previous Initial Descriptor Pointer Register	0x00000000
0x30FFF02C	DMA30_ADDR_CUR	DMA30 Current Address Register	0x00000000
0x30FFF030	DMA30_STAT	DMA30 Status Register	0x00006000
0x30FFF034	DMA30_XCNT_CUR	DMA30 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x30FFF038	DMA30_YCNT_CUR	DMA30 Current Row Count (2D only) Register	0x00000000
0x30FFF040	DMA30_BWLCNT	DMA30 Bandwidth Limit Count Register	0x00000000
0x30FFF044	DMA30_BWLCNT_CUR	DMA30 Bandwidth Limit Count Current Register	0x00000000
0x30FFF048	DMA30_BWMCNT	DMA30 Bandwidth Monitor Count Register	0x00000000
0x30FFF04C	DMA30_BWMCNT_CUR	DMA30 Bandwidth Monitor Count Current Register	0x00000000

Table A-55: ADSP-SC59x DMA34 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026180	DMA34_DSCPTR_NXT	DMA34 Pointer to Next Initial Descriptor Register	0x00000000
0x31026184	DMA34_ADDRSTART	DMA34 Start Address of Current Buffer Register	0x00000000
0x31026188	DMA34_CFG	DMA34 Configuration Register	0x00000000
0x3102618C	DMA34_XCNT	DMA34 Inner Loop Count Start Value Register	0x00000000
0x31026190	DMA34_XMOD	DMA34 Inner Loop Address Increment Register	0x00000000
0x31026194	DMA34_YCNT	DMA34 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026198	DMA34_YMOD	DMA34 Outer Loop Address Increment (2D only) Register	0x00000000
0x310261A4	DMA34_DSCPTR_CUR	DMA34 Current Descriptor Pointer Register	0x00000000
0x310261A8	DMA34_DSCPTR_PRV	DMA34 Previous Initial Descriptor Pointer Register	0x00000000
0x310261AC	DMA34_ADDR_CUR	DMA34 Current Address Register	0x00000000
0x310261B0	DMA34_STAT	DMA34 Status Register	0x00006000
0x310261B4	DMA34_XCNT_CUR	DMA34 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310261B8	DMA34_YCNT_CUR	DMA34 Current Row Count (2D only) Register	0x00000000
0x310261C0	DMA34_BWLCNT	DMA34 Bandwidth Limit Count Register	0x00000000
0x310261C4	DMA34_BWLCNT_CUR	DMA34 Bandwidth Limit Count Current Register	0x00000000
0x310261C8	DMA34_BWMCNT	DMA34 Bandwidth Monitor Count Register	0x00000000

Table A-55: ADSP-SC59x DMA34 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310261CC	DMA34_BWMCNT_CUR	DMA34 Bandwidth Monitor Count Current Register	0x00000000

Table A-56: ADSP-SC59x DMA35 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026100	DMA35_DSCPTR_NXT	DMA35 Pointer to Next Initial Descriptor Register	0x00000000
0x31026104	DMA35_ADDRSTART	DMA35 Start Address of Current Buffer Register	0x00000000
0x31026108	DMA35_CFG	DMA35 Configuration Register	0x00000000
0x3102610C	DMA35_XCNT	DMA35 Inner Loop Count Start Value Register	0x00000000
0x31026110	DMA35_XMOD	DMA35 Inner Loop Address Increment Register	0x00000000
0x31026114	DMA35_YCNT	DMA35 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026118	DMA35_YMOD	DMA35 Outer Loop Address Increment (2D only) Register	0x00000000
0x31026124	DMA35_DSCPTR_CUR	DMA35 Current Descriptor Pointer Register	0x00000000
0x31026128	DMA35_DSCPTR_PRV	DMA35 Previous Initial Descriptor Pointer Register	0x00000000
0x3102612C	DMA35_ADDR_CUR	DMA35 Current Address Register	0x00000000
0x31026130	DMA35_STAT	DMA35 Status Register	0x00006000
0x31026134	DMA35_XCNT_CUR	DMA35 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31026138	DMA35_YCNT_CUR	DMA35 Current Row Count (2D only) Register	0x00000000
0x31026140	DMA35_BWLCNT	DMA35 Bandwidth Limit Count Register	0x00000000
0x31026144	DMA35_BWLCNT_CUR	DMA35 Bandwidth Limit Count Current Register	0x00000000
0x31026148	DMA35_BWMCNT	DMA35 Bandwidth Monitor Count Register	0x00000000
0x3102614C	DMA35_BWMCNT_CUR	DMA35 Bandwidth Monitor Count Current Register	0x00000000

Table A-57: ADSP-SC59x DMA36 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFF080	DMA36_DSCPTR_NXT	DMA36 Pointer to Next Initial Descriptor Register	0x00000000
0x30FFF084	DMA36_ADDRSTART	DMA36 Start Address of Current Buffer Register	0x00000000
0x30FFF088	DMA36_CFG	DMA36 Configuration Register	0x00000000
0x30FFF08C	DMA36_XCNT	DMA36 Inner Loop Count Start Value Register	0x00000000

Table A-57: ADSP-SC59x DMA36 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFF090	DMA36_XMOD	DMA36 Inner Loop Address Increment Register	0x00000000
0x30FFF094	DMA36_YCNT	DMA36 Outer Loop Count Start Value (2D only) Register	0x00000000
0x30FFF098	DMA36_YMOD	DMA36 Outer Loop Address Increment (2D only) Register	0x00000000
0x30FFF0A4	DMA36_DSCPTR_CUR	DMA36 Current Descriptor Pointer Register	0x00000000
0x30FFF0A8	DMA36_DSCPTR_PRV	DMA36 Previous Initial Descriptor Pointer Register	0x00000000
0x30FFF0AC	DMA36_ADDR_CUR	DMA36 Current Address Register	0x00000000
0x30FFF0B0	DMA36_STAT	DMA36 Status Register	0x00006000
0x30FFF0B4	DMA36_XCNT_CUR	DMA36 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x30FFF0B8	DMA36_YCNT_CUR	DMA36 Current Row Count (2D only) Register	0x00000000
0x30FFF0C0	DMA36_BWLCNT	DMA36 Bandwidth Limit Count Register	0x00000000
0x30FFF0C4	DMA36_BWLCNT_CUR	DMA36 Bandwidth Limit Count Current Register	0x00000000
0x30FFF0C8	DMA36_BWMCNT	DMA36 Bandwidth Monitor Count Register	0x00000000
0x30FFF0CC	DMA36_BWMCNT_CUR	DMA36 Bandwidth Monitor Count Current Register	0x00000000

Table A-58: ADSP-SC59x DMA37 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026280	DMA37_DSCPTR_NXT	DMA37 Pointer to Next Initial Descriptor Register	0x00000000
0x31026284	DMA37_ADDRSTART	DMA37 Start Address of Current Buffer Register	0x00000000
0x31026288	DMA37_CFG	DMA37 Configuration Register	0x00000000
0x3102628C	DMA37_XCNT	DMA37 Inner Loop Count Start Value Register	0x00000000
0x31026290	DMA37_XMOD	DMA37 Inner Loop Address Increment Register	0x00000000
0x31026294	DMA37_YCNT	DMA37 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026298	DMA37_YMOD	DMA37 Outer Loop Address Increment (2D only) Register	0x00000000
0x310262A4	DMA37_DSCPTR_CUR	DMA37 Current Descriptor Pointer Register	0x00000000
0x310262A8	DMA37_DSCPTR_PRV	DMA37 Previous Initial Descriptor Pointer Register	0x00000000
0x310262AC	DMA37_ADDR_CUR	DMA37 Current Address Register	0x00000000
0x310262B0	DMA37_STAT	DMA37 Status Register	0x00006000

Table A-58: ADSP-SC59x DMA37 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310262B4	DMA37_XCNT_CUR	DMA37 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310262B8	DMA37_YCNT_CUR	DMA37 Current Row Count (2D only) Register	0x00000000
0x310262C0	DMA37_BWLCNT	DMA37 Bandwidth Limit Count Register	0x00000000
0x310262C4	DMA37_BWLCNT_CUR	DMA37 Bandwidth Limit Count Current Register	0x00000000
0x310262C8	DMA37_BWMCNT	DMA37 Bandwidth Monitor Count Register	0x00000000
0x310262CC	DMA37_BWMCNT_CUR	DMA37 Bandwidth Monitor Count Current Register	0x00000000

Table A-59: ADSP-SC59x DMA38 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026200	DMA38_DSCPTR_NXT	DMA38 Pointer to Next Initial Descriptor Register	0x00000000
0x31026204	DMA38_ADDRSTART	DMA38 Start Address of Current Buffer Register	0x00000000
0x31026208	DMA38_CFG	DMA38 Configuration Register	0x00000000
0x3102620C	DMA38_XCNT	DMA38 Inner Loop Count Start Value Register	0x00000000
0x31026210	DMA38_XMOD	DMA38 Inner Loop Address Increment Register	0x00000000
0x31026214	DMA38_YCNT	DMA38 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026218	DMA38_YMOD	DMA38 Outer Loop Address Increment (2D only) Register	0x00000000
0x31026224	DMA38_DSCPTR_CUR	DMA38 Current Descriptor Pointer Register	0x00000000
0x31026228	DMA38_DSCPTR_PRV	DMA38 Previous Initial Descriptor Pointer Register	0x00000000
0x3102622C	DMA38_ADDR_CUR	DMA38 Current Address Register	0x00000000
0x31026230	DMA38_STAT	DMA38 Status Register	0x00006000
0x31026234	DMA38_XCNT_CUR	DMA38 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31026238	DMA38_YCNT_CUR	DMA38 Current Row Count (2D only) Register	0x00000000
0x31026240	DMA38_BWLCNT	DMA38 Bandwidth Limit Count Register	0x00000000
0x31026244	DMA38_BWLCNT_CUR	DMA38 Bandwidth Limit Count Current Register	0x00000000
0x31026248	DMA38_BWMCNT	DMA38 Bandwidth Monitor Count Register	0x00000000
0x3102624C	DMA38_BWMCNT_CUR	DMA38 Bandwidth Monitor Count Current Register	0x00000000

Table A-60: ADSP-SC59x DMA39 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109A000	DMA39_DSCPTR_NXT	DMA39 Pointer to Next Initial Descriptor Register	0x00000000
0x3109A004	DMA39_ADDRSTART	DMA39 Start Address of Current Buffer Register	0x00000000
0x3109A008	DMA39_CFG	DMA39 Configuration Register	0x00000220
0x3109A00C	DMA39_XCNT	DMA39 Inner Loop Count Start Value Register	0x00000000
0x3109A010	DMA39_XMOD	DMA39 Inner Loop Address Increment Register	0x00000000
0x3109A014	DMA39_YCNT	DMA39 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109A018	DMA39_YMOD	DMA39 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109A024	DMA39_DSCPTR_CUR	DMA39 Current Descriptor Pointer Register	0x00000000
0x3109A028	DMA39_DSCPTR_PRV	DMA39 Previous Initial Descriptor Pointer Register	0x00000000
0x3109A02C	DMA39_ADDR_CUR	DMA39 Current Address Register	0x00000000
0x3109A030	DMA39_STAT	DMA39 Status Register	0x00006000
0x3109A034	DMA39_XCNT_CUR	DMA39 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109A038	DMA39_YCNT_CUR	DMA39 Current Row Count (2D only) Register	0x00000000
0x3109A040	DMA39_BWLCNT	DMA39 Bandwidth Limit Count Register	0x00000000
0x3109A044	DMA39_BWLCNT_CUR	DMA39 Bandwidth Limit Count Current Register	0x00000000
0x3109A048	DMA39_BWMCNT	DMA39 Bandwidth Monitor Count Register	0x00000000
0x3109A04C	DMA39_BWMCNT_CUR	DMA39 Bandwidth Monitor Count Current Register	0x00000000

Table A-61: ADSP-SC59x DMA4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022200	DMA4_DSCPTR_NXT	DMA4 Pointer to Next Initial Descriptor Register	0x00000000
0x31022204	DMA4_ADDRSTART	DMA4 Start Address of Current Buffer Register	0x00000000
0x31022208	DMA4_CFG	DMA4 Configuration Register	0x00000000
0x3102220C	DMA4_XCNT	DMA4 Inner Loop Count Start Value Register	0x00000000
0x31022210	DMA4_XMOD	DMA4 Inner Loop Address Increment Register	0x00000000
0x31022214	DMA4_YCNT	DMA4 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022218	DMA4_YMOD	DMA4 Outer Loop Address Increment (2D only) Register	0x00000000
0x31022224	DMA4_DSCPTR_CUR	DMA4 Current Descriptor Pointer Register	0x00000000

Table A-61: ADSP-SC59x DMA4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31022228	DMA4_DSCPTR_PRV	DMA4 Previous Initial Descriptor Pointer Register	0x00000000
0x3102222C	DMA4_ADDR_CUR	DMA4 Current Address Register	0x00000000
0x31022230	DMA4_STAT	DMA4 Status Register	0x00006000
0x31022234	DMA4_XCNT_CUR	DMA4 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31022238	DMA4_YCNT_CUR	DMA4 Current Row Count (2D only) Register	0x00000000
0x31022240	DMA4_BWLCNT	DMA4 Bandwidth Limit Count Register	0x00000000
0x31022244	DMA4_BWLCNT_CUR	DMA4 Bandwidth Limit Count Current Register	0x00000000
0x31022248	DMA4_BWMCNT	DMA4 Bandwidth Monitor Count Register	0x00000000
0x3102224C	DMA4_BWMCNT_CUR	DMA4 Bandwidth Monitor Count Current Register	0x00000000

Table A-62: ADSP-SC59x DMA40 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109A080	DMA40_DSCPTR_NXT	DMA40 Pointer to Next Initial Descriptor Register	0x00000000
0x3109A084	DMA40_ADDRSTART	DMA40 Start Address of Current Buffer Register	0x00000000
0x3109A088	DMA40_CFG	DMA40 Configuration Register	0x00000222
0x3109A08C	DMA40_XCNT	DMA40 Inner Loop Count Start Value Register	0x00000000
0x3109A090	DMA40_XMOD	DMA40 Inner Loop Address Increment Register	0x00000000
0x3109A094	DMA40_YCNT	DMA40 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109A098	DMA40_YMOD	DMA40 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109A0A4	DMA40_DSCPTR_CUR	DMA40 Current Descriptor Pointer Register	0x00000000
0x3109A0A8	DMA40_DSCPTR_PRV	DMA40 Previous Initial Descriptor Pointer Register	0x00000000
0x3109A0AC	DMA40_ADDR_CUR	DMA40 Current Address Register	0x00000000
0x3109A0B0	DMA40_STAT	DMA40 Status Register	0x00006000
0x3109A0B4	DMA40_XCNT_CUR	DMA40 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109A0B8	DMA40_YCNT_CUR	DMA40 Current Row Count (2D only) Register	0x00000000
0x3109A0C0	DMA40_BWLCNT	DMA40 Bandwidth Limit Count Register	0x00000000
0x3109A0C4	DMA40_BWLCNT_CUR	DMA40 Bandwidth Limit Count Current Register	0x00000000
0x3109A0C8	DMA40_BWMCNT	DMA40 Bandwidth Monitor Count Register	0x00000000

Table A-62: ADSP-SC59x DMA40 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109A0CC	DMA40_BWMCNT_CUR	DMA40 Bandwidth Monitor Count Current Register	0x00000000

Table A-63: ADSP-SC59x DMA43 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109B000	DMA43_DSCPTR_NXT	DMA43 Pointer to Next Initial Descriptor Register	0x00000000
0x3109B004	DMA43_ADDRSTART	DMA43 Start Address of Current Buffer Register	0x00000000
0x3109B008	DMA43_CFG	DMA43 Configuration Register	0x00000220
0x3109B00C	DMA43_XCNT	DMA43 Inner Loop Count Start Value Register	0x00000000
0x3109B010	DMA43_XMOD	DMA43 Inner Loop Address Increment Register	0x00000000
0x3109B014	DMA43_YCNT	DMA43 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109B018	DMA43_YMOD	DMA43 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109B024	DMA43_DSCPTR_CUR	DMA43 Current Descriptor Pointer Register	0x00000000
0x3109B028	DMA43_DSCPTR_PRV	DMA43 Previous Initial Descriptor Pointer Register	0x00000000
0x3109B02C	DMA43_ADDR_CUR	DMA43 Current Address Register	0x00000000
0x3109B030	DMA43_STAT	DMA43 Status Register	0x0000B000
0x3109B034	DMA43_XCNT_CUR	DMA43 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109B038	DMA43_YCNT_CUR	DMA43 Current Row Count (2D only) Register	0x00000000
0x3109B040	DMA43_BWLCNT	DMA43 Bandwidth Limit Count Register	0x00000000
0x3109B044	DMA43_BWLCNT_CUR	DMA43 Bandwidth Limit Count Current Register	0x00000000
0x3109B048	DMA43_BWMCNT	DMA43 Bandwidth Monitor Count Register	0x00000000
0x3109B04C	DMA43_BWMCNT_CUR	DMA43 Bandwidth Monitor Count Current Register	0x00000000

Table A-64: ADSP-SC59x DMA44 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109B080	DMA44_DSCPTR_NXT	DMA44 Pointer to Next Initial Descriptor Register	0x00000000
0x3109B084	DMA44_ADDRSTART	DMA44 Start Address of Current Buffer Register	0x00000000
0x3109B088	DMA44_CFG	DMA44 Configuration Register	0x00000222
0x3109B08C	DMA44_XCNT	DMA44 Inner Loop Count Start Value Register	0x00000000

Table A-64: ADSP-SC59x DMA44 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109B090	DMA44_XMOD	DMA44 Inner Loop Address Increment Register	0x00000000
0x3109B094	DMA44_YCNT	DMA44 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109B098	DMA44_YMOD	DMA44 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109B0A4	DMA44_DSCPTR_CUR	DMA44 Current Descriptor Pointer Register	0x00000000
0x3109B0A8	DMA44_DSCPTR_PRV	DMA44 Previous Initial Descriptor Pointer Register	0x00000000
0x3109B0AC	DMA44_ADDR_CUR	DMA44 Current Address Register	0x00000000
0x3109B0B0	DMA44_STAT	DMA44 Status Register	0x0000B000
0x3109B0B4	DMA44_XCNT_CUR	DMA44 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109B0B8	DMA44_YCNT_CUR	DMA44 Current Row Count (2D only) Register	0x00000000
0x3109B0C0	DMA44_BWLCNT	DMA44 Bandwidth Limit Count Register	0x00000000
0x3109B0C4	DMA44_BWLCNT_CUR	DMA44 Bandwidth Limit Count Current Register	0x00000000
0x3109B0C8	DMA44_BWMCNT	DMA44 Bandwidth Monitor Count Register	0x00000000
0x3109B0CC	DMA44_BWMCNT_CUR	DMA44 Bandwidth Monitor Count Current Register	0x00000000

Table A-65: ADSP-SC59x DMA45 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7200	DMA45_DSCPTR_NXT	DMA45 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7204	DMA45_ADDRSTART	DMA45 Start Address of Current Buffer Register	0x00000000
0x310A7208	DMA45_CFG	DMA45 Configuration Register	0x00000000
0x310A720C	DMA45_XCNT	DMA45 Inner Loop Count Start Value Register	0x00000000
0x310A7210	DMA45_XMOD	DMA45 Inner Loop Address Increment Register	0x00000000
0x310A7214	DMA45_YCNT	DMA45 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7218	DMA45_YMOD	DMA45 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A7224	DMA45_DSCPTR_CUR	DMA45 Current Descriptor Pointer Register	0x00000000
0x310A7228	DMA45_DSCPTR_PRV	DMA45 Previous Initial Descriptor Pointer Register	0x00000000
0x310A722C	DMA45_ADDR_CUR	DMA45 Current Address Register	0x00000000
0x310A7230	DMA45_STAT	DMA45 Status Register	0x00006000

Table A-65: ADSP-SC59x DMA45 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7234	DMA45_XCNT_CUR	DMA45 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A7238	DMA45_YCNT_CUR	DMA45 Current Row Count (2D only) Register	0x00000000
0x310A7240	DMA45_BWLCNT	DMA45 Bandwidth Limit Count Register	0x00000000
0x310A7244	DMA45_BWLCNT_CUR	DMA45 Bandwidth Limit Count Current Register	0x00000000
0x310A7248	DMA45_BWMCNT	DMA45 Bandwidth Monitor Count Register	0x00000000
0x310A724C	DMA45_BWMCNT_CUR	DMA45 Bandwidth Monitor Count Current Register	0x00000000

Table A-66: ADSP-SC59x DMA46 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7280	DMA46_DSCPTR_NXT	DMA46 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7284	DMA46_ADDRSTART	DMA46 Start Address of Current Buffer Register	0x00000000
0x310A7288	DMA46_CFG	DMA46 Configuration Register	0x00000000
0x310A728C	DMA46_XCNT	DMA46 Inner Loop Count Start Value Register	0x00000000
0x310A7290	DMA46_XMOD	DMA46 Inner Loop Address Increment Register	0x00000000
0x310A7294	DMA46_YCNT	DMA46 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7298	DMA46_YMOD	DMA46 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A72A4	DMA46_DSCPTR_CUR	DMA46 Current Descriptor Pointer Register	0x00000000
0x310A72A8	DMA46_DSCPTR_PRV	DMA46 Previous Initial Descriptor Pointer Register	0x00000000
0x310A72AC	DMA46_ADDR_CUR	DMA46 Current Address Register	0x00000000
0x310A72B0	DMA46_STAT	DMA46 Status Register	0x00006000
0x310A72B4	DMA46_XCNT_CUR	DMA46 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A72B8	DMA46_YCNT_CUR	DMA46 Current Row Count (2D only) Register	0x00000000
0x310A72C0	DMA46_BWLCNT	DMA46 Bandwidth Limit Count Register	0x00000000
0x310A72C4	DMA46_BWLCNT_CUR	DMA46 Bandwidth Limit Count Current Register	0x00000000
0x310A72C8	DMA46_BWMCNT	DMA46 Bandwidth Monitor Count Register	0x00000000
0x310A72CC	DMA46_BWMCNT_CUR	DMA46 Bandwidth Monitor Count Current Register	0x00000000

Table A-67: ADSP-SC59x DMA47 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7300	DMA47_DSCPTR_NXT	DMA47 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7304	DMA47_ADDRSTART	DMA47 Start Address of Current Buffer Register	0x00000000
0x310A7308	DMA47_CFG	DMA47 Configuration Register	0x00000000
0x310A730C	DMA47_XCNT	DMA47 Inner Loop Count Start Value Register	0x00000000
0x310A7310	DMA47_XMOD	DMA47 Inner Loop Address Increment Register	0x00000000
0x310A7314	DMA47_YCNT	DMA47 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7318	DMA47_YMOD	DMA47 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A7324	DMA47_DSCPTR_CUR	DMA47 Current Descriptor Pointer Register	0x00000000
0x310A7328	DMA47_DSCPTR_PRV	DMA47 Previous Initial Descriptor Pointer Register	0x00000000
0x310A732C	DMA47_ADDR_CUR	DMA47 Current Address Register	0x00000000
0x310A7330	DMA47_STAT	DMA47 Status Register	0x00006000
0x310A7334	DMA47_XCNT_CUR	DMA47 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A7338	DMA47_YCNT_CUR	DMA47 Current Row Count (2D only) Register	0x00000000
0x310A7340	DMA47_BWLCNT	DMA47 Bandwidth Limit Count Register	0x00000000
0x310A7344	DMA47_BWLCNT_CUR	DMA47 Bandwidth Limit Count Current Register	0x00000000
0x310A7348	DMA47_BWMCNT	DMA47 Bandwidth Monitor Count Register	0x00000000
0x310A734C	DMA47_BWMCNT_CUR	DMA47 Bandwidth Monitor Count Current Register	0x00000000

Table A-68: ADSP-SC59x DMA48 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7380	DMA48_DSCPTR_NXT	DMA48 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7384	DMA48_ADDRSTART	DMA48 Start Address of Current Buffer Register	0x00000000
0x310A7388	DMA48_CFG	DMA48 Configuration Register	0x00000000
0x310A738C	DMA48_XCNT	DMA48 Inner Loop Count Start Value Register	0x00000000
0x310A7390	DMA48_XMOD	DMA48 Inner Loop Address Increment Register	0x00000000
0x310A7394	DMA48_YCNT	DMA48 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7398	DMA48_YMOD	DMA48 Outer Loop Address Increment (2D only) Register	0x00000000

Table A-68: ADSP-SC59x DMA48 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A73A4	DMA48_DSCPTR_CUR	DMA48 Current Descriptor Pointer Register	0x00000000
0x310A73A8	DMA48_DSCPTR_PRV	DMA48 Previous Initial Descriptor Pointer Register	0x00000000
0x310A73AC	DMA48_ADDR_CUR	DMA48 Current Address Register	0x00000000
0x310A73B0	DMA48_STAT	DMA48 Status Register	0x00006000
0x310A73B4	DMA48_XCNT_CUR	DMA48 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A73B8	DMA48_YCNT_CUR	DMA48 Current Row Count (2D only) Register	0x00000000
0x310A73C0	DMA48_BWLCNT	DMA48 Bandwidth Limit Count Register	0x00000000
0x310A73C4	DMA48_BWLCNT_CUR	DMA48 Bandwidth Limit Count Current Register	0x00000000
0x310A73C8	DMA48_BWMCNT	DMA48 Bandwidth Monitor Count Register	0x00000000
0x310A73CC	DMA48_BWMCNT_CUR	DMA48 Bandwidth Monitor Count Current Register	0x00000000

Table A-69: ADSP-SC59x DMA49 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AC000	DMA49_DSCPTR_NXT	DMA49 Pointer to Next Initial Descriptor Register	0x00000000
0x310AC004	DMA49_ADDRSTART	DMA49 Start Address of Current Buffer Register	0x00000000
0x310AC008	DMA49_CFG	DMA49 Configuration Register	0x00000220
0x310AC00C	DMA49_XCNT	DMA49 Inner Loop Count Start Value Register	0x00000000
0x310AC010	DMA49_XMOD	DMA49 Inner Loop Address Increment Register	0x00000000
0x310AC014	DMA49_YCNT	DMA49 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310AC018	DMA49_YMOD	DMA49 Outer Loop Address Increment (2D only) Register	0x00000000
0x310AC024	DMA49_DSCPTR_CUR	DMA49 Current Descriptor Pointer Register	0x00000000
0x310AC028	DMA49_DSCPTR_PRV	DMA49 Previous Initial Descriptor Pointer Register	0x00000000
0x310AC02C	DMA49_ADDR_CUR	DMA49 Current Address Register	0x00000000
0x310AC030	DMA49_STAT	DMA49 Status Register	0x00006000
0x310AC034	DMA49_XCNT_CUR	DMA49 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310AC038	DMA49_YCNT_CUR	DMA49 Current Row Count (2D only) Register	0x00000000
0x310AC040	DMA49_BWLCNT	DMA49 Bandwidth Limit Count Register	0x00000000
0x310AC044	DMA49_BWLCNT_CUR	DMA49 Bandwidth Limit Count Current Register	0x00000000

Table A-69: ADSP-SC59x DMA49 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310AC048	DMA49_BWMCNT	DMA49 Bandwidth Monitor Count Register	0x00000000
0x310AC04C	DMA49_BWMCNT_CUR	DMA49 Bandwidth Monitor Count Current Register	0x00000000

Table A-70: ADSP-SC59x DMA5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022280	DMA5_DSCPTR_NXT	DMA5 Pointer to Next Initial Descriptor Register	0x00000000
0x31022284	DMA5_ADDRSTART	DMA5 Start Address of Current Buffer Register	0x00000000
0x31022288	DMA5_CFG	DMA5 Configuration Register	0x00000000
0x3102228C	DMA5_XCNT	DMA5 Inner Loop Count Start Value Register	0x00000000
0x31022290	DMA5_XMOD	DMA5 Inner Loop Address Increment Register	0x00000000
0x31022294	DMA5_YCNT	DMA5 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022298	DMA5_YMOD	DMA5 Outer Loop Address Increment (2D only) Register	0x00000000
0x310222A4	DMA5_DSCPTR_CUR	DMA5 Current Descriptor Pointer Register	0x00000000
0x310222A8	DMA5_DSCPTR_PRV	DMA5 Previous Initial Descriptor Pointer Register	0x00000000
0x310222AC	DMA5_ADDR_CUR	DMA5 Current Address Register	0x00000000
0x310222B0	DMA5_STAT	DMA5 Status Register	0x00006000
0x310222B4	DMA5_XCNT_CUR	DMA5 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310222B8	DMA5_YCNT_CUR	DMA5 Current Row Count (2D only) Register	0x00000000
0x310222C0	DMA5_BWLCNT	DMA5 Bandwidth Limit Count Register	0x00000000
0x310222C4	DMA5_BWLCNT_CUR	DMA5 Bandwidth Limit Count Current Register	0x00000000
0x310222C8	DMA5_BWMCNT	DMA5 Bandwidth Monitor Count Register	0x00000000
0x310222CC	DMA5_BWMCNT_CUR	DMA5 Bandwidth Monitor Count Current Register	0x00000000

Table A-71: ADSP-SC59x DMA50 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310AC080	DMA50_DSCPTR_NXT	DMA50 Pointer to Next Initial Descriptor Register	0x00000000
0x310AC084	DMA50_ADDRSTART	DMA50 Start Address of Current Buffer Register	0x00000000
0x310AC088	DMA50_CFG	DMA50 Configuration Register	0x00000222
0x310AC08C	DMA50_XCNT	DMA50 Inner Loop Count Start Value Register	0x00000000

Table A-71: ADSP-SC59x DMA50 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310AC090	DMA50_XMOD	DMA50 Inner Loop Address Increment Register	0x00000000
0x310AC094	DMA50_YCNT	DMA50 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310AC098	DMA50_YMOD	DMA50 Outer Loop Address Increment (2D only) Register	0x00000000
0x310AC0A4	DMA50_DSCPTR_CUR	DMA50 Current Descriptor Pointer Register	0x00000000
0x310AC0A8	DMA50_DSCPTR_PRV	DMA50 Previous Initial Descriptor Pointer Register	0x00000000
0x310AC0AC	DMA50_ADDR_CUR	DMA50 Current Address Register	0x00000000
0x310AC0B0	DMA50_STAT	DMA50 Status Register	0x00006000
0x310AC0B4	DMA50_XCNT_CUR	DMA50 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310AC0B8	DMA50_YCNT_CUR	DMA50 Current Row Count (2D only) Register	0x00000000
0x310AC0C0	DMA50_BWLCNT	DMA50 Bandwidth Limit Count Register	0x00000000
0x310AC0C4	DMA50_BWLCNT_CUR	DMA50 Bandwidth Limit Count Current Register	0x00000000
0x310AC0C8	DMA50_BWMCNT	DMA50 Bandwidth Monitor Count Register	0x00000000
0x310AC0CC	DMA50_BWMCNT_CUR	DMA50 Bandwidth Monitor Count Current Register	0x00000000

Table A-72: ADSP-SC59x DMA51 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109C000	DMA51_DSCPTR_NXT	DMA51 Pointer to Next Initial Descriptor Register	0x00000000
0x3109C004	DMA51_ADDRSTART	DMA51 Start Address of Current Buffer Register	0x00000000
0x3109C008	DMA51_CFG	DMA51 Configuration Register	0x00000220
0x3109C00C	DMA51_XCNT	DMA51 Inner Loop Count Start Value Register	0x00000000
0x3109C010	DMA51_XMOD	DMA51 Inner Loop Address Increment Register	0x00000000
0x3109C014	DMA51_YCNT	DMA51 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109C018	DMA51_YMOD	DMA51 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109C024	DMA51_DSCPTR_CUR	DMA51 Current Descriptor Pointer Register	0x00000000
0x3109C028	DMA51_DSCPTR_PRV	DMA51 Previous Initial Descriptor Pointer Register	0x00000000
0x3109C02C	DMA51_ADDR_CUR	DMA51 Current Address Register	0x00000000
0x3109C030	DMA51_STAT	DMA51 Status Register	0x0000B000

Table A-72: ADSP-SC59x DMA51 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109C034	DMA51_XCNT_CUR	DMA51 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109C038	DMA51_YCNT_CUR	DMA51 Current Row Count (2D only) Register	0x00000000
0x3109C040	DMA51_BWLCNT	DMA51 Bandwidth Limit Count Register	0x00000000
0x3109C044	DMA51_BWLCNT_CUR	DMA51 Bandwidth Limit Count Current Register	0x00000000
0x3109C048	DMA51_BWMCNT	DMA51 Bandwidth Monitor Count Register	0x00000000
0x3109C04C	DMA51_BWMCNT_CUR	DMA51 Bandwidth Monitor Count Current Register	0x00000000

Table A-73: ADSP-SC59x DMA52 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109C080	DMA52_DSCPTR_NXT	DMA52 Pointer to Next Initial Descriptor Register	0x00000000
0x3109C084	DMA52_ADDRSTART	DMA52 Start Address of Current Buffer Register	0x00000000
0x3109C088	DMA52_CFG	DMA52 Configuration Register	0x00000222
0x3109C08C	DMA52_XCNT	DMA52 Inner Loop Count Start Value Register	0x00000000
0x3109C090	DMA52_XMOD	DMA52 Inner Loop Address Increment Register	0x00000000
0x3109C094	DMA52_YCNT	DMA52 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3109C098	DMA52_YMOD	DMA52 Outer Loop Address Increment (2D only) Register	0x00000000
0x3109C0A4	DMA52_DSCPTR_CUR	DMA52 Current Descriptor Pointer Register	0x00000000
0x3109C0A8	DMA52_DSCPTR_PRV	DMA52 Previous Initial Descriptor Pointer Register	0x00000000
0x3109C0AC	DMA52_ADDR_CUR	DMA52 Current Address Register	0x00000000
0x3109C0B0	DMA52_STAT	DMA52 Status Register	0x0000B000
0x3109C0B4	DMA52_XCNT_CUR	DMA52 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3109C0B8	DMA52_YCNT_CUR	DMA52 Current Row Count (2D only) Register	0x00000000
0x3109C0C0	DMA52_BWLCNT	DMA52 Bandwidth Limit Count Register	0x00000000
0x3109C0C4	DMA52_BWLCNT_CUR	DMA52 Bandwidth Limit Count Current Register	0x00000000
0x3109C0C8	DMA52_BWMCNT	DMA52 Bandwidth Monitor Count Register	0x00000000
0x3109C0CC	DMA52_BWMCNT_CUR	DMA52 Bandwidth Monitor Count Current Register	0x00000000

Table A-74: ADSP-SC59x DMA53 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026380	DMA53_DSCPTR_NXT	DMA53 Pointer to Next Initial Descriptor Register	0x00000000
0x31026384	DMA53_ADDRSTART	DMA53 Start Address of Current Buffer Register	0x00000000
0x31026388	DMA53_CFG	DMA53 Configuration Register	0x00000000
0x3102638C	DMA53_XCNT	DMA53 Inner Loop Count Start Value Register	0x00000000
0x31026390	DMA53_XMOD	DMA53 Inner Loop Address Increment Register	0x00000000
0x31026394	DMA53_YCNT	DMA53 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026398	DMA53_YMOD	DMA53 Outer Loop Address Increment (2D only) Register	0x00000000
0x310263A4	DMA53_DSCPTR_CUR	DMA53 Current Descriptor Pointer Register	0x00000000
0x310263A8	DMA53_DSCPTR_PRV	DMA53 Previous Initial Descriptor Pointer Register	0x00000000
0x310263AC	DMA53_ADDR_CUR	DMA53 Current Address Register	0x00000000
0x310263B0	DMA53_STAT	DMA53 Status Register	0x00006000
0x310263B4	DMA53_XCNT_CUR	DMA53 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310263B8	DMA53_YCNT_CUR	DMA53 Current Row Count (2D only) Register	0x00000000
0x310263C0	DMA53_BWLCNT	DMA53 Bandwidth Limit Count Register	0x00000000
0x310263C4	DMA53_BWLCNT_CUR	DMA53 Bandwidth Limit Count Current Register	0x00000000
0x310263C8	DMA53_BWMCNT	DMA53 Bandwidth Monitor Count Register	0x00000000
0x310263CC	DMA53_BWMCNT_CUR	DMA53 Bandwidth Monitor Count Current Register	0x00000000

Table A-75: ADSP-SC59x DMA54 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31026300	DMA54_DSCPTR_NXT	DMA54 Pointer to Next Initial Descriptor Register	0x00000000
0x31026304	DMA54_ADDRSTART	DMA54 Start Address of Current Buffer Register	0x00000000
0x31026308	DMA54_CFG	DMA54 Configuration Register	0x00000000
0x3102630C	DMA54_XCNT	DMA54 Inner Loop Count Start Value Register	0x00000000
0x31026310	DMA54_XMOD	DMA54 Inner Loop Address Increment Register	0x00000000
0x31026314	DMA54_YCNT	DMA54 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31026318	DMA54_YMOD	DMA54 Outer Loop Address Increment (2D only) Register	0x00000000

Table A-75: ADSP-SC59x DMA54 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31026324	DMA54_DSCPTR_CUR	DMA54 Current Descriptor Pointer Register	0x00000000
0x31026328	DMA54_DSCPTR_PRV	DMA54 Previous Initial Descriptor Pointer Register	0x00000000
0x3102632C	DMA54_ADDR_CUR	DMA54 Current Address Register	0x00000000
0x31026330	DMA54_STAT	DMA54 Status Register	0x00006000
0x31026334	DMA54_XCNT_CUR	DMA54 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31026338	DMA54_YCNT_CUR	DMA54 Current Row Count (2D only) Register	0x00000000
0x31026340	DMA54_BWLCNT	DMA54 Bandwidth Limit Count Register	0x00000000
0x31026344	DMA54_BWLCNT_CUR	DMA54 Bandwidth Limit Count Current Register	0x00000000
0x31026348	DMA54_BWMCNT	DMA54 Bandwidth Monitor Count Register	0x00000000
0x3102634C	DMA54_BWMCNT_CUR	DMA54 Bandwidth Monitor Count Current Register	0x00000000

Table A-76: ADSP-SC59x DMA55 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D300	DMA55_DSCPTR_NXT	DMA55 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D304	DMA55_ADDRSTART	DMA55 Start Address of Current Buffer Register	0x00000000
0x3102D308	DMA55_CFG	DMA55 Configuration Register	0x00000000
0x3102D30C	DMA55_XCNT	DMA55 Inner Loop Count Start Value Register	0x00000000
0x3102D310	DMA55_XMOD	DMA55 Inner Loop Address Increment Register	0x00000000
0x3102D314	DMA55_YCNT	DMA55 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D318	DMA55_YMOD	DMA55 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D324	DMA55_DSCPTR_CUR	DMA55 Current Descriptor Pointer Register	0x00000000
0x3102D328	DMA55_DSCPTR_PRV	DMA55 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D32C	DMA55_ADDR_CUR	DMA55 Current Address Register	0x00000000
0x3102D330	DMA55_STAT	DMA55 Status Register	0x00006000
0x3102D334	DMA55_XCNT_CUR	DMA55 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D338	DMA55_YCNT_CUR	DMA55 Current Row Count (2D only) Register	0x00000000
0x3102D340	DMA55_BWLCNT	DMA55 Bandwidth Limit Count Register	0x00000000
0x3102D344	DMA55_BWLCNT_CUR	DMA55 Bandwidth Limit Count Current Register	0x00000000

Table A-76: ADSP-SC59x DMA55 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D348	DMA55_BWMCNT	DMA55 Bandwidth Monitor Count Register	0x00000000
0x3102D34C	DMA55_BWMCNT_CUR	DMA55 Bandwidth Monitor Count Current Register	0x00000000

Table A-77: ADSP-SC59x DMA56 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102D380	DMA56_DSCPTR_NXT	DMA56 Pointer to Next Initial Descriptor Register	0x00000000
0x3102D384	DMA56_ADDRSTART	DMA56 Start Address of Current Buffer Register	0x00000000
0x3102D388	DMA56_CFG	DMA56 Configuration Register	0x00000000
0x3102D38C	DMA56_XCNT	DMA56 Inner Loop Count Start Value Register	0x00000000
0x3102D390	DMA56_XMOD	DMA56 Inner Loop Address Increment Register	0x00000000
0x3102D394	DMA56_YCNT	DMA56 Outer Loop Count Start Value (2D only) Register	0x00000000
0x3102D398	DMA56_YMOD	DMA56 Outer Loop Address Increment (2D only) Register	0x00000000
0x3102D3A4	DMA56_DSCPTR_CUR	DMA56 Current Descriptor Pointer Register	0x00000000
0x3102D3A8	DMA56_DSCPTR_PRV	DMA56 Previous Initial Descriptor Pointer Register	0x00000000
0x3102D3AC	DMA56_ADDR_CUR	DMA56 Current Address Register	0x00000000
0x3102D3B0	DMA56_STAT	DMA56 Status Register	0x00006000
0x3102D3B4	DMA56_XCNT_CUR	DMA56 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x3102D3B8	DMA56_YCNT_CUR	DMA56 Current Row Count (2D only) Register	0x00000000
0x3102D3C0	DMA56_BWLCNT	DMA56 Bandwidth Limit Count Register	0x00000000
0x3102D3C4	DMA56_BWLCNT_CUR	DMA56 Bandwidth Limit Count Current Register	0x00000000
0x3102D3C8	DMA56_BWMCNT	DMA56 Bandwidth Monitor Count Register	0x00000000
0x3102D3CC	DMA56_BWMCNT_CUR	DMA56 Bandwidth Monitor Count Current Register	0x00000000

Table A-78: ADSP-SC59x DMA6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022300	DMA6_DSCPTR_NXT	DMA6 Pointer to Next Initial Descriptor Register	0x00000000
0x31022304	DMA6_ADDRSTART	DMA6 Start Address of Current Buffer Register	0x00000000
0x31022308	DMA6_CFG	DMA6 Configuration Register	0x00000000

Table A-78: ADSP-SC59x DMA6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102230C	DMA6_XCNT	DMA6 Inner Loop Count Start Value Register	0x00000000
0x31022310	DMA6_XMOD	DMA6 Inner Loop Address Increment Register	0x00000000
0x31022314	DMA6_YCNT	DMA6 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022318	DMA6_YMOD	DMA6 Outer Loop Address Increment (2D only) Register	0x00000000
0x31022324	DMA6_DSCPTR_CUR	DMA6 Current Descriptor Pointer Register	0x00000000
0x31022328	DMA6_DSCPTR_PRV	DMA6 Previous Initial Descriptor Pointer Register	0x00000000
0x3102232C	DMA6_ADDR_CUR	DMA6 Current Address Register	0x00000000
0x31022330	DMA6_STAT	DMA6 Status Register	0x00006000
0x31022334	DMA6_XCNT_CUR	DMA6 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x31022338	DMA6_YCNT_CUR	DMA6 Current Row Count (2D only) Register	0x00000000
0x31022340	DMA6_BWLCNT	DMA6 Bandwidth Limit Count Register	0x00000000
0x31022344	DMA6_BWLCNT_CUR	DMA6 Bandwidth Limit Count Current Register	0x00000000
0x31022348	DMA6_BWMCNT	DMA6 Bandwidth Monitor Count Register	0x00000000
0x3102234C	DMA6_BWMCNT_CUR	DMA6 Bandwidth Monitor Count Current Register	0x00000000

Table A-79: ADSP-SC59x DMA7 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31022380	DMA7_DSCPTR_NXT	DMA7 Pointer to Next Initial Descriptor Register	0x00000000
0x31022384	DMA7_ADDRSTART	DMA7 Start Address of Current Buffer Register	0x00000000
0x31022388	DMA7_CFG	DMA7 Configuration Register	0x00000000
0x3102238C	DMA7_XCNT	DMA7 Inner Loop Count Start Value Register	0x00000000
0x31022390	DMA7_XMOD	DMA7 Inner Loop Address Increment Register	0x00000000
0x31022394	DMA7_YCNT	DMA7 Outer Loop Count Start Value (2D only) Register	0x00000000
0x31022398	DMA7_YMOD	DMA7 Outer Loop Address Increment (2D only) Register	0x00000000
0x310223A4	DMA7_DSCPTR_CUR	DMA7 Current Descriptor Pointer Register	0x00000000
0x310223A8	DMA7_DSCPTR_PRV	DMA7 Previous Initial Descriptor Pointer Register	0x00000000
0x310223AC	DMA7_ADDR_CUR	DMA7 Current Address Register	0x00000000
0x310223B0	DMA7_STAT	DMA7 Status Register	0x00006000
0x310223B4	DMA7_XCNT_CUR	DMA7 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000

Table A-79: ADSP-SC59x DMA7 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310223B8	DMA7_YCNT_CUR	DMA7 Current Row Count (2D only) Register	0x00000000
0x310223C0	DMA7_BWLCNT	DMA7 Bandwidth Limit Count Register	0x00000000
0x310223C4	DMA7_BWLCNT_CUR	DMA7 Bandwidth Limit Count Current Register	0x00000000
0x310223C8	DMA7_BWMCNT	DMA7 Bandwidth Monitor Count Register	0x00000000
0x310223CC	DMA7_BWMCNT_CUR	DMA7 Bandwidth Monitor Count Current Register	0x00000000

Table A-80: ADSP-SC59x DMA8 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7000	DMA8_DSCPTR_NXT	DMA8 Pointer to Next Initial Descriptor Register	0x00000000
0x310A7004	DMA8_ADDRSTART	DMA8 Start Address of Current Buffer Register	0x00000000
0x310A7008	DMA8_CFG	DMA8 Configuration Register	0x00000000
0x310A700C	DMA8_XCNT	DMA8 Inner Loop Count Start Value Register	0x00000000
0x310A7010	DMA8_XMOD	DMA8 Inner Loop Address Increment Register	0x00000000
0x310A7014	DMA8_YCNT	DMA8 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7018	DMA8_YMOD	DMA8 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A7024	DMA8_DSCPTR_CUR	DMA8 Current Descriptor Pointer Register	0x00000000
0x310A7028	DMA8_DSCPTR_PRV	DMA8 Previous Initial Descriptor Pointer Register	0x00000000
0x310A702C	DMA8_ADDR_CUR	DMA8 Current Address Register	0x00000000
0x310A7030	DMA8_STAT	DMA8 Status Register	0x00006000
0x310A7034	DMA8_XCNT_CUR	DMA8 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A7038	DMA8_YCNT_CUR	DMA8 Current Row Count (2D only) Register	0x00000000
0x310A7040	DMA8_BWLCNT	DMA8 Bandwidth Limit Count Register	0x00000000
0x310A7044	DMA8_BWLCNT_CUR	DMA8 Bandwidth Limit Count Current Register	0x00000000
0x310A7048	DMA8_BWMCNT	DMA8 Bandwidth Monitor Count Register	0x00000000
0x310A704C	DMA8_BWMCNT_CUR	DMA8 Bandwidth Monitor Count Current Register	0x00000000

Table A-81: ADSP-SC59x DMA9 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7080	DMA9_DSCPTR_NXT	DMA9 Pointer to Next Initial Descriptor Register	0x00000000

Table A-81: ADSP-SC59x DMA9 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A7084	DMA9_ADDRSTART	DMA9 Start Address of Current Buffer Register	0x00000000
0x310A7088	DMA9_CFG	DMA9 Configuration Register	0x00000000
0x310A708C	DMA9_XCNT	DMA9 Inner Loop Count Start Value Register	0x00000000
0x310A7090	DMA9_XMOD	DMA9 Inner Loop Address Increment Register	0x00000000
0x310A7094	DMA9_YCNT	DMA9 Outer Loop Count Start Value (2D only) Register	0x00000000
0x310A7098	DMA9_YMOD	DMA9 Outer Loop Address Increment (2D only) Register	0x00000000
0x310A70A4	DMA9_DSCPTR_CUR	DMA9 Current Descriptor Pointer Register	0x00000000
0x310A70A8	DMA9_DSCPTR_PRV	DMA9 Previous Initial Descriptor Pointer Register	0x00000000
0x310A70AC	DMA9_ADDR_CUR	DMA9 Current Address Register	0x00000000
0x310A70B0	DMA9_STAT	DMA9 Status Register	0x00006000
0x310A70B4	DMA9_XCNT_CUR	DMA9 Current Count (1D) or Intra-row XCNT (2D) Register	0x00000000
0x310A70B8	DMA9_YCNT_CUR	DMA9 Current Row Count (2D only) Register	0x00000000
0x310A70C0	DMA9_BWLCNT	DMA9 Bandwidth Limit Count Register	0x00000000
0x310A70C4	DMA9_BWLCNT_CUR	DMA9 Bandwidth Limit Count Current Register	0x00000000
0x310A70C8	DMA9_BWMCNT	DMA9 Bandwidth Monitor Count Register	0x00000000
0x310A70CC	DMA9_BWMCNT_CUR	DMA9 Bandwidth Monitor Count Current Register	0x00000000

Table A-82: ADSP-SC59x DMC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31070004	DMC0_CTL	DMC0 Control Register	0x00000000
0x31070008	DMC0_STAT	DMC0 Status Register	0x00000001
0x3107000C	DMC0_EFFCTL	DMC0 Efficiency Control Register	0x00440000
0x31070010	DMC0_PRIO	DMC0 Priority ID Register 1	0x00000000
0x31070014	DMC0_PRIOMSK	DMC0 Priority ID Mask Register 1	0x00000000
0x31070018	DMC0_PRIO2	DMC0 Priority ID Register 2	0x00000000
0x3107001C	DMC0_PRIOMSK2	DMC0 Priority ID Mask Register 2	0x00000000
0x31070040	DMC0_CFG	DMC0 Configuration Register	0x00000000
0x31070044	DMC0_TR0	DMC0 Timing 0 Register	0x00000000
0x31070048	DMC0_TR1	DMC0 Timing 1 Register	0x00000000
0x3107004C	DMC0_TR2	DMC0 Timing 2 Register	0x00000000

Table A-82: ADSP-SC59x DMC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3107005C	DMC0_MSK	DMC0 Mask (Mode Register Shadow) Register	0x00000000
0x31070060	DMC0_MR	DMC0 Shadow MR0 Register (DDR3)	0x00000000
0x31070064	DMC0_MR1	DMC0 Shadow MR1 Register (DDR3)	0x00000000
0x31070068	DMC0_MR2	DMC0 Shadow MR2 Register (DDR3)	0x00000000
0x3107006C	DMC0_EMR3	DMC0 Shadow EMR3 Register	0x00000000
0x31070080	DMC0_DLLCTL	DMC0 DLL Control Register	0x0000054B
0x31070090	DMC0_DT_CALIB_ADDR	DMC0 Data Calibration Address Register	0x00000000
0x31070094	DMC0_DT_DATA_CALIB_DATA0	DMC0 Data Calibration Data 0 Register	0x00000000
0x31070098	DMC0_DT_DATA_CALIB_DATA1	DMC0 Data Calibration Data 1 Register	0xFFFFFFFF
0x31070100	DMC0_RDDATABUFID1	DMC0 DMC Read Data Buffer ID Register 1	0x00000000
0x31070104	DMC0_RDDATABUFMSK1	DMC0 DMC Read Data Buffer Mask Register 1	0x00000000
0x31070108	DMC0_RDDATABUFID2	DMC0 DMC Read Data Buffer ID Register 2	0x00000000
0x3107010C	DMC0_RDDATABUFMSK2	DMC0 DMC Read Data Buffer Mask Register 2	0x00000000
0x31070140	DMC0_PM_CTL	DMC0 DMC Performance Control Register	0x00000000
0x31070144	DMC0_PM_IDPG_CNT	DMC0 Performance Monitor Idle Page Count Register	0x00000000
0x31070148	DMC0_PM_PGMS_CNT	DMC0 Performance Monitor Page Miss Count	0x00000000
0x3107014C	DMC0_PM_PGHT_CNT	DMC0 Performance Monitor Page Hit Count	0x00000000
0x31070150	DMC0_PM_IDCYDQ_CNT	DMC0 Performance Monitor Idle Cycle DQ Count Register	0x00000000
0x31070154	DMC0_PM_RD2WRTA_CNT	DMC0 Performance Monitor Read to Write Turn Around Count Register	0x00000000
0x31070158	DMC0_PM_WR2RDTA_CNT	DMC0 Performance Monitor Write to Read Turn Around Count Register	0x00000000
0x3107015C	DMC0_PM_REF_CNT	DMC0 Performance Monitor Refresh Count Register	0x00000000
0x31070164	DMC0_SCB_PM_IDWRDCY_CNT	DMC0 SCB Performance Monitor Idle Cycle Count Register	0x00000000
0x31070168	DMC0_SCB_PM_IDRDDCY_CNT	DMC0 SCB Performance Monitor Idle Cycle Count Register	0x00000000
0x31070174	DMC0_RDD_BUFHT_CNT	DMC0 Performance Monitor Buffer Hit Count Register	0x00000000
0x31070178	DMC0_SCB_PM_BRLN_CNT	DMC0 SCB Performance AXI Transfer Count Register	0x00000000

Table A-82: ADSP-SC59x DMC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3107017C	DMC0_SCB_PM_BRSZ_CNT	DMC0 SCB Performance AXI Transfer Count Register	0x00000000

Table A-83: ADSP-SC59x DPM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31090000	DPM0_CTL	DPM0 Control Register	0x00000000
0x31090004	DPM0_STAT	DPM0 Status Register	0x00000001
0x31090070	DPM0_PER_DIS0	DPM0 Peripherals Disable Register 0	0x00000000
0x31090074	DPM0_PER_DIS1	DPM0 Peripherals Disable Register 1	0x00000000
0x31090084	DPM0_REVID	DPM0 Revision ID	0x00000020

Table A-84: ADSP-SC59x EMDMA0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310E002C	EMDMA0_CTL	EMDMA0 External Memory DMA Control Register	0x00000000
0x310E002C	EMDMA_CTL	EMDMA External Memory DMA Control Register	0x00000000
0x310E0080	EMDMA0_INDX1	EMDMA0 External Index Register	0x00000000
0x310E0080	EMDMA_INDX1	EMDMA External Index Register	0x00000000
0x310E0084	EMDMA0_MOD1	EMDMA0 External Modifier Register	0x00000000
0x310E0084	EMDMA_MOD1	EMDMA External Modifier Register	0x00000000
0x310E0088	EMDMA0_CNT1	EMDMA0 External Count Register	0x00000000
0x310E0088	EMDMA_CNT1	EMDMA External Count Register	0x00000000
0x310E008C	EMDMA0_INDX0	EMDMA0 Internal Index Register	0x00000000
0x310E008C	EMDMA_INDX0	EMDMA Internal Index Register	0x00000000
0x310E0090	EMDMA0_MOD0	EMDMA0 Internal Modifier Register	0x00000000
0x310E0090	EMDMA_MOD0	EMDMA Internal Modifier Register	0x00000000
0x310E0094	EMDMA0_CNT0	EMDMA0 Internal Count Register	0x00000000
0x310E0094	EMDMA_CNT0	EMDMA Internal Count Register	0x00000000
0x310E0098	EMDMA0_CHNPTR	EMDMA0 Chain Pointer Register	0x00000000
0x310E0098	EMDMA_CHNPTR	EMDMA Chain Pointer Register	0x00000000
0x310E009C	EMDMA0_BASE	EMDMA0 External Base Address Register	0x00000000

Table A-84: ADSP-SC59x EMDMA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310E009C	EMDMA_BASE	EMDMA External Base Address Register	0x00000000
0x310E00A0	EMDMA0_TPTR	EMDMA0 Tap List Pointer Register	0x00000000
0x310E00A0	EMDMA_TPTR	EMDMA Tap List Pointer Register	0x00000000
0x310E00A4	EMDMA0_BUFLEN	EMDMA0 Circular Buffer Length Register	0x00000000
0x310E00A4	EMDMA_BUFLEN	EMDMA Circular Buffer Length Register	0x00000000
0x310E00AC	EMDMA0_TCNT	EMDMA0 Delay Line Tap Count Register	0x00000000
0x310E00AC	EMDMA_TCNT	EMDMA Delay Line Tap Count Register	0x00000000

Table A-85: ADSP-SC59x EMDMA1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310E0030	EMDMA1_CTL	EMDMA1 External Memory DMA Control Register	0x00000000
0x310E00C0	EMDMA1_INDX1	EMDMA1 External Index Register	0x00000000
0x310E00C4	EMDMA1_MOD1	EMDMA1 External Modifier Register	0x00000000
0x310E00C8	EMDMA1_CNT1	EMDMA1 External Count Register	0x00000000
0x310E00CC	EMDMA1_INDX0	EMDMA1 Internal Index Register	0x00000000
0x310E00D0	EMDMA1_MOD0	EMDMA1 Internal Modifier Register	0x00000000
0x310E00D4	EMDMA1_CNT0	EMDMA1 Internal Count Register	0x00000000
0x310E00D8	EMDMA1_CHNPTR	EMDMA1 Chain Pointer Register	0x00000000
0x310E00DC	EMDMA1_BASE	EMDMA1 External Base Address Register	0x00000000
0x310E00E0	EMDMA1_TPTR	EMDMA1 Tap List Pointer Register	0x00000000
0x310E00E4	EMDMA1_BUFLEN	EMDMA1 Circular Buffer Length Register	0x00000000
0x310E00EC	EMDMA1_TCNT	EMDMA1 Delay Line Tap Count Register	0x00000000

Table A-86: ADSP-SC59x EPPI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31029000	EPPI0_STAT	EPPI0 Status Register	0x00000000
0x31029004	EPPI0_HCNT	EPPI0 Horizontal Transfer Count Register	0x00000000
0x31029008	EPPI0_HDLY	EPPI0 Horizontal Delay Count Register	0x00000000
0x3102900C	EPPI0_VCNT	EPPI0 Vertical Transfer Count Register	0x00000000
0x31029010	EPPI0_VDLY	EPPI0 Vertical Delay Count Register	0x00000000

Table A-86: ADSP-SC59x EPPI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31029014	EPPI0_FRAME	EPPI0 Lines Per Frame Register	0x00000000
0x31029018	EPPI0_LINE	EPPI0 Samples Per Line Register	0x00000000
0x3102901C	EPPI0_CLKDIV	EPPI0 Clock Divide Register	0x00000000
0x31029020	EPPI0_CTL	EPPI0 Control Register	0x00000000
0x31029024	EPPI0_FS1_WLHB	EPPI0 FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register	0x00000000
0x31029028	EPPI0_FS1_PASPL	EPPI0 FS1 Period Register / EPPI Active Samples Per Line Register	0x00000000
0x3102902C	EPPI0_FS2_WLVB	EPPI0 FS2 Width Register / EPPI Lines Of Vertical Blanking Register	0x00000000
0x31029030	EPPI0_FS2_PALPF	EPPI0 FS2 Period Register / EPPI Active Lines Per Field Register	0x00000000
0x31029034	EPPI0_IMSK	EPPI0 Interrupt Mask Register	0x00000000
0x3102903C	EPPI0_ODDCLIP	EPPI0 Clipping Register for ODD (Chroma) Data Register	0xFFFF0000
0x31029040	EPPI0_EVENCLIP	EPPI0 Clipping Register for EVEN (Luma) Data Register	0xFFFF0000
0x31029044	EPPI0_FS1_DLY	EPPI0 Frame Sync 1 Delay Value Register	0x00000000
0x31029048	EPPI0_FS2_DLY	EPPI0 Frame Sync 2 Delay Value Register	0x00000000
0x3102904C	EPPI0_CTL2	EPPI0 Control Register 2 Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31040000	EQOS_MAC_CFG	EQOS MAC Configuration Register	0x00000000
0x31040000	EQOS_Q[i]_TXFLOW_CTL	EQOS Queue [i] Transmit Flow Control Register	0x00000000
0x31040000	EMAC0_MAC_CFG	EMAC0 MAC Configuration Register	0x00000000
0x31040004	EQOS_MAC_EXT_CFG	EQOS MAC Extended Configuration Register	0x00000000
0x31040004	EQOS_Q[i]_TXFLOW_CTL	EQOS Queue [i] Transmit Flow Control Register	0x00000000
0x31040004	EMAC0_MAC_EXT_CFG	EMAC0 MAC Extended Configuration Register	0x00000000
0x31040008	EQOS_Q[i]_TXFLOW_CTL	EQOS Queue [i] Transmit Flow Control Register	0x00000000
0x31040008	EQOS_MACPKT_FILT	EQOS MAC Packet Filter Register	0x00000000
0x31040008	EMAC0_MACPKT_FILT	EMAC0 MAC Packet Filter Register	0x00000000
0x3104000C	EQOS_WDT_TMOUT	EQOS Watchdog Timeout Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3104000C	EQOS_Q[i]_TXFLOW_CTL	EQOS Queue [i] Transmit Flow Control Register	0x00000000
0x3104000C	EMAC0_WDT_TMOUT	EMAC0 Watchdog Timeout Register	0x00000000
0x31040010	EQOS_HASHTBL_REG0	EQOS Hash Table Register 0	0x00000000
0x31040010	EQOS_Q[i]_TXFLOW_CTL	EQOS Queue [i] Transmit Flow Control Register	0x00000000
0x31040010	EMAC0_HASHTBL_REG0	EMAC0 Hash Table Register 0	0x00000000
0x31040014	EQOS_HASHTBL_REG1	EQOS Hash Table Register 1	0x00000000
0x31040014	EQOS_Q[i]_TXFLOW_CTL	EQOS Queue [i] Transmit Flow Control Register	0x00000000
0x31040014	EMAC0_HASHTBL_REG1	EMAC0 Hash Table Register 1	0x00000000
0x31040018	EQOS_Q[i]_TXFLOW_CTL	EQOS Queue [i] Transmit Flow Control Register	0x00000000
0x31040018	EQOS_HASHTBL_REG2	EQOS Hash Table Register 2	0x00000000
0x31040018	EMAC0_HASHTBL_REG2	EMAC0 Hash Table Register 2	0x00000000
0x3104001C	EQOS_HASHTBL_REG3	EQOS Hash Table Register 3	0x00000000
0x3104001C	EMAC0_HASHTBL_REG3	EMAC0 Hash Table Register 3	0x00000000
0x31040020	EQOS_HASHTBL_REG4	EQOS Hash Table Register 4	0x00000000
0x31040020	EMAC0_HASHTBL_REG4	EMAC0 Hash Table Register 4	0x00000000
0x31040024	EQOS_HASHTBL_REG5	EQOS Hash Table Register 5	0x00000000
0x31040024	EMAC0_HASHTBL_REG5	EMAC0 Hash Table Register 5	0x00000000
0x31040028	EQOS_HASHTBL_REG6	EQOS Hash Table Register 6	0x00000000
0x31040028	EMAC0_HASHTBL_REG6	EMAC0 Hash Table Register 6	0x00000000
0x3104002C	EQOS_HASHTBL_REG7	EQOS Hash Table Register 7	0x00000000
0x3104002C	EMAC0_HASHTBL_REG7	EMAC0 Hash Table Register 7	0x00000000
0x31040050	EQOS_VLANTAG_CTL	EQOS VLAN Tag Control Register	0x00000000
0x31040050	EMAC0_VLANTAG_CTL	EMAC0 VLAN Tag Control Register	0x00000000
0x31040054	EQOS_VLANTAG_DAT	EQOS VLAN Tag Indirect Access Data Register	0x00000000
0x31040054	EMAC0_VLANTAG_DAT	EMAC0 VLAN Tag Indirect Access Data Register	0x00000000
0x31040058	EQOS_VLAN_HASHTBL	EQOS VLAN Hash Table Register	0x00000000
0x31040058	EMAC0_VLAN_HASHTBL	EMAC0 VLAN Hash Table Register	0x00000000
0x31040060	EQOS_VLAN_INCL	EQOS VLAN Tag Inclusion or Replacement Register	0x00000000
0x31040060	EMAC0_VLAN_INCL	EMAC0 VLAN Tag Inclusion or Replacement Register	0x00000000
0x31040064	EQOS_INVLAN_INCL	EQOS Inner VLAN Tag Inclusion or Replacement Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040064	EMAC0_INVLAN_INCL	EMAC0 Inner VLAN Tag Inclusion or Replacement Register	0x00000000
0x31040070	EQOS_Q0_TXFLOW_CTL	EQOS Queue 0 Transmit Flow Control Register	0x00000000
0x31040070	EMAC0_Q0_TXFLOW_CTL	EMAC0 Queue 0 Transmit Flow Control Register	0x00000000
0x31040074	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x31040078	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x3104007C	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x31040080	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x31040084	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x31040088	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x3104008C	EMAC0_Q[i]_TXFLOW_CTL	EMAC0 Queue [i] Transmit Flow Control Register	0x00000000
0x31040090	EQOS_RXFLOW_CTL	EQOS Receive Flow Control Register	0x00000000
0x31040090	EMAC0_RXFLOW_CTL	EMAC0 Receive Flow Control Register	0x00000000
0x31040094	EQOS_RXQ_CTL4	EQOS Receive Queue Control 4 Register	0x00000000
0x31040094	EMAC0_RXQ_CTL4	EMAC0 Receive Queue Control 4 Register	0x00000000
0x31040098	EQOS_TXQ_PRTY_MAP0	EQOS Transmit Queue Priority Mapping 0 Register	0x00000000
0x31040098	EMAC0_TXQ_PRTY_MAP0	EMAC0 Transmit Queue Priority Mapping 0 Register	0x00000000
0x3104009C	EQOS_TXQ_PRTY_MAP1	EQOS Transmit Queue Priority Mapping 1 Register	0x00000000
0x3104009C	EMAC0_TXQ_PRTY_MAP1	EMAC0 Transmit Queue Priority Mapping 1 Register	0x00000000
0x310400A0	EQOS_RXQ_CTL0	EQOS Receive Queue Control 0 Register	0x00000000
0x310400A0	EMAC0_RXQ_CTL0	EMAC0 Receive Queue Control 0 Register	0x00000000
0x310400A4	EQOS_RXQ_CTL1	EQOS Receive Queue Control 1 Register	0x00000000
0x310400A4	EMAC0_RXQ_CTL1	EMAC0 Receive Queue Control 1 Register	0x00000000
0x310400A8	EQOS_RXQ_CTL2	EQOS Receive Queue Control 2 Register	0x00000000
0x310400A8	EMAC0_RXQ_CTL2	EMAC0 Receive Queue Control 2 Register	0x00000000
0x310400AC	EQOS_RXQ_CTL3	EQOS Receive Queue Control 3 Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310400AC	EMAC0_RXQ_CTL3	EMAC0 Receive Queue Control 3 Register	0x00000000
0x310400B0	EQOS_MAC_ISTAT	EQOS MAC Interrupt Status Register	0x00000000
0x310400B0	EMAC0_MAC_ISTAT	EMAC0 MAC Interrupt Status Register	0x00000000
0x310400B4	EQOS_MAC_IEN	EQOS MAC Interrupt Enable Register	0x00000000
0x310400B4	EMAC0_MAC_IEN	EMAC0 MAC Interrupt Enable Register	0x00000000
0x310400B8	EQOS_RXTX_STAT	EQOS Receive Transmit Status Register	0x00000000
0x310400B8	EMAC0_RXTX_STAT	EMAC0 Receive Transmit Status Register	0x00000000
0x310400D0	EQOS_LPI_CTLSTAT	EQOS LPI Control and Status Register	0x00000000
0x310400D0	EMAC0_LPI_CTLSTAT	EMAC0 LPI Control and Status Register	0x00000000
0x310400D4	EQOS_LPI_TMRSCTL	EQOS LPI Timers Control Register	0x03E80000
0x310400D4	EMAC0_LPI_TMRSCTL	EMAC0 LPI Timers Control Register	0x03E80000
0x310400D8	EQOS_LPI_ENTR_TMR	EQOS Transmit LPI Entry Timer Register	0x00000000
0x310400D8	EMAC0_LPI_ENTR_TMR	EMAC0 Transmit LPI Entry Timer Register	0x00000000
0x310400DC	EQOS_ONEUS_TIC_CNT	EQOS LPI One Microsecond Tic Counter Register	0x00000063
0x310400DC	EMAC0_ONEUS_TIC_CNT	EMAC0 LPI One Microsecond Tic Counter Register	0x00000063
0x310400F8	EQOS_PHYIF_CTLSTAT	EQOS PHY Interface Control and Status Register	0x00000000
0x310400F8	EMAC0_PHYIF_CTLSTAT	EMAC0 PHY Interface Control and Status Register	0x00000000
0x31040114	EQOS_MAC_DBG	EQOS Debug Register	0x00000000
0x31040114	EMAC0_MAC_DBG	EMAC0 Debug Register	0x00000000
0x3104011C	EQOS_HW_FTR_PSNT0	EQOS Hardware Feature Presence Register 0	0x0A7D73F7
0x3104011C	EMAC0_HW_FTR_PSNT0	EMAC0 Hardware Feature Presence Register 0	0x0A7D73F7
0x31040120	EQOS_HW_FTR_PSNT1	EQOS Hardware Feature Presence Register 1	0x439739E7
0x31040120	EMAC0_HW_FTR_PSNT1	EMAC0 Hardware Feature Presence Register 1	0x439739E7
0x31040124	EQOS_HW_FTR_PSNT2	EQOS Hardware Feature Presence Register 2	0x441C71C7
0x31040124	EMAC0_HW_FTR_PSNT2	EMAC0 Hardware Feature Presence Register 2	0x441C71C7
0x31040128	EQOS_HW_FTR_PSNT3	EQOS Hardware Feature Presence Register 3	0x0C370231
0x31040128	EMAC0_HW_FTR_PSNT3	EMAC0 Hardware Feature Presence Register 3	0x0C370231
0x31040200	EQOS_MDIO_ADDR	EQOS MDIO Address Register	0x00000000
0x31040200	EMAC0_MDIO_ADDR	EMAC0 MDIO Address Register	0x00000000
0x31040204	EQOS_MDIO_DATA	EQOS MDIO Data Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040204	EMAC0_MDIO_DATA	EMAC0 MDIO Data Register	0x00000000
0x31040210	EQOS_ARP_ADDR	EQOS Address Resolution Protocol (ARP) Register	0x00000000
0x31040210	EMAC0_ARP_ADDR	EMAC0 Address Resolution Protocol (ARP) Register	0x00000000
0x31040230	EQOS_CSR_SW_CTL	EQOS CSR Software Control Register	0x00000000
0x31040230	EMAC0_CSR_SW_CTL	EMAC0 CSR Software Control Register	0x00000000
0x31040234	EQOS_FPE_CTLSTAT	EQOS Frame Preemption Control and Status Register	0x00000000
0x31040234	EMAC0_FPE_CTLSTAT	EMAC0 Frame Preemption Control and Status Register	0x00000000
0x31040238	EQOS_SPLM_OFST_CFG	EQOS MAC Split Mode and Offset Configuration Register	0x00020002
0x31040238	EMAC0_SPLM_OFST_CFG	EMAC0 MAC Split Mode and Offset Configuration Register	0x00020002
0x31040240	EQOS_PRSNTM_NSEC	EQOS Presentation Time Nanosecond Register	0x00000000
0x31040240	EMAC0_PRSNTM_NSEC	EMAC0 Presentation Time Nanosecond Register	0x00000000
0x31040244	EQOS_PRSNTM_UPDT	EQOS Presentation Time Update Register	0x00000000
0x31040244	EMAC0_PRSNTM_UPDT	EMAC0 Presentation Time Update Register	0x00000000
0x31040300	EQOS_ADDR0_HI	EQOS MAC Address 0 High Register	0x8000FFFF
0x31040300	EMAC0_ADDR0_HI	EMAC0 MAC Address 0 High Register	0x8000FFFF
0x31040304	EQOS_ADDR0_LO	EQOS MAC Address 0 Low Register	0xFFFFFFFF
0x31040304	EMAC0_ADDR0_LO	EMAC0 MAC Address 0 Low Register	0xFFFFFFFF
0x31040308	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040308	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104030C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x3104030C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040310	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040310	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040314	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x31040314	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040318	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040318	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104031C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x3104031C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040320	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040320	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040324	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x31040324	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040328	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040328	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104032C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x3104032C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040330	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040330	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040334	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x31040334	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040338	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040338	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104033C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x3104033C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040340	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040340	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040344	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x31040344	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040348	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040348	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104034C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x3104034C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040350	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040350	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040354	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x31040354	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040358	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040358	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104035C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3104035C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040360	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040360	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040364	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x31040364	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040368	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040368	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104036C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x3104036C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040370	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040370	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040374	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x31040374	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040378	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040378	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104037C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x3104037C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040380	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040380	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040384	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x31040384	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040388	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040388	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x3104038C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x3104038C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040390	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040390	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x31040394	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x31040394	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040398	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x31040398	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3104039C	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x3104039C	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403A0	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403A0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403A4	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403A4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403A8	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403A8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403AC	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403AC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403B0	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403B0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403B4	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403B4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403B8	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403B8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403BC	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403BC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403C0	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403C0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403C4	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403C4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403C8	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403C8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403CC	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403CC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403D0	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403D0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403D4	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403D4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403D8	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310403D8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403DC	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403DC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403E0	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403E0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403E4	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403E4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403E8	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403E8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403EC	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403EC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403F0	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403F0	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403F4	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403F4	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x310403F8	EQOS_ADDR[i]_HI	EQOS MAC Address n High Register	0x0000FFFF
0x310403F8	EMAC0_ADDR[i]_HI	EMAC0 MAC Address n High Register	0x0000FFFF
0x310403FC	EQOS_ADDR[i]_LO	EQOS MAC Address n Low Register	0xFFFFFFFF
0x310403FC	EMAC0_ADDR[i]_LO	EMAC0 MAC Address n Low Register	0xFFFFFFFF
0x31040700	EQOS_MMC_CTRL	EQOS MMC Control Register	0x00000000
0x31040700	EMAC0_MMC_CTRL	EMAC0 MMC Control Register	0x00000000
0x31040704	EQOS_MMC_RXINT	EQOS MMC Receive Interrupt Status Register	0x00000000
0x31040704	EMAC0_MMC_RXINT	EMAC0 MMC Receive Interrupt Status Register	0x00000000
0x31040708	EQOS_MMC_TXINT	EQOS MMC Transmit Interrupt Status Register	0x00000000
0x31040708	EMAC0_MMC_TXINT	EMAC0 MMC Transmit Interrupt Status Register	0x00000000
0x3104070C	EQOS_MMC_RXIMSK	EQOS MMC Receive Interrupt Mask Register	0x00000000
0x3104070C	EMAC0_MMC_RXIMSK	EMAC0 MMC Receive Interrupt Mask Register	0x00000000
0x31040710	EQOS_MMC_TXIMSK	EQOS MMC Transmit Interrupt Mask Register	0x00000000
0x31040710	EMAC0_MMC_TXIMSK	EMAC0 MMC Transmit Interrupt Mask Register	0x00000000
0x31040714	EQOS_TX_OCNT_GB	EQOS Transmit Octet Good Bad Counter Register	0x00000000
0x31040714	EMAC0_TX_OCNT_GB	EMAC0 Transmit Octet Good Bad Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040718	EQOS_TX_PCNT_GB	EQOS Transmit Packet Good Bad Counter Register	0x00000000
0x31040718	EMAC0_TX_PCNT_GB	EMAC0 Transmit Packet Good Bad Counter Register	0x00000000
0x3104071C	EQOS_TX_BCST_PCNT_G	EQOS Transmit Broadcast Packets Good Counter Register	0x00000000
0x3104071C	EMAC0_TX_BCST_PCNT_G	EMAC0 Transmit Broadcast Packets Good Counter Register	0x00000000
0x31040720	EQOS_TX_MCST_PCNT_G	EQOS Transmit Multicast Packets Good Counter Register	0x00000000
0x31040720	EMAC0_TX_MCST_PCNT_G	EMAC0 Transmit Multicast Packets Good Counter Register	0x00000000
0x31040724	EQOS_TX64_GB	EQOS Transmit 64 Octets Packets Good Bad Counter Register	0x00000000
0x31040724	EMAC0_TX64_GB	EMAC0 Transmit 64 Octets Packets Good Bad Counter Register	0x00000000
0x31040728	EQOS_TX65TO127_GB	EQOS Transmit 65-127 Octets Packets Good Bad Counter Register	0x00000000
0x31040728	EMAC0_TX65TO127_GB	EMAC0 Transmit 65-127 Octets Packets Good Bad Counter Register	0x00000000
0x3104072C	EQOS_TX128TO255_GB	EQOS Transmit 128-255 Octets Packets Good Bad Counter Register	0x00000000
0x3104072C	EMAC0_TX128TO255_GB	EMAC0 Transmit 128-255 Octets Packets Good Bad Counter Register	0x00000000
0x31040730	EQOS_TX256TO511_GB	EQOS Transmit 256-511 Octets Packets Good Bad Counter Register	0x00000000
0x31040730	EMAC0_TX256TO511_GB	EMAC0 Transmit 256-511 Octets Packets Good Bad Counter Register	0x00000000
0x31040734	EQOS_TX512TO1023_GB	EQOS Transmit 512-1023 Octets Packets Good Bad Counter Register	0x00000000
0x31040734	EMAC0_TX512TO1023_GB	EMAC0 Transmit 512-1023 Octets Packets Good Bad Counter Register	0x00000000
0x31040738	EQOS_TX1024TOMAX_GB	EQOS Transmit 1024-Max Octets Packets Good Bad Counter Register	0x00000000
0x31040738	EMAC0_TX1024TOMAX_GB	EMAC0 Transmit 1024-Max Octets Packets Good Bad Counter Register	0x00000000
0x3104073C	EQOS_TX_UCST_PCNT_GB	EQOS Transmit Unicast Packets Good Bad Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3104073C	EMAC0_TX_UCST_PCNT_GB	EMAC0 Transmit Unicast Packets Good Bad Counter Register	0x00000000
0x31040740	EQOS_TX_MCST_PCNT_GB	EQOS Transmit Multicast Packets Good Bad Counter Register	0x00000000
0x31040740	EMAC0_TX_MCST_PCNT_GB	EMAC0 Transmit Multicast Packets Good Bad Counter Register	0x00000000
0x31040744	EQOS_TX_BCST_PCNT_GB	EQOS Transmit Broadcast Packets Good Bad Counter Register	0x00000000
0x31040744	EMAC0_TX_BCST_PCNT_GB	EMAC0 Transmit Broadcast Packets Good Bad Counter Register	0x00000000
0x31040748	EQOS_TX_URFL_PCNT	EQOS Transmit Underflow Error Packets Counter Register	0x00000000
0x31040748	EMAC0_TX_URFL_PCNT	EMAC0 Transmit Underflow Error Packets Counter Register	0x00000000
0x3104074C	EQOS_TX_SNGLCOL_PCNT_G	EQOS Transmit Single Collision Packets Good Counter Register	0x00000000
0x3104074C	EMAC0_TX_SNGLCOL_PCNT_G	EMAC0 Transmit Single Collision Packets Good Counter Register	0x00000000
0x31040750	EQOS_TX_MCOL_PCNT_G	EQOS Transmit Multiple Collision Packets Good Counter Register	0x00000000
0x31040750	EMAC0_TX_MCOL_PCNT_G	EMAC0 Transmit Multiple Collision Packets Good Counter Register	0x00000000
0x31040754	EQOS_TX_DFRD_PCNT	EQOS Transmit Deferred Packets Counter Register	0x00000000
0x31040754	EMAC0_TX_DFRD_PCNT	EMAC0 Transmit Deferred Packets Counter Register	0x00000000
0x31040758	EQOS_TX_LTCOL_PCNT	EQOS Transmit Late Collision Packets Counter Register	0x00000000
0x31040758	EMAC0_TX_LTCOL_PCNT	EMAC0 Transmit Late Collision Packets Counter Register	0x00000000
0x3104075C	EQOS_TX_EXCOL_PCNT	EQOS Transmit Excessive Collision Packets Counter Register	0x00000000
0x3104075C	EMAC0_TX_EXCOL_PCNT	EMAC0 Transmit Excessive Collision Packets Counter Register	0x00000000
0x31040760	EQOS_TX_CRERR_PCNT	EQOS Transmit Carrier Error Packets Counter Register	0x00000000
0x31040760	EMAC0_TX_CRERR_PCNT	EMAC0 Transmit Carrier Error Packets Counter Register	0x00000000
0x31040764	EQOS_TX_OCNT_G	EQOS Transmit Octet Good Counter Register	0x00000000
0x31040764	EMAC0_TX_OCNT_G	EMAC0 Transmit Octet Good Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040768	EQOS_TX_PCNT_G	EQOS Transmit Packet Good Counter Register	0x00000000
0x31040768	EMAC0_TX_PCNT_G	EMAC0 Transmit Packet Good Counter Register	0x00000000
0x3104076C	EQOS_TX_EXDFRL_PCNT	EQOS Transmit Excessive Deferral Error Counter Register	0x00000000
0x3104076C	EMAC0_TX_EXDFRL_PCNT	EMAC0 Transmit Excessive Deferral Error Counter Register	0x00000000
0x31040770	EQOS_TX_PAUSE_PCNT	EQOS Transmit Pause Packets Counter Register	0x00000000
0x31040770	EMAC0_TX_PAUSE_PCNT	EMAC0 Transmit Pause Packets Counter Register	0x00000000
0x31040774	EQOS_TX_VLAN_PCNT_G	EQOS Transmit VLAN Packets Good Counter Register	0x00000000
0x31040774	EMAC0_TX_VLAN_PCNT_G	EMAC0 Transmit VLAN Packets Good Counter Register	0x00000000
0x31040778	EQOS_TX_OSIZE_PCNT_G	EQOS Transmit Osize Packets Good Counter Register	0x00000000
0x31040778	EMAC0_TX_OSIZE_PCNT_G	EMAC0 Transmit Osize Packets Good Counter Register	0x00000000
0x31040780	EQOS_RX_PCNT_GB	EQOS Receive Packets Good Bad Counter Register	0x00000000
0x31040780	EMAC0_RX_PCNT_GB	EMAC0 Receive Packets Good Bad Counter Register	0x00000000
0x31040784	EQOS_RX_OCNT_GB	EQOS Receive Octet Good Bad Counter Register	0x00000000
0x31040784	EMAC0_RX_OCNT_GB	EMAC0 Receive Octet Good Bad Counter Register	0x00000000
0x31040788	EQOS_RX_OCNT_G	EQOS Receive Octet Good Counter Register	0x00000000
0x31040788	EMAC0_RX_OCNT_G	EMAC0 Receive Octet Good Counter Register	0x00000000
0x3104078C	EQOS_RX_BCERR_PCNT_G	EQOS Receive Broadcast Packets Good Counter Register	0x00000000
0x3104078C	EMAC0_RX_BCERR_PCNT_G	EMAC0 Receive Broadcast Packets Good Counter Register	0x00000000
0x31040790	EQOS_RX_MCST_PCNT_G	EQOS Receive Multicast Packets Good Counter Register	0x00000000
0x31040790	EMAC0_RX_MCST_PCNT_G	EMAC0 Receive Multicast Packets Good Counter Register	0x00000000
0x31040794	EQOS_RX_CRCERR_PCNT	EQOS Receive CRC Error Packets Counter Register	0x00000000
0x31040794	EMAC0_RX_CRCERR_PCNT	EMAC0 Receive CRC Error Packets Counter Register	0x00000000
0x31040798	EQOS_RX_ALGNERR_PCNT	EQOS Receive Alignment Error Packets Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040798	EMAC0_RX_ALGNERR_PCNT	EMAC0 Receive Alignment Error Packets Counter Register	0x00000000
0x3104079C	EQOS_RX_RNTERR_PCNT	EQOS Receive Runt Error Packets Counter Register	0x00000000
0x3104079C	EMAC0_RX_RNTERR_PCNT	EMAC0 Receive Runt Error Packets Counter Register	0x00000000
0x310407A0	EQOS_RX_JBERR_PCNT	EQOS Receive Jabber Error Packets Counter Register	0x00000000
0x310407A0	EMAC0_RX_JBERR_PCNT	EMAC0 Receive Jabber Error Packets Counter Register	0x00000000
0x310407A4	EQOS_RX_UNDRSZ_PCNT_G	EQOS Receive Undersize Packets Good Counter Register	0x00000000
0x310407A4	EMAC0_RX_UNDRSZ_PCNT_G	EMAC0 Receive Undersize Packets Good Counter Register	0x00000000
0x310407A8	EQOS_RX_OVRSZ_PCNT	EQOS Receive Oversize Packets Good Counter Register	0x00000000
0x310407A8	EMAC0_RX_OVRSZ_PCNT	EMAC0 Receive Oversize Packets Good Counter Register	0x00000000
0x310407AC	EQOS_RX64_GB	EQOS Receive 64 Octets Packets Good Bad Counter Register	0x00000000
0x310407AC	EMAC0_RX64_GB	EMAC0 Receive 64 Octets Packets Good Bad Counter Register	0x00000000
0x310407B0	EQOS_RX65TO127_GB	EQOS Receive 65 - 127 Octets Packets Good Bad Counter Register	0x00000000
0x310407B0	EMAC0_RX65TO127_GB	EMAC0 Receive 65 - 127 Octets Packets Good Bad Counter Register	0x00000000
0x310407B4	EQOS_RX128TO255_GB	EQOS Receive 128 - 255 Octets Packets Good Bad Counter Register	0x00000000
0x310407B4	EMAC0_RX128TO255_GB	EMAC0 Receive 128 - 255 Octets Packets Good Bad Counter Register	0x00000000
0x310407B8	EQOS_RX256TO511_GB	EQOS Receive 256 - 511 Octets Packets Good Bad Counter Register	0x00000000
0x310407B8	EMAC0_RX256TO511_GB	EMAC0 Receive 256 - 511 Octets Packets Good Bad Counter Register	0x00000000
0x310407BC	EQOS_RX512TO1023_GB	EQOS Receive 512 - 1023 Octets Packets Good Bad Counter Register	0x00000000
0x310407BC	EMAC0_RX512TO1023_GB	EMAC0 Receive 512 - 1023 Octets Packets Good Bad Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310407C0	EQOS_RX1024TOMAX_GB	EQOS Receive 1024 - Maximum Octets Packets Good Bad Counter Register	0x00000000
0x310407C0	EMAC0_RX1024TOMAX_GB	EMAC0 Receive 1024 - Maximum Octets Packets Good Bad Counter Register	0x00000000
0x310407C4	EQOS_RX_UCST_PCNT_G	EQOS Receive Unicast Packets Good Counter Register	0x00000000
0x310407C4	EMAC0_RX_UCST_PCNT_G	EMAC0 Receive Unicast Packets Good Counter Register	0x00000000
0x310407C8	EQOS_RX_LENERR_PCNT	EQOS Receive Length Error Packets Counter Register	0x00000000
0x310407C8	EMAC0_RX_LENERR_PCNT	EMAC0 Receive Length Error Packets Counter Register	0x00000000
0x310407CC	EQOS_RX_ORTYPE_PCNT	EQOS Receive Out of Range Type Packets Counter Register	0x00000000
0x310407CC	EMAC0_RX_ORTYPE_PCNT	EMAC0 Receive Out of Range Type Packets Counter Register	0x00000000
0x310407D0	EQOS_RX_PAUSE_PCNT	EQOS Receive Pause Packets Counter Register	0x00000000
0x310407D0	EMAC0_RX_PAUSE_PCNT	EMAC0 Receive Pause Packets Counter Register	0x00000000
0x310407D4	EQOS_RX_FIFOOVFL_PCNT	EQOS Receive FIFO Overflow Packets Counter Register	0x00000000
0x310407D4	EMAC0_RX_FIFOOVFL_PCNT	EMAC0 Receive FIFO Overflow Packets Counter Register	0x00000000
0x310407D8	EQOS_RX_VLAN_PCNT_GB	EQOS Receive VLAN Packets Good Bad Counter Register	0x00000000
0x310407D8	EMAC0_RX_VLAN_PCNT_GB	EMAC0 Receive VLAN Packets Good Bad Counter Register	0x00000000
0x310407DC	EQOS_RX_WDTERR_PCNT	EQOS Receive Watchdog Error Packets Counter Register	0x00000000
0x310407DC	EMAC0_RX_WDTERR_PCNT	EMAC0 Receive Watchdog Error Packets Counter Register	0x00000000
0x310407E0	EQOS_RX_ERR_PCNT	EQOS Receive Error Packets Counter Register	0x00000000
0x310407E0	EMAC0_RX_ERR_PCNT	EMAC0 Receive Error Packets Counter Register	0x00000000
0x310407E4	EQOS_RX_CTL_PCNT_G	EQOS Receive Control Packets Good Counter Register	0x00000000
0x310407E4	EMAC0_RX_CTL_PCNT_G	EMAC0 Receive Control Packets Good Counter Register	0x00000000
0x310407EC	EQOS_TX_LPIUSEC_CNT	EQOS Transmit LPI Microseconds Counter Register	0x00000000
0x310407EC	EMAC0_TX_LPIUSEC_CNT	EMAC0 Transmit LPI Microseconds Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310407F0	EQOS_TX_LPITRAN_CNT	EQOS Transmit LPI Transition Counter Register	0x00000000
0x310407F0	EMAC0_TX_LPITRAN_CNT	EMAC0 Transmit LPI Transition Counter Register	0x00000000
0x310407F4	EQOS_RX_LPIUSEC_CNT	EQOS Receive LPI Microseconds Counter Register	0x00000000
0x310407F4	EMAC0_RX_LPIUSEC_CNT	EMAC0 Receive LPI Microseconds Counter Register	0x00000000
0x310407F8	EQOS_RX_LPITRAN_CNT	EQOS Receive LPI Transition Counter Register	0x00000000
0x310407F8	EMAC0_RX_LPITRAN_CNT	EMAC0 Receive LPI Transition Counter Register	0x00000000
0x31040800	EQOS_MMC_IPC_RXIMSK	EQOS MMC Receive IPC Interrupt Mask Register	0x00000000
0x31040800	EMAC0_MMC_IPC_RXIMSK	EMAC0 MMC Receive IPC Interrupt Mask Register	0x00000000
0x31040808	EQOS_MMC_IPC_RXINT	EQOS MMC Receive IPC Interrupt Status Register	0x00000000
0x31040808	EMAC0_MMC_IPC_RXINT	EMAC0 MMC Receive IPC Interrupt Status Register	0x00000000
0x31040810	EQOS_RXIPV4_PCNT_G	EQOS Receive IPv4 Good Packets Counter Register	0x00000000
0x31040810	EMAC0_RXIPV4_PCNT_G	EMAC0 Receive IPv4 Good Packets Counter Register	0x00000000
0x31040814	EQOS_RXIPV4_HDRERR_PCNT	EQOS Receive IPv4 Header Error Packets Counter Register	0x00000000
0x31040814	EMAC0_RXIPV4_HDRERR_PCNT	EMAC0 Receive IPv4 Header Error Packets Counter Register	0x00000000
0x31040818	EQOS_RXIPV4_PYLD_PCNT	EQOS Receive IPv4 Payload Packets Counter Register	0x00000000
0x31040818	EMAC0_RXIPV4_PYLD_PCNT	EMAC0 Receive IPv4 Payload Packets Counter Register	0x00000000
0x3104081C	EQOS_RXIPV4_FRGMT_PCNT	EQOS Receive IPv4 Fragmented Packets Counter Register	0x00000000
0x3104081C	EMAC0_RXIPV4_FRGMT_PCNT	EMAC0 Receive IPv4 Fragmented Packets Counter Register	0x00000000
0x31040820	EQOS_RXIPV4_UDPCSD_PCNT	EQOS Receive IPv4 UDP Checksum Disable Packets Counter Register	0x00000000
0x31040820	EMAC0_RXIPV4_UDPCSD_PCNT	EMAC0 Receive IPv4 UDP Checksum Disable Packets Counter Register	0x00000000
0x31040824	EQOS_RXIPV6_PCNT_G	EQOS Receive IPv6 Good Packets Counter Register	0x00000000
0x31040824	EMAC0_RXIPV6_PCNT_G	EMAC0 Receive IPv6 Good Packets Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040828	EQOS_RXIPV6_HDRERR_PCNT	EQOS Receive IPv6 Header Error Packets Counter Register	0x00000000
0x31040828	EMAC0_RXIPV6_HDRERR_PCNT	EMAC0 Receive IPv6 Header Error Packets Counter Register	0x00000000
0x3104082C	EQOS_RXIPV6_PYLD_PCNT	EQOS Receive IPv6 Payload Packets Counter Register	0x00000000
0x3104082C	EMAC0_RXIPV6_PYLD_PCNT	EMAC0 Receive IPv6 Payload Packets Counter Register	0x00000000
0x31040830	EQOS_RXUDP_PCNT_G	EQOS Receive UDP Good Packets Counter Register	0x00000000
0x31040830	EMAC0_RXUDP_PCNT_G	EMAC0 Receive UDP Good Packets Counter Register	0x00000000
0x31040834	EQOS_RXUDP_ERR_PCNT	EQOS Receive UDP Error Packets Counter Register	0x00000000
0x31040834	EMAC0_RXUDP_ERR_PCNT	EMAC0 Receive UDP Error Packets Counter Register	0x00000000
0x31040838	EQOS_RXTCP_PCNT_G	EQOS Receive TCP Good Packets Counter Register	0x00000000
0x31040838	EMAC0_RXTCP_PCNT_G	EMAC0 Receive TCP Good Packets Counter Register	0x00000000
0x3104083C	EQOS_RXTCP_ERR_PCNT	EQOS Receive TCP Error Packets Counter Register	0x00000000
0x3104083C	EMAC0_RXTCP_ERR_PCNT	EMAC0 Receive TCP Error Packets Counter Register	0x00000000
0x31040840	EQOS_RXICMP_PCNT_G	EQOS Receive ICMP Good Packets Counter Register	0x00000000
0x31040840	EMAC0_RXICMP_PCNT_G	EMAC0 Receive ICMP Good Packets Counter Register	0x00000000
0x31040844	EQOS_RXICMP_ERR_PCNT	EQOS Receive ICMP Error Packets Counter Register	0x00000000
0x31040844	EMAC0_RXICMP_ERR_PCNT	EMAC0 Receive ICMP Error Packets Counter Register	0x00000000
0x31040850	EQOS_RXIPV4_OCNT_G	EQOS Receive IPv4 Good Octets Counter Register	0x00000000
0x31040850	EMAC0_RXIPV4_OCNT_G	EMAC0 Receive IPv4 Good Octets Counter Register	0x00000000
0x31040854	EQOS_RXIPV4_HDRERR_OCNT	EQOS Receive IPv4 Header Error Octets Counter Register	0x00000000
0x31040854	EMAC0_RXIPV4_HDRERR_OCNT	EMAC0 Receive IPv4 Header Error Octets Counter Register	0x00000000
0x31040858	EQOS_RXIPV4_PYLD_OCNT	EQOS Receive IPv4 Payload Octets Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040858	EMAC0_RXIPV4_PYLD_OCNT	EMAC0 Receive IPv4 Payload Octets Counter Register	0x00000000
0x3104085C	EQOS_RXIPV4_FRGMT_OCNT	EQOS Receive IPv4 Fragmented Octets Counter Register	0x00000000
0x3104085C	EMAC0_RXIPV4_FRGMT_OCNT	EMAC0 Receive IPv4 Fragmented Octets Counter Register	0x00000000
0x31040860	EQOS_RXIPV4_UDPCSD_OCNT	EQOS Receive IPv4 UDP Checksum Disable Octets Counter Register	0x00000000
0x31040860	EMAC0_RXIPV4_UDPCSD_OCNT	EMAC0 Receive IPv4 UDP Checksum Disable Octets Counter Register	0x00000000
0x31040864	EQOS_RXIPV6_OCNT_G	EQOS Receive IPv6 Good Octets Counter Register	0x00000000
0x31040864	EMAC0_RXIPV6_OCNT_G	EMAC0 Receive IPv6 Good Octets Counter Register	0x00000000
0x31040868	EQOS_RXIPV6_HDRERR_OCNT	EQOS Receive IPv6 Header Error Octets Counter Register	0x00000000
0x31040868	EMAC0_RXIPV6_HDRERR_OCNT	EMAC0 Receive IPv6 Header Error Octets Counter Register	0x00000000
0x3104086C	EQOS_RXIPV6_PYLD_OCNT	EQOS Receive IPv6 Payload Octets Counter Register	0x00000000
0x3104086C	EMAC0_RXIPV6_PYLD_OCNT	EMAC0 Receive IPv6 Payload Octets Counter Register	0x00000000
0x31040870	EQOS_RXUDP_OCNT_G	EQOS Receive UDP Good Octets Counter Register	0x00000000
0x31040870	EMAC0_RXUDP_OCNT_G	EMAC0 Receive UDP Good Octets Counter Register	0x00000000
0x31040874	EQOS_RXUDP_ERR_OCNT	EQOS Receive UDP Error Octets Counter Register	0x00000000
0x31040874	EMAC0_RXUDP_ERR_OCNT	EMAC0 Receive UDP Error Octets Counter Register	0x00000000
0x31040878	EQOS_RXTCP_OCNT_G	EQOS Receive TCP Good Octets Counter Register	0x00000000
0x31040878	EMAC0_RXTCP_OCNT_G	EMAC0 Receive TCP Good Octets Counter Register	0x00000000
0x3104087C	EQOS_RXTCP_ERR_OCNT	EQOS Receive TCP Error Octets Counter Register	0x00000000
0x3104087C	EMAC0_RXTCP_ERR_OCNT	EMAC0 Receive TCP Error Octets Counter Register	0x00000000
0x31040880	EQOS_RXICMP_OCNT_G	EQOS Receive ICMP Good Octets Counter Register	0x00000000
0x31040880	EMAC0_RXICMP_OCNT_G	EMAC0 Receive ICMP Good Octets Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040884	EQOS_RXICMP_ERR_OCNT	EQOS Receive ICMP Error Octets Counter Register	0x00000000
0x31040884	EMAC0_RXICMP_ERR_OCNT	EMAC0 Receive ICMP Error Octets Counter Register	0x00000000
0x310408A0	EQOS_MMC_FPE_TXINT	EQOS MMC Transmit FPE Interrupt Status Register	0x00000000
0x310408A0	EMAC0_MMC_FPE_TXINT	EMAC0 MMC Transmit FPE Interrupt Status Register	0x00000000
0x310408A4	EQOS_MMC_FPE_TXIMSK	EQOS MMC Transmit FPE Interrupt Mask Register	0x00000000
0x310408A4	EMAC0_MMC_FPE_TXIMSK	EMAC0 MMC Transmit FPE Interrupt Mask Register	0x00000000
0x310408A8	EQOS_TXFPE_FRGMNT_CNT	EQOS MMC Transmit FPE Packet Fragment Counter Register	0x00000000
0x310408A8	EMAC0_TXFPE_FRGMNT_CNT	EMAC0 MMC Transmit FPE Packet Fragment Counter Register	0x00000000
0x310408AC	EQOS_TXHLDRQ_CNT	EQOS MMC Transmit FPE Hold Request Counter Register	0x00000000
0x310408AC	EMAC0_TXHLDRQ_CNT	EMAC0 MMC Transmit FPE Hold Request Counter Register	0x00000000
0x310408C0	EQOS_MMC_FPE_RXINT	EQOS MMC Receive FPE Interrupt Status Register	0x00000000
0x310408C0	EMAC0_MMC_FPE_RXINT	EMAC0 MMC Receive FPE Interrupt Status Register	0x00000000
0x310408C4	EQOS_MMC_FPE_RXIMSK	EQOS MMC Receive FPE Interrupt Mask Register	0x00000000
0x310408C4	EMAC0_MMC_FPE_RXIMSK	EMAC0 MMC Receive FPE Interrupt Mask Register	0x00000000
0x310408C8	EQOS_RXPKT_ASERR_CNT	EQOS MMC Receive FPE Packet Assembly Error Counter Register	0x00000000
0x310408C8	EMAC0_RXPKT_ASERR_CNT	EMAC0 MMC Receive FPE Packet Assembly Error Counter Register	0x00000000
0x310408CC	EQOS_RXPKT_SMDERR_CNT	EQOS MMC Receive FPE Packet SMD Error Counter Register	0x00000000
0x310408CC	EMAC0_RXPKT_SMDERR_CNT	EMAC0 MMC Receive FPE Packet SMD Error Counter Register	0x00000000
0x310408D0	EQOS_RXPKT_ASOK_CNT	EQOS MMC Receive FPE Packet Assembly Okay Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310408D0	EMAC0_RXPKT_ASOK_CNT	EMAC0 MMC Receive FPE Packet Assembly Okay Counter Register	0x00000000
0x310408D4	EQOS_RXFPE_FRGMNT_CNT	EQOS MMC Receive FPE Fragment Counter Register	0x00000000
0x310408D4	EMAC0_RXFPE_FRGMNT_CNT	EMAC0 MMC Receive FPE Fragment Counter Register	0x00000000
0x31040900	EQOS_L3L4_CTL[i]	EQOS Layer 3 and Layer 4 Control Register	0x00000000
0x31040900	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040904	EQOS_L4_ADDR[i]	EQOS Layer 4 Address Register [i]	0x00000000
0x31040904	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040910	EQOS_L3_ADDR0_REG[i]	EQOS Layer 3 Address 0 Register [i]	0x00000000
0x31040910	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040914	EQOS_L3_ADDR1_REG[i]	EQOS Layer 3 Address 1 Register [i]	0x00000000
0x31040914	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040918	EQOS_L3_ADDR2_REG[i]	EQOS Layer 3 Address 2 Register [i]	0x00000000
0x31040918	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x3104091C	EQOS_L3_ADDR3_REG[i]	EQOS Layer 3 Address 3 Register [i]	0x00000000
0x3104091C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040930	EQOS_L3L4_CTL[i]	EQOS Layer 3 and Layer 4 Control Register	0x00000000
0x31040930	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040934	EQOS_L4_ADDR[i]	EQOS Layer 4 Address Register [i]	0x00000000
0x31040934	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040940	EQOS_L3_ADDR0_REG[i]	EQOS Layer 3 Address 0 Register [i]	0x00000000
0x31040940	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040944	EQOS_L3_ADDR1_REG[i]	EQOS Layer 3 Address 1 Register [i]	0x00000000
0x31040944	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040948	EQOS_L3_ADDR2_REG[i]	EQOS Layer 3 Address 2 Register [i]	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040948	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x3104094C	EQOS_L3_ADDR3_REG[i]	EQOS Layer 3 Address 3 Register [i]	0x00000000
0x3104094C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040960	EQOS_L3L4_CTL[i]	EQOS Layer 3 and Layer 4 Control Register	0x00000000
0x31040960	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040964	EQOS_L4_ADDR[i]	EQOS Layer 4 Address Register [i]	0x00000000
0x31040964	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040970	EQOS_L3_ADDR0_REG[i]	EQOS Layer 3 Address 0 Register [i]	0x00000000
0x31040970	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040974	EQOS_L3_ADDR1_REG[i]	EQOS Layer 3 Address 1 Register [i]	0x00000000
0x31040974	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040978	EQOS_L3_ADDR2_REG[i]	EQOS Layer 3 Address 2 Register [i]	0x00000000
0x31040978	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x3104097C	EQOS_L3_ADDR3_REG[i]	EQOS Layer 3 Address 3 Register [i]	0x00000000
0x3104097C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040990	EQOS_L3L4_CTL[i]	EQOS Layer 3 and Layer 4 Control Register	0x00000000
0x31040990	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040994	EQOS_L4_ADDR[i]	EQOS Layer 4 Address Register [i]	0x00000000
0x31040994	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x310409A0	EQOS_L3_ADDR0_REG[i]	EQOS Layer 3 Address 0 Register [i]	0x00000000
0x310409A0	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x310409A4	EQOS_L3_ADDR1_REG[i]	EQOS Layer 3 Address 1 Register [i]	0x00000000
0x310409A4	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x310409A8	EQOS_L3_ADDR2_REG[i]	EQOS Layer 3 Address 2 Register [i]	0x00000000
0x310409A8	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310409AC	EQOS_L3_ADDR3_REG[i]	EQOS Layer 3 Address 3 Register [i]	0x00000000
0x310409AC	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x310409C0	EQOS_L3L4_CTL[i]	EQOS Layer 3 and Layer 4 Control Register	0x00000000
0x310409C0	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x310409C4	EQOS_L4_ADDR[i]	EQOS Layer 4 Address Register [i]	0x00000000
0x310409C4	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x310409D0	EQOS_L3_ADDR0_REG[i]	EQOS Layer 3 Address 0 Register [i]	0x00000000
0x310409D0	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x310409D4	EQOS_L3_ADDR1_REG[i]	EQOS Layer 3 Address 1 Register [i]	0x00000000
0x310409D4	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x310409D8	EQOS_L3_ADDR2_REG[i]	EQOS Layer 3 Address 2 Register [i]	0x00000000
0x310409D8	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x310409DC	EQOS_L3_ADDR3_REG[i]	EQOS Layer 3 Address 3 Register [i]	0x00000000
0x310409DC	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x310409F0	EQOS_L3L4_CTL[i]	EQOS Layer 3 and Layer 4 Control Register	0x00000000
0x310409F0	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x310409F4	EQOS_L4_ADDR[i]	EQOS Layer 4 Address Register [i]	0x00000000
0x310409F4	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040A00	EQOS_L3_ADDR0_REG[i]	EQOS Layer 3 Address 0 Register [i]	0x00000000
0x31040A00	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040A04	EQOS_L3_ADDR1_REG[i]	EQOS Layer 3 Address 1 Register [i]	0x00000000
0x31040A04	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040A08	EQOS_L3_ADDR2_REG[i]	EQOS Layer 3 Address 2 Register [i]	0x00000000
0x31040A08	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x31040A0C	EQOS_L3_ADDR3_REG[i]	EQOS Layer 3 Address 3 Register [i]	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040A0C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040A20	EQOS_L3L4_CTL[i]	EQOS Layer 3 and Layer 4 Control Register	0x00000000
0x31040A20	EMAC0_L3L4_CTL[i]	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040A24	EQOS_L4_ADDR[i]	EQOS Layer 4 Address Register [i]	0x00000000
0x31040A24	EMAC0_L4_ADDR[i]	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040A30	EQOS_L3_ADDR0_REG[i]	EQOS Layer 3 Address 0 Register [i]	0x00000000
0x31040A30	EMAC0_L3_ADDR0_REG[i]	EMAC0 Layer 3 Address 0 Register [i]	0x00000000
0x31040A34	EQOS_L3_ADDR1_REG[i]	EQOS Layer 3 Address 1 Register [i]	0x00000000
0x31040A34	EMAC0_L3_ADDR1_REG[i]	EMAC0 Layer 3 Address 1 Register [i]	0x00000000
0x31040A38	EQOS_L3_ADDR2_REG[i]	EQOS Layer 3 Address 2 Register [i]	0x00000000
0x31040A38	EMAC0_L3_ADDR2_REG[i]	EMAC0 Layer 3 Address 2 Register [i]	0x00000000
0x31040A3C	EQOS_L3_ADDR3_REG[i]	EQOS Layer 3 Address 3 Register [i]	0x00000000
0x31040A3C	EMAC0_L3_ADDR3_REG[i]	EMAC0 Layer 3 Address 3 Register [i]	0x00000000
0x31040A50	EQOS_L3L4_CTL7	EQOS Layer 3 and Layer 4 Control Register	0x00000000
0x31040A50	EMAC0_L3L4_CTL7	EMAC0 Layer 3 and Layer 4 Control Register	0x00000000
0x31040A54	EQOS_L4_ADDR7	EQOS Layer 4 Address Register [i]	0x00000000
0x31040A54	EMAC0_L4_ADDR7	EMAC0 Layer 4 Address Register [i]	0x00000000
0x31040A60	EQOS_L3_ADDR0_REG7	EQOS Layer 3 Address 0 Register 7	0x00000000
0x31040A60	EMAC0_L3_ADDR0_REG7	EMAC0 Layer 3 Address 0 Register 7	0x00000000
0x31040A64	EQOS_L3_ADDR1_REG7	EQOS Layer 3 Address 1 Register 7	0x00000000
0x31040A64	EMAC0_L3_ADDR1_REG7	EMAC0 Layer 3 Address 1 Register 7	0x00000000
0x31040A68	EQOS_L3_ADDR2_REG7	EQOS Layer 3 Address 2 Register 7	0x00000000
0x31040A68	EMAC0_L3_ADDR2_REG7	EMAC0 Layer 3 Address 2 Register 7	0x00000000
0x31040A6C	EQOS_L3_ADDR3_REG7	EQOS Layer 3 Address 3 Register 7	0x00000000
0x31040A6C	EMAC0_L3_ADDR3_REG7	EMAC0 Layer 3 Address 3 Register 7	0x00000000
0x31040A70	EQOS_INDR_ACC_CTL	EQOS Indirect Access Control Register	0x00000000
0x31040A70	EMAC0_INDR_ACC_CTL	EMAC0 Indirect Access Control Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040A74	EQOS_INDR_ACC_DAT	EQOS Indirect Access Data Register	0x00000000
0x31040A74	EMAC0_INDR_ACC_DAT	EMAC0 Indirect Access Data Register	0x00000000
0x31040B00	EQOS_TM_CTL	EQOS Time Stamp Control Register	0x00002000
0x31040B00	EMAC0_TM_CTL	EMAC0 Time Stamp Control Register	0x00002000
0x31040B04	EQOS_TM_SUBSEC	EQOS Subsecond Increment Register	0x00000000
0x31040B04	EMAC0_TM_SUBSEC	EMAC0 Subsecond Increment Register	0x00000000
0x31040B08	EQOS_TM_SEC	EQOS System Time Seconds Register	0x00000000
0x31040B08	EMAC0_TM_SEC	EMAC0 System Time Seconds Register	0x00000000
0x31040B0C	EQOS_TM_NSEC	EQOS System Time Nanoseconds Register	0x00000000
0x31040B0C	EMAC0_TM_NSEC	EMAC0 System Time Nanoseconds Register	0x00000000
0x31040B10	EQOS_TM_SECUPDT	EQOS System Time Seconds Update Register	0x00000000
0x31040B10	EMAC0_TM_SECUPDT	EMAC0 System Time Seconds Update Register	0x00000000
0x31040B14	EQOS_TM_NSECUPDT	EQOS System Time Nanoseconds Update Register	0x00000000
0x31040B14	EMAC0_TM_NSECUPDT	EMAC0 System Time Nanoseconds Update Register	0x00000000
0x31040B18	EQOS_TM_ADDEND	EQOS Time Stamp Addend Register	0x00000000
0x31040B18	EMAC0_TM_ADDEND	EMAC0 Time Stamp Addend Register	0x00000000
0x31040B1C	EQOS_TM_HISEC	EQOS System Time Higher Word Seconds Register	0x00000000
0x31040B1C	EMAC0_TM_HISEC	EMAC0 System Time Higher Word Seconds Register	0x00000000
0x31040B20	EQOS_TM_STAT	EQOS Time Stamp Status Register	0x00000000
0x31040B20	EMAC0_TM_STAT	EMAC0 Time Stamp Status Register	0x00000000
0x31040B30	EQOS_TXTMSTAT_NSEC	EQOS Transmit Time Stamp Status Nanoseconds Register	0x00000000
0x31040B30	EMAC0_TXTMSTAT_NSEC	EMAC0 Transmit Time Stamp Status Nanoseconds Register	0x00000000
0x31040B34	EQOS_TXTMSTAT_SEC	EQOS Transmit Time Stamp Status Seconds Register	0x00000000
0x31040B34	EMAC0_TXTMSTAT_SEC	EMAC0 Transmit Time Stamp Status Seconds Register	0x00000000
0x31040B40	EQOS_TM_AUX_CTL	EQOS Auxiliary Time Stamp Control Register	0x00000000
0x31040B40	EMAC0_TM_AUX_CTL	EMAC0 Auxiliary Time Stamp Control Register	0x00000000
0x31040B48	EQOS_TM_AUXTS_NSEC	EQOS Auxiliary Time Stamp Nanoseconds Register	0x00000000
0x31040B48	EMAC0_TM_AUXTS_NSEC	EMAC0 Auxiliary Time Stamp Nanoseconds Register	0x00000000
0x31040B4C	EQOS_TM_AUXTS_SEC	EQOS Auxiliary Time Stamp Seconds Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040B4C	EMAC0_TM_AUXTS_SEC	EMAC0 Auxiliary Time Stamp Seconds Register	0x00000000
0x31040B50	EQOS_TM_INGASYM_CORR	EQOS Time Stamp Ingress Asymmetry Correction Register	0x00000000
0x31040B50	EMAC0_TM_INGASYM_CORR	EMAC0 Time Stamp Ingress Asymmetry Correction Register	0x00000000
0x31040B54	EQOS_TM_EGASYM_CORR	EQOS Time Stamp Egress Asymmetry Correction Register	0x00000000
0x31040B54	EMAC0_TM_EGASYM_CORR	EMAC0 Time Stamp Egress Asymmetry Correction Register	0x00000000
0x31040B58	EQOS_TM_INGCORR_NSEC	EQOS Time Stamp Ingress Correction Nanosecond Register	0x00000000
0x31040B58	EMAC0_TM_INGCORR_NSEC	EMAC0 Time Stamp Ingress Correction Nanosecond Register	0x00000000
0x31040B5C	EQOS_TM_EGCORR_NSEC	EQOS Time Stamp Egress Correction Nanosecond Register	0x00000000
0x31040B5C	EMAC0_TM_EGCORR_NSEC	EMAC0 Time Stamp Egress Correction Nanosecond Register	0x00000000
0x31040B60	EQOS_TM_INGCORR_SNSEC	EQOS Time Stamp Ingress Correction Sub Nanosecond Register	0x00000000
0x31040B60	EMAC0_TM_INGCORR_SNSEC	EMAC0 Time Stamp Ingress Correction Sub Nanosecond Register	0x00000000
0x31040B64	EQOS_TM_EGCORR_SNSEC	EQOS Time Stamp Egress Correction Sub Nanosecond Register	0x00000000
0x31040B64	EMAC0_TM_EGCORR_SNSEC	EMAC0 Time Stamp Egress Correction Sub Nanosecond Register	0x00000000
0x31040B68	EQOS_TM_ING_LTNCY	EQOS Time Stamp Ingress Latency Register	0x00000000
0x31040B68	EMAC0_TM_ING_LTNCY	EMAC0 Time Stamp Ingress Latency Register	0x00000000
0x31040B6C	EQOS_TM_EG_LTNCY	EQOS Time Stamp Egress Latency Register	0x00000000
0x31040B6C	EMAC0_TM_EG_LTNCY	EMAC0 Time Stamp Egress Latency Register	0x00000000
0x31040B70	EQOS_PPS_CTL	EQOS PPS Control Register	0x00000000
0x31040B70	EMAC0_PPS_CTL	EMAC0 PPS Control Register	0x00000000
0x31040B80	EQOS_PPS[n]_TGTM_SEC	EQOS PPS n Target Time Seconds Register	0x00000000
0x31040B80	EMAC0_PPS[n]_TGTM_SEC	EMAC0 PPS n Target Time Seconds Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040B84	EQOS_PPS[n]_TGTM_NSEC	EQOS PPS n Target Time Nanoseconds Register	0x00000000
0x31040B84	EMAC0_PPS[n]_TGTM_NSEC	EMAC0 PPS n Target Time Nanoseconds Register	0x00000000
0x31040B88	EQOS_PPS[n]_INTVL	EQOS PPS n Interval Register	0x00000000
0x31040B88	EMAC0_PPS[n]_INTVL	EMAC0 PPS n Interval Register	0x00000000
0x31040B8C	EQOS_PPS[n]_WID	EQOS PPS n Width Register	0x00000000
0x31040B8C	EMAC0_PPS[n]_WID	EMAC0 PPS n Width Register	0x00000000
0x31040B90	EQOS_PPS[n]_TGTM_SEC	EQOS PPS n Target Time Seconds Register	0x00000000
0x31040B90	EMAC0_PPS[n]_TGTM_SEC	EMAC0 PPS n Target Time Seconds Register	0x00000000
0x31040B94	EQOS_PPS[n]_TGTM_NSEC	EQOS PPS n Target Time Nanoseconds Register	0x00000000
0x31040B94	EMAC0_PPS[n]_TGTM_NSEC	EMAC0 PPS n Target Time Nanoseconds Register	0x00000000
0x31040B98	EQOS_PPS[n]_INTVL	EQOS PPS n Interval Register	0x00000000
0x31040B98	EMAC0_PPS[n]_INTVL	EMAC0 PPS n Interval Register	0x00000000
0x31040B9C	EQOS_PPS[n]_WID	EQOS PPS n Width Register	0x00000000
0x31040B9C	EMAC0_PPS[n]_WID	EMAC0 PPS n Width Register	0x00000000
0x31040BA0	EQOS_PPS[n]_TGTM_SEC	EQOS PPS n Target Time Seconds Register	0x00000000
0x31040BA0	EMAC0_PPS[n]_TGTM_SEC	EMAC0 PPS n Target Time Seconds Register	0x00000000
0x31040BA4	EQOS_PPS[n]_TGTM_NSEC	EQOS PPS n Target Time Nanoseconds Register	0x00000000
0x31040BA4	EMAC0_PPS[n]_TGTM_NSEC	EMAC0 PPS n Target Time Nanoseconds Register	0x00000000
0x31040BA8	EQOS_PPS[n]_INTVL	EQOS PPS n Interval Register	0x00000000
0x31040BA8	EMAC0_PPS[n]_INTVL	EMAC0 PPS n Interval Register	0x00000000
0x31040BAC	EQOS_PPS[n]_WID	EQOS PPS n Width Register	0x00000000
0x31040BAC	EMAC0_PPS[n]_WID	EMAC0 PPS n Width Register	0x00000000
0x31040BB0	EQOS_PPS[n]_TGTM_SEC	EQOS PPS n Target Time Seconds Register	0x00000000
0x31040BB0	EMAC0_PPS[n]_TGTM_SEC	EMAC0 PPS n Target Time Seconds Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040BB4	EQOS_PPS[n]_TGTM_NSEC	EQOS PPS n Target Time Nanoseconds Register	0x00000000
0x31040BB4	EMAC0_PPS[n]_TGTM_NSEC	EMAC0 PPS n Target Time Nanoseconds Register	0x00000000
0x31040BB8	EQOS_PPS[n]_INTVL	EQOS PPS n Interval Register	0x00000000
0x31040BB8	EMAC0_PPS[n]_INTVL	EMAC0 PPS n Interval Register	0x00000000
0x31040BBC	EQOS_PPS[n]_WID	EQOS PPS n Width Register	0x00000000
0x31040BBC	EMAC0_PPS[n]_WID	EMAC0 PPS n Width Register	0x00000000
0x31040BC0	EQOS_PTO_CTL	EQOS PTP Time Stamp Offload Control Register	0x00000000
0x31040BC0	EMAC0_PTO_CTL	EMAC0 PTP Time Stamp Offload Control Register	0x00000000
0x31040BC4	EQOS_SRCPRT_IDNTY0	EQOS Source Port Identity 0 Register	0x00000000
0x31040BC4	EMAC0_SRCPRT_IDNTY0	EMAC0 Source Port Identity 0 Register	0x00000000
0x31040BC8	EQOS_SRCPRT_IDNTY1	EQOS Source Port Identity 1 Register	0x00000000
0x31040BC8	EMAC0_SRCPRT_IDNTY1	EMAC0 Source Port Identity 1 Register	0x00000000
0x31040BCC	EQOS_SRCPRT_IDNTY2	EQOS Source Port Identity 2 Register	0x00000000
0x31040BCC	EMAC0_SRCPRT_IDNTY2	EMAC0 Source Port Identity 2 Register	0x00000000
0x31040BD0	EQOS_LOG_MSG_INTVL	EQOS PTP Log Message Interval Register	0x00000000
0x31040BD0	EMAC0_LOG_MSG_INTVL	EMAC0 PTP Log Message Interval Register	0x00000000
0x31040C00	EQOS_MTL_OPMODE	EQOS MTL Operation Mode Register	0x00000000
0x31040C00	EMAC0_MTL_OPMODE	EMAC0 MTL Operation Mode Register	0x00000000
0x31040C20	EQOS_MTL_ISTAT	EQOS MTL Interrupt Status Register	0x00000000
0x31040C20	EMAC0_MTL_ISTAT	EMAC0 MTL Interrupt Status Register	0x00000000
0x31040C30	EQOS_RXQ_DMA_MAP0	EQOS Receive Queue to DMA Channel Mapping Register 0	0x00000000
0x31040C30	EMAC0_RXQ_DMA_MAP0	EMAC0 Receive Queue to DMA Channel Mapping Register 0	0x00000000
0x31040C34	EQOS_RXQ_DMA_MAP1	EQOS Receive Queue to DMA Channel Mapping Register 1	0x00000000
0x31040C34	EMAC0_RXQ_DMA_MAP1	EMAC0 Receive Queue to DMA Channel Mapping Register 1	0x00000000
0x31040C40	EQOS_TBS_CTRL	EQOS MTL Time Based Scheduling Control Register	0x00000000
0x31040C40	EMAC0_TBS_CTRL	EMAC0 MTL Time Based Scheduling Control Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040C50	EQOS_EST_CTL	EQOS EST Control Register	0x00000000
0x31040C50	EMAC0_EST_CTL	EMAC0 EST Control Register	0x00000000
0x31040C54	EQOS_EST_EXT_CTL	EQOS EST Extended Control Register	0x00000000
0x31040C54	EMAC0_EST_EXT_CTL	EMAC0 EST Extended Control Register	0x00000000
0x31040C58	EQOS_EST_STAT	EQOS EST Status Register	0x00000000
0x31040C58	EMAC0_EST_STAT	EMAC0 EST Status Register	0x00000000
0x31040C60	EQOS_EST_SCH_ERR	EQOS EST Scheduling Error Register	0x00000000
0x31040C60	EMAC0_EST_SCH_ERR	EMAC0 EST Scheduling Error Register	0x00000000
0x31040C64	EQOS_EST_FRMSZ_ERR	EQOS EST Frame Size Error Register	0x00000000
0x31040C64	EMAC0_EST_FRMSZ_ERR	EMAC0 EST Frame Size Error Register	0x00000000
0x31040C68	EQOS_EST_FRMSZ_CAP	EQOS EST Frame Size Capture Register	0x00000000
0x31040C68	EMAC0_EST_FRMSZ_CAP	EMAC0 EST Frame Size Capture Register	0x00000000
0x31040C70	EQOS_EST_IEN	EQOS EST Interrupt Enable Register	0x00000000
0x31040C70	EMAC0_EST_IEN	EMAC0 EST Interrupt Enable Register	0x00000000
0x31040C80	EQOS_EST_GCL_CTL	EQOS EST Gate Control List Control Register	0x00000000
0x31040C80	EMAC0_EST_GCL_CTL	EMAC0 EST Gate Control List Control Register	0x00000000
0x31040C84	EQOS_EST_GCL_DAT	EQOS EST Gate Control List Data Register	0x00000000
0x31040C84	EMAC0_EST_GCL_DAT	EMAC0 EST Gate Control List Data Register	0x00000000
0x31040C90	EQOS_FPE_CTRLSTS	EQOS MTL Frame Preemption Control and Status Register	0x00000000
0x31040C90	EMAC0_FPE_CTRLSTS	EMAC0 MTL Frame Preemption Control and Status Register	0x00000000
0x31040C94	EQOS_FPE_ADV	EQOS Frame Preemption Advance Register	0x00000000
0x31040C94	EMAC0_FPE_ADV	EMAC0 Frame Preemption Advance Register	0x00000000
0x31040D00	EQOS_TQ0_OPMODE	EQOS Transmit Queue 0 Operation Mode Register	0x00000000
0x31040D00	EMAC0_TQ0_OPMODE	EMAC0 Transmit Queue 0 Operation Mode Register	0x00000000
0x31040D04	EQOS_TQ0_URFL	EQOS Transmit Queue 0 Underflow Packet Counter Register	0x00000000
0x31040D04	EMAC0_TQ0_URFL	EMAC0 Transmit Queue 0 Underflow Packet Counter Register	0x00000000
0x31040D08	EQOS_TQ0_DBG	EQOS Transmit Queue 0 Debug Register	0x00000000
0x31040D08	EMAC0_TQ0_DBG	EMAC0 Transmit Queue 0 Debug Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040D14	EQOS_TQ0_ETS_STAT	EQOS Transmit Queue 0 ETS Status Register	0x00000000
0x31040D14	EMAC0_TQ0_ETS_STAT	EMAC0 Transmit Queue 0 ETS Status Register	0x00000000
0x31040D18	EQOS_TQ0_QTMWGT	EQOS Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040D18	EMAC0_TQ0_QTMWGT	EMAC0 Transmit Queue 0 IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040D2C	EQOS_Q0_INT_CTLSTAT	EQOS Queue 0 Interrupt Control and Status Register	0x00000000
0x31040D2C	EMAC0_Q0_INT_CTLSTAT	EMAC0 Queue 0 Interrupt Control and Status Register	0x00000000
0x31040D30	EQOS_RQ0_OPMODE	EQOS Receive Queue 0 Operation Mode Register	0x00000000
0x31040D30	EMAC0_RQ0_OPMODE	EMAC0 Receive Queue 0 Operation Mode Register	0x00000000
0x31040D34	EQOS_RQ0_MSPKTOF_CNT	EQOS Receive Queue 0 Missed Packet and Overflow Counter Register	0x00000000
0x31040D34	EMAC0_RQ0_MSPKTOF_CNT	EMAC0 Receive Queue 0 Missed Packet and Overflow Counter Register	0x00000000
0x31040D38	EQOS_RQ0_DBG	EQOS Receive Queue 0 Debug Register	0x00000000
0x31040D38	EMAC0_RQ0_DBG	EMAC0 Receive Queue 0 Debug Register	0x00000000
0x31040D3C	EQOS_RQ0_CTL	EQOS Receive Queue Control Register	0x00000000
0x31040D3C	EMAC0_RQ0_CTL	EMAC0 Receive Queue Control Register	0x00000000
0x31040D40	EQOS_TQ[n]_OPMODE	EQOS Transmit Queue n Operation Mode Register	0x00000000
0x31040D40	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040D44	EQOS_TQ[n]_URFL	EQOS Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040D44	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040D48	EQOS_TQ[n]_DBG	EQOS Transmit Queue n Debug Register	0x00000000
0x31040D48	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040D50	EQOS_TQ[n]_ETS_CTL	EQOS Transmit Queue n ETS Control Register	0x00000000
0x31040D50	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040D54	EQOS_TQ[n]_ETS_STAT	EQOS Transmit Queue n ETS Status Register	0x00000000
0x31040D54	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040D58	EQOS_TQ[n]_QTMWGT	EQOS Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040D58	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040D5C	EQOS_TQ[n]_SSCRDT	EQOS Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040D5C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040D60	EQOS_TQ[n]_HICRDT	EQOS Transmit Queue n CBS HiCredit Register	0x00000000
0x31040D60	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040D64	EQOS_TQ[n]_LOCRDT	EQOS Transmit Queue n CBS LoCredit Register	0x00000000
0x31040D64	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040D6C	EQOS_Q[n]_INT_CTLSTAT	EQOS Queue n Interrupt Control and Status Register	0x00000000
0x31040D6C	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040D70	EQOS_RQ[n]_OPMODE	EQOS Receive Queue n Operation Mode Register	0x00000000
0x31040D70	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040D74	EQOS_RQ[n]_MSPKTOF_CNT	EQOS Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040D74	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040D78	EQOS_RQ[n]_DBG	EQOS Receive Queue n Debug Register	0x00000000
0x31040D78	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040D7C	EQOS_RQ[n]_CTL	EQOS Receive Queue n Control Register	0x00000000
0x31040D7C	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040D80	EQOS_TQ[n]_OPMODE	EQOS Transmit Queue n Operation Mode Register	0x00000000
0x31040D80	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040D84	EQOS_TQ[n]_URFL	EQOS Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040D84	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040D88	EQOS_TQ[n]_DBG	EQOS Transmit Queue n Debug Register	0x00000000
0x31040D88	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040D90	EQOS_TQ[n]_ETS_CTL	EQOS Transmit Queue n ETS Control Register	0x00000000
0x31040D90	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040D94	EQOS_TQ[n]_ETS_STAT	EQOS Transmit Queue n ETS Status Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040D94	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040D98	EQOS_TQ[n]_QTMWGT	EQOS Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040D98	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040D9C	EQOS_TQ[n]_SSCRDT	EQOS Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040D9C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040DA0	EQOS_TQ[n]_HICRDT	EQOS Transmit Queue n CBS HiCredit Register	0x00000000
0x31040DA0	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040DA4	EQOS_TQ[n]_LOCRDT	EQOS Transmit Queue n CBS LoCredit Register	0x00000000
0x31040DA4	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040DAC	EQOS_Q[n]_INT_CTLSTAT	EQOS Queue n Interrupt Control and Status Register	0x00000000
0x31040DAC	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040DB0	EQOS_RQ[n]_OPMODE	EQOS Receive Queue n Operation Mode Register	0x00000000
0x31040DB0	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040DB4	EQOS_RQ[n]_MSPKTOF_CNT	EQOS Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040DB4	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040DB8	EQOS_RQ[n]_DBG	EQOS Receive Queue n Debug Register	0x00000000
0x31040DB8	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040DBC	EQOS_RQ[n]_CTL	EQOS Receive Queue n Control Register	0x00000000
0x31040DBC	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040DC0	EQOS_TQ[n]_OPMODE	EQOS Transmit Queue n Operation Mode Register	0x00000000
0x31040DC0	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040DC4	EQOS_TQ[n]_URFL	EQOS Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040DC4	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040DC8	EQOS_TQ[n]_DBG	EQOS Transmit Queue n Debug Register	0x00000000
0x31040DC8	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040DD0	EQOS_TQ[n]_ETS_CTL	EQOS Transmit Queue n ETS Control Register	0x00000000
0x31040DD0	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040DD4	EQOS_TQ[n]_ETS_STAT	EQOS Transmit Queue n ETS Status Register	0x00000000
0x31040DD4	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040DD8	EQOS_TQ[n]_QTMWGT	EQOS Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040DD8	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040DDC	EQOS_TQ[n]_SSCRDT	EQOS Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040DDC	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040DE0	EQOS_TQ[n]_HICRDT	EQOS Transmit Queue n CBS HiCredit Register	0x00000000
0x31040DE0	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040DE4	EQOS_TQ[n]_LOCRDT	EQOS Transmit Queue n CBS LoCredit Register	0x00000000
0x31040DE4	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040DEC	EQOS_Q[n]_INT_CTLSTAT	EQOS Queue n Interrupt Control and Status Register	0x00000000
0x31040DEC	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040DF0	EQOS_RQ[n]_OPMODE	EQOS Receive Queue n Operation Mode Register	0x00000000
0x31040DF0	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040DF4	EQOS_RQ[n]_MSPKTOF_CNT	EQOS Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040DF4	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040DF8	EQOS_RQ[n]_DBG	EQOS Receive Queue n Debug Register	0x00000000
0x31040DF8	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040DFC	EQOS_RQ[n]_CTL	EQOS Receive Queue n Control Register	0x00000000
0x31040DFC	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040E00	EQOS_TQ[n]_OPMODE	EQOS Transmit Queue n Operation Mode Register	0x00000000
0x31040E00	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040E04	EQOS_TQ[n]_URFL	EQOS Transmit Queue n Underflow Packet Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040E04	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040E08	EQOS_TQ[n]_DBG	EQOS Transmit Queue n Debug Register	0x00000000
0x31040E08	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040E10	EQOS_TQ[n]_ETS_CTL	EQOS Transmit Queue n ETS Control Register	0x00000000
0x31040E10	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040E14	EQOS_TQ[n]_ETS_STAT	EQOS Transmit Queue n ETS Status Register	0x00000000
0x31040E14	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040E18	EQOS_TQ[n]_QTMWGT	EQOS Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040E18	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040E1C	EQOS_TQ[n]_SSCRDT	EQOS Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040E1C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040E20	EQOS_TQ[n]_HICRDT	EQOS Transmit Queue n CBS HiCredit Register	0x00000000
0x31040E20	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040E24	EQOS_TQ[n]_LOCRDT	EQOS Transmit Queue n CBS LoCredit Register	0x00000000
0x31040E24	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040E2C	EQOS_Q[n]_INT_CTLSTAT	EQOS Queue n Interrupt Control and Status Register	0x00000000
0x31040E2C	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040E30	EQOS_RQ[n]_OPMODE	EQOS Receive Queue n Operation Mode Register	0x00000000
0x31040E30	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040E34	EQOS_RQ[n]_MSPKTOF_CNT	EQOS Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040E34	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040E38	EQOS_RQ[n]_DBG	EQOS Receive Queue n Debug Register	0x00000000
0x31040E38	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040E3C	EQOS_RQ[n]_CTL	EQOS Receive Queue n Control Register	0x00000000
0x31040E3C	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040E40	EQOS_TQ[n]_OPMODE	EQOS Transmit Queue n Operation Mode Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040E40	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040E44	EQOS_TQ[n]_URFL	EQOS Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040E44	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040E48	EQOS_TQ[n]_DBG	EQOS Transmit Queue n Debug Register	0x00000000
0x31040E48	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040E50	EQOS_TQ[n]_ETS_CTL	EQOS Transmit Queue n ETS Control Register	0x00000000
0x31040E50	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040E54	EQOS_TQ[n]_ETS_STAT	EQOS Transmit Queue n ETS Status Register	0x00000000
0x31040E54	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040E58	EQOS_TQ[n]_QTMWGT	EQOS Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040E58	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040E5C	EQOS_TQ[n]_SSCRDT	EQOS Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040E5C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040E60	EQOS_TQ[n]_HICRDT	EQOS Transmit Queue n CBS HiCredit Register	0x00000000
0x31040E60	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040E64	EQOS_TQ[n]_LOCRDT	EQOS Transmit Queue n CBS LoCredit Register	0x00000000
0x31040E64	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040E6C	EQOS_Q[n]_INT_CTLSTAT	EQOS Queue n Interrupt Control and Status Register	0x00000000
0x31040E6C	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040E70	EQOS_RQ[n]_OPMODE	EQOS Receive Queue n Operation Mode Register	0x00000000
0x31040E70	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040E74	EQOS_RQ[n]_MSPKTOF_CNT	EQOS Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040E74	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040E78	EQOS_RQ[n]_DBG	EQOS Receive Queue n Debug Register	0x00000000
0x31040E78	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040E7C	EQOS_RQ[n]_CTL	EQOS Receive Queue n Control Register	0x00000000
0x31040E7C	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040E80	EQOS_TQ[n]_OPMODE	EQOS Transmit Queue n Operation Mode Register	0x00000000
0x31040E80	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040E84	EQOS_TQ[n]_URFL	EQOS Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040E84	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040E88	EQOS_TQ[n]_DBG	EQOS Transmit Queue n Debug Register	0x00000000
0x31040E88	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040E90	EQOS_TQ[n]_ETS_CTL	EQOS Transmit Queue n ETS Control Register	0x00000000
0x31040E90	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040E94	EQOS_TQ[n]_ETS_STAT	EQOS Transmit Queue n ETS Status Register	0x00000000
0x31040E94	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040E98	EQOS_TQ[n]_QTMWGT	EQOS Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040E98	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040E9C	EQOS_TQ[n]_SSCRDT	EQOS Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040E9C	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040EA0	EQOS_TQ[n]_HICRDT	EQOS Transmit Queue n CBS HiCredit Register	0x00000000
0x31040EA0	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040EA4	EQOS_TQ[n]_LOCRDT	EQOS Transmit Queue n CBS LoCredit Register	0x00000000
0x31040EA4	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040EAC	EQOS_Q[n]_INT_CTLSTAT	EQOS Queue n Interrupt Control and Status Register	0x00000000
0x31040EAC	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040EB0	EQOS_RQ[n]_OPMODE	EQOS Receive Queue n Operation Mode Register	0x00000000
0x31040EB0	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040EB4	EQOS_RQ[n]_MSPKTOF_CNT	EQOS Receive Queue n Missed Packet and Overflow Counter Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040EB4	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040EB8	EQOS_RQ[n]_DBG	EQOS Receive Queue n Debug Register	0x00000000
0x31040EB8	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040EBC	EQOS_RQ[n]_CTL	EQOS Receive Queue n Control Register	0x00000000
0x31040EBC	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31040EC0	EQOS_TQ[n]_OPMODE	EQOS Transmit Queue n Operation Mode Register	0x00000000
0x31040EC0	EMAC0_TQ[n]_OPMODE	EMAC0 Transmit Queue n Operation Mode Register	0x00000000
0x31040EC4	EQOS_TQ[n]_URFL	EQOS Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040EC4	EMAC0_TQ[n]_URFL	EMAC0 Transmit Queue n Underflow Packet Counter Register	0x00000000
0x31040EC8	EQOS_TQ[n]_DBG	EQOS Transmit Queue n Debug Register	0x00000000
0x31040EC8	EMAC0_TQ[n]_DBG	EMAC0 Transmit Queue n Debug Register	0x00000000
0x31040ED0	EQOS_TQ[n]_ETS_CTL	EQOS Transmit Queue n ETS Control Register	0x00000000
0x31040ED0	EMAC0_TQ[n]_ETS_CTL	EMAC0 Transmit Queue n ETS Control Register	0x00000000
0x31040ED4	EQOS_TQ[n]_ETS_STAT	EQOS Transmit Queue n ETS Status Register	0x00000000
0x31040ED4	EMAC0_TQ[n]_ETS_STAT	EMAC0 Transmit Queue n ETS Status Register	0x00000000
0x31040ED8	EQOS_TQ[n]_QTMWGT	EQOS Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040ED8	EMAC0_TQ[n]_QTMWGT	EMAC0 Transmit Queue n IdleSlopeCredit, Quantum or Weights Register	0x00000000
0x31040EDC	EQOS_TQ[n]_SSCRDT	EQOS Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040EDC	EMAC0_TQ[n]_SSCRDT	EMAC0 Transmit Queue n SendSlopeCredit Register	0x00000000
0x31040EE0	EQOS_TQ[n]_HICRDT	EQOS Transmit Queue n CBS HiCredit Register	0x00000000
0x31040EE0	EMAC0_TQ[n]_HICRDT	EMAC0 Transmit Queue n CBS HiCredit Register	0x00000000
0x31040EE4	EQOS_TQ[n]_LOCRDT	EQOS Transmit Queue n CBS LoCredit Register	0x00000000
0x31040EE4	EMAC0_TQ[n]_LOCRDT	EMAC0 Transmit Queue n CBS LoCredit Register	0x00000000
0x31040EEC	EQOS_Q[n]_INT_CTLSTAT	EQOS Queue n Interrupt Control and Status Register	0x00000000
0x31040EEC	EMAC0_Q[n]_INT_CTLSTAT	EMAC0 Queue n Interrupt Control and Status Register	0x00000000
0x31040EF0	EQOS_RQ[n]_OPMODE	EQOS Receive Queue n Operation Mode Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31040EF0	EMAC0_RQ[n]_OPMODE	EMAC0 Receive Queue n Operation Mode Register	0x00000000
0x31040EF4	EQOS_RQ[n]_MSPKTOF_CNT	EQOS Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040EF4	EMAC0_RQ[n]_MSPKTOF_CNT	EMAC0 Receive Queue n Missed Packet and Overflow Counter Register	0x00000000
0x31040EF8	EQOS_RQ[n]_DBG	EQOS Receive Queue n Debug Register	0x00000000
0x31040EF8	EMAC0_RQ[n]_DBG	EMAC0 Receive Queue n Debug Register	0x00000000
0x31040EFC	EQOS_RQ[n]_CTL	EQOS Receive Queue n Control Register	0x00000000
0x31040EFC	EMAC0_RQ[n]_CTL	EMAC0 Receive Queue n Control Register	0x00000000
0x31041000	EQOS_DMA_MODE	EQOS DMA Mode Register	0x00000000
0x31041000	EMAC0_DMA_MODE	EMAC0 DMA Mode Register	0x00000000
0x31041004	EQOS_DMA_SYSBMODE	EQOS DMA System Bus Mode Register	0x01010000
0x31041004	EMAC0_DMA_SYSBMODE	EMAC0 DMA System Bus Mode Register	0x01010000
0x31041008	EQOS_DMA_ISTAT	EQOS DMA Interrupt Status Register	0x00000000
0x31041008	EMAC0_DMA_ISTAT	EMAC0 DMA Interrupt Status Register	0x00000000
0x3104100C	EQOS_DMA_DBG_STAT0	EQOS DMA Debug Status 0 Register	0x00000000
0x3104100C	EMAC0_DMA_DBG_STAT0	EMAC0 DMA Debug Status 0 Register	0x00000000
0x31041010	EQOS_DMA_DBG_STAT1	EQOS DMA Debug Status 1 Register	0x00000000
0x31041010	EMAC0_DMA_DBG_STAT1	EMAC0 DMA Debug Status 1 Register	0x00000000
0x31041014	EQOS_DMA_DBG_STAT2	EQOS DMA Debug Status 2 Register	0x00000000
0x31041014	EMAC0_DMA_DBG_STAT2	EMAC0 DMA Debug Status 2 Register	0x00000000
0x31041050	EQOS_DMA_TBS_CTL0	EQOS DMA TBS Control 0 Register	0x00000000
0x31041050	EMAC0_DMA_TBS_CTL0	EMAC0 DMA TBS Control 0 Register	0x00000000
0x31041054	EQOS_DMA_TBS_CTL1	EQOS DMA TBS Control 1 Register	0x00000000
0x31041054	EMAC0_DMA_TBS_CTL1	EMAC0 DMA TBS Control 1 Register	0x00000000
0x31041058	EQOS_DMA_TBS_CTL2	EQOS DMA TBS Control 2 Register	0x00000000
0x31041058	EMAC0_DMA_TBS_CTL2	EMAC0 DMA TBS Control 2 Register	0x00000000
0x3104105C	EQOS_DMA_TBS_CTL3	EQOS DMA TBS Control 3 Register	0x00000000
0x3104105C	EMAC0_DMA_TBS_CTL3	EMAC0 DMA TBS Control 3 Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041100	EQOS_DMA[n]_CTL	EQOS DMA Channel n Control Register	0x00000000
0x31041100	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041104	EQOS_DMA[n]_TXCTL	EQOS DMA Channel n Transmit Control Register	0x00000000
0x31041104	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041108	EQOS_DMA[n]_RXCTL	EQOS DMA Channel n Receive Control Register	0x00000000
0x31041108	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041114	EQOS_DMA[n]_TXDSC_ADDR	EQOS DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x31041114	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104111C	EQOS_DMA[n]_RXDSC_ADDR	EQOS DMA Channel n Receive Descriptor List Address Register	0x00000000
0x3104111C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x31041120	EQOS_DMA[n]_TXDSC_TLPTR	EQOS DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041120	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041128	EQOS_DMA[n]_RXDSC_TLPTR	EQOS DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x31041128	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x3104112C	EQOS_DMA[n]_TXDSC_RLEN	EQOS DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x3104112C	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x31041130	EQOS_DMA[n]_RXCTL2	EQOS DMA Channel n Extended Receive Control Register	0x00000000
0x31041130	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x31041134	EQOS_DMA[n]_IEN	EQOS DMA Channel n Interrupt Enable Register	0x00000000
0x31041134	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x31041138	EQOS_DMA[n]_RXINTWDTMR	EQOS DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041138	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x3104113C	EQOS_DMA[n]_SFN_CTLSTAT	EQOS DMA Channel n Slot Function Control and Status register	0x000007C0
0x3104113C	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x31041144	EQOS_DMA[n]_TXDSC_CUR	EQOS DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x31041144	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x3104114C	EQOS_DMA[n]_RXDSC_CUR	EQOS DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x3104114C	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x31041154	EQOS_DMA[n]_TXBUF_CUR	EQOS DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x31041154	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x3104115C	EQOS_DMA[n]_RXBUF_CUR	EQOS DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x3104115C	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x31041160	EQOS_DMA[n]_STAT	EQOS DMA Channel n Status Register	0x00000000
0x31041160	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x31041164	EQOS_DMA[n]_MISSFRM_CNT	EQOS DMA Channel n Missed Frame Counter Register	0x00000000
0x31041164	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041180	EQOS_DMA[n]_CTL	EQOS DMA Channel n Control Register	0x00000000
0x31041180	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041184	EQOS_DMA[n]_TXCTL	EQOS DMA Channel n Transmit Control Register	0x00000000
0x31041184	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041188	EQOS_DMA[n]_RXCTL	EQOS DMA Channel n Receive Control Register	0x00000000
0x31041188	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041194	EQOS_DMA[n]_TXDSC_ADDR	EQOS DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x31041194	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104119C	EQOS_DMA[n]_RXDSC_ADDR	EQOS DMA Channel n Receive Descriptor List Address Register	0x00000000
0x3104119C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x310411A0	EQOS_DMA[n]_TXDSC_TLPTR	EQOS DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310411A0	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310411A8	EQOS_DMA[n]_RXDSC_TLPTR	EQOS DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x310411A8	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x310411AC	EQOS_DMA[n]_TXDSC_RLEN	EQOS DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310411AC	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310411B0	EQOS_DMA[n]_RXCTL2	EQOS DMA Channel n Extended Receive Control Register	0x00000000
0x310411B0	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x310411B4	EQOS_DMA[n]_IEN	EQOS DMA Channel n Interrupt Enable Register	0x00000000
0x310411B4	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x310411B8	EQOS_DMA[n]_RXINTWDTMR	EQOS DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310411B8	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310411BC	EQOS_DMA[n]_SFN_CTLSTAT	EQOS DMA Channel n Slot Function Control and Status register	0x000007C0
0x310411BC	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x310411C4	EQOS_DMA[n]_TXDSC_CUR	EQOS DMA Channel n Current Application Transmit Descriptor Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310411C4	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310411CC	EQOS_DMA[n]_RXDSC_CUR	EQOS DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310411CC	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310411D4	EQOS_DMA[n]_TXBUF_CUR	EQOS DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310411D4	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310411DC	EQOS_DMA[n]_RXBUF_CUR	EQOS DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310411DC	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310411E0	EQOS_DMA[n]_STAT	EQOS DMA Channel n Status Register	0x00000000
0x310411E0	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x310411E4	EQOS_DMA[n]_MISSFRM_CNT	EQOS DMA Channel n Missed Frame Counter Register	0x00000000
0x310411E4	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041200	EQOS_DMA[n]_CTL	EQOS DMA Channel n Control Register	0x00000000
0x31041200	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041204	EQOS_DMA[n]_TXCTL	EQOS DMA Channel n Transmit Control Register	0x00000000
0x31041204	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041208	EQOS_DMA[n]_RXCTL	EQOS DMA Channel n Receive Control Register	0x00000000
0x31041208	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041214	EQOS_DMA[n]_TXDSC_ADDR	EQOS DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x31041214	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104121C	EQOS_DMA[n]_RXDSC_ADDR	EQOS DMA Channel n Receive Descriptor List Address Register	0x00000000
0x3104121C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041220	EQOS_DMA[n]_TXDSC_TLPTR	EQOS DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041220	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041228	EQOS_DMA[n]_RXDSC_TLPTR	EQOS DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x31041228	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x3104122C	EQOS_DMA[n]_TXDSC_RLEN	EQOS DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x3104122C	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x31041230	EQOS_DMA[n]_RXCTL2	EQOS DMA Channel n Extended Receive Control Register	0x00000000
0x31041230	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x31041234	EQOS_DMA[n]_IEN	EQOS DMA Channel n Interrupt Enable Register	0x00000000
0x31041234	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x31041238	EQOS_DMA[n]_RXINTWDTMR	EQOS DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x31041238	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x3104123C	EQOS_DMA[n]_SFN_CTLSTAT	EQOS DMA Channel n Slot Function Control and Status register	0x000007C0
0x3104123C	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x31041244	EQOS_DMA[n]_TXDSC_CUR	EQOS DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x31041244	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x3104124C	EQOS_DMA[n]_RXDSC_CUR	EQOS DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x3104124C	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x31041254	EQOS_DMA[n]_TXBUF_CUR	EQOS DMA Channel n Current Application Transmit Buffer Address Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041254	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x3104125C	EQOS_DMA[n]_RXBUF_CUR	EQOS DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x3104125C	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x31041260	EQOS_DMA[n]_STAT	EQOS DMA Channel n Status Register	0x00000000
0x31041260	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x31041264	EQOS_DMA[n]_MISSFRM_CNT	EQOS DMA Channel n Missed Frame Counter Register	0x00000000
0x31041264	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041280	EQOS_DMA[n]_CTL	EQOS DMA Channel n Control Register	0x00000000
0x31041280	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041284	EQOS_DMA[n]_TXCTL	EQOS DMA Channel n Transmit Control Register	0x00000000
0x31041284	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041288	EQOS_DMA[n]_RXCTL	EQOS DMA Channel n Receive Control Register	0x00000000
0x31041288	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041294	EQOS_DMA[n]_TXDSC_ADDR	EQOS DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x31041294	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104129C	EQOS_DMA[n]_RXDSC_ADDR	EQOS DMA Channel n Receive Descriptor List Address Register	0x00000000
0x3104129C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x310412A0	EQOS_DMA[n]_TXDSC_TLPTR	EQOS DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310412A0	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310412A8	EQOS_DMA[n]_RXDSC_TLPTR	EQOS DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x310412A8	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310412AC	EQOS_DMA[n]_TXDSC_RLEN	EQOS DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310412AC	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310412B0	EQOS_DMA[n]_RXCTL2	EQOS DMA Channel n Extended Receive Control Register	0x00000000
0x310412B0	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x310412B4	EQOS_DMA[n]_IEN	EQOS DMA Channel n Interrupt Enable Register	0x00000000
0x310412B4	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x310412B8	EQOS_DMA[n]_RXINTWDTMR	EQOS DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310412B8	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310412BC	EQOS_DMA[n]_SFN_CTLSTAT	EQOS DMA Channel n Slot Function Control and Status register	0x000007C0
0x310412BC	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x310412C4	EQOS_DMA[n]_TXDSC_CUR	EQOS DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310412C4	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310412CC	EQOS_DMA[n]_RXDSC_CUR	EQOS DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310412CC	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310412D4	EQOS_DMA[n]_TXBUF_CUR	EQOS DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310412D4	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310412DC	EQOS_DMA[n]_RXBUF_CUR	EQOS DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310412DC	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310412E0	EQOS_DMA[n]_STAT	EQOS DMA Channel n Status Register	0x00000000
0x310412E0	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310412E4	EQOS_DMA[n]_MISSFRM_CNT	EQOS DMA Channel n Missed Frame Counter Register	0x00000000
0x310412E4	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041300	EQOS_DMA[n]_CTL	EQOS DMA Channel n Control Register	0x00000000
0x31041300	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041304	EQOS_DMA[n]_TXCTL	EQOS DMA Channel n Transmit Control Register	0x00000000
0x31041304	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041308	EQOS_DMA[n]_RXCTL	EQOS DMA Channel n Receive Control Register	0x00000000
0x31041308	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041314	EQOS_DMA[n]_TXDSC_ADDR	EQOS DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x31041314	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104131C	EQOS_DMA[n]_RXDSC_ADDR	EQOS DMA Channel n Receive Descriptor List Address Register	0x00000000
0x3104131C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x31041320	EQOS_DMA[n]_TXDSC_TLPTR	EQOS DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041320	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041328	EQOS_DMA[n]_RXDSC_TLPTR	EQOS DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x31041328	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x3104132C	EQOS_DMA[n]_TXDSC_RLEN	EQOS DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x3104132C	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x31041330	EQOS_DMA[n]_RXCTL2	EQOS DMA Channel n Extended Receive Control Register	0x00000000
0x31041330	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x31041334	EQOS_DMA[n]_IEN	EQOS DMA Channel n Interrupt Enable Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041334	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x31041338	EQOS_DMA[n]_RXINTWDTMR	EQOS DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x31041338	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x3104133C	EQOS_DMA[n]_SFN_CTLSTAT	EQOS DMA Channel n Slot Function Control and Status register	0x000007C0
0x3104133C	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x31041344	EQOS_DMA[n]_TXDSC_CUR	EQOS DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x31041344	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x3104134C	EQOS_DMA[n]_RXDSC_CUR	EQOS DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x3104134C	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x31041354	EQOS_DMA[n]_TXBUF_CUR	EQOS DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x31041354	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x3104135C	EQOS_DMA[n]_RXBUF_CUR	EQOS DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x3104135C	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x31041360	EQOS_DMA[n]_STAT	EQOS DMA Channel n Status Register	0x00000000
0x31041360	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x31041364	EQOS_DMA[n]_MISSFRM_CNT	EQOS DMA Channel n Missed Frame Counter Register	0x00000000
0x31041364	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041380	EQOS_DMA[n]_CTL	EQOS DMA Channel n Control Register	0x00000000
0x31041380	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041384	EQOS_DMA[n]_TXCTL	EQOS DMA Channel n Transmit Control Register	0x00000000
0x31041384	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041388	EQOS_DMA[n]_RXCTL	EQOS DMA Channel n Receive Control Register	0x00000000
0x31041388	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041394	EQOS_DMA[n]_TXDSC_ADDR	EQOS DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x31041394	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104139C	EQOS_DMA[n]_RXDSC_ADDR	EQOS DMA Channel n Receive Descriptor List Address Register	0x00000000
0x3104139C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x310413A0	EQOS_DMA[n]_TXDSC_TLPTR	EQOS DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310413A0	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310413A8	EQOS_DMA[n]_RXDSC_TLPTR	EQOS DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x310413A8	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x310413AC	EQOS_DMA[n]_TXDSC_RLEN	EQOS DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310413AC	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310413B0	EQOS_DMA[n]_RXCTL2	EQOS DMA Channel n Extended Receive Control Register	0x00000000
0x310413B0	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x310413B4	EQOS_DMA[n]_IEN	EQOS DMA Channel n Interrupt Enable Register	0x00000000
0x310413B4	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x310413B8	EQOS_DMA[n]_RXINTWDTMR	EQOS DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310413B8	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310413BC	EQOS_DMA[n]_SFN_CTLSTAT	EQOS DMA Channel n Slot Function Control and Status register	0x000007C0
0x310413BC	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310413C4	EQOS_DMA[n]_TXDSC_CUR	EQOS DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310413C4	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310413CC	EQOS_DMA[n]_RXDSC_CUR	EQOS DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310413CC	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310413D4	EQOS_DMA[n]_TXBUF_CUR	EQOS DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310413D4	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310413DC	EQOS_DMA[n]_RXBUF_CUR	EQOS DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310413DC	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310413E0	EQOS_DMA[n]_STAT	EQOS DMA Channel n Status Register	0x00000000
0x310413E0	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x310413E4	EQOS_DMA[n]_MISSFRM_CNT	EQOS DMA Channel n Missed Frame Counter Register	0x00000000
0x310413E4	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041400	EQOS_DMA[n]_CTL	EQOS DMA Channel n Control Register	0x00000000
0x31041400	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041404	EQOS_DMA[n]_TXCTL	EQOS DMA Channel n Transmit Control Register	0x00000000
0x31041404	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041408	EQOS_DMA[n]_RXCTL	EQOS DMA Channel n Receive Control Register	0x00000000
0x31041408	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041414	EQOS_DMA[n]_TXDSC_ADDR	EQOS DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x31041414	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104141C	EQOS_DMA[n]_RXDSC_ADDR	EQOS DMA Channel n Receive Descriptor List Address Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3104141C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x31041420	EQOS_DMA[n]_TXDSC_TLPTR	EQOS DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041420	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31041428	EQOS_DMA[n]_RXDSC_TLPTR	EQOS DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x31041428	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x3104142C	EQOS_DMA[n]_TXDSC_RLEN	EQOS DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x3104142C	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x31041430	EQOS_DMA[n]_RXCTL2	EQOS DMA Channel n Extended Receive Control Register	0x00000000
0x31041430	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x31041434	EQOS_DMA[n]_IEN	EQOS DMA Channel n Interrupt Enable Register	0x00000000
0x31041434	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x31041438	EQOS_DMA[n]_RXINTWDTMR	EQOS DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x31041438	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x3104143C	EQOS_DMA[n]_SFN_CTLSTAT	EQOS DMA Channel n Slot Function Control and Status register	0x000007C0
0x3104143C	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x31041444	EQOS_DMA[n]_TXDSC_CUR	EQOS DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x31041444	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x3104144C	EQOS_DMA[n]_RXDSC_CUR	EQOS DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x3104144C	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31041454	EQOS_DMA[n]_TXBUF_CUR	EQOS DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x31041454	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x3104145C	EQOS_DMA[n]_RXBUF_CUR	EQOS DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x3104145C	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x31041460	EQOS_DMA[n]_STAT	EQOS DMA Channel n Status Register	0x00000000
0x31041460	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x31041464	EQOS_DMA[n]_MISSFRM_CNT	EQOS DMA Channel n Missed Frame Counter Register	0x00000000
0x31041464	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000
0x31041480	EQOS_DMA[n]_CTL	EQOS DMA Channel n Control Register	0x00000000
0x31041480	EMAC0_DMA[n]_CTL	EMAC0 DMA Channel n Control Register	0x00000000
0x31041484	EQOS_DMA[n]_TXCTL	EQOS DMA Channel n Transmit Control Register	0x00000000
0x31041484	EMAC0_DMA[n]_TXCTL	EMAC0 DMA Channel n Transmit Control Register	0x00000000
0x31041488	EQOS_DMA[n]_RXCTL	EQOS DMA Channel n Receive Control Register	0x00000000
0x31041488	EMAC0_DMA[n]_RXCTL	EMAC0 DMA Channel n Receive Control Register	0x00000000
0x31041494	EQOS_DMA[n]_TXDSC_ADDR	EQOS DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x31041494	EMAC0_DMA[n]_TXDSC_ADDR	EMAC0 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104149C	EQOS_DMA[n]_RXDSC_ADDR	EQOS DMA Channel n Receive Descriptor List Address Register	0x00000000
0x3104149C	EMAC0_DMA[n]_RXDSC_ADDR	EMAC0 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x310414A0	EQOS_DMA[n]_TXDSC_TLPTR	EQOS DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310414A0	EMAC0_DMA[n]_TXDSC_TLPTR	EMAC0 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x310414A8	EQOS_DMA[n]_RXDSC_TLPTR	EQOS DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310414A8	EMAC0_DMA[n]_RXDSC_TLPTR	EMAC0 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x310414AC	EQOS_DMA[n]_TXDSC_RLEN	EQOS DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310414AC	EMAC0_DMA[n]_TXDSC_RLEN	EMAC0 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x310414B0	EQOS_DMA[n]_RXCTL2	EQOS DMA Channel n Extended Receive Control Register	0x00000000
0x310414B0	EMAC0_DMA[n]_RXCTL2	EMAC0 DMA Channel n Extended Receive Control Register	0x00000000
0x310414B4	EQOS_DMA[n]_IEN	EQOS DMA Channel n Interrupt Enable Register	0x00000000
0x310414B4	EMAC0_DMA[n]_IEN	EMAC0 DMA Channel n Interrupt Enable Register	0x00000000
0x310414B8	EQOS_DMA[n]_RXINTWDTMR	EQOS DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310414B8	EMAC0_DMA[n]_RXINTWDTMR	EMAC0 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x310414BC	EQOS_DMA[n]_SFN_CTLSTAT	EQOS DMA Channel n Slot Function Control and Status register	0x000007C0
0x310414BC	EMAC0_DMA[n]_SFN_CTLSTAT	EMAC0 DMA Channel n Slot Function Control and Status register	0x000007C0
0x310414C4	EQOS_DMA[n]_TXDSC_CUR	EQOS DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310414C4	EMAC0_DMA[n]_TXDSC_CUR	EMAC0 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x310414CC	EQOS_DMA[n]_RXDSC_CUR	EQOS DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310414CC	EMAC0_DMA[n]_RXDSC_CUR	EMAC0 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x310414D4	EQOS_DMA[n]_TXBUF_CUR	EQOS DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310414D4	EMAC0_DMA[n]_TXBUF_CUR	EMAC0 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x310414DC	EQOS_DMA[n]_RXBUF_CUR	EQOS DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x310414DC	EMAC0_DMA[n]_RXBUF_CUR	EMAC0 DMA Channel n Current Application Receive Buffer Address Register	0x00000000

Table A-87: ADSP-SC59x EQOS MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310414E0	EQOS_DMA[n]_STAT	EQOS DMA Channel n Status Register	0x00000000
0x310414E0	EMAC0_DMA[n]_STAT	EMAC0 DMA Channel n Status Register	0x00000000
0x310414E4	EQOS_DMA[n]_MISSFRM_CNT	EQOS DMA Channel n Missed Frame Counter Register	0x00000000
0x310414E4	EMAC0_DMA[n]_MISSFRM_CNT	EMAC0 DMA Channel n Missed Frame Counter Register	0x00000000

Table A-88: ADSP-SC59x EMAC1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31042000	EMAC1_MAC_CFG	EMAC1 MAC Configuration Register	0x00008000
0x31042004	EMAC1_MAC_EXT_CFG	EMAC1 MAC Extended Configuration Register	0x00000000
0x31042008	EMAC1_MACPKT_FILT	EMAC1 MAC Packet Filter Register	0x00000000
0x3104200C	EMAC1_WDT_TMOUT	EMAC1 Watchdog Timeout Register	0x00000000
0x31042010	EMAC1_HASHTBL_REG0	EMAC1 Hash Table Register 0	0x00000000
0x31042014	EMAC1_HASHTBL_REG1	EMAC1 Hash Table Register 1	0x00000000
0x31042050	EMAC1_VLANTAG_CTL	EMAC1 VLAN Tag Control Register	0x00000000
0x31042054	EMAC1_VLANTAG_DAT	EMAC1 VLAN Tag Indirect Access Data Register	0x00000000
0x31042058	EMAC1_VLAN_HASHTBL	EMAC1 VLAN Hash Table Register	0x00000000
0x31042060	EMAC1_VLAN_INCL	EMAC1 VLAN Tag Inclusion or Replacement Register	0x00000000
0x31042070	EMAC1_Q0_TXFLOW_CTL	EMAC1 Queue 0 Transmit Flow Control Register	0x00000000
0x31042090	EMAC1_RXFLOW_CTL	EMAC1 Receive Flow Control Register	0x00000000
0x310420B0	EMAC1_MAC_ISTAT	EMAC1 MAC Interrupt Status Register	0x00000000
0x310420B4	EMAC1_MAC_IEN	EMAC1 MAC Interrupt Enable Register	0x00000000
0x310420B8	EMAC1_RXTX_STAT	EMAC1 Receive Transmit Status Register	0x00000000
0x31042114	EMAC1_MAC_DBG	EMAC1 Debug Register	0x00000000
0x3104211C	EMAC1_HW_FTR_PSNT0	EMAC1 Hardware Feature Presence Register 0	0x480541F5
0x31042120	EMAC1_HW_FTR_PSNT1	EMAC1 Hardware Feature Presence Register 1	0x090000C2
0x31042124	EMAC1_HW_FTR_PSNT2	EMAC1 Hardware Feature Presence Register 2	0x00000000
0x31042128	EMAC1_HW_FTR_PSNT3	EMAC1 Hardware Feature Presence Register 3	0x00000001
0x31042200	EMAC1_MDIO_ADDR	EMAC1 MDIO Address Register	0x00000000

Table A-88: ADSP-SC59x EMAC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31042204	EMAC1_MDIO_DATA	EMAC1 MDIO Data Register	0x00000000
0x31042230	EMAC1_CSR_SW_CTL	EMAC1 CSR Software Control Register	0x00000000
0x31042300	EMAC1_ADDR0_HI	EMAC1 MAC Address 0 High Register	0x8000FFFF
0x31042304	EMAC1_ADDR0_LO	EMAC1 MAC Address 0 Low Register	0xFFFFFFFF
0x31042308	EMAC1_ADDR[i]_HI	EMAC1 MAC Address n High Register	0x0000FFFF
0x3104230C	EMAC1_ADDR[i]_LO	EMAC1 MAC Address n Low Register	0xFFFFFFFF
0x31042700	EMAC1_MMC_CTRL	EMAC1 MMC Control Register	0x00000000
0x31042704	EMAC1_MMC_RXINT	EMAC1 MMC Receive Interrupt Status Register	0x00000000
0x31042708	EMAC1_MMC_TXINT	EMAC1 MMC Transmit Interrupt Status Register	0x00000000
0x3104270C	EMAC1_MMC_RXIMSK	EMAC1 MMC Receive Interrupt Mask Register	0x00000000
0x31042710	EMAC1_MMC_TXIMSK	EMAC1 MMC Transmit Interrupt Mask Register	0x00000000
0x31042714	EMAC1_TX_OCNT_GB	EMAC1 Transmit Octet Good Bad Counter Register	0x00000000
0x31042718	EMAC1_TX_PCNT_GB	EMAC1 Transmit Packet Good Bad Counter Register	0x00000000
0x3104271C	EMAC1_TX_BCST_PCNT_G	EMAC1 Transmit Broadcast Packets Good Counter Register	0x00000000
0x31042720	EMAC1_TX_MCST_PCNT_G	EMAC1 Transmit Multicast Packets Good Counter Register	0x00000000
0x31042724	EMAC1_TX64_GB	EMAC1 Transmit 64 Octets Packets Good Bad Counter Register	0x00000000
0x31042728	EMAC1_TX65TO127_GB	EMAC1 Transmit 65-127 Octets Packets Good Bad Counter Register	0x00000000
0x3104272C	EMAC1_TX128TO255_GB	EMAC1 Transmit 128-255 Octets Packets Good Bad Counter Register	0x00000000
0x31042730	EMAC1_TX256TO511_GB	EMAC1 Transmit 256-511 Octets Packets Good Bad Counter Register	0x00000000
0x31042734	EMAC1_TX512TO1023_GB	EMAC1 Transmit 512-1023 Octets Packets Good Bad Counter Register	0x00000000
0x31042738	EMAC1_TX1024TOMAX_GB	EMAC1 Transmit 1024-Max Octets Packets Good Bad Counter Register	0x00000000
0x3104273C	EMAC1_TX_UCST_PCNT_GB	EMAC1 Transmit Unicast Packets Good Bad Counter Register	0x00000000
0x31042740	EMAC1_TX_MCST_PCNT_GB	EMAC1 Transmit Multicast Packets Good Bad Counter Register	0x00000000

Table A-88: ADSP-SC59x EMAC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31042744	EMAC1_TX_BCST_PCNT_GB	EMAC1 Transmit Broadcast Packets Good Bad Counter Register	0x00000000
0x31042748	EMAC1_TX_URFL_PCNT	EMAC1 Transmit Underflow Error Packets Counter Register	0x00000000
0x3104274C	EMAC1_TX_SNGLCOL_PCNT_G	EMAC1 Transmit Single Collision Packets Good Counter Register	0x00000000
0x31042750	EMAC1_TX_MCOL_PCNT_G	EMAC1 Transmit Multiple Collision Packets Good Counter Register	0x00000000
0x31042754	EMAC1_TX_DFRD_PCNT	EMAC1 Transmit Deferred Packets Counter Register	0x00000000
0x31042758	EMAC1_TX_LTCOL_PCNT	EMAC1 Transmit Late Collision Packets Counter Register	0x00000000
0x3104275C	EMAC1_TX_EXCOL_PCNT	EMAC1 Transmit Excessive Collision Packets Counter Register	0x00000000
0x31042760	EMAC1_TX_CRERR_PCNT	EMAC1 Transmit Carrier Error Packets Counter Register	0x00000000
0x31042764	EMAC1_TX_OCNT_G	EMAC1 Transmit Octet Good Counter Register	0x00000000
0x31042768	EMAC1_TX_PCNT_G	EMAC1 Transmit Packet Good Counter Register	0x00000000
0x3104276C	EMAC1_TX_EXDFRL_PCNT	EMAC1 Transmit Excessive Deferral Error Counter Register	0x00000000
0x31042770	EMAC1_TX_PAUSE_PCNT	EMAC1 Transmit Pause Packets Counter Register	0x00000000
0x31042774	EMAC1_TX_VLAN_PCNT_G	EMAC1 Transmit VLAN Packets Good Counter Register	0x00000000
0x31042778	EMAC1_TX_OSIZE_PCNT_G	EMAC1 Transmit Osize Packets Good Counter Register	0x00000000
0x31042780	EMAC1_RX_PCNT_GB	EMAC1 Receive Packets Good Bad Counter Register	0x00000000
0x31042784	EMAC1_RX_OCNT_GB	EMAC1 Receive Octet Good Bad Counter Register	0x00000000
0x31042788	EMAC1_RX_OCNT_G	EMAC1 Receive Octet Good Counter Register	0x00000000
0x3104278C	EMAC1_RX_BCERR_PCNT_G	EMAC1 Receive Broadcast Packets Good Counter Register	0x00000000
0x31042790	EMAC1_RX_MCST_PCNT_G	EMAC1 Receive Multicast Packets Good Counter Register	0x00000000
0x31042794	EMAC1_RX_CRCERR_PCNT	EMAC1 Receive CRC Error Packets Counter Register	0x00000000
0x31042798	EMAC1_RX_ALGNERR_PCNT	EMAC1 Receive Alignment Error Packets Counter Register	0x00000000

Table A-88: ADSP-SC59x EMAC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3104279C	EMAC1_RX_RNTERR_PCNT	EMAC1 Receive Runt Error Packets Counter Register	0x00000000
0x310427A0	EMAC1_RX_JBERR_PCNT	EMAC1 Receive Jabber Error Packets Counter Register	0x00000000
0x310427A4	EMAC1_RX_UNDRSZ_PCNT_G	EMAC1 Receive Undersize Packets Good Counter Register	0x00000000
0x310427A8	EMAC1_RX_OVRSZ_PCNT	EMAC1 Receive Oversize Packets Good Counter Register	0x00000000
0x310427AC	EMAC1_RX64_GB	EMAC1 Receive 64 Octets Packets Good Bad Counter Register	0x00000000
0x310427B0	EMAC1_RX65TO127_GB	EMAC1 Receive 65 - 127 Octets Packets Good Bad Counter Register	0x00000000
0x310427B4	EMAC1_RX128TO255_GB	EMAC1 Receive 128 - 255 Octets Packets Good Bad Counter Register	0x00000000
0x310427B8	EMAC1_RX256TO511_GB	EMAC1 Receive 256 - 511 Octets Packets Good Bad Counter Register	0x00000000
0x310427BC	EMAC1_RX512TO1023_GB	EMAC1 Receive 512 - 1023 Octets Packets Good Bad Counter Register	0x00000000
0x310427C0	EMAC1_RX1024TOMAX_GB	EMAC1 Receive 1024 - Maximum Octets Packets Good Bad Counter Register	0x00000000
0x310427C4	EMAC1_RX_UCST_PCNT_G	EMAC1 Receive Unicast Packets Good Counter Register	0x00000000
0x310427C8	EMAC1_RX_LENERR_PCNT	EMAC1 Receive Length Error Packets Counter Register	0x00000000
0x310427CC	EMAC1_RX_ORTYPE_PCNT	EMAC1 Receive Out of Range Type Packets Counter Register	0x00000000
0x310427D0	EMAC1_RX_PAUSE_PCNT	EMAC1 Receive Pause Packets Counter Register	0x00000000
0x310427D4	EMAC1_RX_FIFOOVFL_PCNT	EMAC1 Receive FIFO Overflow Packets Counter Register	0x00000000
0x310427D8	EMAC1_RX_VLAN_PCNT_GB	EMAC1 Receive VLAN Packets Good Bad Counter Register	0x00000000
0x310427DC	EMAC1_RX_WDTERR_PCNT	EMAC1 Receive Watchdog Error Packets Counter Register	0x00000000
0x310427E0	EMAC1_RX_ERR_PCNT	EMAC1 Receive Error Packets Counter Register	0x00000000
0x310427E4	EMAC1_RX_CTL_PCNT_G	EMAC1 Receive Control Packets Good Counter Register	0x00000000
0x31042800	EMAC1_MMC_IPC_RXIMSK	EMAC1 MMC Receive IPC Interrupt Mask Register	0x00000000

Table A-88: ADSP-SC59x EMAC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31042808	EMAC1_MMC_IPC_RXINT	EMAC1 MMC Receive IPC Interrupt Status Register	0x00000000
0x31042810	EMAC1_RXIPV4_PCNT_G	EMAC1 Receive IPv4 Good Packets Counter Register	0x00000000
0x31042814	EMAC1_RXIPV4_HDRERR_PCNT	EMAC1 Receive IPv4 Header Error Packets Counter Register	0x00000000
0x31042818	EMAC1_RXIPV4_PYLD_PCNT	EMAC1 Receive IPv4 Payload Packets Counter Register	0x00000000
0x3104281C	EMAC1_RXIPV4_FRGMT_PCNT	EMAC1 Receive IPv4 Fragmented Packets Counter Register	0x00000000
0x31042820	EMAC1_RXIPV4_UDPCSD_PCNT	EMAC1 Receive IPv4 UDP Checksum Disable Packets Counter Register	0x00000000
0x31042824	EMAC1_RXIPV6_PCNT_G	EMAC1 Receive IPv6 Good Packets Counter Register	0x00000000
0x31042828	EMAC1_RXIPV6_HDRERR_PCNT	EMAC1 Receive IPv6 Header Error Packets Counter Register	0x00000000
0x3104282C	EMAC1_RXIPV6_PYLD_PCNT	EMAC1 Receive IPv6 Payload Packets Counter Register	0x00000000
0x31042830	EMAC1_RXUDP_PCNT_G	EMAC1 Receive UDP Good Packets Counter Register	0x00000000
0x31042834	EMAC1_RXUDP_ERR_PCNT	EMAC1 Receive UDP Error Packets Counter Register	0x00000000
0x31042838	EMAC1_RXTCP_PCNT_G	EMAC1 Receive TCP Good Packets Counter Register	0x00000000
0x3104283C	EMAC1_RXTCP_ERR_PCNT	EMAC1 Receive TCP Error Packets Counter Register	0x00000000
0x31042840	EMAC1_RXICMP_PCNT_G	EMAC1 Receive ICMP Good Packets Counter Register	0x00000000
0x31042844	EMAC1_RXICMP_ERR_PCNT	EMAC1 Receive ICMP Error Packets Counter Register	0x00000000
0x31042850	EMAC1_RXIPV4_OCNT_G	EMAC1 Receive IPv4 Good Octets Counter Register	0x00000000
0x31042854	EMAC1_RXIPV4_HDRERR_OCNT	EMAC1 Receive IPv4 Header Error Octets Counter Register	0x00000000
0x31042858	EMAC1_RXIPV4_PYLD_OCNT	EMAC1 Receive IPv4 Payload Octets Counter Register	0x00000000
0x3104285C	EMAC1_RXIPV4_FRGMT_OCNT	EMAC1 Receive IPv4 Fragmented Octets Counter Register	0x00000000
0x31042860	EMAC1_RXIPV4_UDPCSD_OCNT	EMAC1 Receive IPv4 UDP Checksum Disable Octets Counter Register	0x00000000
0x31042864	EMAC1_RXIPV6_OCNT_G	EMAC1 Receive IPv6 Good Octets Counter Register	0x00000000

Table A-88: ADSP-SC59x EMAC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31042868	EMAC1_RXIPV6_HDRERR_OCNT	EMAC1 Receive IPv6 Header Error Octets Counter Register	0x00000000
0x3104286C	EMAC1_RXIPV6_PYLD_OCNT	EMAC1 Receive IPv6 Payload Octets Counter Register	0x00000000
0x31042870	EMAC1_RXUDP_OCNT_G	EMAC1 Receive UDP Good Octets Counter Register	0x00000000
0x31042874	EMAC1_RXUDP_ERR_OCNT	EMAC1 Receive UDP Error Octets Counter Register	0x00000000
0x31042878	EMAC1_RXTCP_OCNT_G	EMAC1 Receive TCP Good Octets Counter Register	0x00000000
0x3104287C	EMAC1_RXTCP_ERR_OCNT	EMAC1 Receive TCP Error Octets Counter Register	0x00000000
0x31042880	EMAC1_RXICMP_OCNT_G	EMAC1 Receive ICMP Good Octets Counter Register	0x00000000
0x31042884	EMAC1_RXICMP_ERR_OCNT	EMAC1 Receive ICMP Error Octets Counter Register	0x00000000
0x31042900	EMAC1_L3L4_CTL[i]	EMAC1 Layer 3 and Layer 4 Control Register	0x00000000
0x31042904	EMAC1_L4_ADDR[i]	EMAC1 Layer 4 Address Register [i]	0x00000000
0x31042910	EMAC1_L3_ADDR0_REG[i]	EMAC1 Layer 3 Address 0 Register [i]	0x00000000
0x31042914	EMAC1_L3_ADDR1_REG[i]	EMAC1 Layer 3 Address 1 Register [i]	0x00000000
0x31042918	EMAC1_L3_ADDR2_REG[i]	EMAC1 Layer 3 Address 2 Register [i]	0x00000000
0x3104291C	EMAC1_L3_ADDR3_REG[i]	EMAC1 Layer 3 Address 3 Register [i]	0x00000000
0x31042C00	EMAC1_MTL_OPMODE	EMAC1 MTL Operation Mode Register	0x00000000
0x31042C20	EMAC1_MTL_ISTAT	EMAC1 MTL Interrupt Status Register	0x00000000
0x31042D00	EMAC1_TQ0_OPMODE	EMAC1 Transmit Queue 0 Operation Mode Register	0x00030000
0x31042D04	EMAC1_TQ0_URFL	EMAC1 Transmit Queue 0 Underflow Packet Counter Register	0x00000000
0x31042D08	EMAC1_TQ0_DBG	EMAC1 Transmit Queue 0 Debug Register	0x00000000
0x31042D2C	EMAC1_Q0_INT_CTLSTAT	EMAC1 Queue 0 Interrupt Control and Status Register	0x00000000
0x31042D30	EMAC1_RQ0_OPMODE	EMAC1 Receive Queue 0 Operation Mode Register	0x00000000
0x31042D34	EMAC1_RQ0_MSPKTOF_CNT	EMAC1 Receive Queue 0 Missed Packet and Overflow Counter Register	0x00000000

Table A-88: ADSP-SC59x EMAC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31042D38	EMAC1_RQ0_DBG	EMAC1 Receive Queue 0 Debug Register	0x00000000
0x31043000	EMAC1_DMA_MODE	EMAC1 DMA Mode Register	0x00000000
0x31043004	EMAC1_DMA_SYSBMODE	EMAC1 DMA System Bus Mode Register	0x01010000
0x31043008	EMAC1_DMA_ISTAT	EMAC1 DMA Interrupt Status Register	0x00000000
0x3104300C	EMAC1_DMA_DBG_STAT 0	EMAC1 DMA Debug Status 0 Register	0x00000000
0x31043100	EMAC1_DMA[n]_CTL	EMAC1 DMA Channel n Control Register	0x00000000
0x31043104	EMAC1_DMA[n]_TXCTL	EMAC1 DMA Channel n Transmit Control Register	0x00000000
0x31043108	EMAC1_DMA[n]_RXCTL	EMAC1 DMA Channel n Receive Control Register	0x00000000
0x31043114	EMAC1_DMA[n]_TXDSC_ ADDR	EMAC1 DMA Channel n Transmit Descriptor List Address Register	0x00000000
0x3104311C	EMAC1_DMA[n]_RXDSC_ ADDR	EMAC1 DMA Channel n Receive Descriptor List Address Register	0x00000000
0x31043120	EMAC1_DMA[n]_TXDSC_ TLPTR	EMAC1 DMA Channel n Transmit Descriptor Tail Pointer Register	0x00000000
0x31043128	EMAC1_DMA[n]_RXDSC_ TLPTR	EMAC1 DMA Channel n Receive Descriptor Tail Pointer Register	0x00000000
0x3104312C	EMAC1_DMA[n]_TXDSC_ RLEN	EMAC1 DMA Channel n Transmit Descriptor Ring Length Register	0x00000000
0x31043130	EMAC1_DMA[n]_RXCTL2	EMAC1 DMA Channel n Extended Receive Control Register	0x00000000
0x31043134	EMAC1_DMA[n]_IEN	EMAC1 DMA Channel n Interrupt Enable Register	0x00000000
0x31043138	EMAC1_DMA[n]_RXINTW DTMR	EMAC1 DMA Channel n Receive Interrupt Watchdog Timer Register	0x00000000
0x31043144	EMAC1_DMA[n]_TXDSC_ CUR	EMAC1 DMA Channel n Current Application Transmit Descriptor Register	0x00000000
0x3104314C	EMAC1_DMA[n]_RXDSC_ CUR	EMAC1 DMA Channel n Current Application Receive Descriptor Register	0x00000000
0x31043154	EMAC1_DMA[n]_TXBUF_ CUR	EMAC1 DMA Channel n Current Application Transmit Buffer Address Register	0x00000000
0x3104315C	EMAC1_DMA[n]_RXBUF_ CUR	EMAC1 DMA Channel n Current Application Receive Buffer Address Register	0x00000000
0x31043160	EMAC1_DMA[n]_STAT	EMAC1 DMA Channel n Status Register	0x00000000
0x31043164	EMAC1_DMA[n]_MISSFR M_CNT	EMAC1 DMA Channel n Missed Frame Counter Register	0x00000000

Table A-89: ADSP-SC59x FIR MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310BD000	FIR_CTL1	FIR FIR Global Control Register	0x00000000
0x310BD004	FIR_DMASTAT	FIR FIR DMA Status Register	0x00000000
0x310BD008	FIR_MACSTAT	FIR FIR MAC Status Register	0x00000000
0x310BD010	FIR_DBG_CTL	FIR FIR Debug Control Register	0x00000000
0x310BD014	FIR_DBG_ADDR	FIR Debug Address Register	0x00000000
0x310BD018	FIR_DBG_WRDAT	FIR FIR Debug Data Write Register	0x00000000
0x310BD01C	FIR_DBG_RDDAT	FIR FIR Debug Data Read Register	0x00000000
0x310BD040	FIR_CTL2	FIR FIR Channel Control Register	0x00000000
0x310BD044	FIR_INIDX	FIR FIR Input Data Index Register	0x00000000
0x310BD048	FIR_INMOD	FIR FIR Input Data Modifier Register	0x00000000
0x310BD04C	FIR_INCNT	FIR FIR Input Data Count Register	0x00000000
0x310BD050	FIR_INBASE	FIR FIR Input Data Base Register	0x00000000
0x310BD054	FIR_OUTIDX	FIR FIR Output Data Index Register	0x00000000
0x310BD058	FIR_OUTMOD	FIR FIR Output Data Modifier Register	0x00000000
0x310BD05C	FIR_OUTCNT	FIR FIR Output Data Count Register	0x00000000
0x310BD060	FIR_OUTBASE	FIR FIR Output Data Base Register	0x00000000
0x310BD064	FIR_COEFIDX	FIR FIR Coefficient Index Register	0x00000000
0x310BD068	FIR_COEFMOD	FIR FIR Coefficient Modifier Register	0x00000000
0x310BD06C	FIR_COEFCNT	FIR FIR Coefficient Count Register	0x00000000
0x310BD070	FIR_CHNPTR	FIR FIR Chain Pointer Register	0x00000000
0x310BD074	FIR_SCTL1	FIR Software Control Register 1	0x00000000
0x310BD078	FIR_SCTL2	FIR Software Control Register 2	0x00000000
0x310BD07C	FIR_SGCTL	FIR Secondary Global Control Register	0x00000000
0x310C3000	FIR0_CTL1	FIR0 FIR Global Control Register	0x00000000
0x310C3004	FIR0_DMASTAT	FIR0 FIR DMA Status Register	0x00000000
0x310C3008	FIR0_MACSTAT	FIR0 FIR MAC Status Register	0x00000000
0x310C3010	FIR0_DBG_CTL	FIR0 FIR Debug Control Register	0x00000000
0x310C3014	FIR0_DBG_ADDR	FIR0 Debug Address Register	0x00000000
0x310C3018	FIR0_DBG_WRDAT	FIR0 FIR Debug Data Write Register	0x00000000
0x310C301C	FIR0_DBG_RDDAT	FIR0 FIR Debug Data Read Register	0x00000000
0x310C3040	FIR0_CTL2	FIR0 FIR Channel Control Register	0x00000000

Table A-89: ADSP-SC59x FIR MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C3044	FIR0_INIDX	FIR0 FIR Input Data Index Register	0x00000000
0x310C3048	FIR0_INMOD	FIR0 FIR Input Data Modifier Register	0x00000000
0x310C304C	FIR0_INCNT	FIR0 FIR Input Data Count Register	0x00000000
0x310C3050	FIR0_INBASE	FIR0 FIR Input Data Base Register	0x00000000
0x310C3054	FIR0_OUTIDX	FIR0 FIR Output Data Index Register	0x00000000
0x310C3058	FIR0_OUTMOD	FIR0 FIR Output Data Modifier Register	0x00000000
0x310C305C	FIR0_OUTCNT	FIR0 FIR Output Data Count Register	0x00000000
0x310C3060	FIR0_OUTBASE	FIR0 FIR Output Data Base Register	0x00000000
0x310C3064	FIR0_COEFIDX	FIR0 FIR Coefficient Index Register	0x00000000
0x310C3068	FIR0_COEFMOD	FIR0 FIR Coefficient Modifier Register	0x00000000
0x310C306C	FIR0_COEFCNT	FIR0 FIR Coefficient Count Register	0x00000000
0x310C3070	FIR0_CHNPTR	FIR0 FIR Chain Pointer Register	0x00000000
0x310C3074	FIR0_SCTL1	FIR0 Software Control Register 1	0x00000000
0x310C3078	FIR0_SCTL2	FIR0 Software Control Register 2	0x00000000
0x310C307C	FIR0_SGCTL	FIR0 Secondary Global Control Register	0x00000000

Table A-90: ADSP-SC59x FIR1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310BD000	FIR1_CTL1	FIR1 FIR Global Control Register	0x00000000
0x310BD004	FIR1_DMASTAT	FIR1 FIR DMA Status Register	0x00000000
0x310BD008	FIR1_MACSTAT	FIR1 FIR MAC Status Register	0x00000000
0x310BD010	FIR1_DBG_CTL	FIR1 FIR Debug Control Register	0x00000000
0x310BD014	FIR1_DBG_ADDR	FIR1 Debug Address Register	0x00000000
0x310BD018	FIR1_DBG_WRDAT	FIR1 FIR Debug Data Write Register	0x00000000
0x310BD01C	FIR1_DBG_RDDAT	FIR1 FIR Debug Data Read Register	0x00000000
0x310BD040	FIR1_CTL2	FIR1 FIR Channel Control Register	0x00000000
0x310BD044	FIR1_INIDX	FIR1 FIR Input Data Index Register	0x00000000
0x310BD048	FIR1_INMOD	FIR1 FIR Input Data Modifier Register	0x00000000
0x310BD04C	FIR1_INCNT	FIR1 FIR Input Data Count Register	0x00000000
0x310BD050	FIR1_INBASE	FIR1 FIR Input Data Base Register	0x00000000

Table A-90: ADSP-SC59x FIR1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310BD054	FIR1_OUTIDX	FIR1 FIR Output Data Index Register	0x00000000
0x310BD058	FIR1_OUTMOD	FIR1 FIR Output Data Modifier Register	0x00000000
0x310BD05C	FIR1_OUTCNT	FIR1 FIR Output Data Count Register	0x00000000
0x310BD060	FIR1_OUTBASE	FIR1 FIR Output Data Base Register	0x00000000
0x310BD064	FIR1_COEFIDX	FIR1 FIR Coefficient Index Register	0x00000000
0x310BD068	FIR1_COEFMOD	FIR1 FIR Coefficient Modifier Register	0x00000000
0x310BD06C	FIR1_COEFCNT	FIR1 FIR Coefficient Count Register	0x00000000
0x310BD070	FIR1_CHNPTR	FIR1 FIR Chain Pointer Register	0x00000000
0x310BD074	FIR1_SCTL1	FIR1 Software Control Register 1	0x00000000
0x310BD078	FIR1_SCTL2	FIR1 Software Control Register 2	0x00000000
0x310BD07C	FIR1_SGCTL	FIR1 Secondary Global Control Register	0x00000000

Table A-91: ADSP-SC59x GICA0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31210040	GICA0_GICA_SETSPI_NSR	GICA0 GICA_SETSPI_NSR	0x00000000
0x31210048	GICA0_GICA_CLRSPI_NSR	GICA0 GICA_CLRSPI_NSR	0x00000000
0x31210050	GICA0_GICA_SETSPI_SR	GICA0 GICA_SETSPI_SR	0x00000000
0x31210058	GICA0_GICA_CLRSPI_SR	GICA0 GICA_CLRSPI_SR	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31200000	GICD0_GICD_CTLR	GICD0 GICD_CTLR	0x00000030
0x31200004	GICD0_GICD_TYPER	GICD0 GICD_TYPER	0x0179040C
0x31200008	GICD0_GICD_IIDR	GICD0 GICD_IIDR	0x0201743B
0x31200020	GICD0_GICD_FCTLR	GICD0 GICD_FCTLR	0x00000000
0x31200024	GICD0_GICD_SAC	GICD0 GICD_SAC	0x00000000
0x31200040	GICD0_GICD_SETSPI_NSR	GICD0 GICD_SETSPI_NSR	0x00000000
0x31200048	GICD0_GICD_CLRSPI_NSR	GICD0 GICD_CLRSPI_NSR	0x00000000
0x31200050	GICD0_GICD_SETSPI_SR	GICD0 GICD_SETSPI_SR	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31200058	GICD0_GICD_CLRSPI_SR	GICD0 GICD_CLRSPI_SR	0x00000000
0x31200084	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x31200088	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x3120008C	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x31200090	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x31200094	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x31200098	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x3120009C	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x312000A0	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x312000A4	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x312000A8	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x312000AC	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x312000B0	GICD0_GICD_IGROUPR[n]	GICD0 GICD_IGROUPR[n]	0x00000000
0x31200104	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x31200108	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x3120010C	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x31200110	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x31200114	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x31200118	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3120011C	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x31200120	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x31200124	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x31200128	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x3120012C	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x31200130	GICD0_GICD_ISENABLER[n]	GICD0 GICD_ISENABLER[n]	0x00000000
0x31200184	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x31200188	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x3120018C	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x31200190	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x31200194	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x31200198	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x3120019C	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x312001A0	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x312001A4	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x312001A8	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x312001AC	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x312001B0	GICD0_GICD_ICENABLER[n]	GICD0 GICD_ICENABLER[n]	0x00000000
0x31200204	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31200208	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x3120020C	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x31200210	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x31200214	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x31200218	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x3120021C	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x31200220	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x31200224	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x31200228	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x3120022C	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x31200230	GICD0_GICD_ISPENDR[n]	GICD0 GICD_ISPENDR[n]	0x00000000
0x31200284	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x31200288	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x3120028C	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x31200290	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x31200294	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x31200298	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x3120029C	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x312002A0	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x312002A4	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x312002A8	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x312002AC	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000
0x312002B0	GICD0_GICD_ICPENDR[n]	GICD0 GICD_ICPENDR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31200304	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x31200308	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x3120030C	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x31200310	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x31200314	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x31200318	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x3120031C	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x31200320	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x31200324	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x31200328	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x3120032C	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x31200330	GICD0_GICD_ISACTIVER[n]	GICD0 GICD_ISACTIVER[n]	0x00000000
0x31200384	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x31200388	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x3120038C	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x31200390	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x31200394	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x31200398	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3120039C	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x312003A0	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x312003A4	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x312003A8	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x312003AC	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x312003B0	GICD0_GICD_ICACTIVER[n]	GICD0 GICD_ICACTIVER[n]	0x00000000
0x31200420	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200424	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200428	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120042C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200430	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200434	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200438	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120043C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200440	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200444	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200448	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120044C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31200450	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200454	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200458	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120045C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200460	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200464	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200468	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120046C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200470	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200474	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200478	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120047C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200480	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200484	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200488	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120048C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200490	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200494	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31200498	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120049C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004A0	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004A4	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004A8	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004AC	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004B0	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004B4	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004B8	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004BC	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004C0	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004C4	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004C8	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004CC	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004D0	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004D4	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004D8	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004DC	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312004E0	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004E4	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004E8	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004EC	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004F0	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004F4	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004F8	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x312004FC	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200500	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200504	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200508	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120050C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200510	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200514	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200518	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120051C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200520	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200524	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31200528	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120052C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200530	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200534	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200538	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120053C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200540	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200544	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200548	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120054C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200550	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200554	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200558	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120055C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200560	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200564	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200568	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120056C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31200570	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200574	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200578	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120057C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200580	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200584	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200588	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120058C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200590	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200594	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200598	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x3120059C	GICD0_GICD_IPRIORITYR[n]	GICD0 GICD_IPRIORITYR[n]	0x00000000
0x31200C08	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C0C	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C10	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C14	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C18	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C1C	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C20	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C24	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C28	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C2C	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C30	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31200C34	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C38	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C3C	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C40	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C44	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C48	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C4C	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C50	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C54	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C58	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C5C	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C60	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200C64	GICD0_GICD_ICFGR[n]	GICD0 GICD_ICFGR[n]	0x00000000
0x31200D04	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D08	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D0C	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D10	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D14	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D18	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D1C	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D20	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D24	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D28	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31200D2C	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200D30	GICD0_GICD_IGRPMODR[n]	GICD0 GICD_IGRPMODR[n]	0x00000000
0x31200E08	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E0C	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E10	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E14	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E18	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E1C	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E20	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E24	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E28	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E2C	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E30	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E34	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E38	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E3C	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E40	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E44	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E48	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E4C	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E50	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E54	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E58	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E5C	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E60	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31200E64	GICD0_GICD_NSACR[n]	GICD0 GICD_NSACR[n]	0x00000000
0x31206100	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206108	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206110	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206118	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206120	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206128	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206130	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206138	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206140	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206148	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206150	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206158	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206160	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206168	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206170	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206178	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206180	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206188	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206190	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206198	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312061A0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061A8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061B0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061B8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061C0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061C8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061D0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061D8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061E0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061E8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061F0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312061F8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206200	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206208	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206210	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206218	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206220	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206228	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206230	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206238	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206240	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206248	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206250	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206258	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206260	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206268	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206270	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206278	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206280	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206288	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206290	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206298	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062A0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062A8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062B0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062B8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312062C0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062C8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062D0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062D8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062E0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062E8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062F0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312062F8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206300	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206308	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206310	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206318	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206320	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206328	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206330	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206338	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206340	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206348	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206350	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206358	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206360	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206368	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206370	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206378	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206380	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206388	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206390	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206398	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063A0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063A8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063B0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063B8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063C0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063C8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063D0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063D8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312063E0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063E8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063F0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312063F8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206400	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206408	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206410	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206418	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206420	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206428	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206430	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206438	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206440	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206448	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206450	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206458	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206460	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206468	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206470	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206478	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206480	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206488	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206490	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206498	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064A0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064A8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064B0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064B8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064C0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064C8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064D0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064D8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064E0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064E8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064F0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312064F8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206500	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206508	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206510	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206518	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206520	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206528	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206530	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206538	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206540	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206548	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206550	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206558	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206560	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206568	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206570	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206578	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206580	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206588	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206590	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206598	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065A0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065A8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065B0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065B8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065C0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065C8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065D0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065D8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065E0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065E8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065F0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312065F8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206600	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206608	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206610	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206618	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206620	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206628	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206630	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206638	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206640	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206648	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206650	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206658	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206660	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206668	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206670	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206678	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206680	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206688	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206690	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206698	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066A0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066A8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312066B0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066B8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066C0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066C8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066D0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066D8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066E0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066E8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066F0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312066F8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206700	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206708	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206710	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206718	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206720	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206728	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206730	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206738	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206740	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206748	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206750	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206758	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206760	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206768	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206770	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206778	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206780	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206788	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206790	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206798	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067A0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067A8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067B0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067B8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067C0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067C8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312067D0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067D8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067E0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067E8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067F0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312067F8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206800	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206808	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206810	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206818	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206820	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206828	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206830	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206838	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206840	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206848	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206850	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206858	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206860	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206868	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206870	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206878	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206880	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206888	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206890	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206898	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068A0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068A8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068B0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068B8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068C0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068C8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068D0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068D8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068E0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068E8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312068F0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312068F8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206900	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206908	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206910	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206918	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206920	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206928	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206930	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206938	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206940	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206948	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206950	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206958	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206960	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206968	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206970	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206978	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206980	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206988	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206990	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206998	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069A0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069A8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069B0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069B8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069C0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069C8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069D0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069D8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069E0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069E8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069F0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x312069F8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A00	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A08	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206A10	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A18	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A20	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A28	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A30	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A38	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A40	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A48	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A50	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A58	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A60	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A68	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A70	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A78	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A80	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A88	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A90	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206A98	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206AA0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AA8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AB0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AB8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AC0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AC8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AD0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AD8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AE0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AE8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AF0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206AF8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B00	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B08	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B10	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B18	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B20	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B28	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206B30	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B38	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B40	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B48	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B50	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B58	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B60	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B68	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B70	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B78	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B80	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B88	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B90	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206B98	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BA0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BA8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BB0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BB8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206BC0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BC8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BD0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BD8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BE0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BE8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BF0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206BF8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C00	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C08	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C10	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C18	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C20	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C28	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C30	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C38	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C40	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C48	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206C50	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C58	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C60	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C68	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C70	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C78	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C80	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C88	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C90	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206C98	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CA0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CA8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CB0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CB8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CC0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CC8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CD0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CD8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31206CE0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CE8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CF0	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x31206CF8	GICD0_GICD_IROUTER[n]	GICD0 GICD_IROUTER[n]	0x80000000
0x3120E008	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E00C	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E010	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E014	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E018	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E01C	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E020	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E024	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E028	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E02C	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E030	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E034	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E038	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E03C	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E040	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E044	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E048	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E04C	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E050	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E054	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E058	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E05C	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E060	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000
0x3120E064	GICD0_GICD_ICLAR[n]	GICD0 GICD_ICLAR[n]	0x00000000

Table A-92: ADSP-SC59x GICD0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3120E104	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E108	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E10C	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E110	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E114	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E118	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E11C	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E120	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E124	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E128	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E12C	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120E130	GICD0_GICD_IERRR[n]	GICD0 GICD_IERRR[n]	0x00000000
0x3120F000	GICD0_GICD_CFGID	GICD0 GICD_CFGID	0x1888400064000
0x3120FFD0	GICD0_GICD_PIDR4	GICD0 GICD_PIDR4	0x00000044
0x3120FFE0	GICD0_GICD_PIDR0	GICD0 GICD_PIDR0	0x00000092
0x3120FFE4	GICD0_GICD_PIDR1	GICD0 GICD_PIDR1	0x000000B4
0x3120FFE8	GICD0_GICD_PIDR2	GICD0 GICD_PIDR2	0x0000003B
0x3120FFEC	GICD0_GICD_PIDR3	GICD0 GICD_PIDR3	0x00000000
0x3120FFF0	GICD0_GICD_CIDR0	GICD0 GICD_CIDR0	0x0000000D
0x3120FFF4	GICD0_GICD_CIDR1	GICD0 GICD_CIDR1	0x000000F0
0x3120FFF8	GICD0_GICD_CIDR2	GICD0 GICD_CIDR2	0x00000005
0x3120FFFC	GICD0_GICD_CIDR3	GICD0 GICD_CIDR3	0x000000B1

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31260000	GICDA0_GICDA_CTLR	GICDA0 GICDA_CTLR	0x00000030
0x31260004	GICDA0_GICDA_TYPER	GICDA0 GICDA_TYPER	0x0179040C
0x31260008	GICDA0_GICDA_IIDR	GICDA0 GICDA_IIDR	0x0201743B
0x31260020	GICDA0_GICDA_FCTLR	GICDA0 GICDA_FCTLR	0x00000000
0x31260024	GICDA0_GICDA_SAC	GICDA0 GICDA_SAC	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260040	GICDA0_GICDA_SETSPI_NSR	GICDA0 GICDA_SETSPI_NSR	0x00000000
0x31260048	GICDA0_GICDA_CLRSPI_NSR	GICDA0 GICDA_CLRSPI_NSR	0x00000000
0x31260050	GICDA0_GICDA_SETSPI_SR	GICDA0 GICDA_SETSPI_SR	0x00000000
0x31260058	GICDA0_GICDA_CLRSPI_SR	GICDA0 GICDA_CLRSPI_SR	0x00000000
0x31260084	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x31260088	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x3126008C	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x31260090	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x31260094	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x31260098	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x3126009C	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x312600A0	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x312600A4	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x312600A8	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x312600AC	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x312600B0	GICDA0_GICDA_IGROUPR[n]	GICDA0 GICDA_IGROUPR[n]	0x00000000
0x31260104	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x31260108	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3126010C	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x31260110	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x31260114	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x31260118	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x3126011C	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x31260120	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x31260124	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x31260128	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x3126012C	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x31260130	GICDA0_GICDA_ISENABLER[n]	GICDA0 GICDA_ISENABLER[n]	0x00000000
0x31260184	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x31260188	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x3126018C	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x31260190	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x31260194	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x31260198	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x3126019C	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x312601A0	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312601A4	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x312601A8	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x312601AC	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x312601B0	GICDA0_GICDA_ICENABLER[n]	GICDA0 GICDA_ICENABLER[n]	0x00000000
0x31260204	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x31260208	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x3126020C	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x31260210	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x31260214	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x31260218	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x3126021C	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x31260220	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x31260224	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x31260228	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x3126022C	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x31260230	GICDA0_GICDA_ISPENDR[n]	GICDA0 GICDA_ISPENDR[n]	0x00000000
0x31260284	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x31260288	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3126028C	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x31260290	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x31260294	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x31260298	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x3126029C	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x312602A0	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x312602A4	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x312602A8	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x312602AC	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x312602B0	GICDA0_GICDA_ICPENDR[n]	GICDA0 GICDA_ICPENDR[n]	0x00000000
0x31260304	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x31260308	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x3126030C	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x31260310	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x31260314	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x31260318	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x3126031C	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x31260320	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260324	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x31260328	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x3126032C	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x31260330	GICDA0_GICDA_ISACTIVER[n]	GICDA0 GICDA_ISACTIVER[n]	0x00000000
0x31260384	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x31260388	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x3126038C	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x31260390	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x31260394	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x31260398	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x3126039C	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x312603A0	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x312603A4	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x312603A8	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x312603AC	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x312603B0	GICDA0_GICDA_ICACTIVER[n]	GICDA0 GICDA_ICACTIVER[n]	0x00000000
0x31260420	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260424	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260428	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126042C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260430	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260434	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260438	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126043C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260440	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260444	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260448	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126044C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260450	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260454	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260458	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126045C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260460	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260464	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260468	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126046C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260470	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260474	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260478	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126047C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260480	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260484	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260488	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126048C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260490	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260494	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260498	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126049C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604A0	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604A4	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604A8	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604AC	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604B0	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604B4	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312604B8	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604BC	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604C0	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604C4	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604C8	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604CC	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604D0	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604D4	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604D8	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604DC	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604E0	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604E4	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604E8	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604EC	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604F0	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604F4	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604F8	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x312604FC	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260500	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260504	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260508	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126050C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260510	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260514	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260518	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126051C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260520	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260524	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260528	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126052C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260530	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260534	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260538	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126053C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260540	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260544	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260548	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126054C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260550	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260554	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260558	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126055C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260560	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260564	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260568	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126056C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260570	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260574	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260578	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126057C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260580	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260584	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260588	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126058C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260590	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260594	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260598	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x3126059C	GICDA0_GICDA_IPRIORITYR[n]	GICDA0 GICDA_IPRIORITYR[n]	0x00000000
0x31260C08	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C0C	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C10	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C14	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C18	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C1C	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C20	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C24	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C28	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C2C	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C30	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C34	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C38	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C3C	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260C40	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C44	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C48	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C4C	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C50	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C54	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C58	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C5C	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C60	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260C64	GICDA0_GICDA_ICFGR[n]	GICDA0 GICDA_ICFGR[n]	0x00000000
0x31260D04	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D08	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D0C	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D10	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D14	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D18	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D1C	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D20	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260D24	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D28	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D2C	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260D30	GICDA0_GICDA_IGRPMODR[n]	GICDA0 GICDA_IGRPMODR[n]	0x00000000
0x31260E08	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E0C	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E10	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E14	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E18	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E1C	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E20	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E24	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E28	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E2C	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E30	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E34	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E38	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E3C	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31260E40	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E44	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E48	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E4C	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E50	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E54	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E58	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E5C	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E60	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31260E64	GICDA0_GICDA_NSACR[n]	GICDA0 GICDA_NSACR[n]	0x00000000
0x31266100	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266108	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266110	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266118	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266120	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266128	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266130	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266138	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266140	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266148	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266150	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266158	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266160	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266168	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266170	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266178	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266180	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266188	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266190	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266198	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661A0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661A8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661B0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661B8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661C0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661C8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312661D0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661D8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661E0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661E8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661F0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312661F8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266200	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266208	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266210	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266218	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266220	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266228	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266230	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266238	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266240	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266248	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266250	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266258	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266260	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266268	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266270	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266278	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266280	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266288	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266290	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266298	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662A0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662A8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662B0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662B8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662C0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662C8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662D0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662D8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662E0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662E8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312662F0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312662F8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266300	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266308	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266310	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266318	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266320	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266328	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266330	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266338	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266340	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266348	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266350	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266358	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266360	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266368	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266370	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266378	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266380	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266388	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266390	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266398	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663A0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663A8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663B0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663B8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663C0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663C8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663D0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663D8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663E0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663E8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663F0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312663F8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266400	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266408	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266410	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266418	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266420	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266428	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266430	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266438	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266440	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266448	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266450	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266458	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266460	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266468	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266470	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266478	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266480	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266488	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266490	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266498	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312664A0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664A8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664B0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664B8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664C0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664C8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664D0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664D8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664E0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664E8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664F0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312664F8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266500	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266508	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266510	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266518	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266520	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266528	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266530	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266538	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266540	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266548	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266550	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266558	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266560	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266568	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266570	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266578	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266580	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266588	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266590	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266598	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665A0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665A8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665B0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665B8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312665C0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665C8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665D0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665D8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665E0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665E8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665F0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312665F8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266600	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266608	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266610	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266618	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266620	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266628	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266630	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266638	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266640	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266648	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266650	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266658	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266660	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266668	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266670	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266678	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266680	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266688	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266690	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266698	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666A0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666A8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666B0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666B8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666C0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666C8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666D0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666D8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312666E0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666E8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666F0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312666F8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266700	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266708	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266710	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266718	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266720	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266728	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266730	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266738	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266740	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266748	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266750	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266758	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266760	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266768	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266770	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266778	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266780	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266788	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266790	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266798	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667A0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667A8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667B0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667B8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667C0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667C8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667D0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667D8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667E0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667E8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667F0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312667F8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266800	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266808	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266810	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266818	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266820	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266828	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266830	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266838	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266840	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266848	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266850	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266858	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266860	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266868	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266870	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266878	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266880	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266888	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266890	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266898	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668A0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668A8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668B0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668B8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668C0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668C8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668D0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668D8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668E0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668E8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668F0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312668F8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266900	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266908	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266910	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266918	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266920	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266928	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266930	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266938	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266940	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266948	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266950	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266958	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266960	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266968	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266970	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266978	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266980	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266988	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266990	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266998	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669A0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669A8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x312669B0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669B8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669C0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669C8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669D0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669D8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669E0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669E8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669F0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x312669F8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A00	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A08	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A10	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A18	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A20	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A28	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A30	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A38	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266A40	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A48	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A50	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A58	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A60	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A68	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A70	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A78	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A80	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A88	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A90	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266A98	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AA0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AA8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AB0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AB8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AC0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AC8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266AD0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AD8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AE0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AE8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AF0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266AF8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B00	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B08	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B10	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B18	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B20	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B28	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B30	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B38	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B40	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B48	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B50	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B58	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266B60	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B68	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B70	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B78	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B80	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B88	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B90	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266B98	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BA0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BA8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BB0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BB8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BC0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BC8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BD0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BD8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BE0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BE8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266BF0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266BF8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C00	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C08	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C10	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C18	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C20	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C28	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C30	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C38	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C40	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C48	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C50	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C58	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C60	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C68	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C70	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C78	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31266C80	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C88	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C90	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266C98	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CA0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CA8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CB0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CB8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CC0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CC8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CD0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CD8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CE0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CE8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CF0	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x31266CF8	GICDA0_GICDA_IROUTER[n]	GICDA0 GICDA_IROUTER[n]	0x80000000
0x3126E008	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E00C	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E010	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E014	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3126E018	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E01C	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E020	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E024	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E028	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E02C	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E030	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E034	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E038	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E03C	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E040	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E044	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E048	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E04C	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E050	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E054	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E058	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E05C	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E060	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E064	GICDA0_GICDA_ICLAR[n]	GICDA0 GICDA_ICLAR[n]	0x00000000
0x3126E104	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E108	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E10C	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E110	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E114	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E118	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E11C	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E120	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E124	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E128	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126E12C	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000

Table A-93: ADSP-SC59x GICDA0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3126E130	GICDA0_GICDA_IERRR[n]	GICDA0 GICDA_IERRR[n]	0x00000000
0x3126F000	GICDA0_GICDA_CFGID	GICDA0 GICDA_CFGID	0x1888400064000
0x3126FFD0	GICDA0_GICDA_PIDR4	GICDA0 GICDA_PIDR4	0x00000044
0x3126FFE0	GICDA0_GICDA_PIDR0	GICDA0 GICDA_PIDR0	0x00000092
0x3126FFE4	GICDA0_GICDA_PIDR1	GICDA0 GICDA_PIDR1	0x000000B4
0x3126FFE8	GICDA0_GICDA_PIDR2	GICDA0 GICDA_PIDR2	0x0000003B
0x3126FFEC	GICDA0_GICDA_PIDR3	GICDA0 GICDA_PIDR3	0x00000000
0x3126FFF0	GICDA0_GICDA_CIDR0	GICDA0 GICDA_CIDR0	0x0000000D
0x3126FFF4	GICDA0_GICDA_CIDR1	GICDA0 GICDA_CIDR1	0x000000F0
0x3126FFF8	GICDA0_GICDA_CIDR2	GICDA0 GICDA_CIDR2	0x00000005
0x3126FFFC	GICDA0_GICDA_CIDR3	GICDA0 GICDA_CIDR3	0x000000B1

Table A-94: ADSP-SC59x GICP0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31230000	GICP0_GICP_EVCNTR[n]	GICP0 GICP_EVCNTR[n]	0x00000000
0x31230004	GICP0_GICP_EVCNTR[n]	GICP0 GICP_EVCNTR[n]	0x00000000
0x31230008	GICP0_GICP_EVCNTR[n]	GICP0 GICP_EVCNTR[n]	0x00000000
0x3123000C	GICP0_GICP_EVCNTR[n]	GICP0 GICP_EVCNTR[n]	0x00000000
0x31230010	GICP0_GICP_EVCNTR[n]	GICP0 GICP_EVCNTR[n]	0x00000000
0x31230400	GICP0_GICP_EVTYPER[n]	GICP0 GICP_EVTYPER[n]	0x00000000
0x31230404	GICP0_GICP_EVTYPER[n]	GICP0 GICP_EVTYPER[n]	0x00000000
0x31230408	GICP0_GICP_EVTYPER[n]	GICP0 GICP_EVTYPER[n]	0x00000000
0x3123040C	GICP0_GICP_EVTYPER[n]	GICP0 GICP_EVTYPER[n]	0x00000000
0x31230410	GICP0_GICP_EVTYPER[n]	GICP0 GICP_EVTYPER[n]	0x00000000
0x31230600	GICP0_GICP_SVR[n]	GICP0 GICP_SVR[n]	0x00000000
0x31230604	GICP0_GICP_SVR[n]	GICP0 GICP_SVR[n]	0x00000000
0x31230608	GICP0_GICP_SVR[n]	GICP0 GICP_SVR[n]	0x00000000
0x3123060C	GICP0_GICP_SVR[n]	GICP0 GICP_SVR[n]	0x00000000
0x31230610	GICP0_GICP_SVR[n]	GICP0 GICP_SVR[n]	0x00000000
0x31230A00	GICP0_GICP_FR[n]	GICP0 GICP_FR[n]	0x00000000

Table A-94: ADSP-SC59x GICP0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31230A04	GICP0_GICP_FR[n]	GICP0 GICP_FR[n]	0x00000000
0x31230A08	GICP0_GICP_FR[n]	GICP0 GICP_FR[n]	0x00000000
0x31230A0C	GICP0_GICP_FR[n]	GICP0 GICP_FR[n]	0x00000000
0x31230A10	GICP0_GICP_FR[n]	GICP0 GICP_FR[n]	0x00000000
0x31230C00	GICP0_GICP_CNTENSET0	GICP0 GICP_CNTENSET0	0x00000000
0x31230C20	GICP0_GICP_CNTENCLR0	GICP0 GICP_CNTENCLR0	0x00000000
0x31230C40	GICP0_GICP_INTENSET0	GICP0 GICP_INTENSET0	0x00000000
0x31230C60	GICP0_GICP_INTENCLR0	GICP0 GICP_INTENCLR0	0x00000000
0x31230C80	GICP0_GICP_OVSCLR0	GICP0 GICP_OVSCLR0	0x00000000
0x31230CC0	GICP0_GICP_OVSSET0	GICP0 GICP_OVSSET0	0x00000000
0x31230D88	GICP0_GICP_CAPR	GICP0 GICP_CAPR	0x00000000
0x31230E00	GICP0_GICP_CFGR	GICP0 GICP_CFGR	0x00401F04
0x31230E04	GICP0_GICP_CR	GICP0 GICP_CR	0x00000000
0x31230E50	GICP0_GICP_IRQCR	GICP0 GICP_IRQCR	0x00000000
0x31230FB8	GICP0_GICP_PMAUTHSTATUS	GICP0 GICP_PMAUTHSTATUS	0x00000088
0x31230FBC	GICP0_GICP_PMDEVARCH	GICP0 GICP_PMDEVARCH	0x47702A56
0x31230FCC	GICP0_GICP_PMDEVTYPE	GICP0 GICP_PMDEVTYPE	0x00000056
0x31230FD0	GICP0_GICP_PIDR4	GICP0 GICP_PIDR4	0x00000044
0x31230FE0	GICP0_GICP_PIDR0	GICP0 GICP_PIDR0	0x00000096
0x31230FE4	GICP0_GICP_PIDR1	GICP0 GICP_PIDR1	0x000000B4
0x31230FE8	GICP0_GICP_PIDR2	GICP0 GICP_PIDR2	0x0000003B
0x31230FEC	GICP0_GICP_PIDR3	GICP0 GICP_PIDR3	0x00000000
0x31230FF0	GICP0_GICP_CIDR0	GICP0 GICP_CIDR0	0x0000000D
0x31230FF4	GICP0_GICP_CIDR1	GICP0 GICP_CIDR1	0x000000F0
0x31230FF8	GICP0_GICP_CIDR2	GICP0 GICP_CIDR2	0x00000005
0x31230FFC	GICP0_GICP_CIDR3	GICP0 GICP_CIDR3	0x000000B1

Table A-95: ADSP-SC59x GICRLPI00 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31240000	GICRLPI00_GICR0_CTLR	GICRLPI00 GICR0_CTLR	0x00000000
0x31240004	GICRLPI00_GICR0_IIDR	GICRLPI00 GICR0_IIDR	0x0201743B
0x31240008	GICRLPI00_GICR0_TYPER	GICRLPI00 GICR0_TYPER	0x00000030
0x31240014	GICRLPI00_GICR0_WAKER	GICRLPI00 GICR0_WAKER	0x00000006
0x31240020	GICRLPI00_GICR0_FCTLR	GICRLPI00 GICR0_FCTLR	0x00000000
0x31240024	GICRLPI00_GICR0_PWRR	GICRLPI00 GICR0_PWRR	0x0000000D
0x31240028	GICRLPI00_GICR0_CLASS	GICRLPI00 GICR0_CLASS	0x00000000
0x3124FFD0	GICRLPI00_GICR0_PIDR4	GICRLPI00 GICR0_PIDR4	0x00000044
0x3124FFE0	GICRLPI00_GICR0_PIDR0	GICRLPI00 GICR0_PIDR0	0x00000093
0x3124FFE4	GICRLPI00_GICR0_PIDR1	GICRLPI00 GICR0_PIDR1	0x000000B4
0x3124FFE8	GICRLPI00_GICR0_PIDR2	GICRLPI00 GICR0_PIDR2	0x0000003B
0x3124FFEC	GICRLPI00_GICR0_PIDR3	GICRLPI00 GICR0_PIDR3	0x00000000
0x3124FFF0	GICRLPI00_GICR0_CIDR0	GICRLPI00 GICR0_CIDR0	0x0000000D
0x3124FFF4	GICRLPI00_GICR0_CIDR1	GICRLPI00 GICR0_CIDR1	0x000000F0
0x3124FFF8	GICRLPI00_GICR0_CIDR2	GICRLPI00 GICR0_CIDR2	0x00000005
0x3124FFFC	GICRLPI00_GICR0_CIDR3	GICRLPI00 GICR0_CIDR3	0x000000B1

Table A-96: ADSP-SC59x GICRSGI00 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31250080	GICRSGI00_GICR0_IGROUPR0	GICRSGI00 GICR0_IGROUPR0	0x00000000
0x31250100	GICRSGI00_GICR0_ISENABLER0	GICRSGI00 GICR0_ISENABLER0	0x00000000
0x31250180	GICRSGI00_GICR0_ICENABLER0	GICRSGI00 GICR0_ICENABLER0	0x00000000
0x31250200	GICRSGI00_GICR0_ISPENDR0	GICRSGI00 GICR0_ISPENDR0	0x00000000
0x31250280	GICRSGI00_GICR0_ICPENDR0	GICRSGI00 GICR0_ICPENDR0	0x00000000
0x31250300	GICRSGI00_GICR0_ISACTIVER0	GICRSGI00 GICR0_ISACTIVER0	0x00000000

Table A-96: ADSP-SC59x GICRSGI00 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31250380	GICRSGI00_GICR0_ICACTIVER0	GICRSGI00 GICR0_ICACTIVER0	0x00000000
0x31250400	GICRSGI00_GICR0_IPRIORITYR[n]	GICRSGI00 GICR0_IPRIORITYR[n]	0x00000000
0x31250404	GICRSGI00_GICR0_IPRIORITYR[n]	GICRSGI00 GICR0_IPRIORITYR[n]	0x00000000
0x31250408	GICRSGI00_GICR0_IPRIORITYR[n]	GICRSGI00 GICR0_IPRIORITYR[n]	0x00000000
0x3125040C	GICRSGI00_GICR0_IPRIORITYR[n]	GICRSGI00 GICR0_IPRIORITYR[n]	0x00000000
0x31250410	GICRSGI00_GICR0_IPRIORITYR[n]	GICRSGI00 GICR0_IPRIORITYR[n]	0x00000000
0x31250414	GICRSGI00_GICR0_IPRIORITYR[n]	GICRSGI00 GICR0_IPRIORITYR[n]	0x00000000
0x31250418	GICRSGI00_GICR0_IPRIORITYR[n]	GICRSGI00 GICR0_IPRIORITYR[n]	0x00000000
0x3125041C	GICRSGI00_GICR0_IPRIORITYR[n]	GICRSGI00 GICR0_IPRIORITYR[n]	0x00000000
0x31250C00	GICRSGI00_GICR0_ICFGR0	GICRSGI00 GICR0_ICFGR0	0xAAAAAAAA
0x31250C04	GICRSGI00_GICR0_ICFGR1	GICRSGI00 GICR0_ICFGR1	0x00000000
0x31250D00	GICRSGI00_GICR0_IGRPMODR0	GICRSGI00 GICR0_IGRPMODR0	0x00000000
0x31250E00	GICRSGI00_GICR0_NSACR	GICRSGI00 GICR0_NSACR	0x00000000
0x3125C000	GICRSGI00_GICR0_MISCSTATUSR	GICRSGI00 GICR0_MISCSTATUSR	0x00000000
0x3125C008	GICRSGI00_GICR0_IERRVR	GICRSGI00 GICR0_IERRVR	0x00000000

Table A-96: ADSP-SC59x GICRSGI00 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3125C010	GICRSGI00_GICR0_SGIDR	GICRSGI00 GICR0_SGIDR	0x00000000
0x3125F000	GICRSGI00_GICR0_CFGID0	GICRSGI00 GICR0_CFGID0	0x88840800
0x3125F004	GICRSGI00_GICR0_CFGID1	GICRSGI00 GICR0_CFGID1	0xA00F1010

Table A-97: ADSP-SC59x GICT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3122E000	GICT0_GICT_ERRGSR0	GICT0 GICT_ERRGSR0	0x00000000
0x3122E800	GICT0_GICT_ERRIRQCR[n]	GICT0 GICT_ERRIRQCR[n]	0x00000000
0x3122E808	GICT0_GICT_ERRIRQCR[n]	GICT0 GICT_ERRIRQCR[n]	0x00000000
0x3122FFBC	GICT0_GICT_DEVARCH	GICT0 GICT_DEVARCH	0x47700A00
0x3122FFC8	GICT0_GICT_DEVID	GICT0 GICT_DEVID	0x00000009
0x3122FFD0	GICT0_GICT_PIDR4	GICT0 GICT_PIDR4	0x00000044
0x3122FFE0	GICT0_GICT_PIDR0	GICT0 GICT_PIDR0	0x00000095
0x3122FFE4	GICT0_GICT_PIDR1	GICT0 GICT_PIDR1	0x000000B4
0x3122FFE8	GICT0_GICT_PIDR2	GICT0 GICT_PIDR2	0x0000003B
0x3122FFEC	GICT0_GICT_PIDR3	GICT0 GICT_PIDR3	0x00000000
0x3122FFF0	GICT0_GICT_CIDR0	GICT0 GICT_CIDR0	0x0000000D
0x3122FFF4	GICT0_GICT_CIDR1	GICT0 GICT_CIDR1	0x000000F0
0x3122FFF8	GICT0_GICT_CIDR2	GICT0 GICT_CIDR2	0x00000005
0x3122FFFC	GICT0_GICT_CIDR3	GICT0 GICT_CIDR3	0x000000B1

Table A-98: ADSP-SC59x HADC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31016000	HADC0_CTL	HADC0 Control Register	0x000001AA
0x31016004	HADC0_CHAN_MSK	HADC0 Channel Mask Register	0x0000FF00
0x31016008	HADC0_IMSK	HADC0 Interrupt Mask Register	0x00000000
0x3101600C	HADC0_STAT	HADC0 Status Register	0x00000000

Table A-98: ADSP-SC59x HADC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31016010	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016014	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016018	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x3101601C	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016020	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016024	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016028	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x3101602C	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016030	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016034	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016038	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x3101603C	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016040	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016044	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x31016048	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000
0x3101604C	HADC0_DATA[nn]	HADC0 Channel Data Registers	0x00000000

Table A-99: ADSP-SC59x IIR MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310BA000	IIR_CTL1	IIR Global Control Register	0x00000000
0x310BA004	IIR_DMASTAT	IIR DMA Status Register	0x00000000
0x310BA008	IIR_MACSTAT	IIR MAC Status Register	0x00000000
0x310BA00C	IIR_DBG_CTL	IIR IIR Debug Control Register	0x00000000
0x310BA010	IIR_DBG_ADDR	IIR IIR Debug Address Register	0x00000000
0x310BA014	IIR_DBG_WRDAT_LO	IIR IIR Debug Write Data Low Register	0x00000000
0x310BA018	IIR_DBG_WRDAT_HI	IIR IIR Debug Write Data High Register	0x00000000
0x310BA01C	IIR_DBG_RDDAT_LO	IIR IIR Debug Read Data Low Register	0x00000000
0x310BA020	IIR_DBG_RDDAT_HI	IIR IIR Debug Read Data High Register	0x00000000
0x310BA040	IIR_CTL2	IIR Channel Control Register	0x00000000
0x310BA044	IIR_INIDX	IIR Input Data Index Register	0x00000000

Table A-99: ADSP-SC59x IIR MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310BA048	IIR_INMOD	IIR Input Data Index Modifier Register	0x00000000
0x310BA04C	IIR_INLEN	IIR Input Data Buffer Length Register	0x00000000
0x310BA050	IIR_INBASE	IIR Input Buffer Base Register	0x00000000
0x310BA054	IIR_OUTIDX	IIR Output Data Buffer Index Register	0x00000000
0x310BA058	IIR_OUTMOD	IIR IIR Output Data Index Modifier Register	0x00000000
0x310BA05C	IIR_OUTLEN	IIR IIR Output Data Buffer Length Register	0x00000000
0x310BA060	IIR_OUTBASE	IIR Output Buffer Base Register	0x00000000
0x310BA064	IIR_COEFIDX	IIR Coefficient Buffer Index Register	0x00000000
0x310BA068	IIR_COEFMOD	IIR Coefficient Index Modifier Register	0x00000000
0x310BA06C	IIR_COEFLEN	IIR Coefficient Buffer Length Register	0x00000000
0x310BA070	IIR_CHNPTR	IIR Chain Pointer Register	0x00000000
0x310BA074	IIR_SCTL1	IIR Software Control Register1	0x00000000
0x310BA078	IIR_SCTL2	IIR Software Control Register2	0x00000000
0x310BA07C	IIR_SGCTL	IIR Secondary Global Control Register	0x00000000
0x310C4000	IIR0_CTL1	IIR0 Global Control Register	0x00000000
0x310C4004	IIR0_DMASTAT	IIR0 DMA Status Register	0x00000000
0x310C4008	IIR0_MACSTAT	IIR0 MAC Status Register	0x00000000
0x310C400C	IIR0_DBG_CTL	IIR0 IIR Debug Control Register	0x00000000
0x310C4010	IIR0_DBG_ADDR	IIR0 IIR Debug Address Register	0x00000000
0x310C4014	IIR0_DBG_WRDAT_LO	IIR0 IIR Debug Write Data Low Register	0x00000000
0x310C4018	IIR0_DBG_WRDAT_HI	IIR0 IIR Debug Write Data High Register	0x00000000
0x310C401C	IIR0_DBG_RDDAT_LO	IIR0 IIR Debug Read Data Low Register	0x00000000
0x310C4020	IIR0_DBG_RDDAT_HI	IIR0 IIR Debug Read Data High Register	0x00000000
0x310C4040	IIR0_CTL2	IIR0 Channel Control Register	0x00000000
0x310C4044	IIR0_INIDX	IIR0 Input Data Index Register	0x00000000
0x310C4048	IIR0_INMOD	IIR0 Input Data Index Modifier Register	0x00000000
0x310C404C	IIR0_INLEN	IIR0 Input Data Buffer Length Register	0x00000000
0x310C4050	IIR0_INBASE	IIR0 Input Buffer Base Register	0x00000000
0x310C4054	IIR0_OUTIDX	IIR0 Output Data Buffer Index Register	0x00000000
0x310C4058	IIR0_OUTMOD	IIR0 IIR Output Data Index Modifier Register	0x00000000
0x310C405C	IIR0_OUTLEN	IIR0 IIR Output Data Buffer Length Register	0x00000000

Table A-99: ADSP-SC59x IIR MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C4060	IIR0_OUTBASE	IIR0 Output Buffer Base Register	0x00000000
0x310C4064	IIR0_COEFIDX	IIR0 Coefficient Buffer Index Register	0x00000000
0x310C4068	IIR0_COEFMOD	IIR0 Coefficient Index Modifier Register	0x00000000
0x310C406C	IIR0_COEFLEN	IIR0 Coefficient Buffer Length Register	0x00000000
0x310C4070	IIR0_CHNPTR	IIR0 Chain Pointer Register	0x00000000
0x310C4074	IIR0_SCTL1	IIR0 Software Control Register1	0x00000000
0x310C4078	IIR0_SCTL2	IIR0 Software Control Register2	0x00000000
0x310C407C	IIR0_SGCTL	IIR0 Secondary Global Control Register	0x00000000

Table A-100: ADSP-SC59x IIR1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C0000	IIR1_CTL1	IIR1 Global Control Register	0x00000000
0x310C0004	IIR1_DMASTAT	IIR1 DMA Status Register	0x00000000
0x310C0008	IIR1_MACSTAT	IIR1 MAC Status Register	0x00000000
0x310C000C	IIR1_DBG_CTL	IIR1 IIR Debug Control Register	0x00000000
0x310C0010	IIR1_DBG_ADDR	IIR1 IIR Debug Address Register	0x00000000
0x310C0014	IIR1_DBG_WRDAT_LO	IIR1 IIR Debug Write Data Low Register	0x00000000
0x310C0018	IIR1_DBG_WRDAT_HI	IIR1 IIR Debug Write Data High Register	0x00000000
0x310C001C	IIR1_DBG_RDDAT_LO	IIR1 IIR Debug Read Data Low Register	0x00000000
0x310C0020	IIR1_DBG_RDDAT_HI	IIR1 IIR Debug Read Data High Register	0x00000000
0x310C0040	IIR1_CTL2	IIR1 Channel Control Register	0x00000000
0x310C0044	IIR1_INIDX	IIR1 Input Data Index Register	0x00000000
0x310C0048	IIR1_INMOD	IIR1 Input Data Index Modifier Register	0x00000000
0x310C004C	IIR1_INLEN	IIR1 Input Data Buffer Length Register	0x00000000
0x310C0050	IIR1_INBASE	IIR1 Input Buffer Base Register	0x00000000
0x310C0054	IIR1_OUTIDX	IIR1 Output Data Buffer Index Register	0x00000000
0x310C0058	IIR1_OUTMOD	IIR1 IIR Output Data Index Modifier Register	0x00000000
0x310C005C	IIR1_OUTLEN	IIR1 IIR Output Data Buffer Length Register	0x00000000
0x310C0060	IIR1_OUTBASE	IIR1 Output Buffer Base Register	0x00000000
0x310C0064	IIR1_COEFIDX	IIR1 Coefficient Buffer Index Register	0x00000000

Table A-100: ADSP-SC59x IIR1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C0068	IIR1_COEFMOD	IIR1 Coefficient Index Modifier Register	0x00000000
0x310C006C	IIR1_COEFLEN	IIR1 Coefficient Buffer Length Register	0x00000000
0x310C0070	IIR1_CHNPTR	IIR1 Chain Pointer Register	0x00000000
0x310C0074	IIR1_SCTL1	IIR1 Software Control Register1	0x00000000
0x310C0078	IIR1_SCTL2	IIR1 Software Control Register2	0x00000000
0x310C007C	IIR1_SGCTL	IIR1 Secondary Global Control Register	0x00000000

Table A-101: ADSP-SC59x IIR2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C1000	IIR2_CTL1	IIR2 Global Control Register	0x00000000
0x310C1004	IIR2_DMASTAT	IIR2 DMA Status Register	0x00000000
0x310C1008	IIR2_MACSTAT	IIR2 MAC Status Register	0x00000000
0x310C100C	IIR2_DBG_CTL	IIR2 IIR Debug Control Register	0x00000000
0x310C1010	IIR2_DBG_ADDR	IIR2 IIR Debug Address Register	0x00000000
0x310C1014	IIR2_DBG_WRDAT_LO	IIR2 IIR Debug Write Data Low Register	0x00000000
0x310C1018	IIR2_DBG_WRDAT_HI	IIR2 IIR Debug Write Data High Register	0x00000000
0x310C101C	IIR2_DBG_RDDAT_LO	IIR2 IIR Debug Read Data Low Register	0x00000000
0x310C1020	IIR2_DBG_RDDAT_HI	IIR2 IIR Debug Read Data High Register	0x00000000
0x310C1040	IIR2_CTL2	IIR2 Channel Control Register	0x00000000
0x310C1044	IIR2_INIDX	IIR2 Input Data Index Register	0x00000000
0x310C1048	IIR2_INMOD	IIR2 Input Data Index Modifier Register	0x00000000
0x310C104C	IIR2_INLEN	IIR2 Input Data Buffer Length Register	0x00000000
0x310C1050	IIR2_INBASE	IIR2 Input Buffer Base Register	0x00000000
0x310C1054	IIR2_OUTIDX	IIR2 Output Data Buffer Index Register	0x00000000
0x310C1058	IIR2_OUTMOD	IIR2 IIR Output Data Index Modifier Register	0x00000000
0x310C105C	IIR2_OUTLEN	IIR2 IIR Output Data Buffer Length Register	0x00000000
0x310C1060	IIR2_OUTBASE	IIR2 Output Buffer Base Register	0x00000000
0x310C1064	IIR2_COEFIDX	IIR2 Coefficient Buffer Index Register	0x00000000
0x310C1068	IIR2_COEFMOD	IIR2 Coefficient Index Modifier Register	0x00000000
0x310C106C	IIR2_COEFLEN	IIR2 Coefficient Buffer Length Register	0x00000000

Table A-101: ADSP-SC59x IIR2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C1070	IIR2_CHNPTR	IIR2 Chain Pointer Register	0x00000000
0x310C1074	IIR2_SCTL1	IIR2 Software Control Register1	0x00000000
0x310C1078	IIR2_SCTL2	IIR2 Software Control Register2	0x00000000
0x310C107C	IIR2_SGCTL	IIR2 Secondary Global Control Register	0x00000000

Table A-102: ADSP-SC59x IIR3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C2000	IIR3_CTL1	IIR3 Global Control Register	0x00000000
0x310C2004	IIR3_DMASTAT	IIR3 DMA Status Register	0x00000000
0x310C2008	IIR3_MACSTAT	IIR3 MAC Status Register	0x00000000
0x310C200C	IIR3_DBG_CTL	IIR3 IIR Debug Control Register	0x00000000
0x310C2010	IIR3_DBG_ADDR	IIR3 IIR Debug Address Register	0x00000000
0x310C2014	IIR3_DBG_WRDAT_LO	IIR3 IIR Debug Write Data Low Register	0x00000000
0x310C2018	IIR3_DBG_WRDAT_HI	IIR3 IIR Debug Write Data High Register	0x00000000
0x310C201C	IIR3_DBG_RDDAT_LO	IIR3 IIR Debug Read Data Low Register	0x00000000
0x310C2020	IIR3_DBG_RDDAT_HI	IIR3 IIR Debug Read Data High Register	0x00000000
0x310C2040	IIR3_CTL2	IIR3 Channel Control Register	0x00000000
0x310C2044	IIR3_INIDX	IIR3 Input Data Index Register	0x00000000
0x310C2048	IIR3_INMOD	IIR3 Input Data Index Modifier Register	0x00000000
0x310C204C	IIR3_INLEN	IIR3 Input Data Buffer Length Register	0x00000000
0x310C2050	IIR3_INBASE	IIR3 Input Buffer Base Register	0x00000000
0x310C2054	IIR3_OUTIDX	IIR3 Output Data Buffer Index Register	0x00000000
0x310C2058	IIR3_OUTMOD	IIR3 IIR Output Data Index Modifier Register	0x00000000
0x310C205C	IIR3_OUTLEN	IIR3 IIR Output Data Buffer Length Register	0x00000000
0x310C2060	IIR3_OUTBASE	IIR3 Output Buffer Base Register	0x00000000
0x310C2064	IIR3_COEFIDX	IIR3 Coefficient Buffer Index Register	0x00000000
0x310C2068	IIR3_COEFMOD	IIR3 Coefficient Index Modifier Register	0x00000000
0x310C206C	IIR3_COEFLEN	IIR3 Coefficient Buffer Length Register	0x00000000
0x310C2070	IIR3_CHNPTR	IIR3 Chain Pointer Register	0x00000000
0x310C2074	IIR3_SCTL1	IIR3 Software Control Register1	0x00000000

Table A-102: ADSP-SC59x IIR3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C2078	IIR3_SCTL2	IIR3 Software Control Register2	0x00000000
0x310C207C	IIR3_SGCTL	IIR3 Secondary Global Control Register	0x00000000

Table A-103: ADSP-SC59x IIR4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310BE000	IIR4_CTL1	IIR4 Global Control Register	0x00000000
0x310BE004	IIR4_DMASTAT	IIR4 DMA Status Register	0x00000000
0x310BE008	IIR4_MACSTAT	IIR4 MAC Status Register	0x00000000
0x310BE00C	IIR4_DBG_CTL	IIR4 IIR Debug Control Register	0x00000000
0x310BE010	IIR4_DBG_ADDR	IIR4 IIR Debug Address Register	0x00000000
0x310BE014	IIR4_DBG_WRDAT_LO	IIR4 IIR Debug Write Data Low Register	0x00000000
0x310BE018	IIR4_DBG_WRDAT_HI	IIR4 IIR Debug Write Data High Register	0x00000000
0x310BE01C	IIR4_DBG_RDDAT_LO	IIR4 IIR Debug Read Data Low Register	0x00000000
0x310BE020	IIR4_DBG_RDDAT_HI	IIR4 IIR Debug Read Data High Register	0x00000000
0x310BE040	IIR4_CTL2	IIR4 Channel Control Register	0x00000000
0x310BE044	IIR4_INIDX	IIR4 Input Data Index Register	0x00000000
0x310BE048	IIR4_INMOD	IIR4 Input Data Index Modifier Register	0x00000000
0x310BE04C	IIR4_INLEN	IIR4 Input Data Buffer Length Register	0x00000000
0x310BE050	IIR4_INBASE	IIR4 Input Buffer Base Register	0x00000000
0x310BE054	IIR4_OUTIDX	IIR4 Output Data Buffer Index Register	0x00000000
0x310BE058	IIR4_OUTMOD	IIR4 IIR Output Data Index Modifier Register	0x00000000
0x310BE05C	IIR4_OUTLEN	IIR4 IIR Output Data Buffer Length Register	0x00000000
0x310BE060	IIR4_OUTBASE	IIR4 Output Buffer Base Register	0x00000000
0x310BE064	IIR4_COEFIDX	IIR4 Coefficient Buffer Index Register	0x00000000
0x310BE068	IIR4_COEFMOD	IIR4 Coefficient Index Modifier Register	0x00000000
0x310BE06C	IIR4_COEFLEN	IIR4 Coefficient Buffer Length Register	0x00000000
0x310BE070	IIR4_CHNPTR	IIR4 Chain Pointer Register	0x00000000
0x310BE074	IIR4_SCTL1	IIR4 Software Control Register1	0x00000000
0x310BE078	IIR4_SCTL2	IIR4 Software Control Register2	0x00000000
0x310BE07C	IIR4_SGCTL	IIR4 Secondary Global Control Register	0x00000000

Table A-104: ADSP-SC59x IIR5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310BA000	IIR5_CTL1	IIR5 Global Control Register	0x00000000
0x310BA004	IIR5_DMASTAT	IIR5 DMA Status Register	0x00000000
0x310BA008	IIR5_MACSTAT	IIR5 MAC Status Register	0x00000000
0x310BA00C	IIR5_DBG_CTL	IIR5 IIR Debug Control Register	0x00000000
0x310BA010	IIR5_DBG_ADDR	IIR5 IIR Debug Address Register	0x00000000
0x310BA014	IIR5_DBG_WRDAT_LO	IIR5 IIR Debug Write Data Low Register	0x00000000
0x310BA018	IIR5_DBG_WRDAT_HI	IIR5 IIR Debug Write Data High Register	0x00000000
0x310BA01C	IIR5_DBG_RDDAT_LO	IIR5 IIR Debug Read Data Low Register	0x00000000
0x310BA020	IIR5_DBG_RDDAT_HI	IIR5 IIR Debug Read Data High Register	0x00000000
0x310BA040	IIR5_CTL2	IIR5 Channel Control Register	0x00000000
0x310BA044	IIR5_INIDX	IIR5 Input Data Index Register	0x00000000
0x310BA048	IIR5_INMOD	IIR5 Input Data Index Modifier Register	0x00000000
0x310BA04C	IIR5_INLEN	IIR5 Input Data Buffer Length Register	0x00000000
0x310BA050	IIR5_INBASE	IIR5 Input Buffer Base Register	0x00000000
0x310BA054	IIR5_OUTIDX	IIR5 Output Data Buffer Index Register	0x00000000
0x310BA058	IIR5_OUTMOD	IIR5 IIR Output Data Index Modifier Register	0x00000000
0x310BA05C	IIR5_OUTLEN	IIR5 IIR Output Data Buffer Length Register	0x00000000
0x310BA060	IIR5_OUTBASE	IIR5 Output Buffer Base Register	0x00000000
0x310BA064	IIR5_COEFIDX	IIR5 Coefficient Buffer Index Register	0x00000000
0x310BA068	IIR5_COEFMOD	IIR5 Coefficient Index Modifier Register	0x00000000
0x310BA06C	IIR5_COEFLEN	IIR5 Coefficient Buffer Length Register	0x00000000
0x310BA070	IIR5_CHNPTR	IIR5 Chain Pointer Register	0x00000000
0x310BA074	IIR5_SCTL1	IIR5 Software Control Register1	0x00000000
0x310BA078	IIR5_SCTL2	IIR5 Software Control Register2	0x00000000
0x310BA07C	IIR5_SGCTL	IIR5 Secondary Global Control Register	0x00000000

Table A-105: ADSP-SC59x IIR6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310BB000	IIR6_CTL1	IIR6 Global Control Register	0x00000000
0x310BB004	IIR6_DMASTAT	IIR6 DMA Status Register	0x00000000

Table A-105: ADSP-SC59x IIR6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310BB008	IIR6_MACSTAT	IIR6 MAC Status Register	0x00000000
0x310BB00C	IIR6_DBG_CTL	IIR6 IIR Debug Control Register	0x00000000
0x310BB010	IIR6_DBG_ADDR	IIR6 IIR Debug Address Register	0x00000000
0x310BB014	IIR6_DBG_WRDAT_LO	IIR6 IIR Debug Write Data Low Register	0x00000000
0x310BB018	IIR6_DBG_WRDAT_HI	IIR6 IIR Debug Write Data High Register	0x00000000
0x310BB01C	IIR6_DBG_RDDAT_LO	IIR6 IIR Debug Read Data Low Register	0x00000000
0x310BB020	IIR6_DBG_RDDAT_HI	IIR6 IIR Debug Read Data High Register	0x00000000
0x310BB040	IIR6_CTL2	IIR6 Channel Control Register	0x00000000
0x310BB044	IIR6_INIDX	IIR6 Input Data Index Register	0x00000000
0x310BB048	IIR6_INMOD	IIR6 Input Data Index Modifier Register	0x00000000
0x310BB04C	IIR6_INLEN	IIR6 Input Data Buffer Length Register	0x00000000
0x310BB050	IIR6_INBASE	IIR6 Input Buffer Base Register	0x00000000
0x310BB054	IIR6_OUTIDX	IIR6 Output Data Buffer Index Register	0x00000000
0x310BB058	IIR6_OUTMOD	IIR6 IIR Output Data Index Modifier Register	0x00000000
0x310BB05C	IIR6_OUTLEN	IIR6 IIR Output Data Buffer Length Register	0x00000000
0x310BB060	IIR6_OUTBASE	IIR6 Output Buffer Base Register	0x00000000
0x310BB064	IIR6_COEFIDX	IIR6 Coefficient Buffer Index Register	0x00000000
0x310BB068	IIR6_COEFMOD	IIR6 Coefficient Index Modifier Register	0x00000000
0x310BB06C	IIR6_COEFLEN	IIR6 Coefficient Buffer Length Register	0x00000000
0x310BB070	IIR6_CHNPTR	IIR6 Chain Pointer Register	0x00000000
0x310BB074	IIR6_SCTL1	IIR6 Software Control Register1	0x00000000
0x310BB078	IIR6_SCTL2	IIR6 Software Control Register2	0x00000000
0x310BB07C	IIR6_SGCTL	IIR6 Secondary Global Control Register	0x00000000

Table A-106: ADSP-SC59x IIR7 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310BC000	IIR7_CTL1	IIR7 Global Control Register	0x00000000
0x310BC004	IIR7_DMASTAT	IIR7 DMA Status Register	0x00000000
0x310BC008	IIR7_MACSTAT	IIR7 MAC Status Register	0x00000000
0x310BC00C	IIR7_DBG_CTL	IIR7 IIR Debug Control Register	0x00000000

Table A-106: ADSP-SC59x IIR7 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310BC010	IIR7_DBG_ADDR	IIR7 IIR Debug Address Register	0x00000000
0x310BC014	IIR7_DBG_WRDAT_LO	IIR7 IIR Debug Write Data Low Register	0x00000000
0x310BC018	IIR7_DBG_WRDAT_HI	IIR7 IIR Debug Write Data High Register	0x00000000
0x310BC01C	IIR7_DBG_RDDAT_LO	IIR7 IIR Debug Read Data Low Register	0x00000000
0x310BC020	IIR7_DBG_RDDAT_HI	IIR7 IIR Debug Read Data High Register	0x00000000
0x310BC040	IIR7_CTL2	IIR7 Channel Control Register	0x00000000
0x310BC044	IIR7_INIDX	IIR7 Input Data Index Register	0x00000000
0x310BC048	IIR7_INMOD	IIR7 Input Data Index Modifier Register	0x00000000
0x310BC04C	IIR7_INLEN	IIR7 Input Data Buffer Length Register	0x00000000
0x310BC050	IIR7_INBASE	IIR7 Input Buffer Base Register	0x00000000
0x310BC054	IIR7_OUTIDX	IIR7 Output Data Buffer Index Register	0x00000000
0x310BC058	IIR7_OUTMOD	IIR7 IIR Output Data Index Modifier Register	0x00000000
0x310BC05C	IIR7_OUTLEN	IIR7 IIR Output Data Buffer Length Register	0x00000000
0x310BC060	IIR7_OUTBASE	IIR7 Output Buffer Base Register	0x00000000
0x310BC064	IIR7_COEFIDX	IIR7 Coefficient Buffer Index Register	0x00000000
0x310BC068	IIR7_COEFMOD	IIR7 Coefficient Index Modifier Register	0x00000000
0x310BC06C	IIR7_COEFLEN	IIR7 Coefficient Buffer Length Register	0x00000000
0x310BC070	IIR7_CHNPTR	IIR7 Chain Pointer Register	0x00000000
0x310BC074	IIR7_SCTL1	IIR7 Software Control Register1	0x00000000
0x310BC078	IIR7_SCTL2	IIR7 Software Control Register2	0x00000000
0x310BC07C	IIR7_SGCTL	IIR7 Secondary Global Control Register	0x00000000

Table A-107: ADSP-SC59x L2CTL0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31080000	L2CTL0_CTL	L2CTL0 Control Register	0x00000000
0x31080010	L2CTL0_STAT	L2CTL0 Status Register	0x00000000
0x31080014	L2CTL0_RPCR0	L2CTL0 Read Priority Count Register	0x0F0F0F0F
0x31080018	L2CTL0_WPCR0	L2CTL0 Write Priority Count Register	0x0F0F0F0F
0x31080024	L2CTL0_INIT	L2CTL0 Initialization Register	0x00000000
0x31080038	L2CTL0_ISTAT	L2CTL0 Initialization Status Register	0x00000000

Table A-107: ADSP-SC59x L2CTL0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108003C	L2CTL0_PCTL	L2CTL0 Power Control Register	0x00000000
0x31080040	L2CTL0_ERRADDR0	L2CTL0 ECC Error Address 0 Register	0x20000000
0x31080044	L2CTL0_ERRADDR1	L2CTL0 ECC Error Address 1 Register	0x20040000
0x31080048	L2CTL0_ERRADDR2	L2CTL0 ECC Error Address 2 Register	0x20080000
0x3108004C	L2CTL0_ERRADDR3	L2CTL0 ECC Error Address 3 Register	0x200C0000
0x31080050	L2CTL0_ERRADDR4	L2CTL0 ECC Error Address 4 Register	0x20100000
0x31080054	L2CTL0_ERRADDR5	L2CTL0 ECC Error Address 5 Register	0x20140000
0x31080058	L2CTL0_ERRADDR6	L2CTL0 ECC Error Address 6 Register	0x20180000
0x3108005C	L2CTL0_ERRADDR7	L2CTL0 ECC Error Address 7 Register	0x201C0000
0x31080060	L2CTL0_ERRADDR8	L2CTL0 ECC Error Address 8 Register	0x20200000
0x31080080	L2CTL0_ET0	L2CTL0 Error Type 0 Register	0x00000000
0x31080084	L2CTL0_EADDR0	L2CTL0 Error Type 0 Address Register	0x00000000
0x31080088	L2CTL0_ET1	L2CTL0 Error Type 1 Register	0x00000000
0x3108008C	L2CTL0_EADDR1	L2CTL0 Error Type 1 Address Register	0x00000000
0x310800EC	L2CTL0_SCTL	L2CTL0 Scrub Control Register	0x00000000
0x310800F0	L2CTL0_SADR	L2CTL0 Scrub Start Address Register	0x00000000
0x310800F4	L2CTL0_SCNT	L2CTL0 Scrub Count Register	0x00000000
0x310800FC	L2CTL0_REVID	L2CTL0 Revision ID Register	0x00000003
0x31080100	L2CTL0_STAT_1	L2CTL0 L2 Port Error Status Register	0x00000000
0x31080104	L2CTL0_ET2	L2CTL0 Error Type 2 Register	0x00000000
0x31080108	L2CTL0_EADDR2	L2CTL0 Error Type 2 Address Register	0x00000000
0x3108010C	L2CTL0_ET3	L2CTL0 Error Type 3 Register	0x00000000
0x31080110	L2CTL0_EADDR3	L2CTL0 Error Type 3 Address Register	0x00000000
0x31080114	L2CTL0_ET4	L2CTL0 Error Type 4 Register	0x00000000
0x31080118	L2CTL0_EADDR4	L2CTL0 Error Type 4 Address Register	0x00000000
0x31080130	L2CTL0_RPCR1	L2CTL0 Read Priority Count Register	0x0000000F
0x31080134	L2CTL0_WPCR1	L2CTL0 Write Priority Count Register	0x0000000F

Table A-108: ADSP-SC59x LP0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFE000	LP0_CTL	LP0 Control Register	0x00000000
0x30FFE004	LP0_STAT	LP0 Status Register	0x00000000
0x30FFE008	LP0_DIV	LP0 Clock Divider Value Register	0x00000000
0x30FFE010	LP0_TX	LP0 Transmit Buffer Register	0x00000000
0x30FFE014	LP0_RX	LP0 Receive Buffer Register	0x00000000
0x30FFE018	LP0_TXIN_SHDW	LP0 Shadow Input Transmit Buffer Register	0x00000000
0x30FFE01C	LP0_TXOUT_SHDW	LP0 Shadow Output Transmit Buffer Register	0x00000000

Table A-109: ADSP-SC59x LP1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30FFE100	LP1_CTL	LP1 Control Register	0x00000000
0x30FFE104	LP1_STAT	LP1 Status Register	0x00000000
0x30FFE108	LP1_DIV	LP1 Clock Divider Value Register	0x00000000
0x30FFE110	LP1_TX	LP1 Transmit Buffer Register	0x00000000
0x30FFE114	LP1_RX	LP1 Receive Buffer Register	0x00000000
0x30FFE118	LP1_TXIN_SHDW	LP1 Shadow Input Transmit Buffer Register	0x00000000
0x30FFE11C	LP1_TXOUT_SHDW	LP1 Shadow Output Transmit Buffer Register	0x00000000

Table A-110: ADSP-SC59x MEC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A2000	MEC0_PEIRQ_GCTL[n]	MEC0 Parity Error Interrupt Request Global Control Register	0x00000000
0x310A2010	MEC0_PEIRQ_GSTAT[n]	MEC0 Parity Error Interrupt Request Global Status Register	0x00000000
0x310A2040	MEC0_PERR_CTL0	MEC0 Parity Error Control Register	0x00000000
0x310A2044	MEC0_PERR_CTL1	MEC0 Parity Error Control Register	0x00000000
0x310A2080	MEC0_PERR_STAT0	MEC0 Parity Error Status Register	0x00000000
0x310A2084	MEC0_PERR_STAT1	MEC0 Parity Error Status Register	0x00000000
0x310A20C0	MEC0_PERR_IMASK0	MEC0 Parity Error Interrupt Mask Register	0x00000000
0x310A20C4	MEC0_PERR_IMASK1	MEC0 Parity Error Interrupt Mask Register	0x00000000

Table A-110: ADSP-SC59x MEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A2100	MEC0_EEIRQ_GCTL[n]	MEC0 ECC Error Interrupt Request Global Control Register	0x00000000
0x310A2110	MEC0_EEIRQ_GSTAT[n]	MEC0 ECC Error Interrupt Request Global Status Register	0x00000000
0x310A2140	MEC0_ECCERR_CTL[n]	MEC0 ECC Error Control Register	0x00000000
0x310A2180	MEC0_ECCERR_STAT[n]	MEC0 ECC Error Status Register	0x00000000
0x310A21C0	MEC0_ECCERR_IMASK[n]	MEC0 ECC Error Interrupt Mask Register	0x00000000
0x310A2300	MEC0_A55EIRQ_GCTL[n]	MEC0 A55 Core Memory Errors Interrupt Request Global Control Register	0x00000000
0x310A2310	MEC0_A55EIRQ_GSTAT[n]	MEC0 A55 Error Interrupt Request Global Status Register	0x00000000
0x310A2340	MEC0_A55ERR_CTL[n]	MEC0 A55 Error Control Register	0x00000000
0x310A2380	MEC0_A55EIRQ_STAT[n]	MEC0 A55 Error Status Register	0x00000000
0x310A23C0	MEC0_A55ERR_IMASK[n]	MEC0 A55 Error Interrupt Mask Register	0x00000000
0x310A2F00	MEC0_CLR	MEC0 Clear Register	0x00000000
0x310A2FD0	MEC0_PID4	MEC0 Peripheral ID4 Register	0x00000000
0x310A2FE0	MEC0_PID0	MEC0 Peripheral ID0 Register	0x00000003
0x310A2FE4	MEC0_PID1	MEC0 Peripheral ID1 Register	0x00000050
0x310A2FE8	MEC0_PID2	MEC0 Peripheral ID2 Register	0x0000000E
0x310A2FEC	MEC0_PID3	MEC0 Peripheral ID3 Register	0x00000000
0x310A2FF0	MEC0_CID0	MEC0 Component ID0 Register	0x0000000D
0x310A2FF4	MEC0_CID1	MEC0 Component ID1 Register	0x000000F0
0x310A2FF8	MEC0_CID2	MEC0 Component ID2 Register	0x00000005
0x310A2FFC	MEC0_CID3	MEC0 Component ID3 Register	0x000000B1

Table A-111: ADSP-SC59x MEC1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A3000	MEC1_PEIRQ_GCTL[n]	MEC1 Parity Error Interrupt Request Global Control Register	0x00000000
0x310A3010	MEC1_PEIRQ_GSTAT[n]	MEC1 Parity Error Interrupt Request Global Status Register	0x00000000
0x310A3040	MEC1_PERR_CTL0	MEC1 Parity Error Control Register	0x00000000
0x310A3044	MEC1_PERR_CTL1	MEC1 Parity Error Control Register	0x00000000

Table A-111: ADSP-SC59x MEC1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A3080	MEC1_PERR_STAT0	MEC1 Parity Error Status Register	0x00000000
0x310A3084	MEC1_PERR_STAT1	MEC1 Parity Error Status Register	0x00000000
0x310A30C0	MEC1_PERR_IMASK0	MEC1 Parity Error Interrupt Mask Register	0x00000000
0x310A30C4	MEC1_PERR_IMASK1	MEC1 Parity Error Interrupt Mask Register	0x00000000
0x310A3100	MEC1_EEIRQ_GCTL[n]	MEC1 ECC Error Interrupt Request Global Control Register	0x00000000
0x310A3110	MEC1_EEIRQ_GSTAT[n]	MEC1 ECC Error Interrupt Request Global Status Register	0x00000000
0x310A3140	MEC1_ECCERR_CTL[n]	MEC1 ECC Error Control Register	0x00000000
0x310A3180	MEC1_ECCERR_STAT[n]	MEC1 ECC Error Status Register	0x00000000
0x310A31C0	MEC1_ECCERR_IMASK[n]	MEC1 ECC Error Interrupt Mask Register	0x00000000
0x310A3300	MEC1_A55EIRQ_GCTL[n]	MEC1 A55 Core Memory Errors Interrupt Request Global Control Register	0x00000000
0x310A3310	MEC1_A55EIRQ_GSTAT[n]	MEC1 A55 Error Interrupt Request Global Status Register	0x00000000
0x310A3340	MEC1_A55ERR_CTL[n]	MEC1 A55 Error Control Register	0x00000000
0x310A3380	MEC1_A55EIRQ_STAT[n]	MEC1 A55 Error Status Register	0x00000000
0x310A33C0	MEC1_A55ERR_IMASK[n]	MEC1 A55 Error Interrupt Mask Register	0x00000000
0x310A3F00	MEC1_CLR	MEC1 Clear Register	0x00000000
0x310A3FD0	MEC1_PID4	MEC1 Peripheral ID4 Register	0x00000000
0x310A3FE0	MEC1_PID0	MEC1 Peripheral ID0 Register	0x00000003
0x310A3FE4	MEC1_PID1	MEC1 Peripheral ID1 Register	0x00000050
0x310A3FE8	MEC1_PID2	MEC1 Peripheral ID2 Register	0x0000000E
0x310A3FEC	MEC1_PID3	MEC1 Peripheral ID3 Register	0x00000000
0x310A3FF0	MEC1_CID0	MEC1 Component ID0 Register	0x0000000D
0x310A3FF4	MEC1_CID1	MEC1 Component ID1 Register	0x000000F0
0x310A3FF8	MEC1_CID2	MEC1 Component ID2 Register	0x00000005
0x310A3FFC	MEC1_CID3	MEC1 Component ID3 Register	0x000000B1

Table A-112: ADSP-SC59x MEC2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A4000	MEC2_PEIRQ_GCTL[n]	MEC2 Parity Error Interrupt Request Global Control Register	0x00000000

Table A-112: ADSP-SC59x MEC2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A4010	MEC2_PEIRQ_GSTAT[n]	MEC2 Parity Error Interrupt Request Global Status Register	0x00000000
0x310A4040	MEC2_PERR_CTL0	MEC2 Parity Error Control Register	0x00000000
0x310A4044	MEC2_PERR_CTL1	MEC2 Parity Error Control Register	0x00000000
0x310A4080	MEC2_PERR_STAT0	MEC2 Parity Error Status Register	0x00000000
0x310A4084	MEC2_PERR_STAT1	MEC2 Parity Error Status Register	0x00000000
0x310A40C0	MEC2_PERR_IMASK0	MEC2 Parity Error Interrupt Mask Register	0x00000000
0x310A40C4	MEC2_PERR_IMASK1	MEC2 Parity Error Interrupt Mask Register	0x00000000
0x310A4100	MEC2_EEIRQ_GCTL[n]	MEC2 ECC Error Interrupt Request Global Control Register	0x00000000
0x310A4110	MEC2_EEIRQ_GSTAT[n]	MEC2 ECC Error Interrupt Request Global Status Register	0x00000000
0x310A4140	MEC2_ECCERR_CTL[n]	MEC2 ECC Error Control Register	0x00000000
0x310A4180	MEC2_ECCERR_STAT[n]	MEC2 ECC Error Status Register	0x00000000
0x310A41C0	MEC2_ECCERR_IMASK[n]	MEC2 ECC Error Interrupt Mask Register	0x00000000
0x310A4300	MEC2_A55EIRQ_GCTL[n]	MEC2 A55 Core Memory Errors Interrupt Request Global Control Register	0x00000000
0x310A4310	MEC2_A55EIRQ_GSTAT[n]	MEC2 A55 Error Interrupt Request Global Status Register	0x00000000
0x310A4340	MEC2_A55ERR_CTL[n]	MEC2 A55 Error Control Register	0x00000000
0x310A4380	MEC2_A55EIRQ_STAT[n]	MEC2 A55 Error Status Register	0x00000000
0x310A43C0	MEC2_A55ERR_IMASK[n]	MEC2 A55 Error Interrupt Mask Register	0x00000000
0x310A4F00	MEC2_CLR	MEC2 Clear Register	0x00000000
0x310A4FD0	MEC2_PID4	MEC2 Peripheral ID4 Register	0x00000000
0x310A4FE0	MEC2_PID0	MEC2 Peripheral ID0 Register	0x00000003
0x310A4FE4	MEC2_PID1	MEC2 Peripheral ID1 Register	0x00000050
0x310A4FE8	MEC2_PID2	MEC2 Peripheral ID2 Register	0x0000000E
0x310A4FEC	MEC2_PID3	MEC2 Peripheral ID3 Register	0x00000000
0x310A4FF0	MEC2_CID0	MEC2 Component ID0 Register	0x0000000D
0x310A4FF4	MEC2_CID1	MEC2 Component ID1 Register	0x000000F0
0x310A4FF8	MEC2_CID2	MEC2 Component ID2 Register	0x00000005
0x310A4FFC	MEC2_CID3	MEC2 Component ID3 Register	0x000000B1

Table A-113: ADSP-SC59x SCB6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30643108	SCB6_A55_M0_IB_FN_MOD	SCB6 A55 M0 Ib.fn Mod	0x00000000

Table A-114: ADSP-SC59x SCB1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30200020	SCB1_MST_IB_SYNC_MODE	SCB1 DMC Fabric (CLK03) Synchronization Mode Register	0x00000004

Table A-115: ADSP-SC59x SCB5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30400000	SCB5_SPI2_OSPI_REMAP	SCB5 SPI2/OSPI Memory Map Address Remap Register	0x00000000

Table A-116: ADSP-SC59x SCB0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30042100	SCB0_SP0A_READ_QOS	SCB0 SP0A Read Quality of Service Register	0x0000000C
0x30042104	SCB0_SP0A_WRITE_QOS	SCB0 SP0A Write Quality of Service Register	0x0000000C
0x30043100	SCB0_SP0B_READ_QOS	SCB0 SP0B Read Quality of Service Register	0x0000000C
0x30043104	SCB0_SP0B_WRITE_QOS	SCB0 SP0B Write Quality of Service Register	0x0000000C
0x30044100	SCB0_SP1A_READ_QOS	SCB0 SP1A Read Quality of Service Register	0x0000000C
0x30044104	SCB0_SP1A_WRITE_QOS	SCB0 SP1A Write Quality of Service Register	0x0000000C
0x30045100	SCB0_SP1B_READ_QOS	SCB0 SP1B Read Quality of Service Register	0x0000000C
0x30045104	SCB0_SP1B_WRITE_QOS	SCB0 SP1B Write Quality of Service Register	0x0000000C
0x30046100	SCB0_SP2A_READ_QOS	SCB0 SP2A Read Quality of Service Register	0x0000000C
0x30046104	SCB0_SP2A_WRITE_QOS	SCB0 SP2A Write Quality of Service Register	0x0000000C
0x30047100	SCB0_SP2B_READ_QOS	SCB0 SP1B Read Quality of Service Register	0x0000000C
0x30047104	SCB0_SP2B_WRITE_QOS	SCB0 SP1B Write Quality of Service Register	0x0000000C
0x30048100	SCB0_SP3A_READ_QOS	SCB0 SP3A Read Quality of Service Register	0x0000000C
0x30048104	SCB0_SP3A_WRITE_QOS	SCB0 SP3A Write Quality of Service Register	0x0000000C
0x30049100	SCB0_SP3B_READ_QOS	SCB0 SP3B Read Quality of Service Register	0x0000000C
0x30049104	SCB0_SP3B_WRITE_QOS	SCB0 SP3B Write Quality of Service Register	0x0000000C

Table A-116: ADSP-SC59x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3004A100	SCB0_CRC0_CH0_IB_READ_QOS	SCB0 CRC0 Channel 0 Read Quality of Service Register	0x00000001
0x3004A104	SCB0_CRC0_CH0_IB_WRITE_QOS	SCB0 CRC0 Channel 0 Write Quality of Service Register	0x00000001
0x3004B100	SCB0_CRC0_CH1_IB_READ_QOS	SCB0 CRC0 Channel 1 Read Quality of Service Register	0x00000001
0x3004B104	SCB0_CRC0_CH1_IB_WRITE_QOS	SCB0 CRC0 Channel 1 Write Quality of Service Register	0x00000001
0x3004C100	SCB0_CRC1_CH0_IB_READ_QOS	SCB0 CRC1 Channel 0 Read Quality of Service Register	0x00000001
0x3004C104	SCB0_CRC1_CH0_IB_WRITE_QOS	SCB0 CRC1 Channel 0 Write Quality of Service Register	0x00000001
0x3004D100	SCB0_CRC1_CH1_IB_READ_QOS	SCB0 CRC1 Channel 1 Read Quality of Service Register	0x00000001
0x3004D104	SCB0_CRC1_CH1_IB_WRITE_QOS	SCB0 CRC1 Channel 1 Write Quality of Service Register	0x00000001
0x3004E100	SCB0_UART2_RX_READ_QOS	SCB0 UART2 RX Read Quality of Service Register	0x0000000C
0x3004E104	SCB0_UART2_RX_WRITE_QOS	SCB0 UART2 RX Write Quality of Service Register	0x0000000C
0x3004F100	SCB0_SH0_DPORT_READ_QOS	SCB0 SH0 DPORT Read Quality of Service Register	0x00000007
0x3004F104	SCB0_SH0_DPORT_WRITE_QOS	SCB0 SH0 DPORT Write Quality of Service Register	0x00000007
0x30050100	SCB0_SH0_IPORT_READ_QOS	SCB0 SH0 IPORT Read Quality of Service Register	0x00000007
0x30050104	SCB0_SH0_IPORT_WRITE_QOS	SCB0 SH0 IPORT Write Quality of Service Register	0x00000007
0x30051100	SCB0_SP4A_READ_QOS	SCB0 SP4A Read Quality of Service Register	0x0000000C
0x30051104	SCB0_SP4A_WRITE_QOS	SCB0 SP4A Write Quality of Service Register	0x0000000C
0x30052100	SCB0_SP4B_READ_QOS	SCB0 SP4B Read Quality of Service Register	0x0000000C
0x30052104	SCB0_SP4B_WRITE_QOS	SCB0 SP4B Write Quality of Service Register	0x0000000C
0x30053100	SCB0_UART3_TX_READ_QOS	SCB0 UART3 TX Read Quality of Service Register	0x0000000C
0x30053104	SCB0_UART3_TX_WRITE_QOS	SCB0 UART3 TX Write Quality of Service Register	0x0000000C

Table A-116: ADSP-SC59x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30054100	SCB0_UART3_RX_READ_QOS	SCB0 UART3 RX Read Quality of Service Register	0x0000000C
0x30054104	SCB0_UART3_RX_WRITE_QOS	SCB0 UART3 RX Write Quality of Service Register	0x0000000C
0x30055100	SCB0_HSMDMA_CH0_IB_READ_QOS	SCB0 HSMDMA Channel 0 Read Quality of Service Register	0x00000001
0x30055104	SCB0_HSMDMA_CH0_IB_WRITE_QOS	SCB0 HSMDMA Channel 0 Write Quality of Service Register	0x00000001
0x30056100	SCB0_SP5A_READ_QOS	SCB0 SP5A Read Quality of Service Register	0x0000000C
0x30056104	SCB0_SP5A_WRITE_QOS	SCB0 SP5A Write Quality of Service Register	0x0000000C
0x30057100	SCB0_MLB_READ_QOS	SCB0 MLB Read Quality of Service Register	0x0000000C
0x30057104	SCB0_MLB_WRITE_QOS	SCB0 MLB Write Quality of Service Register	0x0000000C
0x30058100	SCB0_UART0_TX_READ_QOS	SCB0 UART0 TX Read Quality of Service Register	0x0000000C
0x30058104	SCB0_UART0_TX_WRITE_QOS	SCB0 UART0 TX Write Quality of Service Register	0x0000000C
0x30059100	SCB0_UART0_RX_READ_QOS	SCB0 UART0 RX Read Quality of Service Register	0x0000000C
0x30059104	SCB0_UART0_RX_WRITE_QOS	SCB0 UART0 RX Write Quality of Service Register	0x0000000C
0x3005A100	SCB0_SP5B_READ_QOS	SCB0 SP5B Read Quality of Service Register	0x0000000C
0x3005A104	SCB0_SP5B_WRITE_QOS	SCB0 SP5B Write Quality of Service Register	0x0000000C
0x3005B100	SCB0_HSMDMA_CH1_IB_READ_QOS	SCB0 HSMDMA Channel 1 Read Quality of Service Register	0x00000001
0x3005B104	SCB0_HSMDMA_CH1_IB_WRITE_QOS	SCB0 HSMDMA Channel 1 Write Quality of Service Register	0x00000001
0x3005C100	SCB0_UART2_TX_READ_QOS	SCB0 UART2 TX Read Quality of Service Register	0x0000000C
0x3005C104	SCB0_UART2_TX_WRITE_QOS	SCB0 UART2 TX Write Quality of Service Register	0x0000000C
0x3005D100	SCB0_SPI0TX_READ_QOS	SCB0 SPI0 TX Read Quality of Service Register	0x0000000C
0x3005D104	SCB0_SPI0TX_WRITE_QOS	SCB0 SPI0 TX Write Quality of Service Register	0x0000000C
0x3005E100	SCB0_SPI0RX_READ_QOS	SCB0 SPI0 RX Read Quality of Service Register	0x0000000C

Table A-116: ADSP-SC59x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3005E104	SCB0_SPI0RX_WRITE_QOS	SCB0 SPI0 RX Write Quality of Service Register	0x0000000C
0x3005F100	SCB0_SPI1TX_READ_QOS	SCB0 SPI1 TX Read Quality of Service Register	0x0000000C
0x3005F104	SCB0_SPI1TX_WRITE_QOS	SCB0 SPI1 TX Write Quality of Service Register	0x0000000C
0x30060100	SCB0_SPI1RX_READ_QOS	SCB0 SPI1 RX Read Quality of Service Register	0x0000000C
0x30060104	SCB0_SPI1RX_WRITE_QOS	SCB0 SPI1 RX Write Quality of Service Register	0x0000000C
0x30061100	SCB0_SPI2TX_READ_QOS	SCB0 SPI2 TX Read Quality of Service Register	0x0000000C
0x30061104	SCB0_SPI2TX_WRITE_QOS	SCB0 SPI2 TX Write Quality of Service Register	0x0000000C
0x30062100	SCB0_SPI2RX_READ_QOS	SCB0 SPI2 RX Read Quality of Service Register	0x0000000C
0x30062104	SCB0_SPI2RX_WRITE_QOS	SCB0 SPI2 RX Write Quality of Service Register	0x0000000C
0x30063100	SCB0_SP6A_READ_QOS	SCB0 SP6A Read Quality of Service Register	0x0000000C
0x30063104	SCB0_SP6A_WRITE_QOS	SCB0 SP6A Write Quality of Service Register	0x0000000C
0x30064100	SCB0_SP6B_READ_QOS	SCB0 SP6B Read Quality of Service Register	0x0000000C
0x30064104	SCB0_SP6B_WRITE_QOS	SCB0 SP6B Write Quality of Service Register	0x0000000C
0x30065100	SCB0_LP0_READ_QOS	SCB0 LP0 Read Quality of Service Register	0x00000007
0x30065104	SCB0_LP0_WRITE_QOS	SCB0 LP0 Write Quality of Service Register	0x00000007
0x30066100	SCB0_SP7A_READ_QOS	SCB0 SP7A Read Quality of Service Register	0x0000000C
0x30066104	SCB0_SP7A_WRITE_QOS	SCB0 SP7A Write Quality of Service Register	0x0000000C
0x30067100	SCB0_SP7B_READ_QOS	SCB0 SP7B Read Quality of Service Register	0x0000000C
0x30067104	SCB0_SP7B_WRITE_QOS	SCB0 SP7B Write Quality of Service Register	0x0000000C
0x30068100	SCB0_SH0_MMR_IB_READ_QOS	SCB0 SH0 MMR Read Quality of Service Register	0x00000007
0x30068104	SCB0_SH0_MMR_IB_WRITE_QOS	SCB0 SH0 MMR Write Quality of Service Register	0x00000007
0x30069100	SCB0_SPI3RX_READ_QOS	SCB0 SPI3 RX Read Quality of Service Register	0x0000000C
0x30069104	SCB0_SPI3RX_WRITE_QOS	SCB0 SPI3 RX Write Quality of Service Register	0x0000000C
0x3006A100	SCB0_UART1_TX_READ_QOS	SCB0 UART1 TX Read Quality of Service Register	0x0000000C

Table A-116: ADSP-SC59x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3006A104	SCB0_UART1_TX_WRITE_QOS	SCB0 UART1 TX Write Quality of Service Register	0x0000000C
0x3006B100	SCB0_UART1_RX_READ_QOS	SCB0 UART1 RX Read Quality of Service Register	0x0000000C
0x3006B104	SCB0_UART1_RX_WRITE_QOS	SCB0 UART1 RX Write Quality of Service Register	0x0000000C
0x3006C100	SCB0_SPI3TX_READ_QOS	SCB0 SPI3 TX Read Quality of Service Register	0x0000000C
0x3006C104	SCB0_SPI3TX_WRITE_QOS	SCB0 SPI3 TX Write Quality of Service Register	0x0000000C
0x3006D100	SCB0_LP1_READ_QOS	SCB0 LP1 Read Quality of Service Register	0x00000007
0x3006D104	SCB0_LP1_WRITE_QOS	SCB0 LP1 Write Quality of Service Register	0x00000007
0x3006E100	SCB0_CRC2_CH1_IB_READ_QOS	SCB0 CRC2 Channel 1 Read Quality of Service Register	0x00000001
0x3006E104	SCB0_CRC2_CH1_IB_WRITE_QOS	SCB0 CRC2 Channel 1 Write Quality of Service Register	0x00000001
0x3006F100	SCB0_CRC2_CH0_IB_READ_QOS	SCB0 CRC2 Channel 0 Read Quality of Service Register	0x00000001
0x3006F104	SCB0_CRC2_CH0_IB_WRITE_QOS	SCB0 CRC2 Channel 0 Write Quality of Service Register	0x00000001
0x30070100	SCB0_CRC3_CH1_IB_READ_QOS	SCB0 CRC3 Channel 1 Read Quality of Service Register	0x00000001
0x30070104	SCB0_CRC3_CH1_IB_WRITE_QOS	SCB0 CRC3 Channel 1 Write Quality of Service Register	0x00000001
0x30071100	SCB0_CRYPTO_IB_READ_QOS	SCB0 CRYPTO Read Quality of Service Register	0x00000001
0x30071104	SCB0_CRYPTO_IB_WRITE_QOS	SCB0 CRYPTO Write Quality of Service Register	0x00000001
0x30072100	SCB0_CRC3_CH0_IB_READ_QOS	SCB0 CRC3 Channel 0 Read Quality of Service Register	0x00000001
0x30072104	SCB0_CRC3_CH0_IB_WRITE_QOS	SCB0 CRC3 Channel 0 Write Quality of Service Register	0x00000001
0x30073100	SCB0_MSMDMA1_CH1_IB_READ_QOS	SCB0 MSMDMA1 Channel 1 Read Quality of Service Register	0x00000001
0x30073104	SCB0_MSMDMA1_CH1_IB_WRITE_QOS	SCB0 MSMDMA1 Channel 1 Write Quality of Service Register	0x00000001

Table A-116: ADSP-SC59x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30074100	SCB0_SH0_FIR_CH0_READ_QOS	SCB0 SH0 FIR Channel 0Read Quality of Service Register	0x00000007
0x30074104	SCB0_SH0_FIR_CH0_WRITE_QOS	SCB0 SH0 FIR Channel 0Write Quality of Service Register	0x00000007
0x30075100	SCB0_SH0_FIR_CH1_READ_QOS	SCB0 SH0 FIR Channel 1Read Quality of Service Register	0x00000007
0x30075104	SCB0_SH0_FIR_CH1_WRITE_QOS	SCB0 SH0 FIR Channel 1Write Quality of Service Register	0x00000007
0x30076100	SCB0_MSMDMA1_CH0_IB_READ_QOS	SCB0 MSMDMA1 Channel 0 Read Quality of Service Register	0x00000001
0x30076104	SCB0_MSMDMA1_CH0_IB_WRITE_QOS	SCB0 MSMDMA1 Channel 0 Write Quality of Service Register	0x00000001
0x30077100	SCB0_HSMDMA1_CH1_IB_READ_QOS	SCB0 HSMDMA1 Channel 1 Read Quality of Service Register	0x00000001
0x30077104	SCB0_HSMDMA1_CH1_IB_WRITE_QOS	SCB0 HSMDMA1 Channel 1 Write Quality of Service Register	0x00000001
0x30078100	SCB0_DLDMA0_CH0_IB_READ_QOS	SCB0 DLDMA0 Channel 0 Read Quality of Service Register	0x00000001
0x30078104	SCB0_DLDMA0_CH0_IB_WRITE_QOS	SCB0 DLDMA0 Channel 0 Write Quality of Service Register	0x00000001
0x30079100	SCB0_DLDMA0_CH1_IB_READ_QOS	SCB0 DLDMA0 Channel 1 Read Quality of Service Register	0x00000001
0x30079104	SCB0_DLDMA0_CH1_IB_WRITE_QOS	SCB0 DLDMA0 Channel 1 Write Quality of Service Register	0x00000001
0x3007A100	SCB0_DLDMA1_CH0_IB_READ_QOS	SCB0 DLDMA1 Channel 0 Read Quality of Service Register	0x00000001
0x3007A104	SCB0_DLDMA1_CH0_IB_WRITE_QOS	SCB0 DLDMA1 Channel 0 Write Quality of Service Register	0x00000001
0x3007B100	SCB0_DLDMA1_CH1_IB_READ_QOS	SCB0 DLDMA1 Channel 1 Read Quality of Service Register	0x00000001
0x3007B104	SCB0_DLDMA1_CH1_IB_WRITE_QOS	SCB0 DLDMA1 Channel 1 Write Quality of Service Register	0x00000001
0x3007C100	SCB0_MSMDMA_CH0_IB_READ_QOS	SCB0 MSMDMA Channel 0 Read Quality of Service Register	0x00000001
0x3007C104	SCB0_MSMDMA_CH0_IB_WRITE_QOS	SCB0 MSMDMA Channel 0 Write Quality of Service Register	0x00000001

Table A-116: ADSP-SC59x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3007D100	SCB0_MSMDMA_CH1_IB_READ_QOS	SCB0 MSMDMA Channel 1 Read Quality of Service Register	0x00000001
0x3007D104	SCB0_MSMDMA_CH1_IB_WRITE_QOS	SCB0 MSMDMA Channel 1 Write Quality of Service Register	0x00000001
0x3007E100	SCB0_HSMDMA1_CH0_IB_READ_QOS	SCB0 HSMDMA1 Channel 0 Read Quality of Service Register	0x00000001
0x3007E104	SCB0_HSMDMA1_CH0_IB_WRITE_QOS	SCB0 HSMDMA1 Channel 0 Write Quality of Service Register	0x00000001
0x3007F100	SCB0_PPI_F0_READ_QOS	SCB0 PPI F0 Read Quality of Service Register	0x0000000C
0x3007F104	SCB0_PPI_F0_WRITE_QOS	SCB0 PPI F0 Write Quality of Service Register	0x0000000C
0x30080100	SCB0_PPI_F1_READ_QOS	SCB0 PPI F1 Read Quality of Service Register	0x0000000C
0x30080104	SCB0_PPI_F1_WRITE_QOS	SCB0 PPI F1 Write Quality of Service Register	0x0000000C
0x30081100	SCB0_USB0_IB_READ_QOS	SCB0 USB0 Read Quality of Service Register	0x00000007
0x30081104	SCB0_USB0_IB_WRITE_QOS	SCB0 USB0 Write Quality of Service Register	0x00000007
0x30082100	SCB0_SH1_IPORT_READ_QOS	SCB0 SH1 IPORT Read Quality of Service Register	0x00000007
0x30082104	SCB0_SH1_IPORT_WRITE_QOS	SCB0 SH1 IPORT Write Quality of Service Register	0x00000007
0x30083100	SCB0_SH1_DPORT_READ_QOS	SCB0 SH1 DPORT Read Quality of Service Register	0x00000007
0x30083104	SCB0_SH1_DPORT_WRITE_QOS	SCB0 SH1 DPORT Write Quality of Service Register	0x00000007
0x30084100	SCB0_SH1_MMR_IB_READ_QOS	SCB0 SH1 MMR Read Quality of Service Register	0x00000007
0x30084104	SCB0_SH1_MMR_IB_WRITE_QOS	SCB0 SH1 MMR Write Quality of Service Register	0x00000007
0x30085100	SCB0_PL310_M0_IB_READ_QOS	SCB0 PL310 M0 Read Quality of Service Register	0x00000007
0x30085104	SCB0_PL310_M0_IB_WRITE_QOS	SCB0 PL310 M0 Write Quality of Service Register	0x00000007
0x30086100	SCB0_PL310_MMR_IB_READ_QOS	SCB0 PL310 MMR Read Quality of Service Register	0x00000007

Table A-116: ADSP-SC59x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30086104	SCB0_PL310_MMR_IB_WRITE_QOS	SCB0 PL310 MMR Write Quality of Service Register	0x00000007
0x30087100	SCB0_GIGE_IB_READ_QOS	SCB0 GIGE Read Quality of Service Register	0x00000007
0x30087104	SCB0_GIGE_IB_WRITE_QOS	SCB0 GIGE Write Quality of Service Register	0x00000007
0x30088100	SCB0_DBG_IB_READ_QOS	SCB0 DBG Read Quality of Service Register	0x00000001
0x30088104	SCB0_DBG_IB_WRITE_QOS	SCB0 DBG Write Quality of Service Register	0x00000001
0x30089100	SCB0_ETR_IB_READ_QOS	SCB0 ETR Read Quality of Service Register	0x00000001
0x30089104	SCB0_ETR_IB_WRITE_QOS	SCB0 ETR Write Quality of Service Register	0x00000001
0x3008A100	SCB0_EMAC_IB_READ_QOS	SCB0 EMAC Read Quality of Service Register	0x00000007
0x3008A104	SCB0_EMAC_IB_WRITE_QOS	SCB0 EMAC Write Quality of Service Register	0x00000007
0x3008B100	SCB0_SH1_FIR_CH1_READ_QOS	SCB0 SH1 FIR Channel 1 Read Quality of Service Register	0x00000007
0x3008B104	SCB0_SH1_FIR_CH1_WRITE_QOS	SCB0 SH1 FIR Channel 1 Write Quality of Service Register	0x00000007
0x3008C100	SCB0_SH1_FIR_CH0_READ_QOS	SCB0 SH1 FIR Channel 0 Read Quality of Service Register	0x00000007
0x3008C104	SCB0_SH1_FIR_CH0_WRITE_QOS	SCB0 SH1 FIR Channel 0 Write Quality of Service Register	0x00000007
0x3008D100	SCB0_SH0_IIR_CH0_READ_QOS	SCB0 SH0 IIR Channel 0 Read Quality of Service Register	0x00000007
0x3008D104	SCB0_SH0_IIR_CH0_WRITE_QOS	SCB0 SH0 IIR Channel 0 Write Quality of Service Register	0x00000007
0x3008E100	SCB0_SH0_IIR_CH1_READ_QOS	SCB0 SH0 IIR Channel 1 Read Quality of Service Register	0x00000007
0x3008E104	SCB0_SH0_IIR_CH1_WRITE_QOS	SCB0 SH0 IIR Channel 1 Write Quality of Service Register	0x00000007
0x3008F100	SCB0_SH1_IIR_CH0_READ_QOS	SCB0 SH1 IIR Channel 0 Read Quality of Service Register	0x00000007
0x3008F104	SCB0_SH1_IIR_CH0_WRITE_QOS	SCB0 SH1 IIR Channel 0 Write Quality of Service Register	0x00000007

Table A-116: ADSP-SC59x SCB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30090100	SCB0_SH1_IIR_CH1_READ_QOS	SCB0 SH1 IIR Channel 1 Read Quality of Service Register	0x00000007
0x30090104	SCB0_SH1_IIR_CH1_WRITE_QOS	SCB0 SH1 IIR Channel 1 Write Quality of Service Register	0x00000007
0x30091100	SCB0_SDIO0_IB_READ_QOS	SCB0 SDIO0 Read Quality of Service Register	0x00000007
0x30091104	SCB0_SDIO0_IB_WRITE_QOS	SCB0 SDIO0 Write Quality of Service Register	0x00000007
0x30092100	SCB0_PL310_M1_IB_READ_QOS	SCB0 PL310 M1 Read Quality of Service Register	0x00000007
0x30092104	SCB0_PL310_M1_IB_WRITE_QOS	SCB0 PL310 M1 Write Quality of Service Register	0x00000007
0x300C2020	SCB0_SCB6_IB_SYNC_MODE	SCB0 SCB6 Synchronization Mode Register	0x00000004

Table A-117: ADSP-SC59x SCB4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30342100	SCB4_IIR2_CH0_IB_READ_QOS	SCB4 IIR2 Channel 0 Read Quality of Service Register	0x0000000C
0x30342104	SCB4_IIR2_CH0_IB_WRITE_QOS	SCB4 IIR2 Channel 0 Write Quality of Service Register	0x0000000C
0x30343100	SCB4_SHARC_DPORT_READ_QOS	SCB4 DPORT Read Quality of Service Register	0x0000000C
0x30343104	SCB4_SHARC_DPORT_WRITE_QOS	SCB4 DPORT Write Quality of Service Register	0x0000000C
0x30344100	SCB4_FABRIC_S2PORT_IB_READ_QOS	SCB4 Fabric S2PORT Read Quality of Service Register	0x0000000C
0x30344104	SCB4_FABRIC_S2PORT_IB_WRITE_QOS	SCB4 Fabric S2PORT Write Quality of Service Register	0x0000000C
0x30345100	SCB4_FIR_CH0_IB_READ_QOS	SCB4 FIR Channel 0 Read Quality of Service Register	0x0000000C
0x30345104	SCB4_FIR_CH0_IB_WRITE_QOS	SCB4 FIR Channel 0 Write Quality of Service Register	0x0000000C

Table A-117: ADSP-SC59x SCB4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x30346100	SCB4_FIR_CH1_IB_READ_QOS	SCB4 FIR Channel 1 Read Quality of Service Register	0x0000000C
0x30346104	SCB4_FIR_CH1_IB_WRITE_QOS	SCB4 FIR Channel 1 Write Quality of Service Register	0x0000000C
0x30347100	SCB4_IIR0_CH0_IB_READ_QOS	SCB4 IIR0 Channel 0 Read Quality of Service Register	0x0000000C
0x30347104	SCB4_IIR0_CH0_IB_WRITE_QOS	SCB4 IIR0 Channel 0 Write Quality of Service Register	0x0000000C
0x30348100	SCB4_IIR0_CH1_IB_READ_QOS	SCB4 IIR0 Channel 1 Read Quality of Service Register	0x0000000C
0x30348104	SCB4_IIR0_CH1_IB_WRITE_QOS	SCB4 IIR0 Channel 1 Write Quality of Service Register	0x0000000C
0x30349100	SCB4_IIR2_CH1_IB_READ_QOS	SCB4 IIR2 Channel 1 Read Quality of Service Register	0x0000000C
0x30349104	SCB4_IIR2_CH1_IB_WRITE_QOS	SCB4 IIR2 Channel 1 Write Quality of Service Register	0x0000000C
0x3034A100	SCB4_FABRIC_S1PORT_IB_READ_QOS	SCB4 Fabric S1PORT Read Quality of Service Register	0x0000000C
0x3034A104	SCB4_FABRIC_S1PORT_IB_WRITE_QOS	SCB4 Fabric S1PORT Write Quality of Service Register	0x0000000C
0x3034B100	SCB4_IIR3_CH0_IB_READ_QOS	SCB4 IIR3 Channel 0 Read Quality of Service Register	0x0000000C
0x3034B104	SCB4_IIR3_CH0_IB_WRITE_QOS	SCB4 IIR3 Channel 0 Write Quality of Service Register	0x0000000C
0x3034C100	SCB4_IIR3_CH1_IB_READ_QOS	SCB4 IIR3 Channel 1 Read Quality of Service Register	0x0000000C
0x3034C104	SCB4_IIR3_CH1_IB_WRITE_QOS	SCB4 IIR3 Channel 1 Write Quality of Service Register	0x0000000C
0x3034D100	SCB4_FABRIC_ACC_MMR_IB_READ_QOS	SCB4 Fabric Acc MMR Read Quality of Service Register	0x0000000C
0x3034D104	SCB4_FABRIC_ACC_MMR_IB_WRITE_QOS	SCB4 Fabric Acc MMR Write Quality of Service Register	0x0000000C
0x3034E100	SCB4_IIR1_CH0_IB_READ_QOS	SCB4 IIR1 Channel 0 Read Quality of Service Register	0x0000000C

Table A-117: ADSP-SC59x SCB4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3034E104	SCB4_IIR1_CH0_IB_WRITE_QOS	SCB4 IIR1 Channel 0 Write Quality of Service Register	0x0000000C
0x3034F100	SCB4_IIR1_CH1_IB_READ_QOS	SCB4 IIR1 Channel 1 Read Quality of Service Register	0x0000000C
0x3034F104	SCB4_IIR1_CH1_IB_WRITE_QOS	SCB4 IIR1 Channel 1 Write Quality of Service Register	0x0000000C

Table A-118: ADSP-SC59x SCB3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x30102020	SCB3_APB_CLKO8_IB_SYNC_MODE	SCB3 DDR MMRG Fabric (CLK08) Synchronization Mode Register	0x00000004
0x30105020	SCB3_APB_CLKO3_IB_SYNC_MODE	SCB3 DDR MMRG Fabric (CLK03) Synchronization Mode Register	0x00000004
0x3010B020	SCB3_APB_CLKO4_IB_SYNC_MODE	SCB3 DDR MMRG Fabric (CLK04) Synchronization Mode Register	0x00000004

Table A-119: ADSP-SC59x MISCREG MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A9000	MISCREG_SH0_PFB_RANGE_SELECT	MISCREG SH0 Prefetch Range Selection Register	0xFFFFFFFF
0x310A9010	MISCREG_CAN_SYSCTL	MISCREG CANFD Low Power Selection Mode	0x00000000
0x310A9014	MISCREG_SHARC_BRIDGE_REMAP	MISCREG SPORT Direct Interrupt Enable for SH0 and SH1	0x00000000
0x310A9028	MISCREG_ECO_REG9	MISCREG Error Register	0x00000000
0x310A902C	MISCREG_PLL2_CONTROL	MISCREG PLL2 Control Register	0x000613F0
0x310A9044	MISCREG_SH1_PFB_RANGE_SELECT	MISCREG SH1 Prefetch Range Selection Register	0xFFFFFFFF

Table A-120: ADSP-SC59x MLB0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109D000	MLB0_CTL0	MLB0 MediaLB Control 0 Register	0x00000000
0x3109D008	MLB0_PCTL0	MLB0 MediaLB 6-pin Control 0 Register	0x00000000
0x3109D00C	MLB0_MS0	MLB0 Channel Status 0 Register	0x00000000
0x3109D014	MLB0_MS1	MLB0 Channel Status 1 Register	0x00000000
0x3109D020	MLB0_MSS	MLB0 System Status Register	0x00000000
0x3109D024	MLB0_MSD	MLB0 System Data Register	0x00000000
0x3109D02C	MLB0_MIEN	MLB0 Interrupt Enable Register	0x00000000
0x3109D034	MLB0_GCTL	MLB0 MLB Global Control Register	0x00000000
0x3109D03C	MLB0_CTL1	MLB0 Control 1 Register	0x00000000
0x3109D080	MLB0_HCTL	MLB0 HBI Control Register	0x00000000
0x3109D088	MLB0_HCMR0	MLB0 HBI Channel Mask 0 Register	0x00000000
0x3109D08C	MLB0_HCMR1	MLB0 HBI Channel Mask 1 Register	0x00000000
0x3109D090	MLB0_HCER0	MLB0 HBI Channel Error 0 Register	0x00000000
0x3109D094	MLB0_HCER1	MLB0 HBI Channel Error 1 Register	0x00000000
0x3109D098	MLB0_HCBR0	MLB0 HBI Channel Busy 0 Register	0x00000000
0x3109D09C	MLB0_HCBR1	MLB0 HBI Channel Busy 1 Register	0x00000000
0x3109D0C0	MLB0_MDAT0	MLB0 Memory Interface Control Data 0 Register	0x00000000
0x3109D0C4	MLB0_MDAT1	MLB0 Memory Interface Control Data 1 Register	0x00000000
0x3109D0C8	MLB0_MDAT2	MLB0 Memory Interface Control Data 2 Register	0x00000000
0x3109D0CC	MLB0_MDAT3	MLB0 Memory Interface Control Data 3 Register	0x00000000
0x3109D0D0	MLB0_MDWE0	MLB0 Memory Interface Control Data Write Enable 0 Register	0x00000000
0x3109D0D4	MLB0_MDWE1	MLB0 Memory Interface Control Data Write Enable 1 Register	0x00000000
0x3109D0D8	MLB0_MDWE2	MLB0 Memory Interface Control Data Write Enable 2 Register	0x00000000
0x3109D0DC	MLB0_MDWE3	MLB0 Memory Interface Control Data Write Enable 3 Register	0x00000000
0x3109D0E0	MLB0_MCTL	MLB0 Memory Interface Control Register	0x00000000
0x3109D0E4	MLB0_MADR	MLB0 Memory Interface Address Register	0x00000000
0x3109D3C0	MLB0_ACTL	MLB0 Bus Control Register	0x00000000
0x3109D3D0	MLB0_ACSR0	MLB0 Peripheral Channel Status 0 Register	0x00000000

Table A-120: ADSP-SC59x MLB0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109D3D4	MLB0_ACSR1	MLB0 Peripheral Channel Status 1 Register	0x00000000
0x3109D3D8	MLB0_ACMR0	MLB0 Peripheral Channel Mask 0 Register	0x00000000
0x3109D3DC	MLB0_ACMR1	MLB0 Peripheral Channel Mask 1 Register	0x00000000

Table A-121: ADSP-SC59x EMSI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C7000	EMSI0_SDMA_ADDR	EMSI0 SDMA System Address Register	0x00000000
0x310C7004	EMSI0_BLKSZ	EMSI0 Block Size Register	0x00000000
0x310C7006	EMSI0_BLKCNT	EMSI0 16-bit Block Count Register	0x00000000
0x310C7008	EMSI0_ARG	EMSI0 Argument Register	0x00000000
0x310C700C	EMSI0_TRNSFRMODE	EMSI0 Transfer Mode Register	0x00000000
0x310C700E	EMSI0_CMD	EMSI0 Command Register	0x00000000
0x310C7010	EMSI0_RESP0	EMSI0 Response Register 0	0x00000000
0x310C7014	EMSI0_RESP1	EMSI0 Response Register 1	0x00000000
0x310C7018	EMSI0_RESP2	EMSI0 Response Register 2	0x00000000
0x310C701C	EMSI0_RESP3	EMSI0 Response Register 3	0x00000000
0x310C7024	EMSI0_PSTATE	EMSI0 Present State Register	0x00000000
0x310C7028	EMSI0_CTL1	EMSI0 EMSI Control 1 Register	0x00000000
0x310C702B	EMSI0_WU_CTL	EMSI0 Wakeup Control Register	0x00000000
0x310C702C	EMSI0_CLK_CTL	EMSI0 Clock Control Register	0x00000000
0x310C702E	EMSI0_TO_CTL	EMSI0 Timeout Control Register	0x00000000
0x310C702F	EMSI0_SWRST	EMSI0 Software Reset Register	0x00000000
0x310C7030	EMSI0_ISTAT	EMSI0 Interrupt Status Register	0x00000000
0x310C7032	EMSI0_ERR_STAT	EMSI0 Error Interrupt Status Register	0x00000000
0x310C7034	EMSI0_ISTAT_EN	EMSI0 Interrupt Status Enable Register	0x00000000
0x310C7036	EMSI0_ERR_STAT_EN	EMSI0 Error Interrupt Status Enable Register	0x00000000
0x310C7038	EMSI0_ISTAT_INTEN	EMSI0 Interrupt Signal Enable Register	0x00000000
0x310C703A	EMSI0_ERR_STAT_INTEN	EMSI0 Error Interrupt Signal Enable Register	0x00000000
0x310C703C	EMSI0_AUTOCMD_STAT	EMSI0 Auto CMD Status Register	0x00000000
0x310C703E	EMSI0_CTL2	EMSI0 EMSI Control 2 Register	0x00000000

Table A-121: ADSP-SC59x EMSI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C7040	EMSI0_CAP1	EMSI0 Capabilities 1 Register - 0 to 31	0x216D328A
0x310C7044	EMSI0_CAP2	EMSI0 Capabilities Register - 32 to 63	0x08000000
0x310C7050	EMSI0_FRC_AUTOCMDSTAT	EMSI0 Force Event Register for Auto CMD Error Status Register	0x00000000
0x310C7052	EMSI0_FRC_ERRSTAT	EMSI0 Force Event Register for Error Interrupt Status	0x00000000
0x310C7054	EMSI0_ADMA_ERR_STAT	EMSI0 ADMA Error Status Register	0x00000000
0x310C7058	EMSI0_ADMA_ADDR_LO	EMSI0 ADMA System Address Register - Low	0x00000000
0x310C7060	EMSI0_PRESET_INIT	EMSI0 Preset Value for Initialization	0x00000040
0x310C7062	EMSI0_PRESET_DS	EMSI0 Preset Value for Default Speed	0x00000001
0x310C7066	EMSI0_PRESET_LEGACY	EMSI0 Preset Value for Legacy Mode	0x00000000
0x310C7068	EMSI0_PRESET_HSSDR	EMSI0 Preset Value for HSSDR	0x00000000
0x310C706E	EMSI0_PRESET_HSDDR	EMSI0 Preset Value for HSDDR	0x00000000
0x310C7078	EMSI0_ADMA_DESADDR_LO	EMSI0 ADMA3 Integrated Descriptor Address Register - Low	0x00000000
0x310C70E8	EMSI0_BASEADDR0	EMSI0 Base Address 0 Register	0x00000500
0x310C70EA	EMSI0_BASEADDR1	EMSI0 Base Address 1 Register	0x00000180
0x310C70FE	EMSI0_HOST_CNTRL_VERS	EMSI0 SD Host Controller Specification Version	0x00000005
0x310C7180	EMSI0_CQVER	EMSI0 Command Queuing Version Register	0x00000510
0x310C7184	EMSI0_CQ_CAP	EMSI0 Command Queuing Capabilities Register	0x00004001
0x310C7188	EMSI0_CQ_CFG	EMSI0 Command Queuing Configuration Register	0x00000000
0x310C718C	EMSI0_CQ_CTL	EMSI0 Command Queuing Control Register	0x00000000
0x310C7190	EMSI0_CQ_ISTAT	EMSI0 Command Queuing Interrupt Status Register	0x00000000
0x310C7194	EMSI0_CQ_ISTAT_EN	EMSI0 Command Queuing Interrupt Status Enable Register	0x00000000
0x310C7198	EMSI0_CQ_ISTAT_INTEN	EMSI0 Command Queuing Interrupt Signal Enable Register	0x00000000
0x310C719C	EMSI0_CQ_IC	EMSI0 Command Queuing Interrupt Coalescing Register	0x00000000
0x310C71A0	EMSI0_CQ_TDL_BADDR	EMSI0 Command Queuing Task Descriptor List Base Address Register	0x00000000
0x310C71A8	EMSI0_CQ_TDBR	EMSI0 Command Queuing DoorBell Register	0x00000000
0x310C71AC	EMSI0_CQ_TCN	EMSI0 Command Queuing Task Completion Notification Register	0x00000000

Table A-121: ADSP-SC59x EMSI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C71B0	EMSI0_CQ_DQSTAT	EMSI0 Device Queue Status Register	0x00000000
0x310C71B4	EMSI0_CQ_DPT	EMSI0 Device Pending Tasks Register	0x00000000
0x310C71B8	EMSI0_CQ_TCLR	EMSI0 Command Queuing Task Clear Register	0x00000000
0x310C71C0	EMSI0_CQ_SSCFG1	EMSI0 CQ Send Status Configuration 1 Register	0x00011000
0x310C71C4	EMSI0_CQ_SSCFG2	EMSI0 CQ Send Status Configuration 2 Register	0x00000000
0x310C71C8	EMSI0_CQ_CRDCT	EMSI0 Command Response for Direct Command Register	0x00000000
0x310C71D0	EMSI0_CQ_RMEM	EMSI0 Command Response Mode Error Mask Register	0xFDF9A080
0x310C71D4	EMSI0_CQ_TERRINFO	EMSI0 CQ Task Error Information Register	0x00000000
0x310C71D8	EMSI0_CQ_CRI	EMSI0 CQ Command Response Index	0x00000000
0x310C71DC	EMSI0_CQ_CRARG	EMSI0 CQ Command Response Argument Register	0x00000000
0x310C7500	EMSI0_VER_ID	EMSI0 EMSI Version	0x3138302A
0x310C7504	EMSI0_VER_TYPE	EMSI0 EMSI Version Type	0x67612A2A
0x310C752C	EMSI0_EMMC_CTL	EMSI0 eMMC Control Register	0x0000000C
0x310C752E	EMSI0_BOOT_CTL	EMSI0 eMMC Boot Control Register	0x00000000

Table A-122: ADSP-SC59x OSPI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31027000	OSPI0_CTL	OSPI0 Octal SPI Control Register	0x80780081
0x31027004	OSPI0_DRICTL	OSPI0 Device Read Instruction Control Register	0x00000003
0x31027008	OSPI0_DWICTL	OSPI0 Device Write Instruction Control Register	0x00000002
0x3102700C	OSPI0_DLY	OSPI0 Device Delay Register	0x00000000
0x31027010	OSPI0_RDC	OSPI0 Read Data Capture Register	0x00000001
0x31027014	OSPI0_DSCTL	OSPI0 Device Size Control Register	0x00101002
0x31027024	OSPI0_REMAPADDR	OSPI0 Remap Address Register	0x00000000
0x31027028	OSPI0_MBCTL	OSPI0 Mode Bit Control Register	0x00000000
0x31027038	OSPI0_WCCTL	OSPI0 Write Completion Control Register	0x00010005
0x3102703C	OSPI0_POLLEXP	OSPI0 Polling Expiration Register	0xFFFFFFFF
0x31027040	OSPI0_ISTAT	OSPI0 Interrupt Status Register	0x00000000
0x31027044	OSPI0_IMSK	OSPI0 Interrupt Mask Register	0x00000000
0x31027050	OSPI0_WRPROT_LWR	OSPI0 Lower Write Protection Register	0x00000000

Table A-122: ADSP-SC59x OSPI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31027054	OSPI0_WRPROT_UP	OSPI0 Upper Write Protection Register	0x00000000
0x31027058	OSPI0_WRPROT_CTL	OSPI0 Write Protection Control Register	0x00000000
0x3102708C	OSPI0_FCMCTL	OSPI0 Flash Command Control Memory Register	0x00000000
0x31027090	OSPI0_FCCTL	OSPI0 Flash Command Control Register	0x00000000
0x31027094	OSPI0_FCA	OSPI0 Flash Command Address Register	0x00000000
0x310270A0	OSPI0_FCRD_LWR	OSPI0 Flash Command Read Data Register (Lower)	0x00000000
0x310270A4	OSPI0_FCRD_UP	OSPI0 Flash Command Read Data Register (Upper)	0x00000000
0x310270A8	OSPI0_FCWD_LWR	OSPI0 Flash Command Write Data Register (Lower)	0x00000000
0x310270AC	OSPI0_FCWD_UP	OSPI0 Flash Command Write Data Register (Upper)	0x00000000
0x310270B0	OSPI0_POLSTAT	OSPI0 Polling Flash Status Register	0x00000000
0x310270B4	OSPI0_PHYCTL	OSPI0 PHY Control Register	0x40000000
0x310270B8	OSPI0_PHYMCTL	OSPI0 PHY DLL Master Control Register	0x00800000
0x310270BC	OSPI0_DLLOB_LWR	OSPI0 DLL Observable Register (Lower)	0x00000000
0x310270C0	OSPI0_DLLOB_UP	OSPI0 DLL Observable Register (Upper)	0x00000000
0x310270E0	OSPI0_OE_LWR	OSPI0 Opcode Extension Register (Lower)	0x13EDFA00
0x310270E4	OSPI0_OE_UP	OSPI0 Opcode Extension Register (Upper)	0x06F90000
0x310270FC	OSPI0_MODID	OSPI0 Module ID Register	0x04000300

Table A-123: ADSP-SC59x OTPC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31011004	OTPC0_STAT	OTPC0 OTP Status Register	0x00000000
0x3101102C	OTPC0_SECU_STATE	OTPC0 OTP Security State Register	0x00000001

Table A-124: ADSP-SC59x PADS0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004604	PADS0_PCFG0	PADS0 Peripheral PAD Configuration0 Register	0x005D0009
0x31004608	PADS0_PCFG1	PADS0 Peripheral Configuration1 Register	0x00000000
0x3100460C	PADS0_PORTA0_DS	PADS0 PORTA 0 to 7 pins DS control	0x00249249
0x31004610	PADS0_PORTA1_DS	PADS0 PORTA 8 - 15 pins DS control	0x00249249
0x31004614	PADS0_PORTB0_DS	PADS0 PORTB 0 to 7 pins DS control	0x00249249

Table A-124: ADSP-SC59x PADS0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31004618	PADS0_PORTB1_DS	PADS0 PORTB 8 - 15 pins DS control	0x00249249
0x3100461C	PADS0_PORTC0_DS	PADS0 PORTC 0 to 7 pins DS control	0x00249249
0x31004620	PADS0_PORTC1_DS	PADS0 PORTC 8 - 15 pins DS control	0x00249249
0x31004624	PADS0_PORTD0_DS	PADS0 PORTD 0 to 7 pins DS control	0x00249249
0x31004628	PADS0_PORTD1_DS	PADS0 PORTD 8 to 15 pins DS control	0x00249249
0x3100462C	PADS0_PORTE0_DS	PADS0 PORTE 0 to 7 pins DS control	0x00249249
0x31004630	PADS0_PORTE1_DS	PADS0 PORTE 8 to 15 pins DS control	0x00249249
0x31004634	PADS0_PORTF0_DS	PADS0 PORTF 0 to 7 pins DS control	0x00249249
0x31004638	PADS0_PORTF1_DS	PADS0 PORTF 8 to 15 pins DS control	0x00249249
0x3100463C	PADS0_PORTG0_DS	PADS0 PORTG 0 to 7 pins DS control	0x00249249
0x31004640	PADS0_PORTG1_DS	PADS0 PORTG 8 to 15 pins DS control	0x00249249
0x31004644	PADS0_PORTH0_DS	PADS0 PORTH 0 to 7 pins DS control	0x00249249
0x31004648	PADS0_PORTH1_DS	PADS0 PORTH 8 to 15 pins DS control	0x00249249
0x3100464C	PADS0_PORTI0_DS	PADS0 PORTI 0 to 7 pins DS control	0x00249249
0x31004650	PADS0_NONPORTS_DS	PADS0 Non-GPIO Drive Strength Register	0x0045244A
0x31004678	PADS0_DAI0_0_DS	PADS0 DAI0 1 to 10 pins DS control	0x09249249
0x3100467C	PADS0_DAI0_1_DS	PADS0 DAI0 11 to 20 pins DS control	0x09249249
0x31004680	PADS0_DAI1_0_DS	PADS0 DAI1 1 to 10 pins DS control	0x09249249
0x31004684	PADS0_DAI1_1_DS	PADS0 DAI1 11 to 20 pins DS control	0x09249249
0x31004690	PADS0_DAI0_IE	PADS0 DAI0 Port Input Enable Control Register	0x00000000
0x31004694	PADS0_DAI1_IE	PADS0 DAI1 Port Input Enable Control Register	0x00000000
0x31004698	PADS0_PORT[n]_PUE	PADS0 PORT[n] Pull-Up Enable	0x00000000
0x3100469C	PADS0_PORT[n]_PUE	PADS0 PORT[n] Pull-Up Enable	0x00000000
0x310046A0	PADS0_PORT[n]_PUE	PADS0 PORT[n] Pull-Up Enable	0x00000000
0x310046A4	PADS0_PORT[n]_PUE	PADS0 PORT[n] Pull-Up Enable	0x00000000
0x310046A8	PADS0_PORT[n]_PUE	PADS0 PORT[n] Pull-Up Enable	0x00000000
0x310046AC	PADS0_PORT[n]_PUE	PADS0 PORT[n] Pull-Up Enable	0x00000000
0x310046B0	PADS0_PORT[n]_PUE	PADS0 PORT[n] Pull-Up Enable	0x00000000
0x310046B4	PADS0_PORT[n]_PUE	PADS0 PORT[n] Pull-Up Enable	0x00000000
0x310046B8	PADS0_PORT[n]_PUE	PADS0 PORT[n] Pull-Up Enable	0x00000000
0x310046BC	PADS0_DAI[n]_PUE	PADS0 DAI[n] Pull-Up Enable	0x00000000

Table A-124: ADSP-SC59x PADS0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310046C0	PADS0_DAI[n]_PUE	PADS0 DAI[n] Pull-Up Enable	0x00000000
0x310046C4	PADS0_PORT[n]_PDE	PADS0 PORT[n] Pull-Down Enable	0x00000000
0x310046C8	PADS0_PORT[n]_PDE	PADS0 PORT[n] Pull-Down Enable	0x00000000
0x310046CC	PADS0_PORT[n]_PDE	PADS0 PORT[n] Pull-Down Enable	0x00000000
0x310046D0	PADS0_PORT[n]_PDE	PADS0 PORT[n] Pull-Down Enable	0x00000000
0x310046D4	PADS0_PORT[n]_PDE	PADS0 PORT[n] Pull-Down Enable	0x00000000
0x310046D8	PADS0_PORT[n]_PDE	PADS0 PORT[n] Pull-Down Enable	0x00000000
0x310046DC	PADS0_PORT[n]_PDE	PADS0 PORT[n] Pull-Down Enable	0x00000000
0x310046E0	PADS0_PORT[n]_PDE	PADS0 PORT[n] Pull-Down Enable	0x00000000
0x310046E4	PADS0_PORT[n]_PDE	PADS0 PORT[n] Pull-Down Enable	0x00000000
0x310046FC	PADS0_DAI[n]_PDE	PADS0 DAI[n] Pull-Down Enable	0x00000000
0x31004700	PADS0_DAI[n]_PDE	PADS0 DAI[n] Pull-Down Enable	0x00000000

Table A-125: ADSP-SC59x PCG0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9300	PCG0_CTLA0	PCG0 Precision Clock A Control 0 Register	0x00000000
0x310C9304	PCG0_CTLA1	PCG0 Precision Clock A Control 1 Register	0x00000000
0x310C9308	PCG0_CTLB0	PCG0 Precision Clock B Control 0 Register	0x00000000
0x310C930C	PCG0_CTLB1	PCG0 Precision Clock B Control 1 Register	0x00000000
0x310C9310	PCG0_PW1	PCG0 Precision Clock Pulse Width Control 1 Register	0x00000000
0x310C9314	PCG0_SYNC1	PCG0 Precision Clock Frame Sync Synchronization 1 Register	0x00000000
0x310C9318	PCG0_CTLE0	PCG0 Precision Clock E Control 0 Register	0x00000000
0x310C931C	PCG0_CTLE1	PCG0 Precision Clock E Control 1 Register	0x00000000
0x310C9320	PCG0_CTLF0	PCG0 Precision Clock F Control 0 Register	0x00000000
0x310C9324	PCG0_CTLF1	PCG0 Precision Clock F Control 1 Register	0x00000000
0x310C9328	PCG0_PW3	PCG0 Precision Clock Pulse Width Control 3 Register	0x00000000
0x310C932C	PCG0_SYNC3	PCG0 Precision Clock Frame Sync Synchronization 3 Register	0x00000000
0x310CA300	PCG0_CTLC0	PCG0 Precision Clock C Control 0 Register	0x00000000
0x310CA304	PCG0_CTLC1	PCG0 Precision Clock C Control 1 Register	0x00000000

Table A-125: ADSP-SC59x PCG0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA308	PCG0_CTLD0	PCG0 Precision Clock D Control 0 Register	0x00000000
0x310CA30C	PCG0_CTLD1	PCG0 Precision Clock D Control 1 Register	0x00000000
0x310CA310	PCG0_PW2	PCG0 Precision Clock Pulse Width Control 2 Register	0x00000000
0x310CA314	PCG0_SYNC2	PCG0 Precision Clock Frame Sync Synchronization 2 Register	0x00000000
0x310CA318	PCG0_CTLG0	PCG0 Precision Clock G Control 0 Register	0x00000000
0x310CA31C	PCG0_CTLG1	PCG0 Precision Clock G Control 1 Register	0x00000000
0x310CA320	PCG0_CTLH0	PCG0 Precision Clock H Control 0 Register	0x00000000
0x310CA324	PCG0_CTLH1	PCG0 Precision Clock H Control 1 Register	0x00000000
0x310CA328	PCG0_PW4	PCG0 Precision Clock Pulse Width Control 4 Register	0x00000000
0x310CA32C	PCG0_SYNC4	PCG0 Precision Clock Frame Sync Synchronization 4 Register	0x00000000

Table A-126: ADSP-SC59x PDM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C90A4	PDM0_CTL0	PDM0 PDM Control Register	0x00000013
0x310C90A8	PDM0_HPF_CTL	PDM0 High Pass Filter Control Register	0x000000D0
0x310C90AC	PDM0_SP_CTL0	PDM0 Serial Port Control0 Register	0x00000041
0x310C90B0	PDM0_SP_CTL1	PDM0 Serial Port Control1 Register	0x31211101
0x310C90B8	PDM0_RESET	PDM0 Software Reset Register	0x00000000

Table A-127: ADSP-SC59x PDM1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA0A4	PDM1_CTL0	PDM1 PDM Control Register	0x00000013
0x310CA0A8	PDM1_HPF_CTL	PDM1 High Pass Filter Control Register	0x000000D0
0x310CA0AC	PDM1_SP_CTL0	PDM1 Serial Port Control0 Register	0x00000041
0x310CA0B0	PDM1_SP_CTL1	PDM1 Serial Port Control1 Register	0x31211101
0x310CA0B8	PDM1_RESET	PDM1 Software Reset Register	0x00000000

Table A-128: ADSP-SC59x PE0_CTI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31124000	PE0_CTI0_CTICONTROL	PE0_CTI0 CTI Control register	0x00000000
0x31124010	PE0_CTI0_CTIINTACK	PE0_CTI0 CTI Output Trigger Acknowledge register	0x00000000
0x31124014	PE0_CTI0_CTIAPPSET	PE0_CTI0 CTI Application Trigger Set register	0x00000000
0x31124018	PE0_CTI0_CTIAPPCLEAR	PE0_CTI0 CTI Application Trigger Clear register	0x00000000
0x3112401C	PE0_CTI0_CTIAPPPULSE	PE0_CTI0 CTI Application Pulse register	0x00000000
0x31124020	PE0_CTI0_CTIINEN0	PE0_CTI0 CTI Input Trigger to Output Channel Enable registers	0x00000000
0x31124024	PE0_CTI0_CTIINEN1	PE0_CTI0 CTI Input Trigger to Output Channel Enable registers	0x00000000
0x31124028	PE0_CTI0_CTIINEN2	PE0_CTI0 CTI Input Trigger to Output Channel Enable registers	0x00000000
0x3112402C	PE0_CTI0_CTIINEN3	PE0_CTI0 CTI Input Trigger to Output Channel Enable registers	0x00000000
0x31124030	PE0_CTI0_CTIINEN4	PE0_CTI0 CTI Input Trigger to Output Channel Enable registers	0x00000000
0x31124034	PE0_CTI0_CTIINEN5	PE0_CTI0 CTI Input Trigger to Output Channel Enable registers	0x00000000
0x31124038	PE0_CTI0_CTIINEN6	PE0_CTI0 CTI Input Trigger to Output Channel Enable registers	0x00000000
0x3112403C	PE0_CTI0_CTIINEN7	PE0_CTI0 CTI Input Trigger to Output Channel Enable registers	0x00000000
0x311240A0	PE0_CTI0_CTIOUTEN0	PE0_CTI0 CTI Input Channel to Output Trigger Enable registers	0x00000000
0x311240A4	PE0_CTI0_CTIOUTEN1	PE0_CTI0 CTI Input Channel to Output Trigger Enable registers	0x00000000
0x311240A8	PE0_CTI0_CTIOUTEN2	PE0_CTI0 CTI Input Channel to Output Trigger Enable registers	0x00000000
0x311240AC	PE0_CTI0_CTIOUTEN3	PE0_CTI0 CTI Input Channel to Output Trigger Enable registers	0x00000000
0x311240B0	PE0_CTI0_CTIOUTEN4	PE0_CTI0 CTI Input Channel to Output Trigger Enable registers	0x00000000
0x311240B4	PE0_CTI0_CTIOUTEN5	PE0_CTI0 CTI Input Channel to Output Trigger Enable registers	0x00000000
0x311240B8	PE0_CTI0_CTIOUTEN6	PE0_CTI0 CTI Input Channel to Output Trigger Enable registers	0x00000000

Table A-128: ADSP-SC59x PE0_CTI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311240BC	PE0_CTI0_CTIOUTEN7	PE0_CTI0 CTI Input Channel to Output Trigger Enable registers	0x00000000
0x31124130	PE0_CTI0_CTITRIGINSTATUS	PE0_CTI0 CTI Trigger In Status register	0x00000000
0x31124134	PE0_CTI0_CTITRIGOUTSTATUS	PE0_CTI0 CTI Trigger Out Status register	0x00000000
0x31124138	PE0_CTI0_CTICHINSTATUS	PE0_CTI0 CTI Channel In Status register	0x00000000
0x3112413C	PE0_CTI0_CTICHOUTSTATUS	PE0_CTI0 CTI Channel Out Status register	0x00000000
0x31124140	PE0_CTI0_CTIGATE	PE0_CTI0 CTI Channel Gate Enable register	0x0000000F
0x31124144	PE0_CTI0_ASICCTL	PE0_CTI0 CTI External Multiplexer Control register	0x00000000
0x31124F00	PE0_CTI0_CTIITCTRL	PE0_CTI0 CTI Integration mode Control register	0x00000000
0x31124FA0	PE0_CTI0_CTICLAIMSET	PE0_CTI0 CTI Claim Tag Set register	0x0000000F
0x31124FA4	PE0_CTI0_CTICLAIMCLR	PE0_CTI0 CTI Claim Tag Clear register	0x00000000
0x31124FA8	PE0_CTI0_CTIDEVAFF0	PE0_CTI0 CTI Device Affinity register 0	0x81000000
0x31124FAC	PE0_CTI0_CTIDEVAFF1	PE0_CTI0 CTI Device Affinity register 1	0x00000000
0x31124FB0	PE0_CTI0_CTILAR	PE0_CTI0 CTI Lock Access Register	0x00000000
0x31124FB4	PE0_CTI0_CTILSR	PE0_CTI0 CTI Lock Status Register	0x00000000
0x31124FB8	PE0_CTI0_CTIAUTHSTATUS	PE0_CTI0 CTI Authentication Status register	0x0000000A
0x31124FBC	PE0_CTI0_CTIDEVARCH	PE0_CTI0 CTI Device Architecture register	0x47701A14
0x31124FC0	PE0_CTI0_CTIDEVID2	PE0_CTI0 CTI Device ID register 2	0x00000000
0x31124FC4	PE0_CTI0_CTIDEVID1	PE0_CTI0 CTI Device ID register 1	0x00000000
0x31124FC8	PE0_CTI0_CTIDEVID	PE0_CTI0 CTI Device ID register 0	0x01040A00
0x31124FCC	PE0_CTI0_CTIDEVTYPE	PE0_CTI0 CTI Device Type register	0x00000014
0x31124FD0	PE0_CTI0_CTIPIDR4	PE0_CTI0 CTI Peripheral Identification Register 4	0x00000004
0x31124FE0	PE0_CTI0_CTIPIDR0	PE0_CTI0 CTI Peripheral Identification Register 0	0x00000005
0x31124FE4	PE0_CTI0_CTIPIDR1	PE0_CTI0 CTI Peripheral Identification Register 1	0x000000BD
0x31124FE8	PE0_CTI0_CTIPIDR2	PE0_CTI0 CTI Peripheral Identification Register 2	0x0000007B
0x31124FEC	PE0_CTI0_CTIPIDR3	PE0_CTI0 CTI Peripheral Identification Register 3	0x00000000
0x31124FF0	PE0_CTI0_CTICIDR0	PE0_CTI0 CTI Component Identification Register 0	0x0000000D
0x31124FF4	PE0_CTI0_CTICIDR1	PE0_CTI0 CTI Component Identification Register 1	0x00000090

Table A-128: ADSP-SC59x PE0_CTI0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31124FF8	PE0_CTI0_CTICIDR2	PE0_CTI0 CTI Component Identification Register 2	0x00000005
0x31124FFC	PE0_CTI0_CTICIDR3	PE0_CTI0 CTI Component Identification Register 3	0x000000B1

Table A-129: ADSP-SC59x PE0_DEBUG0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31114024	PE0_DEBUG0_EDECR	PE0_DEBUG0 External Debug Execution Control Register	0x00000000
0x31114D00	PE0_DEBUG0_MIDR_EL1	PE0_DEBUG0 Main ID Register	0x412FD050
0x31114D20	PE0_DEBUG0_EDPFR_31_0	PE0_DEBUG0 External Debug Processor Feature Register	0x00112222
0x31114D24	PE0_DEBUG0_EDPFR_63_32	PE0_DEBUG0 External Debug Processor Feature Register	0x00000000
0x31114D28	PE0_DEBUG0_EDDFR_31_0	PE0_DEBUG0 External Debug Feature Register	0x10305408
0x31114D2C	PE0_DEBUG0_EDDFR_63_32	PE0_DEBUG0 External Debug Feature Register	0x00000000
0x31114FA8	PE0_DEBUG0_EDDEVAFF0	PE0_DEBUG0 External Debug Device Affinity register 0	0x81000000
0x31114FAC	PE0_DEBUG0_EDDEVAFF1	PE0_DEBUG0 External Debug Device Affinity register 1	0x00000000
0x31114FB0	PE0_DEBUG0_EDLAR	PE0_DEBUG0 External Debug Lock Access Register	0x00000000
0x31114FB4	PE0_DEBUG0_EDLSR	PE0_DEBUG0 External Debug Lock Status Register	0x00000000
0x31114FB8	PE0_DEBUG0_DBGAUTHSTATUS_EL1	PE0_DEBUG0 Debug Authentication Status register	0x000000AA
0x31114FBC	PE0_DEBUG0_EDDEVARCH	PE0_DEBUG0 External Debug Device Architecture register	0x47708A15
0x31114FC0	PE0_DEBUG0_EDDEVID2	PE0_DEBUG0 External Debug Device ID register 2	0x00000000
0x31114FC4	PE0_DEBUG0_EDDEVID1	PE0_DEBUG0 External Debug Device ID register 1	0x00000000
0x31114FC8	PE0_DEBUG0_EDDEVID	PE0_DEBUG0 External Debug Device ID register 0	0x00000000
0x31114FCC	PE0_DEBUG0_EDDEVTYPE	PE0_DEBUG0 External Debug Device Type register	0x00000015
0x31114FD0	PE0_DEBUG0_EDPIDR4	PE0_DEBUG0 External Debug Peripheral Identification Register 4	0x00000004

Table A-129: ADSP-SC59x PE0_DEBUG0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31114FE0	PE0_DEBUG0_EDPIDR0	PE0_DEBUG0 External Debug Peripheral Identification Register 0	0x00000005
0x31114FE4	PE0_DEBUG0_EDPIDR1	PE0_DEBUG0 External Debug Peripheral Identification Register 1	0x000000BD
0x31114FE8	PE0_DEBUG0_EDPIDR2	PE0_DEBUG0 External Debug Peripheral Identification Register 2	0x0000003B
0x31114FEC	PE0_DEBUG0_EDPIDR3	PE0_DEBUG0 External Debug Peripheral Identification Register 3	0x00000000
0x31114FF0	PE0_DEBUG0_EDCIDR0	PE0_DEBUG0 External Debug Component Identification Register 0	0x0000000D
0x31114FF4	PE0_DEBUG0_EDCIDR1	PE0_DEBUG0 External Debug Component Identification Register 1	0x00000090
0x31114FF8	PE0_DEBUG0_EDCIDR2	PE0_DEBUG0 External Debug Component Identification Register 2	0x00000005
0x31114FFC	PE0_DEBUG0_EDCIDR3	PE0_DEBUG0 External Debug Component Identification Register 3	0x000000B1

Table A-130: ADSP-SC59x PE0_ETM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3112C310	PE0_ETM0_TRCPDCR	PE0_ETM0 Trace PowerDown Control Register	0x00000000
0x3112CFA8	PE0_ETM0_TRCDEVAFF0	PE0_ETM0 Trace Device Affinity Register 0	0x81000000
0x3112CFAC	PE0_ETM0_TRCDEVAFF1	PE0_ETM0 Trace Device Affinity Register 1	0x00000000
0x3112CFB0	PE0_ETM0_TRCLAR	PE0_ETM0 Trace Software Lock Access Register	0x00000000
0x3112CFB4	PE0_ETM0_TRCLSR	PE0_ETM0 Trace Software Lock Status Register	0x00000000
0x3112CFB8	PE0_ETM0_TRCAUTHSTATUS	PE0_ETM0 Trace Authentication Status Register	0x00000088
0x3112CFBC	PE0_ETM0_TRCDEVARCH	PE0_ETM0 Trace Device Architecture Register	0x47724A13
0x3112CFC8	PE0_ETM0_TRCDEVID	PE0_ETM0 Trace Device ID Register	0x00000000
0x3112CFCC	PE0_ETM0_TRCDEVTYPE	PE0_ETM0 Trace Device Type Register	0x00000013
0x3112CFD0	PE0_ETM0_TRCPIDR4	PE0_ETM0 Trace Peripheral ID 4 Register	0x00000004
0x3112CFE0	PE0_ETM0_TRCPIDR0	PE0_ETM0 Trace Peripheral ID 0 Register	0x00000005
0x3112CFE4	PE0_ETM0_TRCPIDR1	PE0_ETM0 Trace Peripheral ID 1 Register	0x000000BD
0x3112CFE8	PE0_ETM0_TRCPIDR2	PE0_ETM0 Trace Peripheral ID 2 Register	0x0000003B

Table A-130: ADSP-SC59x PE0_ETM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112CFEC	PE0_ETM0_TRCPIDR3	PE0_ETM0 Trace Peripheral ID 3 Register	0x00000000
0x3112CFF0	PE0_ETM0_TRCCIDR0	PE0_ETM0 Trace Component ID 0 Register	0x0000000D
0x3112CFF4	PE0_ETM0_TRCCIDR1	PE0_ETM0 Trace Component ID 1 Register	0x00000090
0x3112CFF8	PE0_ETM0_TRCCIDR2	PE0_ETM0 Trace Component ID 2 Register	0x00000005
0x3112CFFC	PE0_ETM0_TRCCIDR3	PE0_ETM0 Trace Component ID 3 Register	0x000000B1

Table A-131: ADSP-SC59x PE0_PMU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3111CFA8	PE0_PMU0_PMDEVAFF0	PE0_PMU0 Performance Monitors Device Affinity register 0	0x81000000
0x3111CFAC	PE0_PMU0_PMDEVAFF1	PE0_PMU0 Performance Monitors Device Affinity register 1	0x00000000
0x3111CFB0	PE0_PMU0_PMLAR	PE0_PMU0 Performance Monitors Lock Access Register	0x00000000
0x3111CFB4	PE0_PMU0_PMLSR	PE0_PMU0 Performance Monitors Lock Status Register	0x00000000
0x3111CFB8	PE0_PMU0_PMAUTHSTATUS	PE0_PMU0 Performance Monitors Authentication Status register	0x00000088
0x3111CFBC	PE0_PMU0_PMDEVARCH	PE0_PMU0 Performance Monitors Device Architecture register	0x47702A16
0x3111CFCC	PE0_PMU0_PMDEVTYPE	PE0_PMU0 Performance Monitors Device Type register	0x00000016
0x3111CFD0	PE0_PMU0_PMPIDR4	PE0_PMU0 Performance Monitors Peripheral Identification Register 4	0x00000004
0x3111CFE0	PE0_PMU0_PMPIDR0	PE0_PMU0 Performance Monitors Peripheral Identification Register 0	0x00000005
0x3111CFE4	PE0_PMU0_PMPIDR1	PE0_PMU0 Performance Monitors Peripheral Identification Register 1	0x000000BD
0x3111CFE8	PE0_PMU0_PMPIDR2	PE0_PMU0 Performance Monitors Peripheral Identification Register 2	0x0000003B
0x3111CFEC	PE0_PMU0_PMPIDR3	PE0_PMU0 Performance Monitors Peripheral Identification Register 3	0x00000000
0x3111CFF0	PE0_PMU0_PMCIDR0	PE0_PMU0 Performance Monitors Component Identification Register 0	0x0000000D
0x3111CFF4	PE0_PMU0_PMCIDR1	PE0_PMU0 Performance Monitors Component Identification Register 1	0x00000090

Table A-131: ADSP-SC59x PE0_PMU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3111CFF8	PE0_PMU0_PMCIDR2	PE0_PMU0 Performance Monitors Component Identification Register 2	0x00000005
0x3111CFFC	PE0_PMU0_PMCIDR3	PE0_PMU0 Performance Monitors Component Identification Register 3	0x000000B1

Table A-132: ADSP-SC59x PE_DEBUG0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31114020	PE_DEBUG0_EDESR	PE_DEBUG0 External Debug Event Status Register	0x00000000
0x31114030	PE_DEBUG0_EDWAR_31_0	PE_DEBUG0 External Debug Watchpoint Address Register	0x00000000
0x31114034	PE_DEBUG0_EDWAR_63_32	PE_DEBUG0 External Debug Watchpoint Address Register	0x00000000
0x31114080	PE_DEBUG0_DBGDTRRX_EL0	PE_DEBUG0 Debug Data Transfer Register, Receive	0x00000000
0x31114084	PE_DEBUG0_EDITR	PE_DEBUG0 External Debug Instruction Transfer Register	0x00000000
0x31114088	PE_DEBUG0_EDSCR	PE_DEBUG0 External Debug Status and Control Register	0x00000000
0x3111408C	PE_DEBUG0_DBGDTRTX_EL0	PE_DEBUG0 Debug Data Transfer Register, Transmit	0x00000000
0x31114090	PE_DEBUG0_EDRCR	PE_DEBUG0 External Debug Reserve Control Register	0x00000000
0x31114098	PE_DEBUG0_EDECCR	PE_DEBUG0 External Debug Exception Catch Control Register	0x00000000
0x311140A4	PE_DEBUG0_EDCIDSR	PE_DEBUG0 External Debug Context ID Sample Register	0x00000000
0x31114300	PE_DEBUG0_OSLAR_EL1	PE_DEBUG0 OS Lock Access Register	0x00000000
0x31114310	PE_DEBUG0_EDPRCR	PE_DEBUG0 External Debug Power/Reset Control Register	0x00000000
0x31114314	PE_DEBUG0_EDPRSR	PE_DEBUG0 External Debug Processor Status Register	0x00000003
0x31114408	PE_DEBUG0_DBGBCR0_EL1	PE_DEBUG0 Debug Breakpoint Control Registers	0x00000000
0x31114418	PE_DEBUG0_DBGBCR1_EL1	PE_DEBUG0 Debug Breakpoint Control Registers	0x00000000
0x31114428	PE_DEBUG0_DBGBCR2_EL1	PE_DEBUG0 Debug Breakpoint Control Registers	0x00000000

Table A-132: ADSP-SC59x PE_DEBUG0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31114438	PE_DEBUG0_DBGBCR3_EL1	PE_DEBUG0 Debug Breakpoint Control Registers	0x00000000
0x31114448	PE_DEBUG0_DBGBCR4_EL1	PE_DEBUG0 Debug Breakpoint Control Registers	0x00000000
0x31114458	PE_DEBUG0_DBGBCR5_EL1	PE_DEBUG0 Debug Breakpoint Control Registers	0x00000000
0x31114800	PE_DEBUG0_DBGWVR0_EL1_31_0	PE_DEBUG0 Debug Watchpoint Value Registers	0x00000000
0x31114804	PE_DEBUG0_DBGWVR0_EL1_63_32	PE_DEBUG0 Debug Watchpoint Value Registers	0x00000000
0x31114808	PE_DEBUG0_DBGWCR0_EL1	PE_DEBUG0 Debug Watchpoint Control Registers	0x00000000
0x31114810	PE_DEBUG0_DBGWVR1_EL1_31_0	PE_DEBUG0 Debug Watchpoint Value Registers	0x00000000
0x31114814	PE_DEBUG0_DBGWVR1_EL1_63_32	PE_DEBUG0 Debug Watchpoint Value Registers	0x00000000
0x31114818	PE_DEBUG0_DBGWCR1_EL1	PE_DEBUG0 Debug Watchpoint Control Registers	0x00000000
0x31114820	PE_DEBUG0_DBGWVR2_EL1_31_0	PE_DEBUG0 Debug Watchpoint Value Registers	0x00000000
0x31114824	PE_DEBUG0_DBGWVR2_EL1_63_32	PE_DEBUG0 Debug Watchpoint Value Registers	0x00000000
0x31114828	PE_DEBUG0_DBGWCR2_EL1	PE_DEBUG0 Debug Watchpoint Control Registers	0x00000000
0x31114830	PE_DEBUG0_DBGWVR3_EL1_31_0	PE_DEBUG0 Debug Watchpoint Value Registers	0x00000000
0x31114834	PE_DEBUG0_DBGWVR3_EL1_63_32	PE_DEBUG0 Debug Watchpoint Value Registers	0x00000000
0x31114838	PE_DEBUG0_DBGWCR3_EL1	PE_DEBUG0 Debug Watchpoint Control Registers	0x00000000

Table A-132: ADSP-SC59x PE_DEBUG0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31114F00	PE_DEBUG0_EDITCTRL	PE_DEBUG0 External Debug Integration mode Control register	0x00000000
0x31114FA0	PE_DEBUG0_DBGCLAIMSET_EL1	PE_DEBUG0 Debug Claim Tag Set register	0x000000FF
0x31114FA4	PE_DEBUG0_DBGCLAIMCLR_EL1	PE_DEBUG0 Debug Claim Tag Clear register	0x00000000

Table A-133: ADSP-SC59x PE_ETM0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3112C004	PE_ETM0_TRCPRGCTLR	PE_ETM0 Trace Programming Control Register	0x00000000
0x3112C00C	PE_ETM0_TRCSTATR	PE_ETM0 Trace Status Register	0x00000000
0x3112C010	PE_ETM0_TRCCONFIGR	PE_ETM0 Trace Configuration Register	0x00000001
0x3112C018	PE_ETM0_TRCAUXCTLR	PE_ETM0 Trace Auxiliary Control Register	0x00000000
0x3112C020	PE_ETM0_TRCEVENTCTL0R	PE_ETM0 Trace Event Control 0 Register	0x00000000
0x3112C024	PE_ETM0_TRCEVENTCTL1R	PE_ETM0 Trace Event Control 1 Register	0x00000000
0x3112C02C	PE_ETM0_TRCSTALLCTLR	PE_ETM0 Trace Stall Control Register	0x00000000
0x3112C030	PE_ETM0_TRCTSCTLR	PE_ETM0 Trace Global Timestamp Control Register	0x00000000
0x3112C034	PE_ETM0_TRCSYNCPR	PE_ETM0 Trace Synchronization Period Register	0x00000000
0x3112C038	PE_ETM0_TRCCCCTLR	PE_ETM0 Trace Cycle Count Control Register	0x00000000
0x3112C03C	PE_ETM0_TRCBBCTLR	PE_ETM0 Trace Branch Broadcast Control Register	0x00000000
0x3112C040	PE_ETM0_TRCTRACEIDR	PE_ETM0 Trace Trace ID Register	0x00000000
0x3112C080	PE_ETM0_TRCVICTLR	PE_ETM0 Trace ViewInst Main Control Register	0x00000000
0x3112C084	PE_ETM0_TRCVIIECTLR	PE_ETM0 Trace ViewInst Include/Exclude Control Register	0x00000000
0x3112C088	PE_ETM0_TRCVISSCTLR	PE_ETM0 Trace ViewInst Start/Stop Control Register	0x00000000
0x3112C100	PE_ETM0_TRCSEQEVR0	PE_ETM0 Trace Sequencer State Transition Control Registers	0x00000000
0x3112C104	PE_ETM0_TRCSEQEVR1	PE_ETM0 Trace Sequencer State Transition Control Registers	0x00000000

Table A-133: ADSP-SC59x PE_ETM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112C108	PE_ETM0_TRCSEQEVR2	PE_ETM0 Trace Sequencer State Transition Control Registers	0x00000000
0x3112C118	PE_ETM0_TRCSEQRSTEVR	PE_ETM0 Trace Sequencer Reset Control Register	0x00000000
0x3112C11C	PE_ETM0_TRCSEQSTR	PE_ETM0 Trace Sequencer State Register	0x00000000
0x3112C120	PE_ETM0_TRCEXTINSELR	PE_ETM0 Trace External Input Select Register	0x00000000
0x3112C140	PE_ETM0_TRCCNTRLDVR0	PE_ETM0 Trace Counter Reload Value Registers	0x00000000
0x3112C144	PE_ETM0_TRCCNTRLDVR1	PE_ETM0 Trace Counter Reload Value Registers	0x00000000
0x3112C150	PE_ETM0_TRCCNTCTLR0	PE_ETM0 Trace Counter Control Registers	0x00000000
0x3112C154	PE_ETM0_TRCCNTCTLR1	PE_ETM0 Trace Counter Control Registers	0x00000000
0x3112C160	PE_ETM0_TRCCNTVR0	PE_ETM0 Trace Counter Value Registers	0x00000000
0x3112C164	PE_ETM0_TRCCNTVR1	PE_ETM0 Trace Counter Value Registers	0x00000000
0x3112C1C0	PE_ETM0_TRCIMSPEC0	PE_ETM0 Trace IMPLEMENTATION DEFINED Register 0	0x00000000
0x3112C1E0	PE_ETM0_TRCIDR0	PE_ETM0 Trace ID Register 0	0x50000EA1
0x3112C1E4	PE_ETM0_TRCIDR1	PE_ETM0 Trace ID Register 1	0x4100F423
0x3112C1E8	PE_ETM0_TRCIDR2	PE_ETM0 Trace ID Register 2	0x20001088
0x3112C1EC	PE_ETM0_TRCIDR3	PE_ETM0 Trace ID Register 3	0x0D7B0004
0x3112C1F0	PE_ETM0_TRCIDR4	PE_ETM0 Trace ID Register 4	0x11170004
0x3112C1F4	PE_ETM0_TRCIDR5	PE_ETM0 Trace ID Register 5	0x28C7082F
0x3112C208	PE_ETM0_TRCRSCTLR2	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C20C	PE_ETM0_TRCRSCTLR3	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C210	PE_ETM0_TRCRSCTLR4	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C214	PE_ETM0_TRCRSCTLR5	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C218	PE_ETM0_TRCRSCTLR6	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C21C	PE_ETM0_TRCRSCTLR7	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C220	PE_ETM0_TRCRSCTLR8	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C224	PE_ETM0_TRCRSCTLR9	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C228	PE_ETM0_TRCRSCTLR10	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C22C	PE_ETM0_TRCRSCTLR11	PE_ETM0 Trace Resource Selection Control Registers	0x00000000

Table A-133: ADSP-SC59x PE_ETM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112C230	PE_ETM0_TRCRSCTLR12	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C234	PE_ETM0_TRCRSCTLR13	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C238	PE_ETM0_TRCRSCTLR14	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C23C	PE_ETM0_TRCRSCTLR15	PE_ETM0 Trace Resource Selection Control Registers	0x00000000
0x3112C280	PE_ETM0_TRCSSCCR0	PE_ETM0 Trace Single-Shot Comparator Control Registers	0x00000000
0x3112C2A0	PE_ETM0_TRCSSCSR0	PE_ETM0 Trace Single-Shot Comparator Status Registers	0x00000001
0x3112C300	PE_ETM0_TRCOSLAR	PE_ETM0 Trace OS Lock Access Register	0x00000000
0x3112C304	PE_ETM0_TRCOSLSR	PE_ETM0 Trace OS Lock Status Register	0x0000000A
0x3112C314	PE_ETM0_TRCPDSR	PE_ETM0 Trace PowerDown Status Register	0x00000000
0x3112C400	PE_ETM0_TRCACVR0_31_0	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C404	PE_ETM0_TRCACVR0_63_32	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C408	PE_ETM0_TRCACVR1_31_0	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C40C	PE_ETM0_TRCACVR1_63_32	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C410	PE_ETM0_TRCACVR2_31_0	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C414	PE_ETM0_TRCACVR2_63_32	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C418	PE_ETM0_TRCACVR3_31_0	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C41C	PE_ETM0_TRCACVR3_63_32	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C420	PE_ETM0_TRCACVR4_31_0	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C424	PE_ETM0_TRCACVR4_63_32	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C428	PE_ETM0_TRCACVR5_31_0	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C42C	PE_ETM0_TRCACVR5_63_32	PE_ETM0 Trace Address Comparator Value Registers	0x00000000

Table A-133: ADSP-SC59x PE_ETM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112C430	PE_ETM0_TRCACVR6_31_0	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C434	PE_ETM0_TRCACVR6_63_32	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C438	PE_ETM0_TRCACVR7_31_0	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C43C	PE_ETM0_TRCACVR7_63_32	PE_ETM0 Trace Address Comparator Value Registers	0x00000000
0x3112C480	PE_ETM0_TRCACATR0_31_0	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C484	PE_ETM0_TRCACATR0_63_32	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C488	PE_ETM0_TRCACATR1_31_0	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C48C	PE_ETM0_TRCACATR1_63_32	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C490	PE_ETM0_TRCACATR2_31_0	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C494	PE_ETM0_TRCACATR2_63_32	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C498	PE_ETM0_TRCACATR3_31_0	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C49C	PE_ETM0_TRCACATR3_63_32	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C4A0	PE_ETM0_TRCACATR4_31_0	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C4A4	PE_ETM0_TRCACATR4_63_32	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C4A8	PE_ETM0_TRCACATR5_31_0	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C4AC	PE_ETM0_TRCACATR5_63_32	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C4B0	PE_ETM0_TRCACATR6_31_0	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C4B4	PE_ETM0_TRCACATR6_63_32	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000

Table A-133: ADSP-SC59x PE_ETM0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3112C4B8	PE_ETM0_TRCACATR7_31_0	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C4BC	PE_ETM0_TRCACATR7_63_32	PE_ETM0 Trace Address Comparator Access Type Registers	0x00000000
0x3112C600	PE_ETM0_TRCCIDCVR0_31_0	PE_ETM0 Trace Context ID Comparator Value Registers	0x00000000
0x3112C604	PE_ETM0_TRCCIDCVR0_63_32	PE_ETM0 Trace Context ID Comparator Value Registers	0x00000000
0x3112C640	PE_ETM0_TRCVMIDCVR0_31_0	PE_ETM0 Trace VMID Comparator Value Registers	0x00000000
0x3112C644	PE_ETM0_TRCVMIDCVR0_63_32	PE_ETM0 Trace VMID Comparator Value Registers	0x00000000
0x3112C680	PE_ETM0_TRCCIDCCTLR0	PE_ETM0 Trace Context ID Comparator Control Register 0	0x00000000
0x3112CEE4	PE_ETM0_TRCITATBIDR	PE_ETM0 Integration ATB Identification Register	0x00000000
0x3112CEEC	PE_ETM0_TRCITIDATAR	PE_ETM0 Integration Instruction ATB Data Register	0x00000000
0x3112CEF4	PE_ETM0_TRCITIATBINR	PE_ETM0 Integration Instruction ATB In Register	0x00000000
0x3112CEFC	PE_ETM0_TRCITIATBOUTR	PE_ETM0 Integration Instruction ATB Out Register	0x00000000
0x3112CF00	PE_ETM0_TRCITCTRL	PE_ETM0 Trace Integration Mode Control Register	0x00000000
0x3112CFA0	PE_ETM0_TRCCLAIMSET	PE_ETM0 Trace Claim Tag Set Register	0x0000000F
0x3112CFA4	PE_ETM0_TRCCLAIMCLR	PE_ETM0 Trace Claim Tag Clear Register	0x00000000

Table A-134: ADSP-SC59x PE_PMU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3111C000	PE_PMU0_PMEVCNTR0_EL0	PE_PMU0 Performance Monitors Event Count Registers	0x00000000
0x3111C008	PE_PMU0_PMEVCNTR1_EL0	PE_PMU0 Performance Monitors Event Count Registers	0x00000000
0x3111C010	PE_PMU0_PMEVCNTR2_EL0	PE_PMU0 Performance Monitors Event Count Registers	0x00000000
0x3111C018	PE_PMU0_PMEVCNTR3_EL0	PE_PMU0 Performance Monitors Event Count Registers	0x00000000
0x3111C020	PE_PMU0_PMEVCNTR4_EL0	PE_PMU0 Performance Monitors Event Count Registers	0x00000000

Table A-134: ADSP-SC59x PE_PMU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3111C028	PE_PMU0_PMEVCNTR5_EL0	PE_PMU0 Performance Monitors Event Count Registers	0x00000000
0x3111C0F8	PE_PMU0_PMCCNTR_EL0_31_0	PE_PMU0 Performance Monitors Cycle Counter	0x00000000
0x3111C0FC	PE_PMU0_PMCCNTR_EL0_63_32	PE_PMU0 Performance Monitors Cycle Counter	0x00000000
0x3111C400	PE_PMU0_PMEVTYPER0_EL0	PE_PMU0 Performance Monitors Event Type Registers	0x00000000
0x3111C404	PE_PMU0_PMEVTYPER1_EL0	PE_PMU0 Performance Monitors Event Type Registers	0x00000000
0x3111C408	PE_PMU0_PMEVTYPER2_EL0	PE_PMU0 Performance Monitors Event Type Registers	0x00000000
0x3111C40C	PE_PMU0_PMEVTYPER3_EL0	PE_PMU0 Performance Monitors Event Type Registers	0x00000000
0x3111C410	PE_PMU0_PMEVTYPER4_EL0	PE_PMU0 Performance Monitors Event Type Registers	0x00000000
0x3111C414	PE_PMU0_PMEVTYPER5_EL0	PE_PMU0 Performance Monitors Event Type Registers	0x00000000
0x3111C47C	PE_PMU0_PMCCFILTR_EL0	PE_PMU0 Performance Monitors Cycle Counter Filter Register	0x00000000
0x3111CC00	PE_PMU0_PMCNTENSET_EL0	PE_PMU0 Performance Monitors Count Enable Set register	0x00000000
0x3111CC20	PE_PMU0_PMCNTENCLR_EL0	PE_PMU0 Performance Monitors Count Enable Clear register	0x00000000
0x3111CC40	PE_PMU0_PMINTENSET_EL1	PE_PMU0 Performance Monitors Interrupt Enable Set register	0x00000000
0x3111CC60	PE_PMU0_PMINTENCLR_EL1	PE_PMU0 Performance Monitors Interrupt Enable Clear register	0x00000000
0x3111CC80	PE_PMU0_PMOVSCLR_EL0	PE_PMU0 Performance Monitors Overflow Flag Status Clear register	0x00000000
0x3111CCA0	PE_PMU0_PMSWINC_EL0	PE_PMU0 Performance Monitors Software Increment register	0x00000000
0x3111CCC0	PE_PMU0_PMOVSSET_EL0	PE_PMU0 Performance Monitors Overflow Flag Status Set register	0x00000000
0x3111CE00	PE_PMU0_PMCFGR	PE_PMU0 Performance Monitors Configuration Register	0x0001FF06
0x3111CE04	PE_PMU0_PMCR_EL0	PE_PMU0 Performance Monitors Control Register	0x00000000

Table A-134: ADSP-SC59x PE_PMU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3111CE20	PE_PMU0_PMCEID0	PE_PMU0 Performance Monitors Common Event Identification register 0	0x6FFFBFFF
0x3111CE24	PE_PMU0_PMCEID1	PE_PMU0 Performance Monitors Common Event Identification register 1	0x00000000
0x3111CE28	PE_PMU0_PMCEID2	PE_PMU0 Performance Monitors Common Event Identification register 2	0x00000000
0x3111CE2C	PE_PMU0_PMCEID3	PE_PMU0 Performance Monitors Common Event Identification register 3	0x00000000
0x3111CF00	PE_PMU0_PMITCTRL	PE_PMU0 Performance Monitors Integration mode Control register	0x00000000

Table A-135: ADSP-SC59x DDRPFB0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31076000	DDRPFB0_CTL0	DDRPFB0 Control Register 0	0x00000000
0x31076004	DDRPFB0_CTL1	DDRPFB0 Control Register 1	0x00001107
0x31076014	DDRPFB0_ZERO_START	DDRPFB0 Return Zero Start Address Register	0x00000000
0x31076018	DDRPFB0_ZERO_END	DDRPFB0 Return Zero End Address Register	0x00000000
0x3107601C	DDRPFB0_RANGE_SELECT	DDRPFB0 Range Select Register	0x0000FFFF
0x31076020	DDRPFB0_STS0	DDRPFB0 Prefetch Buffer First Status Register	0x00000002
0x31076050	DDRPFB0_RQSTR_ID_EN	DDRPFB0 Requester ID Enable Register	0x0000000F
0x31076054	DDRPFB0_RQSTR0_ID	DDRPFB0 Requester ID 0 Register	0x0000000F
0x31076058	DDRPFB0_RQSTR0_ID_MSK	DDRPFB0 Requester ID 0 Mask Register	0x00000000
0x3107605C	DDRPFB0_RQSTR1_ID	DDRPFB0 Requester ID 1 Register	0x0000000F
0x31076060	DDRPFB0_RQSTR1_ID_MSK	DDRPFB0 Requester ID 1 Mask Register	0x00000000
0x31076064	DDRPFB0_RQSTR2_ID	DDRPFB0 Requester ID 2 Register	0x0000000F
0x31076068	DDRPFB0_RQSTR2_ID_MSK	DDRPFB0 Requester ID 2 Mask Register	0x00000000
0x3107606C	DDRPFB0_RQSTR3_ID	DDRPFB0 Requester ID 3 Register	0x0000000F
0x31076070	DDRPFB0_RQSTR3_ID_MSK	DDRPFB0 Requester ID 3 Mask Register	0x00000000

Table A-136: ADSP-SC59x PINT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005000	PINT0_MSK_SET	PINT0 PINT Mask Set Register	0x00000000
0x31005004	PINT0_MSK_CLR	PINT0 PINT Mask Clear Register	0x00000000
0x31005008	PINT0_REQ	PINT0 PINT Request Register	0x00000000
0x3100500C	PINT0_ASSIGN	PINT0 PINT Assign Register	0x00000101
0x31005010	PINT0_EDGE_SET	PINT0 PINT Edge Set Register	0x00000000
0x31005014	PINT0_EDGE_CLR	PINT0 PINT Edge Clear Register	0x00000000
0x31005018	PINT0_INV_SET	PINT0 PINT Invert Set Register	0x00000000
0x3100501C	PINT0_INV_CLR	PINT0 PINT Invert Clear Register	0x00000000
0x31005020	PINT0_PINSTATE	PINT0 PINT Pin State Register	0x00000000
0x31005024	PINT0_LATCH	PINT0 PINT Latch Register	0x00000000

Table A-137: ADSP-SC59x PINT1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005100	PINT1_MSK_SET	PINT1 PINT Mask Set Register	0x00000000
0x31005104	PINT1_MSK_CLR	PINT1 PINT Mask Clear Register	0x00000000
0x31005108	PINT1_REQ	PINT1 PINT Request Register	0x00000000
0x3100510C	PINT1_ASSIGN	PINT1 PINT Assign Register	0x00000101
0x31005110	PINT1_EDGE_SET	PINT1 PINT Edge Set Register	0x00000000
0x31005114	PINT1_EDGE_CLR	PINT1 PINT Edge Clear Register	0x00000000
0x31005118	PINT1_INV_SET	PINT1 PINT Invert Set Register	0x00000000
0x3100511C	PINT1_INV_CLR	PINT1 PINT Invert Clear Register	0x00000000
0x31005120	PINT1_PINSTATE	PINT1 PINT Pin State Register	0x00000000
0x31005124	PINT1_LATCH	PINT1 PINT Latch Register	0x00000000

Table A-138: ADSP-SC59x PINT2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005200	PINT2_MSK_SET	PINT2 PINT Mask Set Register	0x00000000
0x31005204	PINT2_MSK_CLR	PINT2 PINT Mask Clear Register	0x00000000
0x31005208	PINT2_REQ	PINT2 PINT Request Register	0x00000000
0x3100520C	PINT2_ASSIGN	PINT2 PINT Assign Register	0x00000101

Table A-138: ADSP-SC59x PINT2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31005210	PINT2_EDGE_SET	PINT2 PINT Edge Set Register	0x00000000
0x31005214	PINT2_EDGE_CLR	PINT2 PINT Edge Clear Register	0x00000000
0x31005218	PINT2_INV_SET	PINT2 PINT Invert Set Register	0x00000000
0x3100521C	PINT2_INV_CLR	PINT2 PINT Invert Clear Register	0x00000000
0x31005220	PINT2_PINSTATE	PINT2 PINT Pin State Register	0x00000000
0x31005224	PINT2_LATCH	PINT2 PINT Latch Register	0x00000000

Table A-139: ADSP-SC59x PINT3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005300	PINT3_MSK_SET	PINT3 PINT Mask Set Register	0x00000000
0x31005304	PINT3_MSK_CLR	PINT3 PINT Mask Clear Register	0x00000000
0x31005308	PINT3_REQ	PINT3 PINT Request Register	0x00000000
0x3100530C	PINT3_ASSIGN	PINT3 PINT Assign Register	0x00000101
0x31005310	PINT3_EDGE_SET	PINT3 PINT Edge Set Register	0x00000000
0x31005314	PINT3_EDGE_CLR	PINT3 PINT Edge Clear Register	0x00000000
0x31005318	PINT3_INV_SET	PINT3 PINT Invert Set Register	0x00000000
0x3100531C	PINT3_INV_CLR	PINT3 PINT Invert Clear Register	0x00000000
0x31005320	PINT3_PINSTATE	PINT3 PINT Pin State Register	0x00000000
0x31005324	PINT3_LATCH	PINT3 PINT Latch Register	0x00000000

Table A-140: ADSP-SC59x PINT4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005400	PINT4_MSK_SET	PINT4 PINT Mask Set Register	0x00000000
0x31005404	PINT4_MSK_CLR	PINT4 PINT Mask Clear Register	0x00000000
0x31005408	PINT4_REQ	PINT4 PINT Request Register	0x00000000
0x3100540C	PINT4_ASSIGN	PINT4 PINT Assign Register	0x00000101
0x31005410	PINT4_EDGE_SET	PINT4 PINT Edge Set Register	0x00000000
0x31005414	PINT4_EDGE_CLR	PINT4 PINT Edge Clear Register	0x00000000
0x31005418	PINT4_INV_SET	PINT4 PINT Invert Set Register	0x00000000
0x3100541C	PINT4_INV_CLR	PINT4 PINT Invert Clear Register	0x00000000

Table A-140: ADSP-SC59x PINT4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31005420	PINT4_PINSTATE	PINT4 PINT Pin State Register	0x00000000
0x31005424	PINT4_LATCH	PINT4 PINT Latch Register	0x00000000

Table A-141: ADSP-SC59x PINT5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005500	PINT5_MSK_SET	PINT5 PINT Mask Set Register	0x00000000
0x31005504	PINT5_MSK_CLR	PINT5 PINT Mask Clear Register	0x00000000
0x31005508	PINT5_REQ	PINT5 PINT Request Register	0x00000000
0x3100550C	PINT5_ASSIGN	PINT5 PINT Assign Register	0x00000101
0x31005510	PINT5_EDGE_SET	PINT5 PINT Edge Set Register	0x00000000
0x31005514	PINT5_EDGE_CLR	PINT5 PINT Edge Clear Register	0x00000000
0x31005518	PINT5_INV_SET	PINT5 PINT Invert Set Register	0x00000000
0x3100551C	PINT5_INV_CLR	PINT5 PINT Invert Clear Register	0x00000000
0x31005520	PINT5_PINSTATE	PINT5 PINT Pin State Register	0x00000000
0x31005524	PINT5_LATCH	PINT5 PINT Latch Register	0x00000000

Table A-142: ADSP-SC59x PINT6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005600	PINT6_MSK_SET	PINT6 PINT Mask Set Register	0x00000000
0x31005604	PINT6_MSK_CLR	PINT6 PINT Mask Clear Register	0x00000000
0x31005608	PINT6_REQ	PINT6 PINT Request Register	0x00000000
0x3100560C	PINT6_ASSIGN	PINT6 PINT Assign Register	0x00000101
0x31005610	PINT6_EDGE_SET	PINT6 PINT Edge Set Register	0x00000000
0x31005614	PINT6_EDGE_CLR	PINT6 PINT Edge Clear Register	0x00000000
0x31005618	PINT6_INV_SET	PINT6 PINT Invert Set Register	0x00000000
0x3100561C	PINT6_INV_CLR	PINT6 PINT Invert Clear Register	0x00000000
0x31005620	PINT6_PINSTATE	PINT6 PINT Pin State Register	0x00000000
0x31005624	PINT6_LATCH	PINT6 PINT Latch Register	0x00000000

Table A-143: ADSP-SC59x PINT7 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31005700	PINT7_MSK_SET	PINT7 PINT Mask Set Register	0x00000000
0x31005704	PINT7_MSK_CLR	PINT7 PINT Mask Clear Register	0x00000000
0x31005708	PINT7_REQ	PINT7 PINT Request Register	0x00000000
0x3100570C	PINT7_ASSIGN	PINT7 PINT Assign Register	0x00000101
0x31005710	PINT7_EDGE_SET	PINT7 PINT Edge Set Register	0x00000000
0x31005714	PINT7_EDGE_CLR	PINT7 PINT Edge Clear Register	0x00000000
0x31005718	PINT7_INV_SET	PINT7 PINT Invert Set Register	0x00000000
0x3100571C	PINT7_INV_CLR	PINT7 PINT Invert Clear Register	0x00000000
0x31005720	PINT7_PINSTATE	PINT7 PINT Pin State Register	0x00000000
0x31005724	PINT7_LATCH	PINT7 PINT Latch Register	0x00000000

Table A-144: ADSP-SC59x PKA0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310D4000	PKA0_APTR	PKA0 PKA Vector_A Address	0x00000000
0x310D4004	PKA0_BPTR	PKA0 PKA Vector_B Address	0x00000000
0x310D4008	PKA0_CPTR	PKA0 PKA Vector_C Address	0x00000000
0x310D400C	PKA0_DPTR	PKA0 PKA Vector_D Address	0x00000000
0x310D4010	PKA0_ALEN	PKA0 PKA Vector_A Length	0x00000000
0x310D4014	PKA0_BLEN	PKA0 PKA Vector_B Length	0x00000000
0x310D4018	PKA0_SHIFT	PKA0 PKA Bit Shift Value	0x00000000
0x310D401C	PKA0_FUNC	PKA0 PKA Function	0x00000000
0x310D4020	PKA0_COMPARE	PKA0 PKA Compare Result	0x00000001
0x310D4024	PKA0_RESULTMSW	PKA0 PKA Most-Significant-Word of Result Vector	0x00008000
0x310D4028	PKA0_DIVMSW	PKA0 PKA Most-Significant-Word of Divide Remainder	0x00008000
0x310D6000	PKA0_RAM	PKA0 Start of PKA RAM space	0x00000000

Table A-145: ADSP-SC59x PKIC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310D8000	PKIC0_POL_CTL	PKIC0 Polarity Control Register	0x00000000
0x310D8004	PKIC0_TYPE_CTL	PKIC0 Type Control Register	0x00000000

Table A-145: ADSP-SC59x PKIC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310D8008	PKIC0_EN_CTL	PKIC0 Enable Control Register	0x00000000
0x310D800C	PKIC0_RAW_STAT	PKIC0 Raw Status Register	0x00000000
0x310D800C	PKIC0_EN_SET	PKIC0 Enable Set Register	0x00000000
0x310D8010	PKIC0_ACK	PKIC0 Acknowledge Register	0x00000000
0x310D8010	PKIC0_EN_STAT	PKIC0 Enabled Status Register	0x00000000
0x310D8014	PKIC0_EN_CLR	PKIC0 Enable Clear Register	0x00000000

Table A-146: ADSP-SC59x PKTE0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CD000	PKTE0_CTL_STAT	PKTE0 Packet Engine Control Register	0x00000002
0x310CD004	PKTE0_SRC_ADDR	PKTE0 Packet Engine Source Address	0x00000000
0x310CD008	PKTE0_DEST_ADDR	PKTE0 Packet Engine Destination Address	0x00000000
0x310CD00C	PKTE0_SA_ADDR	PKTE0 Packet Engine SA Address	0x00000000
0x310CD010	PKTE0_STATE_ADDR	PKTE0 Packet Engine State Record Address	0x00000000
0x310CD014	PKTE0_ARC4STATE_ADDR	PKTE0 Packet Engine ARC4 State Record Address	0x00000000
0x310CD018	PKTE0_USERID	PKTE0 Packet Engine User ID	0x00000000
0x310CD01C	PKTE0_LEN	PKTE0 Packet Engine Length Register	0x00800000
0x310CD080	PKTE0_CDRBASE_ADDR	PKTE0 Packet Engine Command Descriptor Ring Base Address	0x00000000
0x310CD084	PKTE0_RDRBASE_ADDR	PKTE0 Packet Engine Result Descriptor Ring Base Address	0x00000000
0x310CD088	PKTE0_RING_CFG	PKTE0 Packet Engine Ring Configuration	0x00000000
0x310CD08C	PKTE0_RING_THRESH	PKTE0 Packet Engine Ring Threshold Registers	0x00000000
0x310CD090	PKTE0_CDSC_INCR	PKTE0 Packet Engine Command Descriptor Count Increment Register	0x00000000
0x310CD090	PKTE0_CDSC_CNT	PKTE0 Packet Engine Command Descriptor Count Register	0x00000000
0x310CD094	PKTE0_RDSC_CNT	PKTE0 Packet Engine Result Descriptor Count Registers	0x00000000
0x310CD094	PKTE0_RDSC_DECR	PKTE0 Packet Engine Result Descriptor Count Decrement Registers	0x00000000
0x310CD098	PKTE0_RING_PTR	PKTE0 Packet Engine Ring Pointer Status	0x00000000

Table A-146: ADSP-SC59x PKTE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CD09C	PKTE0_RING_STAT	PKTE0 Packet Engine Ring Status	0x00000000
0x310CD100	PKTE0_CFG	PKTE0 Packet Engine Configuration Register	0x00000000
0x310CD104	PKTE0_STAT	PKTE0 Packet Engine Status Register	0x00040402
0x310CD10C	PKTE0_BUF_THRESH	PKTE0 Packet Engine Buffer Threshold Register	0x00800080
0x310CD110	PKTE0_INBUF_INCR	PKTE0 Packet Engine Input Buffer Count Increment Register	0x00000000
0x310CD110	PKTE0_INBUF_CNT	PKTE0 Packet Engine Input Buffer Count Register	0x00000000
0x310CD114	PKTE0_OUTBUF_CNT	PKTE0 Packet Engine Output Buffer Count Register	0x00000000
0x310CD114	PKTE0_OUTBUF_DECR	PKTE0 Packet Engine Output Buffer Count Decrement Register	0x00000000
0x310CD118	PKTE0_BUF_PTR	PKTE0 Packet Engine Buffer Pointer Register	0x00000000
0x310CD120	PKTE0_DMA_CFG	PKTE0 Packet Engine DMA Configuration Register	0x00180006
0x310CD1D0	PKTE0_ENDIAN_CFG	PKTE0 Packet Engine Endian Configuration Register	0x00E400E4
0x310CD1E0	PKTE0_HLT_CTL	PKTE0 Packet Engine Halt Control Register	0x00000000
0x310CD1E0	PKTE0_HLT_STAT	PKTE0 Packet Engine Halt Status Register	0x00000000
0x310CD1E4	PKTE0_CONT	PKTE0 PKTE Continue Register	0x00000000
0x310CD1E8	PKTE0_CLK_CTL	PKTE0 PE Clock Control Register	0x0000001F
0x310CD200	PKTE0_IUMSK_STAT	PKTE0 Interrupt Unmasked Status Register	0x00000000
0x310CD204	PKTE0_IMSK_STAT	PKTE0 Interrupt Masked Status Register	0x00000000
0x310CD204	PKTE0_INT_CLR	PKTE0 Interrupt Clear Register	0x00000000
0x310CD208	PKTE0_INT_EN	PKTE0 Interrupt Enable Register	0x00000000
0x310CD20C	PKTE0_INT_CFG	PKTE0 Interrupt Configuration Register	0x00000000
0x310CD210	PKTE0_IMSK_EN	PKTE0 Interrupt Mask Enable Register	0x00000000
0x310CD214	PKTE0_IMSK_DIS	PKTE0 Interrupt Mask Disable Register	0x00000000
0x310CD400	PKTE0_SA_CMD0	PKTE0 SA Command 0	0x00000000
0x310CD404	PKTE0_SA_CMD1	PKTE0 SA Command 1	0x00000000
0x310CD408	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD40C	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD410	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD414	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD418	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000

Table A-146: ADSP-SC59x PKTE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CD41C	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD420	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD424	PKTE0_SA_KEY[n]	PKTE0 SA Key Registers	0x00000000
0x310CD428	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD42C	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD430	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD434	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD438	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD43C	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD440	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD444	PKTE0_SA_IDIGEST[n]	PKTE0 SA Inner Hash Digest Registers	0x00000000
0x310CD448	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD44C	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD450	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD454	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD458	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD45C	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD460	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD464	PKTE0_SA_ODIGEST[n]	PKTE0 SA Outer Hash Digest Registers	0x00000000
0x310CD468	PKTE0_SA_SPI	PKTE0 SA SPI Register	0x00000000
0x310CD46C	PKTE0_SA_SEQNUM[n]	PKTE0 SA Sequence Number Register	0x00000000
0x310CD470	PKTE0_SA_SEQNUM[n]	PKTE0 SA Sequence Number Register	0x00000000
0x310CD474	PKTE0_SA_SEQ-NUM_MSK[n]	PKTE0 SA Sequence Number Mask Registers	0x00000000
0x310CD478	PKTE0_SA_SEQ-NUM_MSK[n]	PKTE0 SA Sequence Number Mask Registers	0x00000000
0x310CD47C	PKTE0_SA_ARC4IJPTR	PKTE0 ARC4 i and j Pointer Register	0x00000000
0x310CD47C	PKTE0_SA_RDY	PKTE0 SA Ready Indicator	0x00000000
0x310CD47C	PKTE0_SA_NONCE	PKTE0 SA Initialization Vector Register	0x00000000
0x310CD500	PKTE0_STATE_IV[n]	PKTE0 State Initialization Vector Registers	0x00000000
0x310CD504	PKTE0_STATE_IV[n]	PKTE0 State Initialization Vector Registers	0x00000000

Table A-146: ADSP-SC59x PKTE0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CD508	PKTE0_STATE_IV[n]	PKTE0 State Initialization Vector Registers	0x00000000
0x310CD50C	PKTE0_STATE_IV[n]	PKTE0 State Initialization Vector Registers	0x00000000
0x310CD510	PKTE0_STATE_BYTE_CNT[n]	PKTE0 State Hash Byte Count Registers	0x00000000
0x310CD514	PKTE0_STATE_BYTE_CNT[n]	PKTE0 State Hash Byte Count Registers	0x00000000
0x310CD518	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD51C	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD520	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD524	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD528	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD52C	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD530	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD534	PKTE0_STATE_IDIGEST[n]	PKTE0 State Inner Digest Registers	0x00000000
0x310CD700	PKTE0_ARC4STATE_BUF	PKTE0 Starting Entry of 256-byte ARC4 State Buffer	0x00000000
0x310CD800	PKTE0_DATAIO_BUF	PKTE0 Starting Entry of 256-byte Data Input/Output Buffer	0x00000000

Table A-147: ADSP-SC59x PORT MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004000	PORT_FER	PORT Port x Function Enable Register	0x00000000
0x31004000	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004004	PORT_FER_SET	PORT Port x Function Enable Set Register	0x00000000
0x31004004	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004008	PORT_FER_CLR	PORT Port x Function Enable Clear Register	0x00000000
0x31004008	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100400C	PORT_DATA	PORT Port x GPIO Data Register	0x00000000
0x3100400C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004010	PORT_DATA_SET	PORT Port x GPIO Data Set Register	0x00000000
0x31004010	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004014	PORT_DATA_CLR	PORT Port x GPIO Data Clear Register	0x00000000

Table A-147: ADSP-SC59x PORT MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31004014	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004018	PORT_DIR	PORT Port x GPIO Direction Register	0x00000000
0x31004018	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100401C	PORT_DIR_SET	PORT Port x GPIO Direction Set Register	0x00000000
0x3100401C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x31004020	PORT_DIR_CLR	PORT Port x GPIO Direction Clear Register	0x00000000
0x31004020	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004024	PORT_INEN	PORT Port x GPIO Input Enable Register	0x00000000
0x31004024	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x31004028	PORT_INEN_SET	PORT Port x GPIO Input Enable Set Register	0x00000000
0x31004028	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100402C	PORT_INEN_CLR	PORT Port x GPIO Input Enable Clear Register	0x00000000
0x3100402C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004030	PORT_MUX	PORT Port x Multiplexer Control Register	0x00000000
0x31004030	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004034	PORT_DATA_TGL	PORT Port x GPIO Output Toggle Register	0x00000000
0x31004034	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004038	PORT_POL	PORT Port x GPIO Polarity Invert Register	0x00000000
0x31004038	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100403C	PORT_POL_SET	PORT Port x GPIO Polarity Invert Set Register	0x00000000
0x3100403C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004040	PORT_POL_CLR	PORT Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004040	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004044	PORT_LOCK	PORT Port x GPIO Lock Register	0x00000000
0x31004044	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004048	PORT_TRIG_TGL	PORT Port x GPIO Trigger Toggle Register	0x00000000
0x31004048	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-148: ADSP-SC59x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004080	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004084	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004088	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100408C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004090	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004094	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004098	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100409C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x310040A0	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x310040A4	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x310040A8	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x310040AC	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x310040B0	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x310040B4	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x310040B8	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x310040BC	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x310040C0	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x310040C4	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x310040C8	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-149: ADSP-SC59x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004100	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004104	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004108	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100410C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004110	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004114	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004118	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100411C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000

Table A-149: ADSP-SC59x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31004120	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004124	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x31004128	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100412C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004130	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004134	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004138	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100413C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004140	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004144	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004148	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-150: ADSP-SC59x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004180	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004184	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004188	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100418C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004190	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004194	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004198	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100419C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x310041A0	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x310041A4	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x310041A8	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x310041AC	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x310041B0	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x310041B4	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x310041B8	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x310041BC	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000

Table A-150: ADSP-SC59x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310041C0	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x310041C4	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x310041C8	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-151: ADSP-SC59x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004200	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004204	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004208	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100420C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004210	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004214	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004218	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100421C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x31004220	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004224	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x31004228	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100422C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004230	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004234	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004238	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100423C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004240	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004244	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004248	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-152: ADSP-SC59x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004280	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004284	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000

Table A-152: ADSP-SC59x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31004288	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100428C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004290	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004294	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004298	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100429C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x310042A0	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x310042A4	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x310042A8	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x310042AC	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x310042B0	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x310042B4	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x310042B8	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x310042BC	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x310042C0	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x310042C4	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x310042C8	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-153: ADSP-SC59x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004300	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004304	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004308	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100430C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004310	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004314	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004318	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100431C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x31004320	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004324	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000

Table A-153: ADSP-SC59x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31004328	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100432C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004330	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004334	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004338	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100433C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004340	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004344	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004348	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-154: ADSP-SC59x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004380	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004384	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004388	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100438C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004390	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004394	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004398	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100439C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x310043A0	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x310043A4	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x310043A8	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x310043AC	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x310043B0	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x310043B4	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x310043B8	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x310043BC	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x310043C0	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x310043C4	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000

Table A-154: ADSP-SC59x PORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310043C8	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-155: ADSP-SC59x PORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31004400	PORT0_FER	PORT0 Port x Function Enable Register	0x00000000
0x31004404	PORT0_FER_SET	PORT0 Port x Function Enable Set Register	0x00000000
0x31004408	PORT0_FER_CLR	PORT0 Port x Function Enable Clear Register	0x00000000
0x3100440C	PORT0_DATA	PORT0 Port x GPIO Data Register	0x00000000
0x31004410	PORT0_DATA_SET	PORT0 Port x GPIO Data Set Register	0x00000000
0x31004414	PORT0_DATA_CLR	PORT0 Port x GPIO Data Clear Register	0x00000000
0x31004418	PORT0_DIR	PORT0 Port x GPIO Direction Register	0x00000000
0x3100441C	PORT0_DIR_SET	PORT0 Port x GPIO Direction Set Register	0x00000000
0x31004420	PORT0_DIR_CLR	PORT0 Port x GPIO Direction Clear Register	0x00000000
0x31004424	PORT0_INEN	PORT0 Port x GPIO Input Enable Register	0x00000000
0x31004428	PORT0_INEN_SET	PORT0 Port x GPIO Input Enable Set Register	0x00000000
0x3100442C	PORT0_INEN_CLR	PORT0 Port x GPIO Input Enable Clear Register	0x00000000
0x31004430	PORT0_MUX	PORT0 Port x Multiplexer Control Register	0x00000000
0x31004434	PORT0_DATA_TGL	PORT0 Port x GPIO Output Toggle Register	0x00000000
0x31004438	PORT0_POL	PORT0 Port x GPIO Polarity Invert Register	0x00000000
0x3100443C	PORT0_POL_SET	PORT0 Port x GPIO Polarity Invert Set Register	0x00000000
0x31004440	PORT0_POL_CLR	PORT0 Port x GPIO Polarity Invert Clear Register	0x00000000
0x31004444	PORT0_LOCK	PORT0 Port x GPIO Lock Register	0x00000000
0x31004448	PORT0_TRIG_TGL	PORT0 Port x GPIO Trigger Toggle Register	0x00000000

Table A-156: ADSP-SC59x RCU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108C000	RCU0_CTL	RCU0 Control Register	0x00000700
0x3108C004	RCU0_STAT	RCU0 Status Register	0x00000021
0x3108C008	RCU0_CRCTL	RCU0 Core Reset Outputs Control Register	0x00000006
0x3108C00C	RCU0_CRSTAT	RCU0 Core Reset Outputs Status Register	0x00000006

Table A-156: ADSP-SC59x RCU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108C018	RCU0_SRRQSTAT	RCU0 System Reset Request Status Register	0x00000000
0x3108C01C	RCU0_SIDIS	RCU0 System Interface Disable Register	0x00000000
0x3108C020	RCU0_SISTAT	RCU0 System Interface Status Register	0x00000000
0x3108C024	RCU0_SVECT_LCK	RCU0 SVECT Lock Register	0x00000000
0x3108C028	RCU0_BCODE	RCU0 Boot Code Register	0x00000000
0x3108C02C	RCU0_SVECT0	RCU0 Software Vector Register 0	0x00000040
0x3108C030	RCU0_SVECT1	RCU0 Software Vector Register 1	0x00500004
0x3108C034	RCU0_SVECT2	RCU0 Software Vector Register 2	0x00500004
0x3108C06C	RCU0_MSG	RCU0 Message Register	0x00000000
0x3108C070	RCU0_MSG_SET	RCU0 Message Set Bits Register	0x00000000
0x3108C074	RCU0_MSG_CLR	RCU0 Message Clear Bits Register	0x00000000

Table A-157: ADSP-SC59x ROMV70 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31110000	ROMV70_ROMENTRY0	ROMV70 ROM Table Entry Register 0	0x00001002
0x31110004	ROMV70_ROMENTRY1	ROMV70 ROM Table Entry Register 1	0x00002002
0x31110008	ROMV70_ROMENTRY2	ROMV70 ROM Table Entry Register 2	0x00004003
0x3111000C	ROMV70_ROMENTRY3	ROMV70 ROM Table Entry Register 3	0x00000002
0x31110010	ROMV70_ROMENTRY4	ROMV70 ROM Table Entry Register 4	0x00000002
0x31110014	ROMV70_ROMENTRY5	ROMV70 ROM Table Entry Register 5	0x00000002
0x31110018	ROMV70_ROMENTRY6	ROMV70 ROM Table Entry Register 6	0x00008002
0x3111001C	ROMV70_ROMENTRY7	ROMV70 ROM Table Entry Register 7	0x00000002
0x31110020	ROMV70_ROMENTRY8	ROMV70 ROM Table Entry Register 8	0x00000002
0x31110024	ROMV70_ROMENTRY9	ROMV70 ROM Table Entry Register 9	0x00000002
0x31110028	ROMV70_ROMENTRY10	ROMV70 ROM Table Entry Register 10	0x0000C003
0x3111002C	ROMV70_ROMENTRY11	ROMV70 ROM Table Entry Register 11	0x00000002
0x31110030	ROMV70_ROMENTRY12	ROMV70 ROM Table Entry Register 12	0x00000002
0x31110034	ROMV70_ROMENTRY13	ROMV70 ROM Table Entry Register 13	0x00000002
0x31110038	ROMV70_ROMENTRY14	ROMV70 ROM Table Entry Register 14	0x00014003
0x3111003C	ROMV70_ROMENTRY15	ROMV70 ROM Table Entry Register 15	0x00000002

Table A-157: ADSP-SC59x ROMV70 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31110040	ROMV70_ROMENTRY16	ROMV70 ROM Table Entry Register 16	0x00000002
0x31110044	ROMV70_ROMENTRY17	ROMV70 ROM Table Entry Register 17	0x00000002
0x31110048	ROMV70_ROMENTRY18	ROMV70 ROM Table Entry Register 18	0x0001C003
0x3111004C	ROMV70_ROMENTRY19	ROMV70 ROM Table Entry Register 19	0x00000002
0x31110050	ROMV70_ROMENTRY20	ROMV70 ROM Table Entry Register 20	0x00000002
0x31110054	ROMV70_ROMENTRY21	ROMV70 ROM Table Entry Register 21	0x00000002
0x31110058	ROMV70_ROMENTRY22	ROMV70 ROM Table Entry Register 22	0x00000002
0x3111005C	ROMV70_ROMENTRY23	ROMV70 ROM Table Entry Register 23	0x00000002
0x31110060	ROMV70_ROMENTRY24	ROMV70 ROM Table Entry Register 24	0x00000002
0x31110064	ROMV70_ROMENTRY25	ROMV70 ROM Table Entry Register 25	0x00000002
0x31110068	ROMV70_ROMENTRY26	ROMV70 ROM Table Entry Register 26	0x00000002
0x3111006C	ROMV70_ROMENTRY27	ROMV70 ROM Table Entry Register 27	0x00000002
0x31110070	ROMV70_ROMENTRY28	ROMV70 ROM Table Entry Register 28	0x00000002
0x31110074	ROMV70_ROMENTRY29	ROMV70 ROM Table Entry Register 29	0x00000002
0x31110078	ROMV70_ROMENTRY30	ROMV70 ROM Table Entry Register 30	0x00000002
0x3111007C	ROMV70_ROMENTRY31	ROMV70 ROM Table Entry Register 31	0x00000002
0x31110080	ROMV70_ROMENTRY32	ROMV70 ROM Table Entry Register 32	0x00000002
0x31110084	ROMV70_ROMENTRY33	ROMV70 ROM Table Entry Register 33	0x00000002
0x31110088	ROMV70_ROMENTRY34	ROMV70 ROM Table Entry Register 34	0x00000002
0x3111008C	ROMV70_ROMENTRY35	ROMV70 ROM Table Entry Register 35	0x00000002
0x31110090	ROMV70_ROMENTRY36	ROMV70 ROM Table Entry Register 36	0x00000002
0x31110094	ROMV70_ROMENTRY37	ROMV70 ROM Table Entry Register 37	0x00000002
0x31110098	ROMV70_ROMENTRY38	ROMV70 ROM Table Entry Register 38	0x00000002
0x3111009C	ROMV70_ROMENTRY39	ROMV70 ROM Table Entry Register 39	0x00000002
0x311100A0	ROMV70_ROMENTRY40	ROMV70 ROM Table Entry Register 40	0x00000002
0x311100A4	ROMV70_ROMENTRY41	ROMV70 ROM Table Entry Register 41	0x00000002
0x311100A8	ROMV70_ROMENTRY42	ROMV70 ROM Table Entry Register 42	0x00000002
0x311100AC	ROMV70_ROMENTRY43	ROMV70 ROM Table Entry Register 43	0x00000002
0x311100B0	ROMV70_ROMENTRY44	ROMV70 ROM Table Entry Register 44	0x00000002
0x311100B4	ROMV70_ROMENTRY45	ROMV70 ROM Table Entry Register 45	0x00000002
0x311100B8	ROMV70_ROMENTRY46	ROMV70 ROM Table Entry Register 46	0x00000002

Table A-157: ADSP-SC59x ROMV70 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x311100BC	ROMV70_ROMENTRY47	ROMV70 ROM Table Entry Register 47	0x00000002
0x311100C0	ROMV70_ROMENTRY48	ROMV70 ROM Table Entry Register 48	0x00000002
0x311100C4	ROMV70_ROMENTRY49	ROMV70 ROM Table Entry Register 49	0x00000002
0x311100C8	ROMV70_ROMENTRY50	ROMV70 ROM Table Entry Register 50	0x00000002
0x311100CC	ROMV70_ROMENTRY51	ROMV70 ROM Table Entry Register 51	0x00000002
0x311100D0	ROMV70_ROMENTRY52	ROMV70 ROM Table Entry Register 52	0x00000002
0x311100D4	ROMV70_ROMENTRY53	ROMV70 ROM Table Entry Register 53	0x00000002
0x311100D8	ROMV70_ROMENTRY54	ROMV70 ROM Table Entry Register 54	0x00000002
0x311100DC	ROMV70_ROMENTRY55	ROMV70 ROM Table Entry Register 55	0x00000002
0x311100E0	ROMV70_ROMENTRY56	ROMV70 ROM Table Entry Register 56	0x00000002
0x311100E4	ROMV70_ROMENTRY57	ROMV70 ROM Table Entry Register 57	0x00000002
0x311100E8	ROMV70_ROMENTRY58	ROMV70 ROM Table Entry Register 58	0x00000002
0x311100EC	ROMV70_ROMENTRY59	ROMV70 ROM Table Entry Register 59	0x00000002
0x311100F0	ROMV70_ROMENTRY60	ROMV70 ROM Table Entry Register 60	0x00000002
0x311100F4	ROMV70_ROMENTRY61	ROMV70 ROM Table Entry Register 61	0x00000002
0x311100F8	ROMV70_ROMENTRY62	ROMV70 ROM Table Entry Register 62	0x00000002
0x311100FC	ROMV70_ROMENTRY63	ROMV70 ROM Table Entry Register 63	0x00000002
0x31110100	ROMV70_ROMENTRY64	ROMV70 ROM Table Entry Register 64	0x00000002
0x31110104	ROMV70_ROMENTRY65	ROMV70 ROM Table Entry Register 65	0x00000002
0x31110108	ROMV70_ROMENTRY66	ROMV70 ROM Table Entry Register 66	0x00000002
0x3111010C	ROMV70_ROMENTRY67	ROMV70 ROM Table Entry Register 67	0x00000002
0x31110110	ROMV70_ROMENTRY68	ROMV70 ROM Table Entry Register 68	0x00000002
0x31110114	ROMV70_ROMENTRY69	ROMV70 ROM Table Entry Register 69	0x00000002
0x31110118	ROMV70_ROMENTRY70	ROMV70 ROM Table Entry Register 70	0x00000002
0x3111011C	ROMV70_ROMENTRY71	ROMV70 ROM Table Entry Register 71	0x00000002
0x31110120	ROMV70_ROMENTRY72	ROMV70 ROM Table Entry Register 72	0x00000002
0x31110124	ROMV70_ROMENTRY73	ROMV70 ROM Table Entry Register 73	0x00000002
0x31110128	ROMV70_ROMENTRY74	ROMV70 ROM Table Entry Register 74	0x00000002
0x3111012C	ROMV70_ROMENTRY75	ROMV70 ROM Table Entry Register 75	0x00000002
0x31110130	ROMV70_ROMENTRY76	ROMV70 ROM Table Entry Register 76	0x00000002
0x31110134	ROMV70_ROMENTRY77	ROMV70 ROM Table Entry Register 77	0x00000002

Table A-157: ADSP-SC59x ROMV70 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31110138	ROMV70_ROMENTRY78	ROMV70 ROM Table Entry Register 78	0x00000002
0x3111013C	ROMV70_ROMENTRY79	ROMV70 ROM Table Entry Register 79	0x00000002
0x31110140	ROMV70_ROMENTRY80	ROMV70 ROM Table Entry Register 80	0x00000002
0x31110144	ROMV70_ROMENTRY81	ROMV70 ROM Table Entry Register 81	0x00000002
0x31110A00	ROMV70_DBGPCR0	ROMV70 Debug Power Control Registers 0	0x00000000
0x31110A04	ROMV70_DBGPCR1	ROMV70 Debug Power Control Registers 1	0x00000000
0x31110A08	ROMV70_DBGPCR2	ROMV70 Debug Power Control Registers 2	0x00000000
0x31110A0C	ROMV70_DBGPCR3	ROMV70 Debug Power Control Registers 3	0x00000000
0x31110A10	ROMV70_DBGPCR4	ROMV70 Debug Power Control Registers 4	0x00000000
0x31110A14	ROMV70_DBGPCR5	ROMV70 Debug Power Control Registers 5	0x00000000
0x31110A18	ROMV70_DBGPCR6	ROMV70 Debug Power Control Registers 6	0x00000000
0x31110A1C	ROMV70_DBGPCR7	ROMV70 Debug Power Control Registers 7	0x00000000
0x31110A20	ROMV70_DBGPCR8	ROMV70 Debug Power Control Registers 8	0x00000000
0x31110A80	ROMV70_DBGPSR0	ROMV70 Debug Power Status Registers 0	0x00000000
0x31110A84	ROMV70_DBGPSR1	ROMV70 Debug Power Status Registers 1	0x00000000
0x31110A88	ROMV70_DBGPSR2	ROMV70 Debug Power Status Registers 2	0x00000000
0x31110A8C	ROMV70_DBGPSR3	ROMV70 Debug Power Status Registers 3	0x00000000
0x31110A90	ROMV70_DBGPSR4	ROMV70 Debug Power Status Registers 4	0x00000000
0x31110A94	ROMV70_DBGPSR5	ROMV70 Debug Power Status Registers 5	0x00000000
0x31110A98	ROMV70_DBGPSR6	ROMV70 Debug Power Status Registers 6	0x00000000
0x31110A9C	ROMV70_DBGPSR7	ROMV70 Debug Power Status Registers 7	0x00000000
0x31110AA0	ROMV70_DBGPSR8	ROMV70 Debug Power Status Registers 8	0x00000000
0x31110C00	ROMV70_PRIDR0	ROMV70 Power Reset Identification Register 0	0x00000001
0x31110FB8	ROMV70_AUTHSTATUS	ROMV70 Authentication Status Register	0x00000008
0x31110FBC	ROMV70_DEVARCH	ROMV70 Device Architecture Register	0x47700AF7
0x31110FC8	ROMV70_DEVID	ROMV70 Device ID Register	0x00000020
0x31110FCC	ROMV70_DEVTYPE	ROMV70 Device Type Register	0x00000000
0x31110FD0	ROMV70_PIDR4	ROMV70 Peripheral Identification Register 4	0x00000004
0x31110FD4	ROMV70_PIDR5	ROMV70 Peripheral Identification Register 5	0x00000000
0x31110FD8	ROMV70_PIDR6	ROMV70 Peripheral Identification Register 6	0x00000000
0x31110FDC	ROMV70_PIDR7	ROMV70 Peripheral Identification Register 7	0x00000000

Table A-157: ADSP-SC59x ROMV70 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31110FE0	ROMV70_PIDR0	ROMV70 Peripheral Identification Register 0	0x000000E3
0x31110FE4	ROMV70_PIDR1	ROMV70 Peripheral Identification Register 1	0x000000B4
0x31110FE8	ROMV70_PIDR2	ROMV70 Peripheral Identification Register 2	0x0000007B
0x31110FEC	ROMV70_PIDR3	ROMV70 Peripheral Identification Register 3	0x00000000
0x31110FF0	ROMV70_CIDR0	ROMV70 Component Identification Register 0	0x0000000D
0x31110FF4	ROMV70_CIDR1	ROMV70 Component Identification Register 1	0x00000090
0x31110FF8	ROMV70_CIDR2	ROMV70 Component Identification Register 2	0x00000005
0x31110FFC	ROMV70_CIDR3	ROMV70 Component Identification Register 3	0x000000B1

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31089000	SEC0_GCTL	SEC0 Global Control Register	0x00000000
0x31089004	SEC0_GSTAT	SEC0 Global Status Register	0x00000000
0x31089008	SEC0_RAISE	SEC0 Global Raise Register	0x00000000
0x3108900C	SEC0_END	SEC0 Global End Register	0x00000000
0x31089010	SEC0_FCTL	SEC0 Fault Control Register	0x00000000
0x31089014	SEC0_FSTAT	SEC0 Fault Status Register	0x00000000
0x31089018	SEC0_FSID	SEC0 Fault Source ID Register	0x00000000
0x3108901C	SEC0_FEND	SEC0 Fault End Register	0x00000000
0x31089020	SEC0_FDLY	SEC0 Fault Delay Register	0x00000000
0x31089024	SEC0_FDLY_CUR	SEC0 Fault Delay Current Register	0x00000000
0x31089028	SEC0_FSRDLY	SEC0 Fault System Reset Delay Register	0x00000000
0x3108902C	SEC0_FSRDLY_CUR	SEC0 Fault System Reset Delay Current Register	0x00000000
0x31089030	SEC0_FCOPP	SEC0 Fault COP Period Register	0x00000000
0x31089034	SEC0_FCOPP_CUR	SEC0 Fault COP Period Current Register	0x00000000
0x31089440	SEC0_CCTL[n]	SEC0 SCI Control Register n	0x00000000
0x31089444	SEC0_CSTAT[n]	SEC0 SCI Status Register n	0x00000000
0x31089448	SEC0_CPND[n]	SEC0 Core Pending Register n	0x00000000
0x3108944C	SEC0_CACT[n]	SEC0 SCI Active Register n	0x00000000
0x31089450	SEC0_CPMSK[n]	SEC0 SCI Priority Mask Register n	0x000000FF

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089454	SEC0_CGMSK[n]	SEC0 SCI Group Mask Register n	0x00000000
0x31089458	SEC0_CPLVL[n]	SEC0 SCI Priority Level Register n	0x00000007
0x3108945C	SEC0_CSID[n]	SEC0 SCI Source ID Register n	0x00000000
0x31089480	SEC0_CCTL[n]	SEC0 SCI Control Register n	0x00000000
0x31089484	SEC0_CSTAT[n]	SEC0 SCI Status Register n	0x00000000
0x31089488	SEC0_CPND[n]	SEC0 Core Pending Register n	0x00000000
0x3108948C	SEC0_CACT[n]	SEC0 SCI Active Register n	0x00000000
0x31089490	SEC0_CPMSK[n]	SEC0 SCI Priority Mask Register n	0x000000FF
0x31089494	SEC0_CGMSK[n]	SEC0 SCI Group Mask Register n	0x00000000
0x31089498	SEC0_CPLVL[n]	SEC0 SCI Priority Level Register n	0x00000007
0x3108949C	SEC0_CSID[n]	SEC0 SCI Source ID Register n	0x00000000
0x31089800	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089804	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089808	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108980C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089810	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089814	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089818	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108981C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089820	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089824	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089828	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108982C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089830	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089834	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089838	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108983C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089840	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089844	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089848	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108984C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089850	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089854	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089858	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108985C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089860	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089864	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089868	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108986C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089870	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089874	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089878	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108987C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089880	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089884	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089888	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108988C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089890	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089894	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089898	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108989C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898A0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898A4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898A8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898AC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898B0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898B4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898B8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898BC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898C0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898C4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898C8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310898CC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898D0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898D4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898D8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898DC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898E0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898E4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898E8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898EC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898F0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898F4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310898F8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310898FC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089900	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089904	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089908	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108990C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089910	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089914	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089918	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108991C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089920	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089924	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089928	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108992C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089930	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089934	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089938	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108993C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089940	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089944	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089948	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108994C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089950	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089954	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089958	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108995C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089960	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089964	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089968	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108996C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089970	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089974	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089978	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108997C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089980	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089984	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089988	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108998C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089990	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089994	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089998	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x3108999C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899A0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899A4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899A8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899AC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899B0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899B4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899B8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899BC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899C0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310899C4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899C8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899CC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899D0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899D4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899D8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899DC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899E0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899E4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899E8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899EC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899F0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899F4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x310899F8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x310899FC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089A40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089A98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089A9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089ABC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089ACC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089ADC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089AF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089AFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089B38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089B98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089B9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089BB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BCC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089BF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089BFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089C30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089C98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089C9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089CAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CCC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089CF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089CFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089D28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089D98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089D9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089DA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DCC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089DF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089DFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089E20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089E94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089E98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089E9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089ECC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089ED0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089ED4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089ED8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089EF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089EFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F00	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F04	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F08	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F0C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F10	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F14	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089F18	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F1C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F20	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F24	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F28	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F2C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F30	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F34	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F38	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F3C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F40	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F44	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F48	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F4C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F50	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F54	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F58	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F5C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F60	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F64	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F68	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F6C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F70	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F74	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F78	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F7C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F80	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F84	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F88	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F8C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F90	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000

Table A-158: ADSP-SC59x SEC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31089F94	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089F98	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089F9C	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FA0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FA4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FA8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FAC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FB0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FB4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FB8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FBC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FC0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FC4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FC8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FCC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FD0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FD4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FD8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FDC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FE0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FE4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FE8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FEC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FF0	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FF4	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000
0x31089FF8	SEC0_SCTL[n]	SEC0 Source Control Register n	0x00000000
0x31089FFC	SEC0_SSTAT[n]	SEC0 Source Status Register n	0x00000000

Table A-159: ADSP-SC59x SMC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3100600C	SMC0_B0CTL	SMC0 Bank 0 Control Register	0x01000000
0x31006010	SMC0_B0TIM	SMC0 Bank 0 Timing Register	0x01010101
0x31006014	SMC0_B0ETIM	SMC0 Bank 0 Extended Timing Register	0x00020200
0x3100601C	SMC0_B1CTL	SMC0 Bank 1 Control Register	0x01000000
0x31006020	SMC0_B1TIM	SMC0 Bank 1 Timing Register	0x01010101
0x31006024	SMC0_B1ETIM	SMC0 Bank 1 Extended Timing Register	0x00020200
0x3100602C	SMC0_B2CTL	SMC0 Bank 2 Control Register	0x01000000
0x31006030	SMC0_B2TIM	SMC0 Bank 2 Timing Register	0x01010101
0x31006034	SMC0_B2ETIM	SMC0 Bank 2 Extended Timing Register	0x00020200
0x3100603C	SMC0_B3CTL	SMC0 Bank 3 Control Register	0x01000000
0x31006040	SMC0_B3TIM	SMC0 Bank 3 Timing Register	0x01010101
0x31006044	SMC0_B3ETIM	SMC0 Bank 3 Extended Timing Register	0x00020200

Table A-160: ADSP-SC59x SMPU MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31007000	SMPU0_CTL	SMPU0 SMPU Control Register	0x00000000
0x31007004	SMPU0_STAT	SMPU0 SMPU Status Register	0x00000000
0x31007008	SMPU0_IADDR	SMPU0 Interrupt Address Register	0x00000000
0x3100700C	SMPU0_IDTLS	SMPU0 Interrupt Details Register	0x00000000
0x31007010	SMPU0_BADDR	SMPU0 Bus Error Address Register	0x00000000
0x31007014	SMPU0_BDTLS	SMPU0 Bus Error Details Register	0x00000000
0x31007020	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x31007024	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x31007028	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x3100702C	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007030	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x31007034	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007038	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x3100703C	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x31007040	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000

Table A-160: ADSP-SC59x SMPU MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31007044	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007048	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x3100704C	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007050	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x31007054	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x31007058	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x3100705C	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007060	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x31007064	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007068	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x3100706C	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x31007070	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x31007074	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007078	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x3100707C	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007080	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x31007084	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x31007088	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x3100708C	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x31007090	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x31007094	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007098	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x3100709C	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x310070A0	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x310070A4	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x310070A8	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x310070AC	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x310070B0	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x310070B4	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x310070B8	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x310070BC	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000

Table A-160: ADSP-SC59x SMPU MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310070C0	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x310070C4	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x310070C8	SMPU0_RCTL[n]	SMPU0 Region n Control Register	0x00000000
0x310070CC	SMPU0_RADDR[n]	SMPU0 Region n Address Register	0x00000000
0x310070D0	SMPU0_RIDA[n]	SMPU0 Region n ID A Register	0x00000000
0x310070D4	SMPU0_RIDMSKA[n]	SMPU0 Region n ID Mask A Register	0x00000000
0x310070D8	SMPU0_RIDB[n]	SMPU0 Region n ID B Register	0x00000000
0x310070DC	SMPU0_RIDMSKB[n]	SMPU0 Region n ID Mask B Register	0x00000000
0x31007220	SMPU0_REVID	SMPU0 SMPU Revision ID Register	0x00000010
0x31007800	SMPU0_SECURECTL	SMPU0 SMPU Control Secure Accesses Register	0x00000000
0x31007820	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007824	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007828	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x3100782C	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007830	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007834	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31007838	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x3100783C	SMPU0_SECURERCTL[n]	SMPU0 Region n Control Secure Accesses Register	0x00000000
0x31012000	SMPU_CTL	SMPU SMPU Control Register	0x00000000
0x31012004	SMPU_STAT	SMPU SMPU Status Register	0x00000000
0x31012008	SMPU_IADDR	SMPU Interrupt Address Register	0x00000000
0x3101200C	SMPU_IDTLS	SMPU Interrupt Details Register	0x00000000
0x31012010	SMPU_BADDR	SMPU Bus Error Address Register	0x00000000
0x31012014	SMPU_BDTLS	SMPU Bus Error Details Register	0x00000000
0x31012020	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x31012024	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012028	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x3101202C	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012030	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x31012034	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x31012038	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000

Table A-160: ADSP-SC59x SMPU MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3101203C	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012040	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x31012044	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012048	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x3101204C	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x31012050	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x31012054	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012058	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x3101205C	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012060	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x31012064	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x31012068	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x3101206C	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012070	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x31012074	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012078	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x3101207C	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x31012080	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x31012084	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x31012088	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x3101208C	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x31012090	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x31012094	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x31012098	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x3101209C	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x310120A0	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x310120A4	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x310120A8	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x310120AC	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x310120B0	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x310120B4	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000

Table A-160: ADSP-SC59x SMPU MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310120B8	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x310120BC	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x310120C0	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x310120C4	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x310120C8	SMPU_RCTL[n]	SMPU Region n Control Register	0x00000000
0x310120CC	SMPU_RADDR[n]	SMPU Region n Address Register	0x00000000
0x310120D0	SMPU_RIDA[n]	SMPU Region n ID A Register	0x00000000
0x310120D4	SMPU_RIDMSKA[n]	SMPU Region n ID Mask A Register	0x00000000
0x310120D8	SMPU_RIDB[n]	SMPU Region n ID B Register	0x00000000
0x310120DC	SMPU_RIDMSKB[n]	SMPU Region n ID Mask B Register	0x00000000
0x310121A0	SMPU_EXACADD[n]	SMPU Exclusive Access IDn Address	0x00000000
0x310121A4	SMPU_EXACSTAT[n]	SMPU Exclusive Access Status	0x00000000
0x310121A8	SMPU_EXACADD[n]	SMPU Exclusive Access IDn Address	0x00000000
0x310121AC	SMPU_EXACSTAT[n]	SMPU Exclusive Access Status	0x00000000
0x310121B0	SMPU_EXACADD[n]	SMPU Exclusive Access IDn Address	0x00000000
0x310121B4	SMPU_EXACSTAT[n]	SMPU Exclusive Access Status	0x00000000
0x31012220	SMPU_REVID	SMPU SMPU Revision ID Register	0x00000010
0x31012800	SMPU_SECURECTL	SMPU SMPU Control Secure Accesses Register	0x00000000
0x31012820	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012824	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012828	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x3101282C	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012830	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012834	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x31012838	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000
0x3101283C	SMPU_SECURERCTL[n]	SMPU Region n Control Secure Accesses Register	0x00000000

Table A-161: ADSP-SC59x SMPU11 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A1000	SMPU11_CTL	SMPU11 SMPU Control Register	0x00000000

Table A-161: ADSP-SC59x SMPU11 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A1004	SMPU11_STAT	SMPU11 SMPU Status Register	0x00000000
0x310A1008	SMPU11_IADDR	SMPU11 Interrupt Address Register	0x00000000
0x310A100C	SMPU11_IDTLS	SMPU11 Interrupt Details Register	0x00000000
0x310A1010	SMPU11_BADDR	SMPU11 Bus Error Address Register	0x00000000
0x310A1014	SMPU11_BDTLS	SMPU11 Bus Error Details Register	0x00000000
0x310A1020	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A1024	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A1028	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A102C	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1030	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A1034	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1038	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A103C	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A1040	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A1044	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1048	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A104C	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1050	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A1054	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A1058	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A105C	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1060	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A1064	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1068	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A106C	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A1070	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A1074	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1078	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A107C	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1080	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A1084	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000

Table A-161: ADSP-SC59x SMPU11 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A1088	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A108C	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A1090	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A1094	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1098	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A109C	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A10A0	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A10A4	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A10A8	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A10AC	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A10B0	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A10B4	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A10B8	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A10BC	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A10C0	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A10C4	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A10C8	SMPU11_RCTL[n]	SMPU11 Region n Control Register	0x00000000
0x310A10CC	SMPU11_RADDR[n]	SMPU11 Region n Address Register	0x00000000
0x310A10D0	SMPU11_RIDA[n]	SMPU11 Region n ID A Register	0x00000000
0x310A10D4	SMPU11_RIDMSKA[n]	SMPU11 Region n ID Mask A Register	0x00000000
0x310A10D8	SMPU11_RIDB[n]	SMPU11 Region n ID B Register	0x00000000
0x310A10DC	SMPU11_RIDMSKB[n]	SMPU11 Region n ID Mask B Register	0x00000000
0x310A1220	SMPU11_REVID	SMPU11 SMPU Revision ID Register	0x00000010
0x310A1800	SMPU11_SECURECTL	SMPU11 SMPU Control Secure Accesses Register	0x00000000
0x310A1820	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1824	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1828	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A182C	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1830	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1834	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000
0x310A1838	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000

Table A-161: ADSP-SC59x SMPU11 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A183C	SMPU11_SECURERCTL[n]	SMPU11 Region n Control Secure Accesses Register	0x00000000

Table A-162: ADSP-SC59x SMPU12 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31012000	SMPU12_CTL	SMPU12 SMPU Control Register	0x00000000
0x31012004	SMPU12_STAT	SMPU12 SMPU Status Register	0x00000000
0x31012008	SMPU12_IADDR	SMPU12 Interrupt Address Register	0x00000000
0x3101200C	SMPU12_IDTLS	SMPU12 Interrupt Details Register	0x00000000
0x31012010	SMPU12_BADDR	SMPU12 Bus Error Address Register	0x00000000
0x31012014	SMPU12_BDTLS	SMPU12 Bus Error Details Register	0x00000000
0x31012020	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x31012024	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x31012028	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x3101202C	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012030	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x31012034	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012038	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x3101203C	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x31012040	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x31012044	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012048	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x3101204C	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012050	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x31012054	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x31012058	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x3101205C	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012060	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x31012064	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012068	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x3101206C	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000

Table A-162: ADSP-SC59x SMPU12 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31012070	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x31012074	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012078	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x3101207C	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012080	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x31012084	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x31012088	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x3101208C	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x31012090	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x31012094	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012098	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x3101209C	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x310120A0	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x310120A4	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x310120A8	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x310120AC	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x310120B0	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x310120B4	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x310120B8	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x310120BC	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x310120C0	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x310120C4	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x310120C8	SMPU12_RCTL[n]	SMPU12 Region n Control Register	0x00000000
0x310120CC	SMPU12_RADDR[n]	SMPU12 Region n Address Register	0x00000000
0x310120D0	SMPU12_RIDA[n]	SMPU12 Region n ID A Register	0x00000000
0x310120D4	SMPU12_RIDMSKA[n]	SMPU12 Region n ID Mask A Register	0x00000000
0x310120D8	SMPU12_RIDB[n]	SMPU12 Region n ID B Register	0x00000000
0x310120DC	SMPU12_RIDMSKB[n]	SMPU12 Region n ID Mask B Register	0x00000000
0x31012220	SMPU12_REVID	SMPU12 SMPU Revision ID Register	0x00000010
0x31012800	SMPU12_SECURECTL	SMPU12 SMPU Control Secure Accesses Register	0x00000000
0x31012820	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000

Table A-162: ADSP-SC59x SMPU12 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31012824	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x31012828	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x3101282C	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x31012830	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x31012834	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x31012838	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000
0x3101283C	SMPU12_SECURERCTL[n]	SMPU12 Region n Control Secure Accesses Register	0x00000000

Table A-163: ADSP-SC59x SMPU2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31083000	SMPU2_CTL	SMPU2 SMPU Control Register	0x00000000
0x31083004	SMPU2_STAT	SMPU2 SMPU Status Register	0x00000000
0x31083008	SMPU2_IADDR	SMPU2 Interrupt Address Register	0x00000000
0x3108300C	SMPU2_IDTLS	SMPU2 Interrupt Details Register	0x00000000
0x31083010	SMPU2_BADDR	SMPU2 Bus Error Address Register	0x00000000
0x31083014	SMPU2_BDTLS	SMPU2 Bus Error Details Register	0x00000000
0x31083020	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x31083024	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x31083028	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x3108302C	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083030	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x31083034	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083038	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x3108303C	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x31083040	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x31083044	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083048	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x3108304C	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083050	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x31083054	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000

Table A-163: ADSP-SC59x SMPU2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31083058	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x3108305C	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083060	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x31083064	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083068	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x3108306C	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x31083070	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x31083074	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083078	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x3108307C	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083080	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x31083084	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x31083088	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x3108308C	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x31083090	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x31083094	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x31083098	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x3108309C	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x310830A0	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x310830A4	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x310830A8	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x310830AC	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x310830B0	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x310830B4	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x310830B8	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000
0x310830BC	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x310830C0	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x310830C4	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x310830C8	SMPU2_RCTL[n]	SMPU2 Region n Control Register	0x00000000
0x310830CC	SMPU2_RADDR[n]	SMPU2 Region n Address Register	0x00000000
0x310830D0	SMPU2_RIDA[n]	SMPU2 Region n ID A Register	0x00000000

Table A-163: ADSP-SC59x SMPU2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310830D4	SMPU2_RIDMSKA[n]	SMPU2 Region n ID Mask A Register	0x00000000
0x310830D8	SMPU2_RIDB[n]	SMPU2 Region n ID B Register	0x00000000
0x310830DC	SMPU2_RIDMSKB[n]	SMPU2 Region n ID Mask B Register	0x00000000
0x310831A0	SMPU2_EXACADD[n]	SMPU2 Exclusive Access IDn Address	0x00000000
0x310831A4	SMPU2_EXACSTAT[n]	SMPU2 Exclusive Access Status	0x00000000
0x310831A8	SMPU2_EXACADD[n]	SMPU2 Exclusive Access IDn Address	0x00000000
0x310831AC	SMPU2_EXACSTAT[n]	SMPU2 Exclusive Access Status	0x00000000
0x310831B0	SMPU2_EXACADD[n]	SMPU2 Exclusive Access IDn Address	0x00000000
0x310831B4	SMPU2_EXACSTAT[n]	SMPU2 Exclusive Access Status	0x00000000
0x31083220	SMPU2_REVID	SMPU2 SMPU Revision ID Register	0x00000010
0x31083800	SMPU2_SECURECTL	SMPU2 SMPU Control Secure Accesses Register	0x00000000
0x31083820	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083824	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083828	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x3108382C	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083830	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083834	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x31083838	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000
0x3108383C	SMPU2_SECURERCTL[n]	SMPU2 Region n Control Secure Accesses Register	0x00000000

Table A-164: ADSP-SC59x SMPU3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31084000	SMPU3_CTL	SMPU3 SMPU Control Register	0x00000000
0x31084004	SMPU3_STAT	SMPU3 SMPU Status Register	0x00000000
0x31084008	SMPU3_IADDR	SMPU3 Interrupt Address Register	0x00000000
0x3108400C	SMPU3_IDTLS	SMPU3 Interrupt Details Register	0x00000000
0x31084010	SMPU3_BADDR	SMPU3 Bus Error Address Register	0x00000000
0x31084014	SMPU3_BDTLS	SMPU3 Bus Error Details Register	0x00000000
0x31084020	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x31084024	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000

Table A-164: ADSP-SC59x SMPU3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31084028	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x3108402C	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084030	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x31084034	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084038	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x3108403C	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x31084040	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x31084044	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084048	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x3108404C	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084050	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x31084054	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x31084058	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x3108405C	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084060	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x31084064	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084068	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x3108406C	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x31084070	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x31084074	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084078	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x3108407C	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084080	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x31084084	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x31084088	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x3108408C	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x31084090	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x31084094	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084098	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x3108409C	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x310840A0	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000

Table A-164: ADSP-SC59x SMPU3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310840A4	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x310840A8	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x310840AC	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x310840B0	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x310840B4	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x310840B8	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x310840BC	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x310840C0	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x310840C4	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x310840C8	SMPU3_RCTL[n]	SMPU3 Region n Control Register	0x00000000
0x310840CC	SMPU3_RADDR[n]	SMPU3 Region n Address Register	0x00000000
0x310840D0	SMPU3_RIDA[n]	SMPU3 Region n ID A Register	0x00000000
0x310840D4	SMPU3_RIDMSKA[n]	SMPU3 Region n ID Mask A Register	0x00000000
0x310840D8	SMPU3_RIDB[n]	SMPU3 Region n ID B Register	0x00000000
0x310840DC	SMPU3_RIDMSKB[n]	SMPU3 Region n ID Mask B Register	0x00000000
0x31084220	SMPU3_REVID	SMPU3 SMPU Revision ID Register	0x00000010
0x31084800	SMPU3_SECURECTL	SMPU3 SMPU Control Secure Accesses Register	0x00000000
0x31084820	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084824	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084828	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x3108482C	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084830	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084834	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x31084838	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000
0x3108483C	SMPU3_SECURERCTL[n]	SMPU3 Region n Control Secure Accesses Register	0x00000000

Table A-165: ADSP-SC59x SMPU4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31085000	SMPU4_CTL	SMPU4 SMPU Control Register	0x00000000
0x31085004	SMPU4_STAT	SMPU4 SMPU Status Register	0x00000000

Table A-165: ADSP-SC59x SMPU4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31085008	SMPU4_IADDR	SMPU4 Interrupt Address Register	0x00000000
0x3108500C	SMPU4_IDTLS	SMPU4 Interrupt Details Register	0x00000000
0x31085010	SMPU4_BADDR	SMPU4 Bus Error Address Register	0x00000000
0x31085014	SMPU4_BDTLS	SMPU4 Bus Error Details Register	0x00000000
0x31085020	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x31085024	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085028	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x3108502C	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085030	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x31085034	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085038	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x3108503C	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085040	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x31085044	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085048	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x3108504C	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085050	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x31085054	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085058	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x3108505C	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085060	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x31085064	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085068	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x3108506C	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085070	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x31085074	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085078	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x3108507C	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085080	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x31085084	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x31085088	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000

Table A-165: ADSP-SC59x SMPU4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108508C	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x31085090	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x31085094	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x31085098	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x3108509C	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x310850A0	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x310850A4	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x310850A8	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x310850AC	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x310850B0	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x310850B4	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x310850B8	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x310850BC	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x310850C0	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x310850C4	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x310850C8	SMPU4_RCTL[n]	SMPU4 Region n Control Register	0x00000000
0x310850CC	SMPU4_RADDR[n]	SMPU4 Region n Address Register	0x00000000
0x310850D0	SMPU4_RIDA[n]	SMPU4 Region n ID A Register	0x00000000
0x310850D4	SMPU4_RIDMSKA[n]	SMPU4 Region n ID Mask A Register	0x00000000
0x310850D8	SMPU4_RIDB[n]	SMPU4 Region n ID B Register	0x00000000
0x310850DC	SMPU4_RIDMSKB[n]	SMPU4 Region n ID Mask B Register	0x00000000
0x310851A0	SMPU4_EXACADD[n]	SMPU4 Exclusive Access IDn Address	0x00000000
0x310851A4	SMPU4_EXACSTAT[n]	SMPU4 Exclusive Access Status	0x00000000
0x310851A8	SMPU4_EXACADD[n]	SMPU4 Exclusive Access IDn Address	0x00000000
0x310851AC	SMPU4_EXACSTAT[n]	SMPU4 Exclusive Access Status	0x00000000
0x310851B0	SMPU4_EXACADD[n]	SMPU4 Exclusive Access IDn Address	0x00000000
0x310851B4	SMPU4_EXACSTAT[n]	SMPU4 Exclusive Access Status	0x00000000
0x31085220	SMPU4_REVID	SMPU4 SMPU Revision ID Register	0x00000010
0x31085800	SMPU4_SECURECTL	SMPU4 SMPU Control Secure Accesses Register	0x00000000
0x31085820	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x31085824	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000

Table A-165: ADSP-SC59x SMPU4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31085828	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x3108582C	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x31085830	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x31085834	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x31085838	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000
0x3108583C	SMPU4_SECURERCTL[n]	SMPU4 Region n Control Secure Accesses Register	0x00000000

Table A-166: ADSP-SC59x SMPU5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31086000	SMPU5_CTL	SMPU5 SMPU Control Register	0x00000000
0x31086004	SMPU5_STAT	SMPU5 SMPU Status Register	0x00000000
0x31086008	SMPU5_IADDR	SMPU5 Interrupt Address Register	0x00000000
0x3108600C	SMPU5_IDTLS	SMPU5 Interrupt Details Register	0x00000000
0x31086010	SMPU5_BADDR	SMPU5 Bus Error Address Register	0x00000000
0x31086014	SMPU5_BDTLS	SMPU5 Bus Error Details Register	0x00000000
0x31086020	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x31086024	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x31086028	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x3108602C	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086030	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x31086034	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086038	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x3108603C	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x31086040	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x31086044	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086048	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x3108604C	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086050	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x31086054	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x31086058	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000

Table A-166: ADSP-SC59x SMPU5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108605C	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086060	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x31086064	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086068	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x3108606C	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x31086070	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x31086074	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086078	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x3108607C	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086080	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x31086084	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x31086088	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x3108608C	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x31086090	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x31086094	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086098	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x3108609C	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x310860A0	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x310860A4	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x310860A8	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x310860AC	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x310860B0	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x310860B4	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x310860B8	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x310860BC	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000
0x310860C0	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x310860C4	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x310860C8	SMPU5_RCTL[n]	SMPU5 Region n Control Register	0x00000000
0x310860CC	SMPU5_RADDR[n]	SMPU5 Region n Address Register	0x00000000
0x310860D0	SMPU5_RIDA[n]	SMPU5 Region n ID A Register	0x00000000
0x310860D4	SMPU5_RIDMSKA[n]	SMPU5 Region n ID Mask A Register	0x00000000

Table A-166: ADSP-SC59x SMPU5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310860D8	SMPU5_RIDB[n]	SMPU5 Region n ID B Register	0x00000000
0x310860DC	SMPU5_RIDMSKB[n]	SMPU5 Region n ID Mask B Register	0x00000000
0x31086220	SMPU5_REVID	SMPU5 SMPU Revision ID Register	0x00000010
0x31086800	SMPU5_SECURECTL	SMPU5 SMPU Control Secure Accesses Register	0x00000000
0x31086820	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x31086824	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x31086828	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x3108682C	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x31086830	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x31086834	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x31086838	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000
0x3108683C	SMPU5_SECURERCTL[n]	SMPU5 Region n Control Secure Accesses Register	0x00000000

Table A-167: ADSP-SC59x SMPU6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31087000	SMPU6_CTL	SMPU6 SMPU Control Register	0x00000000
0x31087004	SMPU6_STAT	SMPU6 SMPU Status Register	0x00000000
0x31087008	SMPU6_IADDR	SMPU6 Interrupt Address Register	0x00000000
0x3108700C	SMPU6_IDTLS	SMPU6 Interrupt Details Register	0x00000000
0x31087010	SMPU6_BADDR	SMPU6 Bus Error Address Register	0x00000000
0x31087014	SMPU6_BDTLS	SMPU6 Bus Error Details Register	0x00000000
0x31087020	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x31087024	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x31087028	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x3108702C	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087030	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x31087034	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087038	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x3108703C	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x31087040	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000

Table A-167: ADSP-SC59x SMPU6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31087044	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087048	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x3108704C	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087050	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x31087054	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x31087058	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x3108705C	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087060	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x31087064	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087068	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x3108706C	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x31087070	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x31087074	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087078	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x3108707C	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087080	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x31087084	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x31087088	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x3108708C	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x31087090	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x31087094	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087098	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x3108709C	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x310870A0	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x310870A4	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x310870A8	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x310870AC	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x310870B0	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x310870B4	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x310870B8	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x310870BC	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000

Table A-167: ADSP-SC59x SMPU6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310870C0	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x310870C4	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x310870C8	SMPU6_RCTL[n]	SMPU6 Region n Control Register	0x00000000
0x310870CC	SMPU6_RADDR[n]	SMPU6 Region n Address Register	0x00000000
0x310870D0	SMPU6_RIDA[n]	SMPU6 Region n ID A Register	0x00000000
0x310870D4	SMPU6_RIDMSKA[n]	SMPU6 Region n ID Mask A Register	0x00000000
0x310870D8	SMPU6_RIDB[n]	SMPU6 Region n ID B Register	0x00000000
0x310870DC	SMPU6_RIDMSKB[n]	SMPU6 Region n ID Mask B Register	0x00000000
0x31087220	SMPU6_REVID	SMPU6 SMPU Revision ID Register	0x00000010
0x31087800	SMPU6_SECURECTL	SMPU6 SMPU Control Secure Accesses Register	0x00000000
0x31087820	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087824	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087828	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x3108782C	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087830	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087834	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x31087838	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000
0x3108783C	SMPU6_SECURERCTL[n]	SMPU6 Region n Control Secure Accesses Register	0x00000000

Table A-168: ADSP-SC59x SMPU9 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A0000	SMPU9_CTL	SMPU9 SMPU Control Register	0x00000000
0x310A0004	SMPU9_STAT	SMPU9 SMPU Status Register	0x00000000
0x310A0008	SMPU9_IADDR	SMPU9 Interrupt Address Register	0x00000000
0x310A000C	SMPU9_IDTLS	SMPU9 Interrupt Details Register	0x00000000
0x310A0010	SMPU9_BADDR	SMPU9 Bus Error Address Register	0x00000000
0x310A0014	SMPU9_BDTLS	SMPU9 Bus Error Details Register	0x00000000
0x310A0020	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A0024	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A0028	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000

Table A-168: ADSP-SC59x SMPU9 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A002C	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0030	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A0034	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0038	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A003C	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A0040	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A0044	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0048	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A004C	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0050	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A0054	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A0058	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A005C	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0060	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A0064	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0068	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A006C	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A0070	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A0074	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0078	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A007C	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0080	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A0084	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A0088	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A008C	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A0090	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A0094	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A0098	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A009C	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A00A0	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A00A4	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000

Table A-168: ADSP-SC59x SMPU9 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A00A8	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A00AC	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A00B0	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A00B4	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A00B8	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A00BC	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A00C0	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A00C4	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A00C8	SMPU9_RCTL[n]	SMPU9 Region n Control Register	0x00000000
0x310A00CC	SMPU9_RADDR[n]	SMPU9 Region n Address Register	0x00000000
0x310A00D0	SMPU9_RIDA[n]	SMPU9 Region n ID A Register	0x00000000
0x310A00D4	SMPU9_RIDMSKA[n]	SMPU9 Region n ID Mask A Register	0x00000000
0x310A00D8	SMPU9_RIDB[n]	SMPU9 Region n ID B Register	0x00000000
0x310A00DC	SMPU9_RIDMSKB[n]	SMPU9 Region n ID Mask B Register	0x00000000
0x310A01A0	SMPU9_EXACADD[n]	SMPU9 Exclusive Access IDn Address	0x00000000
0x310A01A4	SMPU9_EXACSTAT[n]	SMPU9 Exclusive Access Status	0x00000000
0x310A01A8	SMPU9_EXACADD[n]	SMPU9 Exclusive Access IDn Address	0x00000000
0x310A01AC	SMPU9_EXACSTAT[n]	SMPU9 Exclusive Access Status	0x00000000
0x310A01B0	SMPU9_EXACADD[n]	SMPU9 Exclusive Access IDn Address	0x00000000
0x310A01B4	SMPU9_EXACSTAT[n]	SMPU9 Exclusive Access Status	0x00000000
0x310A0220	SMPU9_REVID	SMPU9 SMPU Revision ID Register	0x00000010
0x310A0800	SMPU9_SECURECTL	SMPU9 SMPU Control Secure Accesses Register	0x00000000
0x310A0820	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0824	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0828	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A082C	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0830	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0834	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A0838	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000
0x310A083C	SMPU9_SECURERCTL[n]	SMPU9 Region n Control Secure Accesses Register	0x00000000

Table A-169: ADSP-SC59x SPDIF0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C9280	SPDIF0_TX_CTL	SPDIF0 Transmit Control Register	0x00000000
0x310C9280	SPDIF_TX_CTL	SPDIF Transmit Control Register	0x00000000
0x310C9284	SPDIF0_TX_STAT_A0	SPDIF0 Transmit Status A0 Register	0x00000000
0x310C9284	SPDIF_TX_STAT_A0	SPDIF Transmit Status A0 Register	0x00000000
0x310C9288	SPDIF0_TX_STAT_B0	SPDIF0 Transmit Status B0 Register	0x00000000
0x310C9288	SPDIF_TX_STAT_B0	SPDIF Transmit Status B0 Register	0x00000000
0x310C92A0	SPDIF0_RX_CTL	SPDIF0 Receive Control	0x00000018
0x310C92A0	SPDIF_RX_CTL	SPDIF Receive Control	0x00000018
0x310C92A4	SPDIF0_RX_STAT	SPDIF0 Receive Status Register	0x00000000
0x310C92A4	SPDIF_RX_STAT	SPDIF Receive Status Register	0x00000000
0x310C92A8	SPDIF0_RX_STAT0_A	SPDIF0 Receive Status A0 Register	0x00000000
0x310C92A8	SPDIF_RX_STAT0_A	SPDIF Receive Status A0 Register	0x00000000
0x310C92AC	SPDIF0_RX_STAT0_B	SPDIF0 Receive Status B0 Register	0x00000000
0x310C92AC	SPDIF_RX_STAT0_B	SPDIF Receive Status B0 Register	0x00000000
0x310C92B0	SPDIF0_RX_STAT1_A	SPDIF0 Receive Status A1 Register	0x00000000
0x310C92B0	SPDIF_RX_STAT1_A	SPDIF Receive Status A1 Register	0x00000000
0x310C92B4	SPDIF0_RX_STAT1_B	SPDIF0 Receive Status B1 Register	0x00000000
0x310C92B4	SPDIF_RX_STAT1_B	SPDIF Receive Status B1 Register	0x00000000
0x310C9350	SPDIF0_TX_STAT_A1	SPDIF0 Transmit Status A1 Register	0x00000000
0x310C9350	SPDIF_TX_STAT_A1	SPDIF Transmit Status A1 Register	0x00000000
0x310C9354	SPDIF0_TX_STAT_A2	SPDIF0 Transmit Status A2 Register	0x00000000
0x310C9354	SPDIF_TX_STAT_A2	SPDIF Transmit Status A2 Register	0x00000000
0x310C9358	SPDIF0_TX_STAT_A3	SPDIF0 Transmit Status A3 Register	0x00000000
0x310C9358	SPDIF_TX_STAT_A3	SPDIF Transmit Status A3 Register	0x00000000
0x310C935C	SPDIF0_TX_STAT_A4	SPDIF0 Transmit Status A4 Register	0x00000000
0x310C935C	SPDIF_TX_STAT_A4	SPDIF Transmit Status A4 Register	0x00000000
0x310C9360	SPDIF0_TX_STAT_A5	SPDIF0 Transmit Status A5 Register	0x00000000
0x310C9360	SPDIF_TX_STAT_A5	SPDIF Transmit Status A5 Register	0x00000000
0x310C9368	SPDIF0_TX_STAT_B1	SPDIF0 Transmit Status B1 Register	0x00000000
0x310C9368	SPDIF_TX_STAT_B1	SPDIF Transmit Status B1 Register	0x00000000
0x310C936C	SPDIF0_TX_STAT_B2	SPDIF0 Transmit Status B2 Register	0x00000000

Table A-169: ADSP-SC59x SPDIF0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C936C	SPDIF_TX_STAT_B2	SPDIF Transmit Status B2 Register	0x00000000
0x310C9370	SPDIF0_TX_STAT_B3	SPDIF0 Transmit Status B3 Register	0x00000000
0x310C9370	SPDIF_TX_STAT_B3	SPDIF Transmit Status B3 Register	0x00000000
0x310C9374	SPDIF0_TX_STAT_B4	SPDIF0 Transmit Status B4 Register	0x00000000
0x310C9374	SPDIF_TX_STAT_B4	SPDIF Transmit Status B4 Register	0x00000000
0x310C9378	SPDIF0_TX_STAT_B5	SPDIF0 Transmit Status B5 Register	0x00000000
0x310C9378	SPDIF_TX_STAT_B5	SPDIF Transmit Status B5 Register	0x00000000
0x310C9380	SPDIF0_TX_UBUFF_A0	SPDIF0 Transmit User Buffer A0 Register	0x00000000
0x310C9380	SPDIF_TX_UBUFF_A0	SPDIF Transmit User Buffer A0 Register	0x00000000
0x310C9384	SPDIF0_TX_UBUFF_A1	SPDIF0 Transmit User Buffer A1 Register	0x00000000
0x310C9384	SPDIF_TX_UBUFF_A1	SPDIF Transmit User Buffer A1 Register	0x00000000
0x310C9388	SPDIF0_TX_UBUFF_A2	SPDIF0 Transmit User Buffer A2 Register	0x00000000
0x310C9388	SPDIF_TX_UBUFF_A2	SPDIF Transmit User Buffer A2 Register	0x00000000
0x310C938C	SPDIF0_TX_UBUFF_A3	SPDIF0 Transmit User Buffer A3 Register	0x00000000
0x310C938C	SPDIF_TX_UBUFF_A3	SPDIF Transmit User Buffer A3 Register	0x00000000
0x310C9390	SPDIF0_TX_UBUFF_A4	SPDIF0 Transmit User Buffer A4 Register	0x00000000
0x310C9390	SPDIF_TX_UBUFF_A4	SPDIF Transmit User Buffer A4 Register	0x00000000
0x310C9394	SPDIF0_TX_UBUFF_A5	SPDIF0 Transmit User Buffer A5 Register	0x00000000
0x310C9394	SPDIF_TX_UBUFF_A5	SPDIF Transmit User Buffer A5 Register	0x00000000
0x310C93A0	SPDIF0_TX_UBUFF_B0	SPDIF0 Transmit User Buffer B0 Register	0x00000000
0x310C93A0	SPDIF_TX_UBUFF_B0	SPDIF Transmit User Buffer B0 Register	0x00000000
0x310C93A4	SPDIF0_TX_UBUFF_B1	SPDIF0 Transmit User Buffer B1 Register	0x00000000
0x310C93A4	SPDIF_TX_UBUFF_B1	SPDIF Transmit User Buffer B1 Register	0x00000000
0x310C93A8	SPDIF0_TX_UBUFF_B2	SPDIF0 Transmit User Buffer B2 Register	0x00000000
0x310C93A8	SPDIF_TX_UBUFF_B2	SPDIF Transmit User Buffer B2 Register	0x00000000
0x310C93AC	SPDIF0_TX_UBUFF_B3	SPDIF0 Transmit User Buffer B3 Register	0x00000000
0x310C93AC	SPDIF_TX_UBUFF_B3	SPDIF Transmit User Buffer B3 Register	0x00000000
0x310C93B0	SPDIF0_TX_UBUFF_B4	SPDIF0 Transmit User Buffer B4 Register	0x00000000
0x310C93B0	SPDIF_TX_UBUFF_B4	SPDIF Transmit User Buffer B4 Register	0x00000000
0x310C93B4	SPDIF0_TX_UBUFF_B5	SPDIF0 Transmit User Buffer B5 Register	0x00000000
0x310C93B4	SPDIF_TX_UBUFF_B5	SPDIF Transmit User Buffer B5 Register	0x00000000

Table A-169: ADSP-SC59x SPDIF0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C93BC	SPDIF0_TX_USRUPDT	SPDIF0 User Bit Update Register	0x00000000
0x310C93BC	SPDIF_TX_USRUPDT	SPDIF User Bit Update Register	0x00000000

Table A-170: ADSP-SC59x SPDIF1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA280	SPDIF1_TX_CTL	SPDIF1 Transmit Control Register	0x00000000
0x310CA284	SPDIF1_TX_STAT_A0	SPDIF1 Transmit Status A0 Register	0x00000000
0x310CA288	SPDIF1_TX_STAT_B0	SPDIF1 Transmit Status B0 Register	0x00000000
0x310CA2A0	SPDIF1_RX_CTL	SPDIF1 Receive Control	0x00000018
0x310CA2A4	SPDIF1_RX_STAT	SPDIF1 Receive Status Register	0x00000000
0x310CA2A8	SPDIF1_RX_STAT0_A	SPDIF1 Receive Status A0 Register	0x00000000
0x310CA2AC	SPDIF1_RX_STAT0_B	SPDIF1 Receive Status B0 Register	0x00000000
0x310CA2B0	SPDIF1_RX_STAT1_A	SPDIF1 Receive Status A1 Register	0x00000000
0x310CA2B4	SPDIF1_RX_STAT1_B	SPDIF1 Receive Status B1 Register	0x00000000
0x310CA350	SPDIF1_TX_STAT_A1	SPDIF1 Transmit Status A1 Register	0x00000000
0x310CA354	SPDIF1_TX_STAT_A2	SPDIF1 Transmit Status A2 Register	0x00000000
0x310CA358	SPDIF1_TX_STAT_A3	SPDIF1 Transmit Status A3 Register	0x00000000
0x310CA35C	SPDIF1_TX_STAT_A4	SPDIF1 Transmit Status A4 Register	0x00000000
0x310CA360	SPDIF1_TX_STAT_A5	SPDIF1 Transmit Status A5 Register	0x00000000
0x310CA368	SPDIF1_TX_STAT_B1	SPDIF1 Transmit Status B1 Register	0x00000000
0x310CA36C	SPDIF1_TX_STAT_B2	SPDIF1 Transmit Status B2 Register	0x00000000
0x310CA370	SPDIF1_TX_STAT_B3	SPDIF1 Transmit Status B3 Register	0x00000000
0x310CA374	SPDIF1_TX_STAT_B4	SPDIF1 Transmit Status B4 Register	0x00000000
0x310CA378	SPDIF1_TX_STAT_B5	SPDIF1 Transmit Status B5 Register	0x00000000
0x310CA380	SPDIF1_TX_UBUFF_A0	SPDIF1 Transmit User Buffer A0 Register	0x00000000
0x310CA384	SPDIF1_TX_UBUFF_A1	SPDIF1 Transmit User Buffer A1 Register	0x00000000
0x310CA388	SPDIF1_TX_UBUFF_A2	SPDIF1 Transmit User Buffer A2 Register	0x00000000
0x310CA38C	SPDIF1_TX_UBUFF_A3	SPDIF1 Transmit User Buffer A3 Register	0x00000000
0x310CA390	SPDIF1_TX_UBUFF_A4	SPDIF1 Transmit User Buffer A4 Register	0x00000000
0x310CA394	SPDIF1_TX_UBUFF_A5	SPDIF1 Transmit User Buffer A5 Register	0x00000000

Table A-170: ADSP-SC59x SPDIF1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310CA3A0	SPDIF1_TX_UBUFF_B0	SPDIF1 Transmit User Buffer B0 Register	0x00000000
0x310CA3A4	SPDIF1_TX_UBUFF_B1	SPDIF1 Transmit User Buffer B1 Register	0x00000000
0x310CA3A8	SPDIF1_TX_UBUFF_B2	SPDIF1 Transmit User Buffer B2 Register	0x00000000
0x310CA3AC	SPDIF1_TX_UBUFF_B3	SPDIF1 Transmit User Buffer B3 Register	0x00000000
0x310CA3B0	SPDIF1_TX_UBUFF_B4	SPDIF1 Transmit User Buffer B4 Register	0x00000000
0x310CA3B4	SPDIF1_TX_UBUFF_B5	SPDIF1 Transmit User Buffer B5 Register	0x00000000
0x310CA3BC	SPDIF1_TX_USRUPDT	SPDIF1 User Bit Update Register	0x00000000

Table A-171: ADSP-SC59x SPI MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102E004	SPI_CTL	SPI Control Register	0x00000050
0x3102E004	SPI0_CTL	SPI0 Control Register	0x00000050
0x3102E008	SPI_RXCTL	SPI Receive Control Register	0x00000000
0x3102E008	SPI0_RXCTL	SPI0 Receive Control Register	0x00000000
0x3102E00C	SPI_TXCTL	SPI Transmit Control Register	0x00000000
0x3102E00C	SPI0_TXCTL	SPI0 Transmit Control Register	0x00000000
0x3102E010	SPI_CLK	SPI Clock Rate Register	0x00000000
0x3102E010	SPI0_CLK	SPI0 Clock Rate Register	0x00000000
0x3102E014	SPI_DLY	SPI Delay Register	0x00000301
0x3102E014	SPI0_DLY	SPI0 Delay Register	0x00000301
0x3102E018	SPI_SLVSEL	SPI Slave Select Register	0x0000FE00
0x3102E018	SPI0_SLVSEL	SPI0 Slave Select Register	0x0000FE00
0x3102E01C	SPI_RWC	SPI Received Word Count Register	0x00000000
0x3102E01C	SPI0_RWC	SPI0 Received Word Count Register	0x00000000
0x3102E020	SPI_RWCR	SPI Received Word Count Reload Register	0x00000000
0x3102E020	SPI0_RWCR	SPI0 Received Word Count Reload Register	0x00000000
0x3102E024	SPI_TWC	SPI Transmitted Word Count Register	0x00000000
0x3102E024	SPI0_TWC	SPI0 Transmitted Word Count Register	0x00000000
0x3102E028	SPI_TWCR	SPI Transmitted Word Count Reload Register	0x00000000
0x3102E028	SPI0_TWCR	SPI0 Transmitted Word Count Reload Register	0x00000000

Table A-171: ADSP-SC59x SPI MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102E030	SPI_IMSK	SPI Interrupt Mask Register	0x00000000
0x3102E030	SPI0_IMSK	SPI0 Interrupt Mask Register	0x00000000
0x3102E034	SPI_IMSK_CLR	SPI Interrupt Mask Clear Register	0x00000000
0x3102E034	SPI0_IMSK_CLR	SPI0 Interrupt Mask Clear Register	0x00000000
0x3102E038	SPI_IMSK_SET	SPI Interrupt Mask Set Register	0x00000000
0x3102E038	SPI0_IMSK_SET	SPI0 Interrupt Mask Set Register	0x00000000
0x3102E040	SPI_STAT	SPI Status Register	0x00440001
0x3102E040	SPI0_STAT	SPI0 Status Register	0x00440001
0x3102E044	SPI_ILAT	SPI Masked Interrupt Condition Register	0x00000000
0x3102E044	SPI0_ILAT	SPI0 Masked Interrupt Condition Register	0x00000000
0x3102E048	SPI_ILAT_CLR	SPI Masked Interrupt Clear Register	0x00000000
0x3102E048	SPI0_ILAT_CLR	SPI0 Masked Interrupt Clear Register	0x00000000
0x3102E050	SPI_RFIFO	SPI Receive FIFO Data Register	0x00000000
0x3102E050	SPI0_RFIFO	SPI0 Receive FIFO Data Register	0x00000000
0x3102E058	SPI_TFIFO	SPI Transmit FIFO Data Register	0x00000000
0x3102E058	SPI0_TFIFO	SPI0 Transmit FIFO Data Register	0x00000000
0x3102E060	SPI_MMRDH	SPI Memory Mapped Read Header	0x00000000
0x3102E064	SPI_MMTOP	SPI SPI Memory Top Address	0x00000000

Table A-172: ADSP-SC59x SPI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3102F004	SPI1_CTL	SPI1 Control Register	0x00000050
0x3102F008	SPI1_RXCTL	SPI1 Receive Control Register	0x00000000
0x3102F00C	SPI1_TXCTL	SPI1 Transmit Control Register	0x00000000
0x3102F010	SPI1_CLK	SPI1 Clock Rate Register	0x00000000
0x3102F014	SPI1_DLY	SPI1 Delay Register	0x00000301
0x3102F018	SPI1_SLVSEL	SPI1 Slave Select Register	0x0000FE00
0x3102F01C	SPI1_RWC	SPI1 Received Word Count Register	0x00000000
0x3102F020	SPI1_RWCR	SPI1 Received Word Count Reload Register	0x00000000
0x3102F024	SPI1_TWC	SPI1 Transmitted Word Count Register	0x00000000

Table A-172: ADSP-SC59x SPI1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3102F028	SPI1_TWCR	SPI1 Transmitted Word Count Reload Register	0x00000000
0x3102F030	SPI1_IMSK	SPI1 Interrupt Mask Register	0x00000000
0x3102F034	SPI1_IMSK_CLR	SPI1 Interrupt Mask Clear Register	0x00000000
0x3102F038	SPI1_IMSK_SET	SPI1 Interrupt Mask Set Register	0x00000000
0x3102F040	SPI1_STAT	SPI1 Status Register	0x00440001
0x3102F044	SPI1_ILAT	SPI1 Masked Interrupt Condition Register	0x00000000
0x3102F048	SPI1_ILAT_CLR	SPI1 Masked Interrupt Clear Register	0x00000000
0x3102F050	SPI1_RFIFO	SPI1 Receive FIFO Data Register	0x00000000
0x3102F058	SPI1_TFIFO	SPI1 Transmit FIFO Data Register	0x00000000

Table A-173: ADSP-SC59x SPI2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31030004	SPI2_CTL	SPI2 Control Register	0x00000050
0x31030008	SPI2_RXCTL	SPI2 Receive Control Register	0x00000000
0x3103000C	SPI2_TXCTL	SPI2 Transmit Control Register	0x00000000
0x31030010	SPI2_CLK	SPI2 Clock Rate Register	0x00000000
0x31030014	SPI2_DLY	SPI2 Delay Register	0x00000301
0x31030018	SPI2_SLVSEL	SPI2 Slave Select Register	0x0000FE00
0x3103001C	SPI2_RWC	SPI2 Received Word Count Register	0x00000000
0x31030020	SPI2_RWCR	SPI2 Received Word Count Reload Register	0x00000000
0x31030024	SPI2_TWC	SPI2 Transmitted Word Count Register	0x00000000
0x31030028	SPI2_TWCR	SPI2 Transmitted Word Count Reload Register	0x00000000
0x31030030	SPI2_IMSK	SPI2 Interrupt Mask Register	0x00000000
0x31030034	SPI2_IMSK_CLR	SPI2 Interrupt Mask Clear Register	0x00000000
0x31030038	SPI2_IMSK_SET	SPI2 Interrupt Mask Set Register	0x00000000
0x31030040	SPI2_STAT	SPI2 Status Register	0x00440001
0x31030044	SPI2_ILAT	SPI2 Masked Interrupt Condition Register	0x00000000
0x31030048	SPI2_ILAT_CLR	SPI2 Masked Interrupt Clear Register	0x00000000
0x31030050	SPI2_RFIFO	SPI2 Receive FIFO Data Register	0x00000000
0x31030058	SPI2_TFIFO	SPI2 Transmit FIFO Data Register	0x00000000

Table A-173: ADSP-SC59x SPI2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31030060	SPI2_MMRDH	SPI2 Memory Mapped Read Header	0x00000000
0x31030064	SPI2_MMTOP	SPI2 SPI Memory Top Address	0x00000000

Table A-174: ADSP-SC59x SPI3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31031004	SPI3_CTL	SPI3 Control Register	0x00000050
0x31031008	SPI3_RXCTL	SPI3 Receive Control Register	0x00000000
0x3103100C	SPI3_TXCTL	SPI3 Transmit Control Register	0x00000000
0x31031010	SPI3_CLK	SPI3 Clock Rate Register	0x00000000
0x31031014	SPI3_DLY	SPI3 Delay Register	0x00000301
0x31031018	SPI3_SLVSEL	SPI3 Slave Select Register	0x0000FE00
0x3103101C	SPI3_RWC	SPI3 Received Word Count Register	0x00000000
0x31031020	SPI3_RWCR	SPI3 Received Word Count Reload Register	0x00000000
0x31031024	SPI3_TWC	SPI3 Transmitted Word Count Register	0x00000000
0x31031028	SPI3_TWCR	SPI3 Transmitted Word Count Reload Register	0x00000000
0x31031030	SPI3_IMSK	SPI3 Interrupt Mask Register	0x00000000
0x31031034	SPI3_IMSK_CLR	SPI3 Interrupt Mask Clear Register	0x00000000
0x31031038	SPI3_IMSK_SET	SPI3 Interrupt Mask Set Register	0x00000000
0x31031040	SPI3_STAT	SPI3 Status Register	0x00440001
0x31031044	SPI3_ILAT	SPI3 Masked Interrupt Condition Register	0x00000000
0x31031048	SPI3_ILAT_CLR	SPI3 Masked Interrupt Clear Register	0x00000000
0x31031050	SPI3_RFIFO	SPI3 Receive FIFO Data Register	0x00000000
0x31031058	SPI3_TFIFO	SPI3 Transmit FIFO Data Register	0x00000000

Table A-175: ADSP-SC59x SPORT0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002000	SPORT0_CTL_A	SPORT0 Half SPORT 'A' Control Register	0x00000000
0x31002004	SPORT0_DIV_A	SPORT0 Half SPORT 'A' Divisor Register	0x00000000
0x31002008	SPORT0_MCTL_A	SPORT0 Half SPORT 'A' Multichannel Control Register	0x00000000

Table A-175: ADSP-SC59x SPORT0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3100200C	SPORT0_CS0_A	SPORT0 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002010	SPORT0_CS1_A	SPORT0 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002014	SPORT0_CS2_A	SPORT0 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002018	SPORT0_CS3_A	SPORT0 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002020	SPORT0_ERR_A	SPORT0 Half SPORT 'A' Error Register	0x00000000
0x31002024	SPORT0_MSTAT_A	SPORT0 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002028	SPORT0_CTL2_A	SPORT0 Half SPORT 'A' Control 2 Register	0x00000000
0x31002040	SPORT0_TXPRI_A	SPORT0 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002044	SPORT0_RXPRI_A	SPORT0 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002048	SPORT0_TXSEC_A	SPORT0 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100204C	SPORT0_RXSEC_A	SPORT0 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002080	SPORT0_CTL_B	SPORT0 Half SPORT 'B' Control Register	0x00000000
0x31002084	SPORT0_DIV_B	SPORT0 Half SPORT 'B' Divisor Register	0x00000000
0x31002088	SPORT0_MCTL_B	SPORT0 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100208C	SPORT0_CS0_B	SPORT0 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002090	SPORT0_CS1_B	SPORT0 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002094	SPORT0_CS2_B	SPORT0 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002098	SPORT0_CS3_B	SPORT0 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310020A0	SPORT0_ERR_B	SPORT0 Half SPORT 'B' Error Register	0x00000000
0x310020A4	SPORT0_MSTAT_B	SPORT0 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310020A8	SPORT0_CTL2_B	SPORT0 Half SPORT 'B' Control 2 Register	0x00000000
0x310020C0	SPORT0_TXPRI_B	SPORT0 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310020C4	SPORT0_RXPRI_B	SPORT0 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310020C8	SPORT0_TXSEC_B	SPORT0 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310020CC	SPORT0_RXSEC_B	SPORT0 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-176: ADSP-SC59x SPORT1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002100	SPORT1_CTL_A	SPORT1 Half SPORT 'A' Control Register	0x00000000
0x31002104	SPORT1_DIV_A	SPORT1 Half SPORT 'A' Divisor Register	0x00000000
0x31002108	SPORT1_MCTL_A	SPORT1 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100210C	SPORT1_CS0_A	SPORT1 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002110	SPORT1_CS1_A	SPORT1 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002114	SPORT1_CS2_A	SPORT1 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002118	SPORT1_CS3_A	SPORT1 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002120	SPORT1_ERR_A	SPORT1 Half SPORT 'A' Error Register	0x00000000
0x31002124	SPORT1_MSTAT_A	SPORT1 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002128	SPORT1_CTL2_A	SPORT1 Half SPORT 'A' Control 2 Register	0x00000000
0x31002140	SPORT1_TXPRI_A	SPORT1 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002144	SPORT1_RXPRI_A	SPORT1 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002148	SPORT1_TXSEC_A	SPORT1 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100214C	SPORT1_RXSEC_A	SPORT1 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002180	SPORT1_CTL_B	SPORT1 Half SPORT 'B' Control Register	0x00000000
0x31002184	SPORT1_DIV_B	SPORT1 Half SPORT 'B' Divisor Register	0x00000000
0x31002188	SPORT1_MCTL_B	SPORT1 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100218C	SPORT1_CS0_B	SPORT1 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002190	SPORT1_CS1_B	SPORT1 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002194	SPORT1_CS2_B	SPORT1 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002198	SPORT1_CS3_B	SPORT1 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310021A0	SPORT1_ERR_B	SPORT1 Half SPORT 'B' Error Register	0x00000000
0x310021A4	SPORT1_MSTAT_B	SPORT1 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310021A8	SPORT1_CTL2_B	SPORT1 Half SPORT 'B' Control 2 Register	0x00000000
0x310021C0	SPORT1_TXPRI_B	SPORT1 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000

Table A-176: ADSP-SC59x SPORT1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310021C4	SPORT1_RXPRI_B	SPORT1 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310021C8	SPORT1_TXSEC_B	SPORT1 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310021CC	SPORT1_RXSEC_B	SPORT1 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-177: ADSP-SC59x SPORT2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002200	SPORT2_CTL_A	SPORT2 Half SPORT 'A' Control Register	0x00000000
0x31002204	SPORT2_DIV_A	SPORT2 Half SPORT 'A' Divisor Register	0x00000000
0x31002208	SPORT2_MCTL_A	SPORT2 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100220C	SPORT2_CS0_A	SPORT2 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002210	SPORT2_CS1_A	SPORT2 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002214	SPORT2_CS2_A	SPORT2 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002218	SPORT2_CS3_A	SPORT2 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002220	SPORT2_ERR_A	SPORT2 Half SPORT 'A' Error Register	0x00000000
0x31002224	SPORT2_MSTAT_A	SPORT2 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002228	SPORT2_CTL2_A	SPORT2 Half SPORT 'A' Control 2 Register	0x00000000
0x31002240	SPORT2_TXPRI_A	SPORT2 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002244	SPORT2_RXPRI_A	SPORT2 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002248	SPORT2_TXSEC_A	SPORT2 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100224C	SPORT2_RXSEC_A	SPORT2 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002280	SPORT2_CTL_B	SPORT2 Half SPORT 'B' Control Register	0x00000000
0x31002284	SPORT2_DIV_B	SPORT2 Half SPORT 'B' Divisor Register	0x00000000
0x31002288	SPORT2_MCTL_B	SPORT2 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100228C	SPORT2_CS0_B	SPORT2 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002290	SPORT2_CS1_B	SPORT2 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002294	SPORT2_CS2_B	SPORT2 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000

Table A-177: ADSP-SC59x SPORT2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002298	SPORT2_CS3_B	SPORT2 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310022A0	SPORT2_ERR_B	SPORT2 Half SPORT 'B' Error Register	0x00000000
0x310022A4	SPORT2_MSTAT_B	SPORT2 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310022A8	SPORT2_CTL2_B	SPORT2 Half SPORT 'B' Control 2 Register	0x00000000
0x310022C0	SPORT2_TXPRI_B	SPORT2 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310022C4	SPORT2_RXPRI_B	SPORT2 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310022C8	SPORT2_TXSEC_B	SPORT2 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310022CC	SPORT2_RXSEC_B	SPORT2 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-178: ADSP-SC59x SPORT3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002300	SPORT3_CTL_A	SPORT3 Half SPORT 'A' Control Register	0x00000000
0x31002304	SPORT3_DIV_A	SPORT3 Half SPORT 'A' Divisor Register	0x00000000
0x31002308	SPORT3_MCTL_A	SPORT3 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100230C	SPORT3_CS0_A	SPORT3 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002310	SPORT3_CS1_A	SPORT3 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002314	SPORT3_CS2_A	SPORT3 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002318	SPORT3_CS3_A	SPORT3 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002320	SPORT3_ERR_A	SPORT3 Half SPORT 'A' Error Register	0x00000000
0x31002324	SPORT3_MSTAT_A	SPORT3 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002328	SPORT3_CTL2_A	SPORT3 Half SPORT 'A' Control 2 Register	0x00000000
0x31002340	SPORT3_TXPRI_A	SPORT3 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002344	SPORT3_RXPRI_A	SPORT3 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002348	SPORT3_TXSEC_A	SPORT3 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100234C	SPORT3_RXSEC_A	SPORT3 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002380	SPORT3_CTL_B	SPORT3 Half SPORT 'B' Control Register	0x00000000
0x31002384	SPORT3_DIV_B	SPORT3 Half SPORT 'B' Divisor Register	0x00000000

Table A-178: ADSP-SC59x SPORT3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002388	SPORT3_MCTL_B	SPORT3 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100238C	SPORT3_CS0_B	SPORT3 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002390	SPORT3_CS1_B	SPORT3 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002394	SPORT3_CS2_B	SPORT3 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002398	SPORT3_CS3_B	SPORT3 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310023A0	SPORT3_ERR_B	SPORT3 Half SPORT 'B' Error Register	0x00000000
0x310023A4	SPORT3_MSTAT_B	SPORT3 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310023A8	SPORT3_CTL2_B	SPORT3 Half SPORT 'B' Control 2 Register	0x00000000
0x310023C0	SPORT3_TXPRI_B	SPORT3 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310023C4	SPORT3_RXPRI_B	SPORT3 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310023C8	SPORT3_TXSEC_B	SPORT3 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310023CC	SPORT3_RXSEC_B	SPORT3 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-179: ADSP-SC59x SPORT4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002400	SPORT4_CTL_A	SPORT4 Half SPORT 'A' Control Register	0x00000000
0x31002404	SPORT4_DIV_A	SPORT4 Half SPORT 'A' Divisor Register	0x00000000
0x31002408	SPORT4_MCTL_A	SPORT4 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100240C	SPORT4_CS0_A	SPORT4 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002410	SPORT4_CS1_A	SPORT4 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002414	SPORT4_CS2_A	SPORT4 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002418	SPORT4_CS3_A	SPORT4 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002420	SPORT4_ERR_A	SPORT4 Half SPORT 'A' Error Register	0x00000000
0x31002424	SPORT4_MSTAT_A	SPORT4 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002428	SPORT4_CTL2_A	SPORT4 Half SPORT 'A' Control 2 Register	0x00000000

Table A-179: ADSP-SC59x SPORT4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002440	SPORT4_TXPRI_A	SPORT4 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002444	SPORT4_RXPRI_A	SPORT4 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002448	SPORT4_TXSEC_A	SPORT4 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100244C	SPORT4_RXSEC_A	SPORT4 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002480	SPORT4_CTL_B	SPORT4 Half SPORT 'B' Control Register	0x00000000
0x31002484	SPORT4_DIV_B	SPORT4 Half SPORT 'B' Divisor Register	0x00000000
0x31002488	SPORT4_MCTL_B	SPORT4 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100248C	SPORT4_CS0_B	SPORT4 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002490	SPORT4_CS1_B	SPORT4 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002494	SPORT4_CS2_B	SPORT4 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002498	SPORT4_CS3_B	SPORT4 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310024A0	SPORT4_ERR_B	SPORT4 Half SPORT 'B' Error Register	0x00000000
0x310024A4	SPORT4_MSTAT_B	SPORT4 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310024A8	SPORT4_CTL2_B	SPORT4 Half SPORT 'B' Control 2 Register	0x00000000
0x310024C0	SPORT4_TXPRI_B	SPORT4 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310024C4	SPORT4_RXPRI_B	SPORT4 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310024C8	SPORT4_TXSEC_B	SPORT4 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310024CC	SPORT4_RXSEC_B	SPORT4 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-180: ADSP-SC59x SPORT5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002500	SPORT5_CTL_A	SPORT5 Half SPORT 'A' Control Register	0x00000000
0x31002504	SPORT5_DIV_A	SPORT5 Half SPORT 'A' Divisor Register	0x00000000
0x31002508	SPORT5_MCTL_A	SPORT5 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100250C	SPORT5_CS0_A	SPORT5 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002510	SPORT5_CS1_A	SPORT5 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000

Table A-180: ADSP-SC59x SPORT5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002514	SPORT5_CS2_A	SPORT5 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002518	SPORT5_CS3_A	SPORT5 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002520	SPORT5_ERR_A	SPORT5 Half SPORT 'A' Error Register	0x00000000
0x31002524	SPORT5_MSTAT_A	SPORT5 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002528	SPORT5_CTL2_A	SPORT5 Half SPORT 'A' Control 2 Register	0x00000000
0x31002540	SPORT5_TXPRI_A	SPORT5 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002544	SPORT5_RXPRI_A	SPORT5 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002548	SPORT5_TXSEC_A	SPORT5 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100254C	SPORT5_RXSEC_A	SPORT5 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002580	SPORT5_CTL_B	SPORT5 Half SPORT 'B' Control Register	0x00000000
0x31002584	SPORT5_DIV_B	SPORT5 Half SPORT 'B' Divisor Register	0x00000000
0x31002588	SPORT5_MCTL_B	SPORT5 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100258C	SPORT5_CS0_B	SPORT5 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002590	SPORT5_CS1_B	SPORT5 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002594	SPORT5_CS2_B	SPORT5 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002598	SPORT5_CS3_B	SPORT5 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310025A0	SPORT5_ERR_B	SPORT5 Half SPORT 'B' Error Register	0x00000000
0x310025A4	SPORT5_MSTAT_B	SPORT5 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310025A8	SPORT5_CTL2_B	SPORT5 Half SPORT 'B' Control 2 Register	0x00000000
0x310025C0	SPORT5_TXPRI_B	SPORT5 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310025C4	SPORT5_RXPRI_B	SPORT5 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310025C8	SPORT5_TXSEC_B	SPORT5 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310025CC	SPORT5_RXSEC_B	SPORT5 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-181: ADSP-SC59x SPORT6 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002600	SPORT6_CTL_A	SPORT6 Half SPORT 'A' Control Register	0x00000000
0x31002604	SPORT6_DIV_A	SPORT6 Half SPORT 'A' Divisor Register	0x00000000
0x31002608	SPORT6_MCTL_A	SPORT6 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100260C	SPORT6_CS0_A	SPORT6 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002610	SPORT6_CS1_A	SPORT6 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002614	SPORT6_CS2_A	SPORT6 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002618	SPORT6_CS3_A	SPORT6 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002620	SPORT6_ERR_A	SPORT6 Half SPORT 'A' Error Register	0x00000000
0x31002624	SPORT6_MSTAT_A	SPORT6 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002628	SPORT6_CTL2_A	SPORT6 Half SPORT 'A' Control 2 Register	0x00000000
0x31002640	SPORT6_TXPRI_A	SPORT6 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002644	SPORT6_RXPRI_A	SPORT6 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002648	SPORT6_TXSEC_A	SPORT6 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100264C	SPORT6_RXSEC_A	SPORT6 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002680	SPORT6_CTL_B	SPORT6 Half SPORT 'B' Control Register	0x00000000
0x31002684	SPORT6_DIV_B	SPORT6 Half SPORT 'B' Divisor Register	0x00000000
0x31002688	SPORT6_MCTL_B	SPORT6 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100268C	SPORT6_CS0_B	SPORT6 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002690	SPORT6_CS1_B	SPORT6 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002694	SPORT6_CS2_B	SPORT6 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000
0x31002698	SPORT6_CS3_B	SPORT6 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310026A0	SPORT6_ERR_B	SPORT6 Half SPORT 'B' Error Register	0x00000000
0x310026A4	SPORT6_MSTAT_B	SPORT6 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310026A8	SPORT6_CTL2_B	SPORT6 Half SPORT 'B' Control 2 Register	0x00000000
0x310026C0	SPORT6_TXPRI_B	SPORT6 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000

Table A-181: ADSP-SC59x SPORT6 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310026C4	SPORT6_RXPRI_B	SPORT6 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310026C8	SPORT6_TXSEC_B	SPORT6 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310026CC	SPORT6_RXSEC_B	SPORT6 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-182: ADSP-SC59x SPORT7 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31002700	SPORT7_CTL_A	SPORT7 Half SPORT 'A' Control Register	0x00000000
0x31002704	SPORT7_DIV_A	SPORT7 Half SPORT 'A' Divisor Register	0x00000000
0x31002708	SPORT7_MCTL_A	SPORT7 Half SPORT 'A' Multichannel Control Register	0x00000000
0x3100270C	SPORT7_CS0_A	SPORT7 Half SPORT 'A' Multichannel 0-31 Select Register	0x00000000
0x31002710	SPORT7_CS1_A	SPORT7 Half SPORT 'A' Multichannel 32-63 Select Register	0x00000000
0x31002714	SPORT7_CS2_A	SPORT7 Half SPORT 'A' Multichannel 64-95 Select Register	0x00000000
0x31002718	SPORT7_CS3_A	SPORT7 Half SPORT 'A' Multichannel 96-127 Select Register	0x00000000
0x31002720	SPORT7_ERR_A	SPORT7 Half SPORT 'A' Error Register	0x00000000
0x31002724	SPORT7_MSTAT_A	SPORT7 Half SPORT 'A' Multichannel Status Register	0x00000000
0x31002728	SPORT7_CTL2_A	SPORT7 Half SPORT 'A' Control 2 Register	0x00000000
0x31002740	SPORT7_TXPRI_A	SPORT7 Half SPORT 'A' Tx Buffer (Primary) Register	0x00000000
0x31002744	SPORT7_RXPRI_A	SPORT7 Half SPORT 'A' Rx Buffer (Primary) Register	0x00000000
0x31002748	SPORT7_TXSEC_A	SPORT7 Half SPORT 'A' Tx Buffer (Secondary) Register	0x00000000
0x3100274C	SPORT7_RXSEC_A	SPORT7 Half SPORT 'A' Rx Buffer (Secondary) Register	0x00000000
0x31002780	SPORT7_CTL_B	SPORT7 Half SPORT 'B' Control Register	0x00000000
0x31002784	SPORT7_DIV_B	SPORT7 Half SPORT 'B' Divisor Register	0x00000000
0x31002788	SPORT7_MCTL_B	SPORT7 Half SPORT 'B' Multichannel Control Register	0x00000000
0x3100278C	SPORT7_CS0_B	SPORT7 Half SPORT 'B' Multichannel 0-31 Select Register	0x00000000
0x31002790	SPORT7_CS1_B	SPORT7 Half SPORT 'B' Multichannel 32-63 Select Register	0x00000000
0x31002794	SPORT7_CS2_B	SPORT7 Half SPORT 'B' Multichannel 64-95 Select Register	0x00000000

Table A-182: ADSP-SC59x SPORT7 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31002798	SPORT7_CS3_B	SPORT7 Half SPORT 'B' Multichannel 96-127 Select Register	0x00000000
0x310027A0	SPORT7_ERR_B	SPORT7 Half SPORT 'B' Error Register	0x00000000
0x310027A4	SPORT7_MSTAT_B	SPORT7 Half SPORT 'B' Multichannel Status Register	0x00000000
0x310027A8	SPORT7_CTL2_B	SPORT7 Half SPORT 'B' Control 2 Register	0x00000000
0x310027C0	SPORT7_TXPRI_B	SPORT7 Half SPORT 'B' Tx Buffer (Primary) Register	0x00000000
0x310027C4	SPORT7_RXPRI_B	SPORT7 Half SPORT 'B' Rx Buffer (Primary) Register	0x00000000
0x310027C8	SPORT7_TXSEC_B	SPORT7 Half SPORT 'B' Tx Buffer (Secondary) Register	0x00000000
0x310027CC	SPORT7_RXSEC_B	SPORT7 Half SPORT 'B' Rx Buffer (Secondary) Register	0x00000000

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B000	SPU0_CTL	SPU0 Control Register	0x000000AD
0x3108B004	SPU0_STAT	SPU0 Status Register	0x00000000
0x3108B400	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B404	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B408	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B40C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B410	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B414	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B418	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B41C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B420	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B424	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B428	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B42C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B430	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B434	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B438	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B43C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B440	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B444	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B448	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B44C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B450	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B454	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B458	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B45C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B460	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B464	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B468	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B46C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B470	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B474	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B478	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B47C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B480	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B484	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B488	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B48C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B490	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B494	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B498	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B49C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4A0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4A4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4A8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4AC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4B0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4B4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4B8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4BC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B4C0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4C4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4C8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4CC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4D0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4D4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4D8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4DC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4E0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4E4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4E8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4EC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4F0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4F4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4F8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B4FC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B500	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B504	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B508	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B50C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B510	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B514	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B518	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B51C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B520	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B524	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B528	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B52C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B530	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B534	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B538	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B53C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B540	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B544	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B548	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B54C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B550	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B554	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B558	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B55C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B560	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B564	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B568	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B56C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B570	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B574	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B578	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B57C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B580	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B584	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B588	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B58C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B590	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B594	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B598	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B59C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5A0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5A4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5A8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5AC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5B0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5B4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B5B8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5BC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5C0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5C4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5C8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5CC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5D0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5D4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5D8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5DC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5E0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5E4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5E8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5EC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5F0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5F4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5F8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B5FC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B600	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B604	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B608	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B60C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B610	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B614	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B618	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B61C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B620	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B624	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B628	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B62C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B630	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B634	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B638	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B63C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B640	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B644	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B648	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B64C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B650	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B654	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B658	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B65C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B660	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B664	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B668	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B66C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B670	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B674	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B678	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B67C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B680	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B684	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B688	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B68C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B690	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B694	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B698	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B69C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6A0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6A4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6A8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6AC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B6B0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6B4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6B8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6BC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6C0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6C4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6C8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6CC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6D0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6D4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6D8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6DC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6E0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6E4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6E8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6EC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6F0	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6F4	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6F8	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B6FC	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B700	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B704	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B708	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B70C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B710	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B714	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B718	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B71C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B720	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B724	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B728	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108B72C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B730	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B734	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B738	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B73C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B740	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B744	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B748	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B74C	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B750	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B754	SPU0_WP[n]	SPU0 Write Protect Register n	0x00000000
0x3108B840	SPU0_SECURECTL	SPU0 Secure Control Register	0x00000000
0x3108B84C	SPU0_SECURECHK	SPU0 Secure Check Register	0xFFFFFFFF
0x3108B980	SPU0_SECUREC[n]	SPU0 Secure Core Registers	0x00000001
0x3108B984	SPU0_SECUREC[n]	SPU0 Secure Core Registers	0x00000001
0x3108B988	SPU0_SECUREC[n]	SPU0 Secure Core Registers	0x00000001
0x3108BA00	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA04	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA08	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA0C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA10	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA14	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA18	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA1C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA20	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA24	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA28	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA2C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA30	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA34	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA38	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BA3C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA40	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA44	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA48	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA4C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA50	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA54	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA58	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA5C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA60	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA64	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA68	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA6C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA70	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA74	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA78	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA7C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA80	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA84	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA88	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA8C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA90	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA94	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA98	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BA9C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAA0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAA4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAA8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAAC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAB0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAB4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BAB8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BABC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAC0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAC4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAC8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BACC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAD0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAD4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAD8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BADC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAE0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAE4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAE8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAEC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAF0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAF4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAF8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BAFC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB00	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB04	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB08	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB0C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB10	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB14	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB18	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB1C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB20	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB24	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB28	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB2C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB30	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BB34	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB38	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB3C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB40	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB44	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB48	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB4C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB50	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB54	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB58	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB5C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB60	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB64	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB68	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB6C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB70	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB74	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB78	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB7C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB80	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB84	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB88	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB8C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB90	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB94	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB98	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BB9C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBA0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBA4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBA8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBAC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BBB0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBB4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBB8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBBC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBC0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBC4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBC8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBCC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBD0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBD4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBD8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBDC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBE0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBE4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBE8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBEC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBF0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBF4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBF8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BBFC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC00	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC04	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC08	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC0C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC10	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC14	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC18	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC1C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC20	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC24	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC28	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BC2C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC30	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC34	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC38	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC3C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC40	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC44	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC48	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC4C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC50	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC54	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC58	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC5C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC60	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC64	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC68	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC6C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC70	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC74	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC78	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC7C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC80	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC84	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC88	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC8C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC90	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC94	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC98	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BC9C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCA0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCA4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BCA8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCAC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCB0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCB4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCB8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCBC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCC0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCC4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCC8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCCC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCD0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCD4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCD8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCDC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCE0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCE4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCE8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCEC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCF0	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCF4	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCF8	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BCFC	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD00	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD04	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD08	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD0C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD10	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD14	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD18	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD1C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD20	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-183: ADSP-SC59x SPU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108BD24	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD28	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD2C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD30	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD34	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD38	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD3C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD40	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD44	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD48	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD4C	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD50	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001
0x3108BD54	SPU0_SECUREP[n]	SPU0 Secure Peripheral Register	0x00000001

Table A-184: ADSP-SC59x SWU MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31015000	SWU_GCTL	SWU Global Control Register	0x00000000
0x31015000	SWU0_GCTL	SWU0 Global Control Register	0x00000000
0x31015004	SWU_GSTAT	SWU Global Status Register	0x00000000
0x31015004	SWU0_GSTAT	SWU0 Global Status Register	0x00000000
0x31015010	SWU_CTL[n]	SWU Control Register n	0x00000000
0x31015010	SWU0_CTL[n]	SWU0 Control Register n	0x00000000
0x31015014	SWU_LA[n]	SWU Lower Address Register n	0x00000000
0x31015014	SWU0_LA[n]	SWU0 Lower Address Register n	0x00000000
0x31015018	SWU_UA[n]	SWU Upper Address Register n	0x00000000
0x31015018	SWU0_UA[n]	SWU0 Upper Address Register n	0x00000000
0x3101501C	SWU_ID[n]	SWU ID Register n	0x00000000
0x3101501C	SWU0_ID[n]	SWU0 ID Register n	0x00000000
0x31015020	SWU_CNT[n]	SWU Count Register n	0x00000000
0x31015020	SWU0_CNT[n]	SWU0 Count Register n	0x00000000

Table A-184: ADSP-SC59x SWU MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31015024	SWU_TARG[n]	SWU Target Register n	0x00000000
0x31015024	SWU0_TARG[n]	SWU0 Target Register n	0x00000000
0x31015028	SWU_HIST[n]	SWU Bandwidth History Register n	0x00000000
0x31015028	SWU0_HIST[n]	SWU0 Bandwidth History Register n	0x00000000
0x3101502C	SWU_CUR[n]	SWU Current Register n	0x00000000
0x3101502C	SWU0_CUR[n]	SWU0 Current Register n	0x00000000
0x31015030	SWU_CTL[n]	SWU Control Register n	0x00000000
0x31015030	SWU0_CTL[n]	SWU0 Control Register n	0x00000000
0x31015034	SWU_LA[n]	SWU Lower Address Register n	0x00000000
0x31015034	SWU0_LA[n]	SWU0 Lower Address Register n	0x00000000
0x31015038	SWU_UA[n]	SWU Upper Address Register n	0x00000000
0x31015038	SWU0_UA[n]	SWU0 Upper Address Register n	0x00000000
0x3101503C	SWU_ID[n]	SWU ID Register n	0x00000000
0x3101503C	SWU0_ID[n]	SWU0 ID Register n	0x00000000
0x31015040	SWU_CNT[n]	SWU Count Register n	0x00000000
0x31015040	SWU0_CNT[n]	SWU0 Count Register n	0x00000000
0x31015044	SWU_TARG[n]	SWU Target Register n	0x00000000
0x31015044	SWU0_TARG[n]	SWU0 Target Register n	0x00000000
0x31015048	SWU_HIST[n]	SWU Bandwidth History Register n	0x00000000
0x31015048	SWU0_HIST[n]	SWU0 Bandwidth History Register n	0x00000000
0x3101504C	SWU_CUR[n]	SWU Current Register n	0x00000000
0x3101504C	SWU0_CUR[n]	SWU0 Current Register n	0x00000000
0x31015050	SWU_CTL[n]	SWU Control Register n	0x00000000
0x31015050	SWU0_CTL[n]	SWU0 Control Register n	0x00000000
0x31015054	SWU_LA[n]	SWU Lower Address Register n	0x00000000
0x31015054	SWU0_LA[n]	SWU0 Lower Address Register n	0x00000000
0x31015058	SWU_UA[n]	SWU Upper Address Register n	0x00000000
0x31015058	SWU0_UA[n]	SWU0 Upper Address Register n	0x00000000
0x3101505C	SWU_ID[n]	SWU ID Register n	0x00000000
0x3101505C	SWU0_ID[n]	SWU0 ID Register n	0x00000000
0x31015060	SWU_CNT[n]	SWU Count Register n	0x00000000

Table A-184: ADSP-SC59x SWU MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31015060	SWU0_CNT[n]	SWU0 Count Register n	0x00000000
0x31015064	SWU_TARG[n]	SWU Target Register n	0x00000000
0x31015064	SWU0_TARG[n]	SWU0 Target Register n	0x00000000
0x31015068	SWU_HIST[n]	SWU Bandwidth History Register n	0x00000000
0x31015068	SWU0_HIST[n]	SWU0 Bandwidth History Register n	0x00000000
0x3101506C	SWU_CUR[n]	SWU Current Register n	0x00000000
0x3101506C	SWU0_CUR[n]	SWU0 Current Register n	0x00000000
0x31015070	SWU_CTL[n]	SWU Control Register n	0x00000000
0x31015070	SWU0_CTL[n]	SWU0 Control Register n	0x00000000
0x31015074	SWU_LA[n]	SWU Lower Address Register n	0x00000000
0x31015074	SWU0_LA[n]	SWU0 Lower Address Register n	0x00000000
0x31015078	SWU_UA[n]	SWU Upper Address Register n	0x00000000
0x31015078	SWU0_UA[n]	SWU0 Upper Address Register n	0x00000000
0x3101507C	SWU_ID[n]	SWU ID Register n	0x00000000
0x3101507C	SWU0_ID[n]	SWU0 ID Register n	0x00000000
0x31015080	SWU_CNT[n]	SWU Count Register n	0x00000000
0x31015080	SWU0_CNT[n]	SWU0 Count Register n	0x00000000
0x31015084	SWU_TARG[n]	SWU Target Register n	0x00000000
0x31015084	SWU0_TARG[n]	SWU0 Target Register n	0x00000000
0x31015088	SWU_HIST[n]	SWU Bandwidth History Register n	0x00000000
0x31015088	SWU0_HIST[n]	SWU0 Bandwidth History Register n	0x00000000
0x3101508C	SWU_CUR[n]	SWU Current Register n	0x00000000
0x3101508C	SWU0_CUR[n]	SWU0 Current Register n	0x00000000

Table A-185: ADSP-SC59x SWU1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31093000	SWU1_GCTL	SWU1 Global Control Register	0x00000000
0x31093004	SWU1_GSTAT	SWU1 Global Status Register	0x00000000
0x31093010	SWU1_CTL[n]	SWU1 Control Register n	0x00000000
0x31093014	SWU1_LA[n]	SWU1 Lower Address Register n	0x00000000

Table A-185: ADSP-SC59x SWU1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31093018	SWU1_UA[n]	SWU1 Upper Address Register n	0x00000000
0x3109301C	SWU1_ID[n]	SWU1 ID Register n	0x00000000
0x31093020	SWU1_CNT[n]	SWU1 Count Register n	0x00000000
0x31093024	SWU1_TARG[n]	SWU1 Target Register n	0x00000000
0x31093028	SWU1_HIST[n]	SWU1 Bandwidth History Register n	0x00000000
0x3109302C	SWU1_CUR[n]	SWU1 Current Register n	0x00000000
0x31093030	SWU1_CTL[n]	SWU1 Control Register n	0x00000000
0x31093034	SWU1_LA[n]	SWU1 Lower Address Register n	0x00000000
0x31093038	SWU1_UA[n]	SWU1 Upper Address Register n	0x00000000
0x3109303C	SWU1_ID[n]	SWU1 ID Register n	0x00000000
0x31093040	SWU1_CNT[n]	SWU1 Count Register n	0x00000000
0x31093044	SWU1_TARG[n]	SWU1 Target Register n	0x00000000
0x31093048	SWU1_HIST[n]	SWU1 Bandwidth History Register n	0x00000000
0x3109304C	SWU1_CUR[n]	SWU1 Current Register n	0x00000000
0x31093050	SWU1_CTL[n]	SWU1 Control Register n	0x00000000
0x31093054	SWU1_LA[n]	SWU1 Lower Address Register n	0x00000000
0x31093058	SWU1_UA[n]	SWU1 Upper Address Register n	0x00000000
0x3109305C	SWU1_ID[n]	SWU1 ID Register n	0x00000000
0x31093060	SWU1_CNT[n]	SWU1 Count Register n	0x00000000
0x31093064	SWU1_TARG[n]	SWU1 Target Register n	0x00000000
0x31093068	SWU1_HIST[n]	SWU1 Bandwidth History Register n	0x00000000
0x3109306C	SWU1_CUR[n]	SWU1 Current Register n	0x00000000
0x31093070	SWU1_CTL[n]	SWU1 Control Register n	0x00000000
0x31093074	SWU1_LA[n]	SWU1 Lower Address Register n	0x00000000
0x31093078	SWU1_UA[n]	SWU1 Upper Address Register n	0x00000000
0x3109307C	SWU1_ID[n]	SWU1 ID Register n	0x00000000
0x31093080	SWU1_CNT[n]	SWU1 Count Register n	0x00000000
0x31093084	SWU1_TARG[n]	SWU1 Target Register n	0x00000000
0x31093088	SWU1_HIST[n]	SWU1 Bandwidth History Register n	0x00000000
0x3109308C	SWU1_CUR[n]	SWU1 Current Register n	0x00000000

Table A-186: ADSP-SC59x SWU10 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31143000	SWU10_GCTL	SWU10 Global Control Register	0x00000000
0x31143004	SWU10_GSTAT	SWU10 Global Status Register	0x00000000
0x31143010	SWU10_CTL[n]	SWU10 Control Register n	0x00000000
0x31143014	SWU10_LA[n]	SWU10 Lower Address Register n	0x00000000
0x31143018	SWU10_UA[n]	SWU10 Upper Address Register n	0x00000000
0x3114301C	SWU10_ID[n]	SWU10 ID Register n	0x00000000
0x31143020	SWU10_CNT[n]	SWU10 Count Register n	0x00000000
0x31143024	SWU10_TARG[n]	SWU10 Target Register n	0x00000000
0x31143028	SWU10_HIST[n]	SWU10 Bandwidth History Register n	0x00000000
0x3114302C	SWU10_CUR[n]	SWU10 Current Register n	0x00000000
0x31143030	SWU10_CTL[n]	SWU10 Control Register n	0x00000000
0x31143034	SWU10_LA[n]	SWU10 Lower Address Register n	0x00000000
0x31143038	SWU10_UA[n]	SWU10 Upper Address Register n	0x00000000
0x3114303C	SWU10_ID[n]	SWU10 ID Register n	0x00000000
0x31143040	SWU10_CNT[n]	SWU10 Count Register n	0x00000000
0x31143044	SWU10_TARG[n]	SWU10 Target Register n	0x00000000
0x31143048	SWU10_HIST[n]	SWU10 Bandwidth History Register n	0x00000000
0x3114304C	SWU10_CUR[n]	SWU10 Current Register n	0x00000000
0x31143050	SWU10_CTL[n]	SWU10 Control Register n	0x00000000
0x31143054	SWU10_LA[n]	SWU10 Lower Address Register n	0x00000000
0x31143058	SWU10_UA[n]	SWU10 Upper Address Register n	0x00000000
0x3114305C	SWU10_ID[n]	SWU10 ID Register n	0x00000000
0x31143060	SWU10_CNT[n]	SWU10 Count Register n	0x00000000
0x31143064	SWU10_TARG[n]	SWU10 Target Register n	0x00000000
0x31143068	SWU10_HIST[n]	SWU10 Bandwidth History Register n	0x00000000
0x3114306C	SWU10_CUR[n]	SWU10 Current Register n	0x00000000
0x31143070	SWU10_CTL[n]	SWU10 Control Register n	0x00000000
0x31143074	SWU10_LA[n]	SWU10 Lower Address Register n	0x00000000
0x31143078	SWU10_UA[n]	SWU10 Upper Address Register n	0x00000000
0x3114307C	SWU10_ID[n]	SWU10 ID Register n	0x00000000
0x31143080	SWU10_CNT[n]	SWU10 Count Register n	0x00000000

Table A-186: ADSP-SC59x SWU10 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31143084	SWU10_TARG[n]	SWU10 Target Register n	0x00000000
0x31143088	SWU10_HIST[n]	SWU10 Bandwidth History Register n	0x00000000
0x3114308C	SWU10_CUR[n]	SWU10 Current Register n	0x00000000

Table A-187: ADSP-SC59x SWU11 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31097000	SWU11_GCTL	SWU11 Global Control Register	0x00000000
0x31097004	SWU11_GSTAT	SWU11 Global Status Register	0x00000000
0x31097010	SWU11_CTL[n]	SWU11 Control Register n	0x00000000
0x31097014	SWU11_LA[n]	SWU11 Lower Address Register n	0x00000000
0x31097018	SWU11_UA[n]	SWU11 Upper Address Register n	0x00000000
0x3109701C	SWU11_ID[n]	SWU11 ID Register n	0x00000000
0x31097020	SWU11_CNT[n]	SWU11 Count Register n	0x00000000
0x31097024	SWU11_TARG[n]	SWU11 Target Register n	0x00000000
0x31097028	SWU11_HIST[n]	SWU11 Bandwidth History Register n	0x00000000
0x3109702C	SWU11_CUR[n]	SWU11 Current Register n	0x00000000
0x31097030	SWU11_CTL[n]	SWU11 Control Register n	0x00000000
0x31097034	SWU11_LA[n]	SWU11 Lower Address Register n	0x00000000
0x31097038	SWU11_UA[n]	SWU11 Upper Address Register n	0x00000000
0x3109703C	SWU11_ID[n]	SWU11 ID Register n	0x00000000
0x31097040	SWU11_CNT[n]	SWU11 Count Register n	0x00000000
0x31097044	SWU11_TARG[n]	SWU11 Target Register n	0x00000000
0x31097048	SWU11_HIST[n]	SWU11 Bandwidth History Register n	0x00000000
0x3109704C	SWU11_CUR[n]	SWU11 Current Register n	0x00000000
0x31097050	SWU11_CTL[n]	SWU11 Control Register n	0x00000000
0x31097054	SWU11_LA[n]	SWU11 Lower Address Register n	0x00000000
0x31097058	SWU11_UA[n]	SWU11 Upper Address Register n	0x00000000
0x3109705C	SWU11_ID[n]	SWU11 ID Register n	0x00000000
0x31097060	SWU11_CNT[n]	SWU11 Count Register n	0x00000000
0x31097064	SWU11_TARG[n]	SWU11 Target Register n	0x00000000

Table A-187: ADSP-SC59x SWU11 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31097068	SWU11_HIST[n]	SWU11 Bandwidth History Register n	0x00000000
0x3109706C	SWU11_CUR[n]	SWU11 Current Register n	0x00000000
0x31097070	SWU11_CTL[n]	SWU11 Control Register n	0x00000000
0x31097074	SWU11_LA[n]	SWU11 Lower Address Register n	0x00000000
0x31097078	SWU11_UA[n]	SWU11 Upper Address Register n	0x00000000
0x3109707C	SWU11_ID[n]	SWU11 ID Register n	0x00000000
0x31097080	SWU11_CNT[n]	SWU11 Count Register n	0x00000000
0x31097084	SWU11_TARG[n]	SWU11 Target Register n	0x00000000
0x31097088	SWU11_HIST[n]	SWU11 Bandwidth History Register n	0x00000000
0x3109708C	SWU11_CUR[n]	SWU11 Current Register n	0x00000000

Table A-188: ADSP-SC59x SWU12 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310A8000	SWU12_GCTL	SWU12 Global Control Register	0x00000000
0x310A8004	SWU12_GSTAT	SWU12 Global Status Register	0x00000000
0x310A8010	SWU12_CTL[n]	SWU12 Control Register n	0x00000000
0x310A8014	SWU12_LA[n]	SWU12 Lower Address Register n	0x00000000
0x310A8018	SWU12_UA[n]	SWU12 Upper Address Register n	0x00000000
0x310A801C	SWU12_ID[n]	SWU12 ID Register n	0x00000000
0x310A8020	SWU12_CNT[n]	SWU12 Count Register n	0x00000000
0x310A8024	SWU12_TARG[n]	SWU12 Target Register n	0x00000000
0x310A8028	SWU12_HIST[n]	SWU12 Bandwidth History Register n	0x00000000
0x310A802C	SWU12_CUR[n]	SWU12 Current Register n	0x00000000
0x310A8030	SWU12_CTL[n]	SWU12 Control Register n	0x00000000
0x310A8034	SWU12_LA[n]	SWU12 Lower Address Register n	0x00000000
0x310A8038	SWU12_UA[n]	SWU12 Upper Address Register n	0x00000000
0x310A803C	SWU12_ID[n]	SWU12 ID Register n	0x00000000
0x310A8040	SWU12_CNT[n]	SWU12 Count Register n	0x00000000
0x310A8044	SWU12_TARG[n]	SWU12 Target Register n	0x00000000
0x310A8048	SWU12_HIST[n]	SWU12 Bandwidth History Register n	0x00000000

Table A-188: ADSP-SC59x SWU12 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310A804C	SWU12_CUR[n]	SWU12 Current Register n	0x00000000
0x310A8050	SWU12_CTL[n]	SWU12 Control Register n	0x00000000
0x310A8054	SWU12_LA[n]	SWU12 Lower Address Register n	0x00000000
0x310A8058	SWU12_UA[n]	SWU12 Upper Address Register n	0x00000000
0x310A805C	SWU12_ID[n]	SWU12 ID Register n	0x00000000
0x310A8060	SWU12_CNT[n]	SWU12 Count Register n	0x00000000
0x310A8064	SWU12_TARG[n]	SWU12 Target Register n	0x00000000
0x310A8068	SWU12_HIST[n]	SWU12 Bandwidth History Register n	0x00000000
0x310A806C	SWU12_CUR[n]	SWU12 Current Register n	0x00000000
0x310A8070	SWU12_CTL[n]	SWU12 Control Register n	0x00000000
0x310A8074	SWU12_LA[n]	SWU12 Lower Address Register n	0x00000000
0x310A8078	SWU12_UA[n]	SWU12 Upper Address Register n	0x00000000
0x310A807C	SWU12_ID[n]	SWU12 ID Register n	0x00000000
0x310A8080	SWU12_CNT[n]	SWU12 Count Register n	0x00000000
0x310A8084	SWU12_TARG[n]	SWU12 Target Register n	0x00000000
0x310A8088	SWU12_HIST[n]	SWU12 Bandwidth History Register n	0x00000000
0x310A808C	SWU12_CUR[n]	SWU12 Current Register n	0x00000000

Table A-189: ADSP-SC59x SWU13 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3109E000	SWU13_GCTL	SWU13 Global Control Register	0x00000000
0x3109E004	SWU13_GSTAT	SWU13 Global Status Register	0x00000000
0x3109E010	SWU13_CTL[n]	SWU13 Control Register n	0x00000000
0x3109E014	SWU13_LA[n]	SWU13 Lower Address Register n	0x00000000
0x3109E018	SWU13_UA[n]	SWU13 Upper Address Register n	0x00000000
0x3109E01C	SWU13_ID[n]	SWU13 ID Register n	0x00000000
0x3109E020	SWU13_CNT[n]	SWU13 Count Register n	0x00000000
0x3109E024	SWU13_TARG[n]	SWU13 Target Register n	0x00000000
0x3109E028	SWU13_HIST[n]	SWU13 Bandwidth History Register n	0x00000000
0x3109E02C	SWU13_CUR[n]	SWU13 Current Register n	0x00000000

Table A-189: ADSP-SC59x SWU13 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109E030	SWU13_CTL[n]	SWU13 Control Register n	0x00000000
0x3109E034	SWU13_LA[n]	SWU13 Lower Address Register n	0x00000000
0x3109E038	SWU13_UA[n]	SWU13 Upper Address Register n	0x00000000
0x3109E03C	SWU13_ID[n]	SWU13 ID Register n	0x00000000
0x3109E040	SWU13_CNT[n]	SWU13 Count Register n	0x00000000
0x3109E044	SWU13_TARG[n]	SWU13 Target Register n	0x00000000
0x3109E048	SWU13_HIST[n]	SWU13 Bandwidth History Register n	0x00000000
0x3109E04C	SWU13_CUR[n]	SWU13 Current Register n	0x00000000
0x3109E050	SWU13_CTL[n]	SWU13 Control Register n	0x00000000
0x3109E054	SWU13_LA[n]	SWU13 Lower Address Register n	0x00000000
0x3109E058	SWU13_UA[n]	SWU13 Upper Address Register n	0x00000000
0x3109E05C	SWU13_ID[n]	SWU13 ID Register n	0x00000000
0x3109E060	SWU13_CNT[n]	SWU13 Count Register n	0x00000000
0x3109E064	SWU13_TARG[n]	SWU13 Target Register n	0x00000000
0x3109E068	SWU13_HIST[n]	SWU13 Bandwidth History Register n	0x00000000
0x3109E06C	SWU13_CUR[n]	SWU13 Current Register n	0x00000000
0x3109E070	SWU13_CTL[n]	SWU13 Control Register n	0x00000000
0x3109E074	SWU13_LA[n]	SWU13 Lower Address Register n	0x00000000
0x3109E078	SWU13_UA[n]	SWU13 Upper Address Register n	0x00000000
0x3109E07C	SWU13_ID[n]	SWU13 ID Register n	0x00000000
0x3109E080	SWU13_CNT[n]	SWU13 Count Register n	0x00000000
0x3109E084	SWU13_TARG[n]	SWU13 Target Register n	0x00000000
0x3109E088	SWU13_HIST[n]	SWU13 Bandwidth History Register n	0x00000000
0x3109E08C	SWU13_CUR[n]	SWU13 Current Register n	0x00000000

Table A-190: ADSP-SC59x SWU2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31094000	SWU2_GCTL	SWU2 Global Control Register	0x00000000
0x31094004	SWU2_GSTAT	SWU2 Global Status Register	0x00000000
0x31094010	SWU2_CTL[n]	SWU2 Control Register n	0x00000000

Table A-190: ADSP-SC59x SWU2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31094014	SWU2_LA[n]	SWU2 Lower Address Register n	0x00000000
0x31094018	SWU2_UA[n]	SWU2 Upper Address Register n	0x00000000
0x3109401C	SWU2_ID[n]	SWU2 ID Register n	0x00000000
0x31094020	SWU2_CNT[n]	SWU2 Count Register n	0x00000000
0x31094024	SWU2_TARG[n]	SWU2 Target Register n	0x00000000
0x31094028	SWU2_HIST[n]	SWU2 Bandwidth History Register n	0x00000000
0x3109402C	SWU2_CUR[n]	SWU2 Current Register n	0x00000000
0x31094030	SWU2_CTL[n]	SWU2 Control Register n	0x00000000
0x31094034	SWU2_LA[n]	SWU2 Lower Address Register n	0x00000000
0x31094038	SWU2_UA[n]	SWU2 Upper Address Register n	0x00000000
0x3109403C	SWU2_ID[n]	SWU2 ID Register n	0x00000000
0x31094040	SWU2_CNT[n]	SWU2 Count Register n	0x00000000
0x31094044	SWU2_TARG[n]	SWU2 Target Register n	0x00000000
0x31094048	SWU2_HIST[n]	SWU2 Bandwidth History Register n	0x00000000
0x3109404C	SWU2_CUR[n]	SWU2 Current Register n	0x00000000
0x31094050	SWU2_CTL[n]	SWU2 Control Register n	0x00000000
0x31094054	SWU2_LA[n]	SWU2 Lower Address Register n	0x00000000
0x31094058	SWU2_UA[n]	SWU2 Upper Address Register n	0x00000000
0x3109405C	SWU2_ID[n]	SWU2 ID Register n	0x00000000
0x31094060	SWU2_CNT[n]	SWU2 Count Register n	0x00000000
0x31094064	SWU2_TARG[n]	SWU2 Target Register n	0x00000000
0x31094068	SWU2_HIST[n]	SWU2 Bandwidth History Register n	0x00000000
0x3109406C	SWU2_CUR[n]	SWU2 Current Register n	0x00000000
0x31094070	SWU2_CTL[n]	SWU2 Control Register n	0x00000000
0x31094074	SWU2_LA[n]	SWU2 Lower Address Register n	0x00000000
0x31094078	SWU2_UA[n]	SWU2 Upper Address Register n	0x00000000
0x3109407C	SWU2_ID[n]	SWU2 ID Register n	0x00000000
0x31094080	SWU2_CNT[n]	SWU2 Count Register n	0x00000000
0x31094084	SWU2_TARG[n]	SWU2 Target Register n	0x00000000
0x31094088	SWU2_HIST[n]	SWU2 Bandwidth History Register n	0x00000000
0x3109408C	SWU2_CUR[n]	SWU2 Current Register n	0x00000000

Table A-191: ADSP-SC59x SWU3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31095000	SWU3_GCTL	SWU3 Global Control Register	0x00000000
0x31095004	SWU3_GSTAT	SWU3 Global Status Register	0x00000000
0x31095010	SWU3_CTL[n]	SWU3 Control Register n	0x00000000
0x31095014	SWU3_LA[n]	SWU3 Lower Address Register n	0x00000000
0x31095018	SWU3_UA[n]	SWU3 Upper Address Register n	0x00000000
0x3109501C	SWU3_ID[n]	SWU3 ID Register n	0x00000000
0x31095020	SWU3_CNT[n]	SWU3 Count Register n	0x00000000
0x31095024	SWU3_TARG[n]	SWU3 Target Register n	0x00000000
0x31095028	SWU3_HIST[n]	SWU3 Bandwidth History Register n	0x00000000
0x3109502C	SWU3_CUR[n]	SWU3 Current Register n	0x00000000
0x31095030	SWU3_CTL[n]	SWU3 Control Register n	0x00000000
0x31095034	SWU3_LA[n]	SWU3 Lower Address Register n	0x00000000
0x31095038	SWU3_UA[n]	SWU3 Upper Address Register n	0x00000000
0x3109503C	SWU3_ID[n]	SWU3 ID Register n	0x00000000
0x31095040	SWU3_CNT[n]	SWU3 Count Register n	0x00000000
0x31095044	SWU3_TARG[n]	SWU3 Target Register n	0x00000000
0x31095048	SWU3_HIST[n]	SWU3 Bandwidth History Register n	0x00000000
0x3109504C	SWU3_CUR[n]	SWU3 Current Register n	0x00000000
0x31095050	SWU3_CTL[n]	SWU3 Control Register n	0x00000000
0x31095054	SWU3_LA[n]	SWU3 Lower Address Register n	0x00000000
0x31095058	SWU3_UA[n]	SWU3 Upper Address Register n	0x00000000
0x3109505C	SWU3_ID[n]	SWU3 ID Register n	0x00000000
0x31095060	SWU3_CNT[n]	SWU3 Count Register n	0x00000000
0x31095064	SWU3_TARG[n]	SWU3 Target Register n	0x00000000
0x31095068	SWU3_HIST[n]	SWU3 Bandwidth History Register n	0x00000000
0x3109506C	SWU3_CUR[n]	SWU3 Current Register n	0x00000000
0x31095070	SWU3_CTL[n]	SWU3 Control Register n	0x00000000
0x31095074	SWU3_LA[n]	SWU3 Lower Address Register n	0x00000000
0x31095078	SWU3_UA[n]	SWU3 Upper Address Register n	0x00000000
0x3109507C	SWU3_ID[n]	SWU3 ID Register n	0x00000000
0x31095080	SWU3_CNT[n]	SWU3 Count Register n	0x00000000

Table A-191: ADSP-SC59x SWU3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31095084	SWU3_TARG[n]	SWU3 Target Register n	0x00000000
0x31095088	SWU3_HIST[n]	SWU3 Bandwidth History Register n	0x00000000
0x3109508C	SWU3_CUR[n]	SWU3 Current Register n	0x00000000

Table A-192: ADSP-SC59x SWU4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31096000	SWU4_GCTL	SWU4 Global Control Register	0x00000000
0x31096004	SWU4_GSTAT	SWU4 Global Status Register	0x00000000
0x31096010	SWU4_CTL[n]	SWU4 Control Register n	0x00000000
0x31096014	SWU4_LA[n]	SWU4 Lower Address Register n	0x00000000
0x31096018	SWU4_UA[n]	SWU4 Upper Address Register n	0x00000000
0x3109601C	SWU4_ID[n]	SWU4 ID Register n	0x00000000
0x31096020	SWU4_CNT[n]	SWU4 Count Register n	0x00000000
0x31096024	SWU4_TARG[n]	SWU4 Target Register n	0x00000000
0x31096028	SWU4_HIST[n]	SWU4 Bandwidth History Register n	0x00000000
0x3109602C	SWU4_CUR[n]	SWU4 Current Register n	0x00000000
0x31096030	SWU4_CTL[n]	SWU4 Control Register n	0x00000000
0x31096034	SWU4_LA[n]	SWU4 Lower Address Register n	0x00000000
0x31096038	SWU4_UA[n]	SWU4 Upper Address Register n	0x00000000
0x3109603C	SWU4_ID[n]	SWU4 ID Register n	0x00000000
0x31096040	SWU4_CNT[n]	SWU4 Count Register n	0x00000000
0x31096044	SWU4_TARG[n]	SWU4 Target Register n	0x00000000
0x31096048	SWU4_HIST[n]	SWU4 Bandwidth History Register n	0x00000000
0x3109604C	SWU4_CUR[n]	SWU4 Current Register n	0x00000000
0x31096050	SWU4_CTL[n]	SWU4 Control Register n	0x00000000
0x31096054	SWU4_LA[n]	SWU4 Lower Address Register n	0x00000000
0x31096058	SWU4_UA[n]	SWU4 Upper Address Register n	0x00000000
0x3109605C	SWU4_ID[n]	SWU4 ID Register n	0x00000000
0x31096060	SWU4_CNT[n]	SWU4 Count Register n	0x00000000
0x31096064	SWU4_TARG[n]	SWU4 Target Register n	0x00000000

Table A-192: ADSP-SC59x SWU4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31096068	SWU4_HIST[n]	SWU4 Bandwidth History Register n	0x00000000
0x3109606C	SWU4_CUR[n]	SWU4 Current Register n	0x00000000
0x31096070	SWU4_CTL[n]	SWU4 Control Register n	0x00000000
0x31096074	SWU4_LA[n]	SWU4 Lower Address Register n	0x00000000
0x31096078	SWU4_UA[n]	SWU4 Upper Address Register n	0x00000000
0x3109607C	SWU4_ID[n]	SWU4 ID Register n	0x00000000
0x31096080	SWU4_CNT[n]	SWU4 Count Register n	0x00000000
0x31096084	SWU4_TARG[n]	SWU4 Target Register n	0x00000000
0x31096088	SWU4_HIST[n]	SWU4 Bandwidth History Register n	0x00000000
0x3109608C	SWU4_CUR[n]	SWU4 Current Register n	0x00000000

Table A-193: ADSP-SC59x SWU5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31098000	SWU5_GCTL	SWU5 Global Control Register	0x00000000
0x31098004	SWU5_GSTAT	SWU5 Global Status Register	0x00000000
0x31098010	SWU5_CTL[n]	SWU5 Control Register n	0x00000000
0x31098014	SWU5_LA[n]	SWU5 Lower Address Register n	0x00000000
0x31098018	SWU5_UA[n]	SWU5 Upper Address Register n	0x00000000
0x3109801C	SWU5_ID[n]	SWU5 ID Register n	0x00000000
0x31098020	SWU5_CNT[n]	SWU5 Count Register n	0x00000000
0x31098024	SWU5_TARG[n]	SWU5 Target Register n	0x00000000
0x31098028	SWU5_HIST[n]	SWU5 Bandwidth History Register n	0x00000000
0x3109802C	SWU5_CUR[n]	SWU5 Current Register n	0x00000000
0x31098030	SWU5_CTL[n]	SWU5 Control Register n	0x00000000
0x31098034	SWU5_LA[n]	SWU5 Lower Address Register n	0x00000000
0x31098038	SWU5_UA[n]	SWU5 Upper Address Register n	0x00000000
0x3109803C	SWU5_ID[n]	SWU5 ID Register n	0x00000000
0x31098040	SWU5_CNT[n]	SWU5 Count Register n	0x00000000
0x31098044	SWU5_TARG[n]	SWU5 Target Register n	0x00000000
0x31098048	SWU5_HIST[n]	SWU5 Bandwidth History Register n	0x00000000

Table A-193: ADSP-SC59x SWU5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3109804C	SWU5_CUR[n]	SWU5 Current Register n	0x00000000
0x31098050	SWU5_CTL[n]	SWU5 Control Register n	0x00000000
0x31098054	SWU5_LA[n]	SWU5 Lower Address Register n	0x00000000
0x31098058	SWU5_UA[n]	SWU5 Upper Address Register n	0x00000000
0x3109805C	SWU5_ID[n]	SWU5 ID Register n	0x00000000
0x31098060	SWU5_CNT[n]	SWU5 Count Register n	0x00000000
0x31098064	SWU5_TARG[n]	SWU5 Target Register n	0x00000000
0x31098068	SWU5_HIST[n]	SWU5 Bandwidth History Register n	0x00000000
0x3109806C	SWU5_CUR[n]	SWU5 Current Register n	0x00000000
0x31098070	SWU5_CTL[n]	SWU5 Control Register n	0x00000000
0x31098074	SWU5_LA[n]	SWU5 Lower Address Register n	0x00000000
0x31098078	SWU5_UA[n]	SWU5 Upper Address Register n	0x00000000
0x3109807C	SWU5_ID[n]	SWU5 ID Register n	0x00000000
0x31098080	SWU5_CNT[n]	SWU5 Count Register n	0x00000000
0x31098084	SWU5_TARG[n]	SWU5 Target Register n	0x00000000
0x31098088	SWU5_HIST[n]	SWU5 Bandwidth History Register n	0x00000000
0x3109808C	SWU5_CUR[n]	SWU5 Current Register n	0x00000000

Table A-194: ADSP-SC59x SWU7 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31140000	SWU7_GCTL	SWU7 Global Control Register	0x00000000
0x31140004	SWU7_GSTAT	SWU7 Global Status Register	0x00000000
0x31140010	SWU7_CTL[n]	SWU7 Control Register n	0x00000000
0x31140014	SWU7_LA[n]	SWU7 Lower Address Register n	0x00000000
0x31140018	SWU7_UA[n]	SWU7 Upper Address Register n	0x00000000
0x3114001C	SWU7_ID[n]	SWU7 ID Register n	0x00000000
0x31140020	SWU7_CNT[n]	SWU7 Count Register n	0x00000000
0x31140024	SWU7_TARG[n]	SWU7 Target Register n	0x00000000
0x31140028	SWU7_HIST[n]	SWU7 Bandwidth History Register n	0x00000000
0x3114002C	SWU7_CUR[n]	SWU7 Current Register n	0x00000000

Table A-194: ADSP-SC59x SWU7 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31140030	SWU7_CTL[n]	SWU7 Control Register n	0x00000000
0x31140034	SWU7_LA[n]	SWU7 Lower Address Register n	0x00000000
0x31140038	SWU7_UA[n]	SWU7 Upper Address Register n	0x00000000
0x3114003C	SWU7_ID[n]	SWU7 ID Register n	0x00000000
0x31140040	SWU7_CNT[n]	SWU7 Count Register n	0x00000000
0x31140044	SWU7_TARG[n]	SWU7 Target Register n	0x00000000
0x31140048	SWU7_HIST[n]	SWU7 Bandwidth History Register n	0x00000000
0x3114004C	SWU7_CUR[n]	SWU7 Current Register n	0x00000000
0x31140050	SWU7_CTL[n]	SWU7 Control Register n	0x00000000
0x31140054	SWU7_LA[n]	SWU7 Lower Address Register n	0x00000000
0x31140058	SWU7_UA[n]	SWU7 Upper Address Register n	0x00000000
0x3114005C	SWU7_ID[n]	SWU7 ID Register n	0x00000000
0x31140060	SWU7_CNT[n]	SWU7 Count Register n	0x00000000
0x31140064	SWU7_TARG[n]	SWU7 Target Register n	0x00000000
0x31140068	SWU7_HIST[n]	SWU7 Bandwidth History Register n	0x00000000
0x3114006C	SWU7_CUR[n]	SWU7 Current Register n	0x00000000
0x31140070	SWU7_CTL[n]	SWU7 Control Register n	0x00000000
0x31140074	SWU7_LA[n]	SWU7 Lower Address Register n	0x00000000
0x31140078	SWU7_UA[n]	SWU7 Upper Address Register n	0x00000000
0x3114007C	SWU7_ID[n]	SWU7 ID Register n	0x00000000
0x31140080	SWU7_CNT[n]	SWU7 Count Register n	0x00000000
0x31140084	SWU7_TARG[n]	SWU7 Target Register n	0x00000000
0x31140088	SWU7_HIST[n]	SWU7 Bandwidth History Register n	0x00000000
0x3114008C	SWU7_CUR[n]	SWU7 Current Register n	0x00000000

Table A-195: ADSP-SC59x SWU8 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31141000	SWU8_GCTL	SWU8 Global Control Register	0x00000000
0x31141004	SWU8_GSTAT	SWU8 Global Status Register	0x00000000
0x31141010	SWU8_CTL[n]	SWU8 Control Register n	0x00000000

Table A-195: ADSP-SC59x SWU8 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31141014	SWU8_LA[n]	SWU8 Lower Address Register n	0x00000000
0x31141018	SWU8_UA[n]	SWU8 Upper Address Register n	0x00000000
0x3114101C	SWU8_ID[n]	SWU8 ID Register n	0x00000000
0x31141020	SWU8_CNT[n]	SWU8 Count Register n	0x00000000
0x31141024	SWU8_TARG[n]	SWU8 Target Register n	0x00000000
0x31141028	SWU8_HIST[n]	SWU8 Bandwidth History Register n	0x00000000
0x3114102C	SWU8_CUR[n]	SWU8 Current Register n	0x00000000
0x31141030	SWU8_CTL[n]	SWU8 Control Register n	0x00000000
0x31141034	SWU8_LA[n]	SWU8 Lower Address Register n	0x00000000
0x31141038	SWU8_UA[n]	SWU8 Upper Address Register n	0x00000000
0x3114103C	SWU8_ID[n]	SWU8 ID Register n	0x00000000
0x31141040	SWU8_CNT[n]	SWU8 Count Register n	0x00000000
0x31141044	SWU8_TARG[n]	SWU8 Target Register n	0x00000000
0x31141048	SWU8_HIST[n]	SWU8 Bandwidth History Register n	0x00000000
0x3114104C	SWU8_CUR[n]	SWU8 Current Register n	0x00000000
0x31141050	SWU8_CTL[n]	SWU8 Control Register n	0x00000000
0x31141054	SWU8_LA[n]	SWU8 Lower Address Register n	0x00000000
0x31141058	SWU8_UA[n]	SWU8 Upper Address Register n	0x00000000
0x3114105C	SWU8_ID[n]	SWU8 ID Register n	0x00000000
0x31141060	SWU8_CNT[n]	SWU8 Count Register n	0x00000000
0x31141064	SWU8_TARG[n]	SWU8 Target Register n	0x00000000
0x31141068	SWU8_HIST[n]	SWU8 Bandwidth History Register n	0x00000000
0x3114106C	SWU8_CUR[n]	SWU8 Current Register n	0x00000000
0x31141070	SWU8_CTL[n]	SWU8 Control Register n	0x00000000
0x31141074	SWU8_LA[n]	SWU8 Lower Address Register n	0x00000000
0x31141078	SWU8_UA[n]	SWU8 Upper Address Register n	0x00000000
0x3114107C	SWU8_ID[n]	SWU8 ID Register n	0x00000000
0x31141080	SWU8_CNT[n]	SWU8 Count Register n	0x00000000
0x31141084	SWU8_TARG[n]	SWU8 Target Register n	0x00000000
0x31141088	SWU8_HIST[n]	SWU8 Bandwidth History Register n	0x00000000
0x3114108C	SWU8_CUR[n]	SWU8 Current Register n	0x00000000

Table A-196: ADSP-SC59x SWU9 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31142000	SWU9_GCTL	SWU9 Global Control Register	0x00000000
0x31142004	SWU9_GSTAT	SWU9 Global Status Register	0x00000000
0x31142010	SWU9_CTL[n]	SWU9 Control Register n	0x00000000
0x31142014	SWU9_LA[n]	SWU9 Lower Address Register n	0x00000000
0x31142018	SWU9_UA[n]	SWU9 Upper Address Register n	0x00000000
0x3114201C	SWU9_ID[n]	SWU9 ID Register n	0x00000000
0x31142020	SWU9_CNT[n]	SWU9 Count Register n	0x00000000
0x31142024	SWU9_TARG[n]	SWU9 Target Register n	0x00000000
0x31142028	SWU9_HIST[n]	SWU9 Bandwidth History Register n	0x00000000
0x3114202C	SWU9_CUR[n]	SWU9 Current Register n	0x00000000
0x31142030	SWU9_CTL[n]	SWU9 Control Register n	0x00000000
0x31142034	SWU9_LA[n]	SWU9 Lower Address Register n	0x00000000
0x31142038	SWU9_UA[n]	SWU9 Upper Address Register n	0x00000000
0x3114203C	SWU9_ID[n]	SWU9 ID Register n	0x00000000
0x31142040	SWU9_CNT[n]	SWU9 Count Register n	0x00000000
0x31142044	SWU9_TARG[n]	SWU9 Target Register n	0x00000000
0x31142048	SWU9_HIST[n]	SWU9 Bandwidth History Register n	0x00000000
0x3114204C	SWU9_CUR[n]	SWU9 Current Register n	0x00000000
0x31142050	SWU9_CTL[n]	SWU9 Control Register n	0x00000000
0x31142054	SWU9_LA[n]	SWU9 Lower Address Register n	0x00000000
0x31142058	SWU9_UA[n]	SWU9 Upper Address Register n	0x00000000
0x3114205C	SWU9_ID[n]	SWU9 ID Register n	0x00000000
0x31142060	SWU9_CNT[n]	SWU9 Count Register n	0x00000000
0x31142064	SWU9_TARG[n]	SWU9 Target Register n	0x00000000
0x31142068	SWU9_HIST[n]	SWU9 Bandwidth History Register n	0x00000000
0x3114206C	SWU9_CUR[n]	SWU9 Current Register n	0x00000000
0x31142070	SWU9_CTL[n]	SWU9 Control Register n	0x00000000
0x31142074	SWU9_LA[n]	SWU9 Lower Address Register n	0x00000000
0x31142078	SWU9_UA[n]	SWU9 Upper Address Register n	0x00000000
0x3114207C	SWU9_ID[n]	SWU9 ID Register n	0x00000000
0x31142080	SWU9_CNT[n]	SWU9 Count Register n	0x00000000

Table A-196: ADSP-SC59x SWU9 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31142084	SWU9_TARG[n]	SWU9 Target Register n	0x00000000
0x31142088	SWU9_HIST[n]	SWU9 Bandwidth History Register n	0x00000000
0x3114208C	SWU9_CUR[n]	SWU9 Current Register n	0x00000000

Table A-197: ADSP-SC59x TAPC MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31130000	TAPC_IDCODE	TAPC IDCODE Register	0x0282E0CB
0x31130004	TAPC_USERCODE	TAPC USERCODE Register	0x00000000
0x31130008	TAPC_SDBGKEY_CTL	TAPC Secure Debug Key Control Register	0x00000000
0x3113000C	TAPC_SDBGKEY_STAT	TAPC Secure Debug Key Status Register	0x00000000
0x31130010	TAPC_SDBGKEY0	TAPC Secure Debug Key 0 Register	0x00000000
0x31130014	TAPC_SDBGKEY1	TAPC Secure Debug Key 1 Register	0x00000000
0x31130018	TAPC_SDBGKEY2	TAPC Secure Debug Key 2 Register	0x00000000
0x3113001C	TAPC_SDBGKEY3	TAPC Secure Debug Key 3 Register	0x00000000
0x31131000	TAPC_DBGCTL	TAPC Debug Control Register	0x00000000

Table A-198: ADSP-SC59x TIMER0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31018004	TIMER0_RUN	TIMER0 Run Register	0x00000000
0x31018008	TIMER0_RUN_SET	TIMER0 Run Set Register	0x00000000
0x3101800C	TIMER0_RUN_CLR	TIMER0 Run Clear Register	0x00000000
0x31018010	TIMER0_STOP_CFG	TIMER0 Stop Configuration Register	0x00000000
0x31018014	TIMER0_STOP_CFG_SET	TIMER0 Stop Configuration Set Register	0x00000000
0x31018018	TIMER0_STOP_CFG_CLR	TIMER0 Stop Configuration Clear Register	0x00000000
0x3101801C	TIMER0_DATA_IMSK	TIMER0 Data Interrupt Mask Register	0x0000FFFF
0x31018020	TIMER0_STAT_IMSK	TIMER0 Status Interrupt Mask Register	0x0000FFFF
0x31018024	TIMER0_TRG_MSK	TIMER0 Trigger Generator Mask Register	0x0000FFFF
0x31018028	TIMER0_TRG_IE	TIMER0 Trigger Receiver Enable Register	0x00000000
0x3101802C	TIMER0_DATA_ILAT	TIMER0 Data Interrupt Latch Register	0x00000000
0x31018030	TIMER0_STAT_ILAT	TIMER0 Status Interrupt Latch Register	0x00000000

Table A-198: ADSP-SC59x TIMER0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31018034	TIMER0_ERR_TYPE	TIMER0 Error Type Status Register	0x00000000
0x31018038	TIMER0_BCAST_PER	TIMER0 Broadcast Period Register	0x00000000
0x3101803C	TIMER0_BCAST_WID	TIMER0 Broadcast Width Register	0x00000000
0x31018040	TIMER0_BCAST_DLY	TIMER0 Broadcast Delay Register	0x00000000
0x31018060	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018064	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018068	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101806C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018070	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018080	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018084	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018088	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101808C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018090	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310180A0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310180A4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310180A8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310180AC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310180B0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310180C0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310180C4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310180C8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310180CC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310180D0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310180E0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310180E4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310180E8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310180EC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310180F0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018100	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018104	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001

Table A-198: ADSP-SC59x TIMER0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31018108	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101810C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018110	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018120	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018124	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018128	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101812C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018130	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018140	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018144	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018148	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101814C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018150	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018160	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018164	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018168	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101816C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018170	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018180	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018184	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018188	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101818C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018190	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310181A0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310181A4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310181A8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310181AC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310181B0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310181C0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310181C4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310181C8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000

Table A-198: ADSP-SC59x TIMER0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310181CC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310181D0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x310181E0	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x310181E4	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x310181E8	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x310181EC	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x310181F0	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018200	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018204	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018208	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101820C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018210	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018220	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018224	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018228	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101822C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018230	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000
0x31018240	TIMER0_TMR[n]_CFG	TIMER0 Timer n Configuration Register	0x00000000
0x31018244	TIMER0_TMR[n]_CNT	TIMER0 Timer n Counter Register	0x00000001
0x31018248	TIMER0_TMR[n]_PER	TIMER0 Timer n Period Register	0x00000000
0x3101824C	TIMER0_TMR[n]_WID	TIMER0 Timer n Width Register	0x00000000
0x31018250	TIMER0_TMR[n]_DLY	TIMER0 Timer n Delay Register	0x00000000

Table A-199: ADSP-SC59x TMU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31016800	TMU0_CTL	TMU0 TMU Control Register	0x00000000
0x31016804	TMU0_TEMP	TMU0 Temperature Value Register	0x00000000
0x31016808	TMU0_AVG	TMU0 Averaging Register	0x00000000
0x3101680C	TMU0_FLT_LIM_HI	TMU0 Fault High Limit Register	0x000000FF
0x31016810	TMU0_ALRT_LIM_HI	TMU0 Alert High Limit Register	0x000000FF

Table A-199: ADSP-SC59x TMU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31016814	TMU0_FLT_LIM_LO	TMU0 Fault Low Limit Register	0x00000100
0x31016818	TMU0_ALRT_LIM_LO	TMU0 Alert Low Limit Register	0x00000100
0x3101681C	TMU0_STAT	TMU0 Status Register	0x00000000
0x31016824	TMU0_GAIN	TMU0 Gain Value Register	0x00000000
0x31016828	TMU0_IMSK	TMU0 Interrupt Mask Register	0x0000000F
0x3101682C	TMU0_OFFSET	TMU0 Offset Register	0x00000000
0x31016834	TMU0_CNV_BLANK	TMU0 Temperature Conversion Blank Register	0x03100000
0x31016838	TMU0_REFR_CNTR	TMU0 Temperature Refresh Counter	0x00000000

Table A-200: ADSP-SC59x TRNG0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310D0000	TRNG0_INPUT[n]	TRNG0 TRNG Input Registers	0x00000000
0x310D0000	TRNG0_OUTPUT[n]	TRNG0 TRNG Output Registers	0x00000000
0x310D0004	TRNG0_INPUT[n]	TRNG0 TRNG Input Registers	0x00000000
0x310D0004	TRNG0_OUTPUT[n]	TRNG0 TRNG Output Registers	0x00000000
0x310D0008	TRNG0_OUTPUT[n]	TRNG0 TRNG Output Registers	0x00000000
0x310D000C	TRNG0_OUTPUT[n]	TRNG0 TRNG Output Registers	0x00000000
0x310D0010	TRNG0_STAT	TRNG0 TRNG Status Register	0x00000000
0x310D0010	TRNG0_INTACK	TRNG0 TRNG Interrupt Acknowledge Register	0x00000000
0x310D0014	TRNG0_CTL	TRNG0 TRNG Control Register	0x00000000
0x310D0018	TRNG0_CFG	TRNG0 TRNG Configuration Register	0x00000000
0x310D001C	TRNG0_ALMCNT	TRNG0 TRNG Alarm Counter Register	0x000000FF
0x310D0020	TRNG0_FROEN	TRNG0 TRNG FRO Enable Register	0x000000FF
0x310D0024	TRNG0_FRODETUNE	TRNG0 TRNG FRO De-tune Register	0x00000000
0x310D0028	TRNG0_ALMMSK	TRNG0 TRNG Alarm Mask Register	0x00000000
0x310D002C	TRNG0_ALMSTP	TRNG0 TRNG Alarm Stop Register	0x00000000
0x310D0030	TRNG0_LFSR_L	TRNG0 TRNG LFSR Access Register	0x00000000
0x310D0034	TRNG0_LFSR_M	TRNG0 TRNG LFSR Access Register	0x00000000
0x310D0038	TRNG0_LFSR_H	TRNG0 TRNG LFSR Access Register	0x00000000
0x310D003C	TRNG0_CNT	TRNG0 Counter Register	0x00000000

Table A-200: ADSP-SC59x TRNG0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310D0040	TRNG0_RUNCNT	TRNG0 TRNG Run Count Registers	0x00000000
0x310D0040	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0044	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D0044	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0048	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D0048	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D004C	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D004C	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0050	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D0050	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0054	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D0054	TRNG0_KEY[n]	TRNG0 Post-Process Key Registers	0x00000000
0x310D0058	TRNG0_RUN[n]	TRNG0 TRNG Run Test State and Result Registers	0x00000000
0x310D005C	TRNG0_MONOBITCNT	TRNG0 TRNG Monobit Test Result Register	0x00002710
0x310D0060	TRNG0_POKER[n]	TRNG0 TRNG Poker Test Result Registers	0x00000000
0x310D0060	TRNG0_V[n]	TRNG0 TRNG Post-Process "V" Value Registers	0x00000000
0x310D0064	TRNG0_POKER[n]	TRNG0 TRNG Poker Test Result Registers	0x00000000
0x310D0064	TRNG0_V[n]	TRNG0 TRNG Post-Process "V" Value Registers	0x00000000
0x310D0068	TRNG0_POKER[n]	TRNG0 TRNG Poker Test Result Registers	0x00000000
0x310D006C	TRNG0_POKER[n]	TRNG0 TRNG Poker Test Result Registers	0x00000000
0x310D0070	TRNG0_TEST	TRNG0 TRNG Test Register	0x00000000
0x310D0074	TRNG0_BLKCNT	TRNG0 TRNG Block Count Register	0x00000000

Table A-201: ADSP-SC59x TRU0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A000	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A004	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A008	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A00C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A010	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-201: ADSP-SC59x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A014	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A018	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A01C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A020	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A024	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A028	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A02C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A030	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A034	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A038	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A03C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A040	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A044	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A048	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A04C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A050	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A054	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A058	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A05C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A060	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A064	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A068	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A06C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A070	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A074	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A078	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A07C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A080	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A084	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A088	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A08C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-201: ADSP-SC59x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A090	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A094	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A098	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A09C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0A0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0A4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0A8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0AC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0B0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0B4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0B8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0BC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0C0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0C4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0C8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0CC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0D0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0D4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0D8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0DC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0E0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0E4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0E8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0EC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0F0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0F4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0F8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A0FC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A100	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A104	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A108	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-201: ADSP-SC59x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A10C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A110	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A114	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A118	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A11C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A120	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A124	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A128	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A12C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A130	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A134	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A138	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A13C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A140	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A144	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A148	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A14C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A150	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A154	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A158	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A15C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A160	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A164	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A168	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A16C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A170	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A174	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A178	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A17C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A180	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A184	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-201: ADSP-SC59x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A188	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A18C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A190	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A194	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A198	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A19C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1A0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1A4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1A8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1AC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1B0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1B4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1B8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1BC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1C0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1C4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1C8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1CC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1D0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1D4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1D8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1DC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1E0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1E4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1E8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1EC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1F0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1F4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1F8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A1FC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A200	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-201: ADSP-SC59x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A204	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A208	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A20C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A210	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A214	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A218	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A21C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A220	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A224	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A228	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A22C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A230	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A234	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A238	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A23C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A240	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A244	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A248	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A24C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A250	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A254	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A258	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A25C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A260	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A264	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A268	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A26C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A270	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A274	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A278	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A27C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-201: ADSP-SC59x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A280	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A284	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A288	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A28C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A290	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A294	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A298	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A29C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2A0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2A4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2A8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2AC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2B0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2B4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2B8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2BC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2C0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2C4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2C8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2CC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2D0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2D4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2D8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2DC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2E0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2E4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2E8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2EC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2F0	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2F4	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A2F8	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-201: ADSP-SC59x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A2FC	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A300	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A304	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A308	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A30C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A310	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A314	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A318	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A31C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A320	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A324	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A328	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A32C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A330	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A334	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A338	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A33C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A340	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A344	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A348	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A34C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A350	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A354	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A358	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A35C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A360	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A364	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A368	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A36C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A370	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A374	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000

Table A-201: ADSP-SC59x TRU0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3108A378	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A37C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A380	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A384	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A388	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A38C	TRU0_SSR[n]	TRU0 Receiver Select Register	0x00000000
0x3108A7E0	TRU0_MTR	TRU0 Generator Trigger Register	0x00000000
0x3108A7E8	TRU0_ERRADDR	TRU0 Error Address Register	0x00000000
0x3108A7EC	TRU0_STAT	TRU0 Status Information Register	0x00000000
0x3108A7F4	TRU0_GCTL	TRU0 Global Control Register	0x00000000

Table A-202: ADSP-SC59x TWI0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001400	TWI0_CLKDIV	TWI0 SCL Clock Divider Register	0x00000000
0x31001404	TWI0_CTL	TWI0 Control Register	0x00000000
0x31001408	TWI0_SLVCTL	TWI0 Target Mode Control Register	0x00000000
0x3100140C	TWI0_SLVSTAT	TWI0 Target Mode Status Register	0x00000000
0x31001410	TWI0_SLVADDR	TWI0 Target Mode Address Register	0x00000000
0x31001414	TWI0_MSTRCTL	TWI0 Controller Mode Registers	0x00000000
0x31001418	TWI0_MSTRSTAT	TWI0 Controller Mode Status Register	0x00000000
0x3100141C	TWI0_MSTRADDR	TWI0 Controller Mode Address Register	0x00000000
0x31001420	TWI0_ISTAT	TWI0 Interrupt Status Register	0x00000000
0x31001424	TWI0_IMSK	TWI0 Interrupt Mask Register	0x00000000
0x31001428	TWI0_FIFOCTL	TWI0 FIFO Control Register	0x00000000
0x3100142C	TWI0_FIFOSTAT	TWI0 FIFO Status Register	0x00000000
0x31001480	TWI0_TXDATA8	TWI0 Tx Data Single-Byte Register	0x00000000
0x31001484	TWI0_TXDATA16	TWI0 Tx Data Double-Byte Register	0x00000000
0x31001488	TWI0_RXDATA8	TWI0 Rx Data Single-Byte Register	0x00000000
0x3100148C	TWI0_RXDATA16	TWI0 Rx Data Double-Byte Register	0x00000000

Table A-203: ADSP-SC59x TWI1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001500	TWI1_CLKDIV	TWI1 SCL Clock Divider Register	0x00000000
0x31001504	TWI1_CTL	TWI1 Control Register	0x00000000
0x31001508	TWI1_SLVCTL	TWI1 Target Mode Control Register	0x00000000
0x3100150C	TWI1_SLVSTAT	TWI1 Target Mode Status Register	0x00000000
0x31001510	TWI1_SLVADDR	TWI1 Target Mode Address Register	0x00000000
0x31001514	TWI1_MSTRCTL	TWI1 Controller Mode Registers	0x00000000
0x31001518	TWI1_MSTRSTAT	TWI1 Controller Mode Status Register	0x00000000
0x3100151C	TWI1_MSTRADDR	TWI1 Controller Mode Address Register	0x00000000
0x31001520	TWI1_ISTAT	TWI1 Interrupt Status Register	0x00000000
0x31001524	TWI1_IMSK	TWI1 Interrupt Mask Register	0x00000000
0x31001528	TWI1_FIFOCTL	TWI1 FIFO Control Register	0x00000000
0x3100152C	TWI1_FIFOSTAT	TWI1 FIFO Status Register	0x00000000
0x31001580	TWI1_TXDATA8	TWI1 Tx Data Single-Byte Register	0x00000000
0x31001584	TWI1_TXDATA16	TWI1 Tx Data Double-Byte Register	0x00000000
0x31001588	TWI1_RXDATA8	TWI1 Rx Data Single-Byte Register	0x00000000
0x3100158C	TWI1_RXDATA16	TWI1 Rx Data Double-Byte Register	0x00000000

Table A-204: ADSP-SC59x TWI2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001600	TWI2_CLKDIV	TWI2 SCL Clock Divider Register	0x00000000
0x31001604	TWI2_CTL	TWI2 Control Register	0x00000000
0x31001608	TWI2_SLVCTL	TWI2 Target Mode Control Register	0x00000000
0x3100160C	TWI2_SLVSTAT	TWI2 Target Mode Status Register	0x00000000
0x31001610	TWI2_SLVADDR	TWI2 Target Mode Address Register	0x00000000
0x31001614	TWI2_MSTRCTL	TWI2 Controller Mode Registers	0x00000000
0x31001618	TWI2_MSTRSTAT	TWI2 Controller Mode Status Register	0x00000000
0x3100161C	TWI2_MSTRADDR	TWI2 Controller Mode Address Register	0x00000000
0x31001620	TWI2_ISTAT	TWI2 Interrupt Status Register	0x00000000
0x31001624	TWI2_IMSK	TWI2 Interrupt Mask Register	0x00000000
0x31001628	TWI2_FIFOCTL	TWI2 FIFO Control Register	0x00000000

Table A-204: ADSP-SC59x TWI2 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3100162C	TWI2_FIFOSTAT	TWI2 FIFO Status Register	0x00000000
0x31001680	TWI2_TXDATA8	TWI2 Tx Data Single-Byte Register	0x00000000
0x31001684	TWI2_TXDATA16	TWI2 Tx Data Double-Byte Register	0x00000000
0x31001688	TWI2_RXDATA8	TWI2 Rx Data Single-Byte Register	0x00000000
0x3100168C	TWI2_RXDATA16	TWI2 Rx Data Double-Byte Register	0x00000000

Table A-205: ADSP-SC59x TWI3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001000	TWI3_CLKDIV	TWI3 SCL Clock Divider Register	0x00000000
0x31001004	TWI3_CTL	TWI3 Control Register	0x00000000
0x31001008	TWI3_SLVCTL	TWI3 Target Mode Control Register	0x00000000
0x3100100C	TWI3_SLVSTAT	TWI3 Target Mode Status Register	0x00000000
0x31001010	TWI3_SLVADDR	TWI3 Target Mode Address Register	0x00000000
0x31001014	TWI3_MSTRCTL	TWI3 Controller Mode Registers	0x00000000
0x31001018	TWI3_MSTRSTAT	TWI3 Controller Mode Status Register	0x00000000
0x3100101C	TWI3_MSTRADDR	TWI3 Controller Mode Address Register	0x00000000
0x31001020	TWI3_ISTAT	TWI3 Interrupt Status Register	0x00000000
0x31001024	TWI3_IMSK	TWI3 Interrupt Mask Register	0x00000000
0x31001028	TWI3_FIFOCTL	TWI3 FIFO Control Register	0x00000000
0x3100102C	TWI3_FIFOSTAT	TWI3 FIFO Status Register	0x00000000
0x31001080	TWI3_TXDATA8	TWI3 Tx Data Single-Byte Register	0x00000000
0x31001084	TWI3_TXDATA16	TWI3 Tx Data Double-Byte Register	0x00000000
0x31001088	TWI3_RXDATA8	TWI3 Rx Data Single-Byte Register	0x00000000
0x3100108C	TWI3_RXDATA16	TWI3 Rx Data Double-Byte Register	0x00000000

Table A-206: ADSP-SC59x TWI4 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001100	TWI4_CLKDIV	TWI4 SCL Clock Divider Register	0x00000000
0x31001104	TWI4_CTL	TWI4 Control Register	0x00000000
0x31001108	TWI4_SLVCTL	TWI4 Target Mode Control Register	0x00000000

Table A-206: ADSP-SC59x TWI4 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x3100110C	TWI4_SLVSTAT	TWI4 Target Mode Status Register	0x00000000
0x31001110	TWI4_SLVADDR	TWI4 Target Mode Address Register	0x00000000
0x31001114	TWI4_MSTRCTL	TWI4 Controller Mode Registers	0x00000000
0x31001118	TWI4_MSTRSTAT	TWI4 Controller Mode Status Register	0x00000000
0x3100111C	TWI4_MSTRADDR	TWI4 Controller Mode Address Register	0x00000000
0x31001120	TWI4_ISTAT	TWI4 Interrupt Status Register	0x00000000
0x31001124	TWI4_IMSK	TWI4 Interrupt Mask Register	0x00000000
0x31001128	TWI4_FIFOCTL	TWI4 FIFO Control Register	0x00000000
0x3100112C	TWI4_FIFOSTAT	TWI4 FIFO Status Register	0x00000000
0x31001180	TWI4_TXDATA8	TWI4 Tx Data Single-Byte Register	0x00000000
0x31001184	TWI4_TXDATA16	TWI4 Tx Data Double-Byte Register	0x00000000
0x31001188	TWI4_RXDATA8	TWI4 Rx Data Single-Byte Register	0x00000000
0x3100118C	TWI4_RXDATA16	TWI4 Rx Data Double-Byte Register	0x00000000

Table A-207: ADSP-SC59x TWI5 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31001200	TWI5_CLKDIV	TWI5 SCL Clock Divider Register	0x00000000
0x31001204	TWI5_CTL	TWI5 Control Register	0x00000000
0x31001208	TWI5_SLVCTL	TWI5 Target Mode Control Register	0x00000000
0x3100120C	TWI5_SLVSTAT	TWI5 Target Mode Status Register	0x00000000
0x31001210	TWI5_SLVADDR	TWI5 Target Mode Address Register	0x00000000
0x31001214	TWI5_MSTRCTL	TWI5 Controller Mode Registers	0x00000000
0x31001218	TWI5_MSTRSTAT	TWI5 Controller Mode Status Register	0x00000000
0x3100121C	TWI5_MSTRADDR	TWI5 Controller Mode Address Register	0x00000000
0x31001220	TWI5_ISTAT	TWI5 Interrupt Status Register	0x00000000
0x31001224	TWI5_IMSK	TWI5 Interrupt Mask Register	0x00000000
0x31001228	TWI5_FIFOCTL	TWI5 FIFO Control Register	0x00000000
0x3100122C	TWI5_FIFOSTAT	TWI5 FIFO Status Register	0x00000000
0x31001280	TWI5_TXDATA8	TWI5 Tx Data Single-Byte Register	0x00000000
0x31001284	TWI5_TXDATA16	TWI5 Tx Data Double-Byte Register	0x00000000

Table A-207: ADSP-SC59x TWI5 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31001288	TWI5_RXDATA8	TWI5 Rx Data Single-Byte Register	0x00000000
0x3100128C	TWI5_RXDATA16	TWI5 Rx Data Double-Byte Register	0x00000000

Table A-208: ADSP-SC59x UART0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31003004	UART0_CTL	UART0 Control Register	0x00000000
0x31003008	UART0_STAT	UART0 Status Register	0x000000A0
0x3100300C	UART0_SCR	UART0 Scratch Register	0x00000000
0x31003010	UART0_CLK	UART0 Clock Rate Register	0x0000FFFF
0x31003014	UART0_IMSK	UART0 Interrupt Mask Register	0x00000000
0x31003018	UART0_IMSK_SET	UART0 Interrupt Mask Set Register	0x00000000
0x3100301C	UART0_IMSK_CLR	UART0 Interrupt Mask Clear Register	0x00000000
0x31003020	UART0_RBR	UART0 Receive Buffer Register	0x00000000
0x31003024	UART0_THR	UART0 Transmit Hold Register	0x00000000
0x31003028	UART0_TAIP	UART0 Transmit Address/Insert Pulse Register	0x00000000
0x3100302C	UART0_TSR	UART0 Transmit Shift Register	0x000007FF
0x31003030	UART0_RSR	UART0 Receive Shift Register	0x00000000
0x31003034	UART0_TXCNT	UART0 Transmit Counter Register	0x00000000
0x31003038	UART0_RXCNT	UART0 Receive Counter Register	0x00000000

Table A-209: ADSP-SC59x UART1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31003404	UART1_CTL	UART1 Control Register	0x00000000
0x31003408	UART1_STAT	UART1 Status Register	0x000000A0
0x3100340C	UART1_SCR	UART1 Scratch Register	0x00000000
0x31003410	UART1_CLK	UART1 Clock Rate Register	0x0000FFFF
0x31003414	UART1_IMSK	UART1 Interrupt Mask Register	0x00000000
0x31003418	UART1_IMSK_SET	UART1 Interrupt Mask Set Register	0x00000000
0x3100341C	UART1_IMSK_CLR	UART1 Interrupt Mask Clear Register	0x00000000
0x31003420	UART1_RBR	UART1 Receive Buffer Register	0x00000000

Table A-209: ADSP-SC59x UART1 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31003424	UART1_THR	UART1 Transmit Hold Register	0x00000000
0x31003428	UART1_TAIP	UART1 Transmit Address/Insert Pulse Register	0x00000000
0x3100342C	UART1_TSR	UART1 Transmit Shift Register	0x000007FF
0x31003430	UART1_RSR	UART1 Receive Shift Register	0x00000000
0x31003434	UART1_TXCNT	UART1 Transmit Counter Register	0x00000000
0x31003438	UART1_RXCNT	UART1 Receive Counter Register	0x00000000

Table A-210: ADSP-SC59x UART2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31003804	UART2_CTL	UART2 Control Register	0x00000000
0x31003808	UART2_STAT	UART2 Status Register	0x000000A0
0x3100380C	UART2_SCR	UART2 Scratch Register	0x00000000
0x31003810	UART2_CLK	UART2 Clock Rate Register	0x0000FFFF
0x31003814	UART2_IMSK	UART2 Interrupt Mask Register	0x00000000
0x31003818	UART2_IMSK_SET	UART2 Interrupt Mask Set Register	0x00000000
0x3100381C	UART2_IMSK_CLR	UART2 Interrupt Mask Clear Register	0x00000000
0x31003820	UART2_RBR	UART2 Receive Buffer Register	0x00000000
0x31003824	UART2_THR	UART2 Transmit Hold Register	0x00000000
0x31003828	UART2_TAIP	UART2 Transmit Address/Insert Pulse Register	0x00000000
0x3100382C	UART2_TSR	UART2 Transmit Shift Register	0x000007FF
0x31003830	UART2_RSR	UART2 Receive Shift Register	0x00000000
0x31003834	UART2_TXCNT	UART2 Transmit Counter Register	0x00000000
0x31003838	UART2_RXCNT	UART2 Receive Counter Register	0x00000000

Table A-211: ADSP-SC59x UART3 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31003C04	UART3_CTL	UART3 Control Register	0x00000000
0x31003C08	UART3_STAT	UART3 Status Register	0x000000A0
0x31003C0C	UART3_SCR	UART3 Scratch Register	0x00000000
0x31003C10	UART3_CLK	UART3 Clock Rate Register	0x0000FFFF

Table A-211: ADSP-SC59x UART3 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x31003C14	UART3_IMSK	UART3 Interrupt Mask Register	0x00000000
0x31003C18	UART3_IMSK_SET	UART3 Interrupt Mask Set Register	0x00000000
0x31003C1C	UART3_IMSK_CLR	UART3 Interrupt Mask Clear Register	0x00000000
0x31003C20	UART3_RBR	UART3 Receive Buffer Register	0x00000000
0x31003C24	UART3_THR	UART3 Transmit Hold Register	0x00000000
0x31003C28	UART3_TAIP	UART3 Transmit Address/Insert Pulse Register	0x00000000
0x31003C2C	UART3_TSR	UART3 Transmit Shift Register	0x000007FF
0x31003C30	UART3_RSR	UART3 Receive Shift Register	0x00000000
0x31003C34	UART3_TXCNT	UART3 Transmit Counter Register	0x00000000
0x31003C38	UART3_RXCNT	UART3 Receive Counter Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5000	USBC0_OTG_CTL	USBC0 OTG Control and Status Register	0x00010000
0x310C5004	USBC0_OTG_IRQ	USBC0 OTG Interrupt Register	0x00000000
0x310C5008	USBC0_AHB_CFG	USBC0 Bus Configuration Register	0x00000000
0x310C500C	USBC0_CFG	USBC0 USB Configuration Register	0x00001410
0x310C5010	USBC0_RST_CTL	USBC0 Reset Register	0x80000000
0x310C5014	USBC0_ISTAT	USBC0 Interrupt Status Register	0x04000020
0x310C5018	USBC0_IMSK	USBC0 Interrupt Mask Register	0x00000000
0x310C501C	USBC0_RXDBG_STAT	USBC0 Receive Status Debug Read Register	0x00000000
0x310C5020	USBC0_RXDATA_STAT	USBC0 Receive Status Read/Pop Register	0x00000000
0x310C5024	USBC0_RXFIFOSZ	USBC0 Receive FIFO Size Register	0x00000800
0x310C5028	USBC0_TXFIFOSZ_NP	USBC0 Non-periodic Transmit FIFO Size Register	0x08000800
0x310C502C	USBC0_TXFIFO_STAT_NP	USBC0 Non-periodic Transmit FIFO/Queue Status Register	0x00080800
0x310C5034	USBC0_PHYIF_CTL	USBC0 PHY Interface Control Register	0x00000000
0x310C5040	USBC0_MODID	USBC0 Module ID Register	0x4F54400A
0x310C5044	USBC0_HWCFG1	USBC0 User Hardware Configuration 1 Register	0x00000000
0x310C5048	USBC0_HWCFG2	USBC0 User Hardware Configuration 2 Register	0x228BEC90
0x310C504C	USBC0_HWCFG3	USBC0 User Hardware Configuration 3 Register	0x0FA006E8

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5050	USBC0_HWCFG4	USBC0 User Hardware Configuration 4 Register	0xCE002020
0x310C505C	USBC0_DFIFO_CFG	USBC0 DFIFO Configuration Register	0x0FA01000
0x310C5100	USBC0_TXFIFOSZ_PER_H	USBC0 Host Periodic Transmit FIFO Size Register	0x04001000
0x310C5104	USBC0_TXFIFOSZ1_IEP_D	USBC0 Device IN Endpoint 1 Transmit FIFO Size Register	0x01000900
0x310C5108	USBC0_TXFIFOSZ2_IEP_D	USBC0 Device IN Endpoint 2 Transmit FIFO Size Register	0x01000A00
0x310C510C	USBC0_TXFIFOSZ3_IEP_D	USBC0 Device IN Endpoint 3 Transmit FIFO Size Register	0x01000B00
0x310C5400	USBC0_CFG_H	USBC0 Host Configuration Register	0x00000200
0x310C5404	USBC0_FIR_H	USBC0 Host Frame Interval Register	0x0000EA60
0x310C5408	USBC0_FNUM_H	USBC0 Host Frame Number/Frame Time Remaining Register	0x00003FFF
0x310C5414	USBC0_ISTAT_H	USBC0 Host All Channels Interrupt Register	0x00000000
0x310C5418	USBC0_IMSK_H	USBC0 Host All Channels Interrupt Mask Register	0x00000000
0x310C541C	USBC0_FL_BADDR_H	USBC0 Host Frame List Base Address Register	0x00000000
0x310C5440	USBC0_PORT_CTL_H	USBC0 Host Port Control and Status Register	0x00000000
0x310C5500	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5504	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5508	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C550C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5510	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5514	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C551C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C5520	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5524	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5528	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C552C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5530	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5534	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C553C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5540	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5544	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5548	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C554C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5550	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5554	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C555C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C5560	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5564	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5568	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C556C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5570	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5574	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C557C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C5580	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5584	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5588	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C558C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5590	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5594	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C559C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C55A0	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C55A4	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C55A8	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C55AC	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C55B0	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C55B4	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C55BC	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C55C0	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C55C4	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C55C8	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C55CC	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C55D0	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C55D4	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C55DC	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C55E0	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C55E4	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C55E8	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C55EC	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C55F0	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C55F4	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C55FC	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C5600	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5604	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5608	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C560C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5610	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5614	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C561C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C5620	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5624	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5628	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C562C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5630	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5634	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C563C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5640	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5644	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5648	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C564C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5650	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5654	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C565C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C5660	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5664	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5668	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C566C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5670	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5674	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C567C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C5680	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C5684	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C5688	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C568C	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C5690	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C5694	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C569C	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C56A0	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C56A4	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C56A8	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C56AC	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C56B0	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C56B4	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C56BC	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C56C0	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C56C4	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C56C8	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C56CC	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C56D0	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C56D4	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C56DC	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C56E0	USBC0_CHAR[n]_H	USBC0 Host Channel n Characteristics Register	0x00000000
0x310C56E4	USBC0_SPLT_CTL[n]_H	USBC0 Host Channel n Split Control Register	0x00000000
0x310C56E8	USBC0_ISTAT[n]_H	USBC0 Host Channel n Interrupt Status Register	0x00000000
0x310C56EC	USBC0_IMSK[n]_H	USBC0 Host Channel n Interrupt Mask Register	0x00000000
0x310C56F0	USBC0_TSIZ[n]_H	USBC0 Host Channel n Transfer Size Register	0x00000000
0x310C56F4	USBC0_DMA_ADDR[n]_H	USBC0 Host Channel n DMA Address Register	0x00000000
0x310C56FC	USBC0_DMA_BADDR[n]_H	USBC0 Host Channel n DMA Buffer Address Register	0x00000000
0x310C5800	USBC0_CFG_D	USBC0 Device Configuration Register	0x08100000
0x310C5804	USBC0_CTL_D	USBC0 Device Control Register	0x00000002
0x310C5808	USBC0_STAT_D	USBC0 Device Status Register	0x00000002
0x310C5810	USBC0_IMASK_IEP_D	USBC0 Device IN Endpoint Common Interrupt Mask Register	0x00000000
0x310C5814	USBC0_IMASK_OEP_D	USBC0 Device OUT Endpoint Common Interrupt Mask Register	0x00000000
0x310C5818	USBC0_ISTAT_D	USBC0 Device All Endpoint Interrupt Status Register	0x00000000
0x310C581C	USBC0_IMSK_D	USBC0 Device All Endpoint Interrupt Mask Register	0x00000000
0x310C5828	USBC0_VBUSDIS_D	USBC0 Device VBUS Discharge Time Register	0x000017D7
0x310C582C	USBC0_VBUSPULSE_D	USBC0 Device VBUS Pulsing Time Register	0x000005B8
0x310C5830	USBC0_THR_CTL_D	USBC0 Device Threshold Control Register	0x0C100020
0x310C5834	USBC0_IMSK_IEP_FEMPT_D	USBC0 Device IN Endpoint FIFO Empty Interrupt Mask Register	0x00000000
0x310C5900	USBC0_CTL_IEP0_D	USBC0 Device Control IN Endpoint 0 Control Register	0x00008000
0x310C5908	USBC0_ISTAT_IEP0_D	USBC0 Device Control IN Endpoint 0 Interrupt Control Register	0x00000080

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5910	USBC0_TSIZ_IEP0_D	USBC0 Device Control IN Endpoint 0 Transfer Size Register	0x00000000
0x310C5914	USBC0_DMA_ADDR_IEP0_D	USBC0 Device Control IN Endpoint 0 DMA Address Register	0x00000000
0x310C5918	USBC0_TXFSTAT_IEP0_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C591C	USBC0_DMA_BADDR_IEP0_D	USBC0 Device IN Endpoint 0 Buffer Address Register	0x00000000
0x310C5920	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C5928	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C5930	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C5934	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C5938	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C593C	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C5940	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C5948	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C5950	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C5954	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C5958	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C595C	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C5960	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C5968	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C5970	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5974	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C5978	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C597C	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C5980	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C5988	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C5990	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C5994	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C5998	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C599C	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C59A0	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C59A8	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C59B0	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C59B4	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C59B8	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C59BC	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C59C0	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C59C8	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C59D0	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C59D4	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C59D8	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C59DC	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C59E0	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C59E8	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C59F0	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C59F4	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C59F8	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C59FC	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C5A00	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C5A08	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C5A10	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C5A14	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C5A18	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C5A1C	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C5A20	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C5A28	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C5A30	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C5A34	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C5A38	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5A3C	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C5A40	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C5A48	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C5A50	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C5A54	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C5A58	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C5A5C	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C5A60	USBC0_CTL_IEP[n]_D	USBC0 Device Control IN Endpoint n Control Register	0x00000000
0x310C5A68	USBC0_ISTAT_IEP[n]_D	USBC0 Device Control IN Endpoint n Interrupt Control Register	0x00000080
0x310C5A70	USBC0_TSIZ_IEP[n]_D	USBC0 Device Control IN Endpoint n Transfer Size Register	0x00000000
0x310C5A74	USBC0_DMA_ADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Address Register	0x00000000
0x310C5A78	USBC0_TXFSTAT_IEP[n]_D	USBC0 Device Control IN Endpoint Transmit FIFO Status Register	0x00000800
0x310C5A7C	USBC0_DMA_BADDR_IEP[n]_D	USBC0 Device Control IN Endpoint n DMA Buffer Address Register	0x00000000
0x310C5B00	USBC0_CTL_OEP0_D	USBC0 Device OUT Endpoint 0 Control Register	0x00008000
0x310C5B08	USBC0_ISTAT_OEP0_D	USBC0 Device OUT Endpoint 0 Interrupt Register	0x00000000
0x310C5B10	USBC0_TSIZ_OEP0_D	USBC0 Device OUT Endpoint 0 Transfer Size Register	0x00000000
0x310C5B14	USBC0_DMA_ADDR_OEP0_D	USBC0 Device OUT Endpoint 0 DMA Address Register	0x00000000
0x310C5B1C	USBC0_DMA_BADDR_OEP0_D	USBC0 Device OUT Endpoint 0 Buffer Address Register	0x00000000
0x310C5B20	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5B28	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5B30	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5B34	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5B3C	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5B40	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5B48	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5B50	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5B54	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5B5C	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5B60	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5B68	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5B70	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5B74	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5B7C	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5B80	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5B88	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5B90	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5B94	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5B9C	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5BA0	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5BA8	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5BB0	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5BB4	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5BBC	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5BC0	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5BC8	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5BD0	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5BD4	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5BDC	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5BE0	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5BE8	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5BF0	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5BF4	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5BFC	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5C00	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5C08	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5C10	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5C14	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5C1C	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5C20	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5C28	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5C30	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5C34	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5C3C	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5C40	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000
0x310C5C48	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5C50	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5C54	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5C5C	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5C60	USBC0_CTL_OEP[n]_D	USBC0 Device OUT Endpoint n Control Register	0x00000000

Table A-212: ADSP-SC59x USBC0 MMR Register Addresses (Continued)

Memory Mapped Address	Register Name	Description	Reset Value
0x310C5C68	USBC0_ISTAT_OEP[n]_D	USBC0 Device OUT Endpoint n Interrupt Register	0x00000000
0x310C5C70	USBC0_TSIZ_OEP[n]_D	USBC0 Device OUT Endpoint n Transfer Size Register	0x00000000
0x310C5C74	USBC0_DMA_ADDR_OEP[n]_D	USBC0 Device OUT Endpoint n DMA Address Register	0x00000000
0x310C5C7C	USBC0_DMA_BADDR_OEP[n]_D	USBC0 Device OUT Endpoint n Buffer Address Register	0x00000000
0x310C5E00	USBC0_PWR_CTL	USBC0 Power and Clock Gating Control Register	0x00000000

Table A-213: ADSP-SC59x WDOG0 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31008000	WDOG0_CTL	WDOG0 Control Register	0x00000AD0
0x31008004	WDOG0_CNT	WDOG0 Count Register	0x00000000
0x31008008	WDOG0_STAT	WDOG0 Watchdog Timer Status Register	0x00000000
0x3100800C	WDOG0_WIN	WDOG0 Watchdog Timer Window Register	0xFFFFFFFF

Table A-214: ADSP-SC59x WDOG1 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31008800	WDOG1_CTL	WDOG1 Control Register	0x00000AD0
0x31008804	WDOG1_CNT	WDOG1 Count Register	0x00000000
0x31008808	WDOG1_STAT	WDOG1 Watchdog Timer Status Register	0x00000000
0x3100880C	WDOG1_WIN	WDOG1 Watchdog Timer Window Register	0xFFFFFFFF

Table A-215: ADSP-SC59x WDOG2 MMR Register Addresses

Memory Mapped Address	Register Name	Description	Reset Value
0x31009000	WDOG2_CTL	WDOG2 Control Register	0x00000AD0
0x31009004	WDOG2_CNT	WDOG2 Count Register	0x00000000
0x31009008	WDOG2_STAT	WDOG2 Watchdog Timer Status Register	0x00000000
0x3100900C	WDOG2_WIN	WDOG2 Watchdog Timer Window Register	0xFFFFFFFF