SECTION IX

INTERFACING ADCs AND DACs TO DIGITAL SIGNAL PROCESSORS

INTERFACING ADCs AND DACs TO DIGITAL SIGNAL PROCESSORS

SECTION IX

INTERFACING ADCs AND DACs TO DIGITAL SIGNAL PROCESSORS

As the technology in the rapidly growing field of Mixed Signal Processing evolves, more highly integrated DSP products are being introduced (such as the ADSP-21msp50) which contain on-chip ADCs and DACs as well as the DSP, thereby eliminating most component-level interface problems. Stand-alone ADCs and DACs are now available with interfaces especially designed for DSP chips, thereby minimizing or eliminating external interface support or *glue* logic. High performance sigma-delta ADCs and DACs are currently available in the same package (called a codec or coder/decoder) such as the ADSP-28msp02. These products are also designed to require minimum glue logic when interfacing to the most common DSP chips. This section discusses the various data transfer and timing issues associated with the various interfaces.

PARALLEL INTERFACING TO DSP PROCESSORS: *READING* DATA FROM MEMORY-MAPPED PERIPHERAL ADCs

Interfacing an ADC or a DAC to a fast DSP parallel bus (such as the ADSP-2101, ADSP-2100, or the TMS320C25) requires an understanding of how the DSP processor reads data from a memory-mapped peripheral (the ADC) and how the DSP processor writes data to a memory-mapped peripheral (the DAC). We will first consider some general timing requirements for reading and writing data.

A block diagram of a typical parallel DSP interface to an external ADC is shown in Figure 9.1. This diagram has been greatly simplified to show only those signals associated with *reading* data from an external memory-mapped peripheral device. The timing diagram for the ADSP-2101 read-cycle is shown in Figure 9.2.

The *read* process begins when the peripheral device (such as an ADC) asserts the *processor interrupt request line* ($\overline{\text{IRQ}}$). The processor then places the address of the peripheral initiating the interrupt request on the *memory address bus* (A0 - A13). At the same time, the processor asserts the *data memory select line* ($\overline{\text{DMS}}$). The two internal address buses of the ADSP-2101 (program memory address bus and data memory address bus) share a single external address bus, and the two internal data buses (program memory data bus and data memory data bus) share a single external data bus. The *boot memory select* ($\overline{\text{BMS}}$), *data memory select* ($\overline{\text{DMS}}$), and *program memory select* ($\overline{\text{PMS}}$) signals indicate which memory space the external buses are being used for. These signals are typically used to enable an external address decoder as shown in Figure 9.1. The output of the address decoder drives the *chip select* input of the peripheral device.

The *memory read* ($\overline{\text{RD}}$) is asserted t_{ASR} ns after the DMSbar line is asserted. The sum of the address decode delay plus the peripheral chip select setup time should be less than t_{ASR} in order to take full advantage of the time the $\overline{\text{RD}}$ low-time. The $\overline{\text{RD}}$ line remains low for t_{RP} ns. The *memory read* signal is used to enable the tri-state parallel data outputs of the peripheral device. The $\overline{\text{RD}}$ line is connected to the appropriate pin

ADC/ADSP-2101 PARALLEL INTERFACE

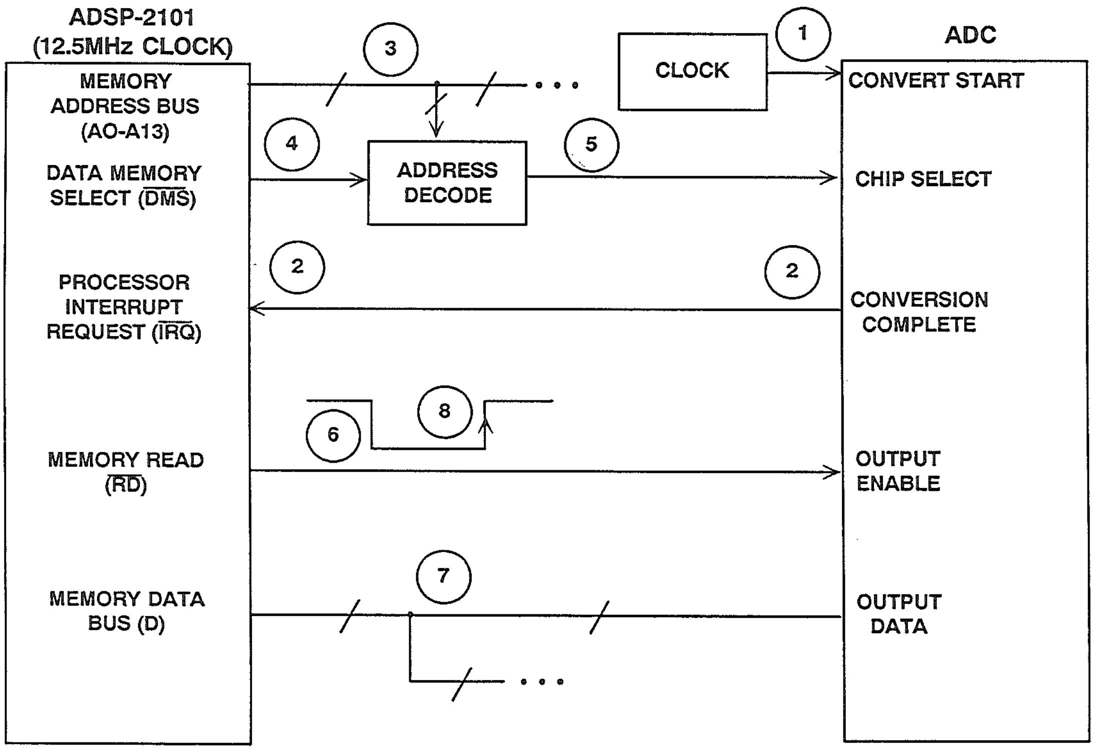


Figure 9.1

ADSP-2101 MEMORY READ TIMING

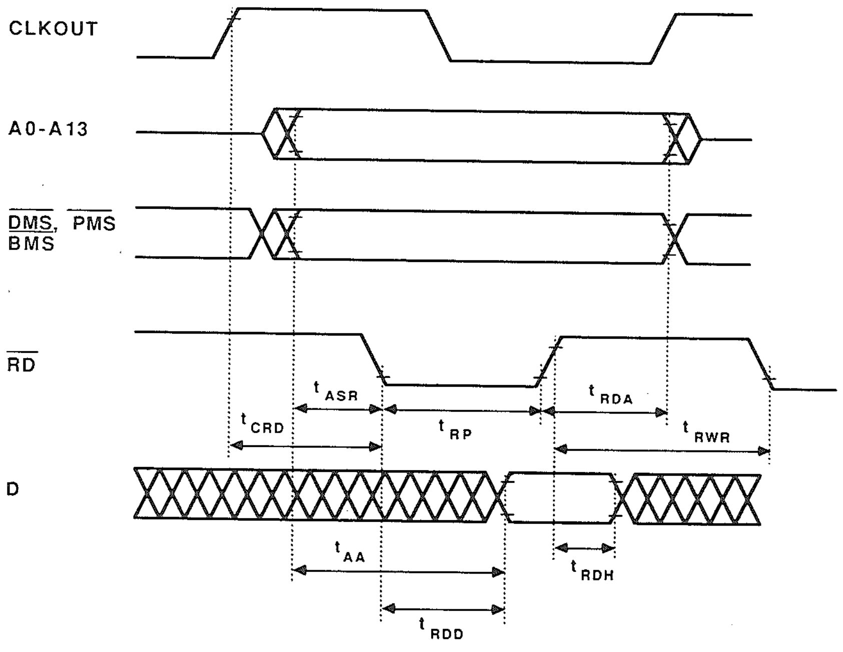


Figure 9.2

on the peripheral device usually called *output enable* or *read*. The rising edge of the $\overline{RD}$ signal is used to clock the data on the data bus into the DSP processor. After the rising edge of the $\overline{RD}$ signal, the data on the data bus must remain valid for t_{RDH} ns, the data hold time. In the case of the ADSP-2101, this value is 0ns.

The key timing requirements for the peripheral device are shown in Figure 9.3.

PARALLEL PERIPHERAL DEVICE READ INTERFACE KEY REQUIREMENTS

- Peripheral Device Data Outputs Must Be Tri-State
- Address Decode Delay Plus Peripheral Chip Select Setup Time Must Be Less Than Address and Data Memory Setup Times (5ns min for ADSP-2101)
- Access Time from Negative-Going Edge of Memory Read Pulse ($\overline{RD}$) Until Output Data Valid Must be Less than t_{RDD} (25ns max for ADSP-2101 Operating at 12.5MHz). Otherwise Software Wait States Must be Added, or Processor Clock Frequency Reduced.
- Output Data Must Remain Valid for t_{RDH} (0ns for ADSP-2101)
- Peripheral Device Must Accept Minimum Output Enable PulseWidth of t_{RP} (30ns for ADSP-2101 Operating at 12.5MHz).

Figure 9.3

The t_{RDD} specification determines the peripheral device data access requirement. In the case of the ADSP-2101, t_{RDD} = 25ns minimum. If the access time of the peripheral is greater than this, wait states must be added or the processor speed reduced. The relationship between these parameters for the ADSP-2101 is given by the following equations:

ADSP-2101 PARALLEL READ TIMING

- t_{ASR} = Address and Data Memory Select Setup Before READ DATA low
- $t_{ASR} = 0.25t_{CK} - 15\text{ns}$ Minimum
- t_{RDD} = READ DATA LOW to Data Valid
- t_{CK} = Processor Clock Period (80ns Minimum)
- $t_{RDD} = 0.5t_{CK} - 15\text{ns} + \#\text{ wait states} * t_{CK}$ Maximum

Figure 9.4

9

The ADSP-2101 can easily be interfaced to slow peripheral devices using its programmable wait state generation capability. Three registers control wait state generation for Boot, Program, and Data Memory interfaces. You can specify 0 to 7 wait states for each parallel memory interface. Each wait state added increases the allowable external data memory access time by an amount equal to the processor clock period (80ns for the ADSP-2101 operating at 12.5MHz). The Data Memory Address, $\overline{\text{DMS}}$, and $\overline{\text{RD}}$ lines are all held stable for an additional amount of time equal to the duration of the wait states.

Parallel ADC to DSP Interface

The conversion process in a sampling ADC is initiated by a pulse often called the *encode command, or start-convert.* The leading (or trailing) edge of this pulse causes the internal ADC sample-and-hold to switch from the sample mode to the hold mode so that the conversion process can take place. Extreme care must be taken in order to insure that this pulse is both jitter-free and noise-free. Any sample-to-sample variation in the occurrence of this edge has the same effect as aperture jitter and will produce a corresponding degradation in the overall ADC signal-to-noise ratio. For this reason, the start-convert signal is usually generated by a stable source external to the DSP processor.

The various timing pulses required to carry out the actual internal conversion process (after receipt of the *convert-start* command) may be generated in several ways depending upon the individual ADC design. In some ADCs, the *convert-start* pulse triggers an internal oscillator or timing chain which in turn controls the conversion. In other ADCs, a user-supplied asynchronous external clock is required.

At some point in time after the convert-start pulse edge, the internal ADC conversion process is completed. In the case of a parallel-output ADC, a single pulse called *data valid, data ready, read data, conversion complete, end-of-conversion,or busy / interrupt* is asserted. This pulse is used to drive an *interrupt request* input of the DSP processor as shown in the ADC/DSP parallel interface in Figure 9.5. The DSP then places the address of the ADC on the *data memory address bus* and asserts the *data memory select* line which in turn enables the address decoder. The *chip select* input to the ADC is then asserted along with the *read data* line from the DSP. The *read data* line is connected to the *read* input of the ADC. Asserting the *read* line to the ADC enables the tri-state parallel outputs which are connected to the *data memory data bus*. The DSP then reads the ADC data into an internal register on the rising edge of the *read data* pulse. In order for the circuit shown in Figure 9.5 to operate properly, the timing between the two devices must be made compatible. This will be illustrated by considering a representative example of the ADSP-2101 processor interfaced to the AD7871 ADC.

The AD7871 is a 14 bit, 83kSPS ADC which can operate in either the parallel or serial mode. A functional block diagram of the AD7871 is shown in Figure 9.6. The key interface timing specifications for the two devices are compared in Figure 9.7. Specifications for the ADSP-2101 are given for a clock frequency of 12.5MHz.

ADC/ADSP-2101 PARALLEL INTERFACE

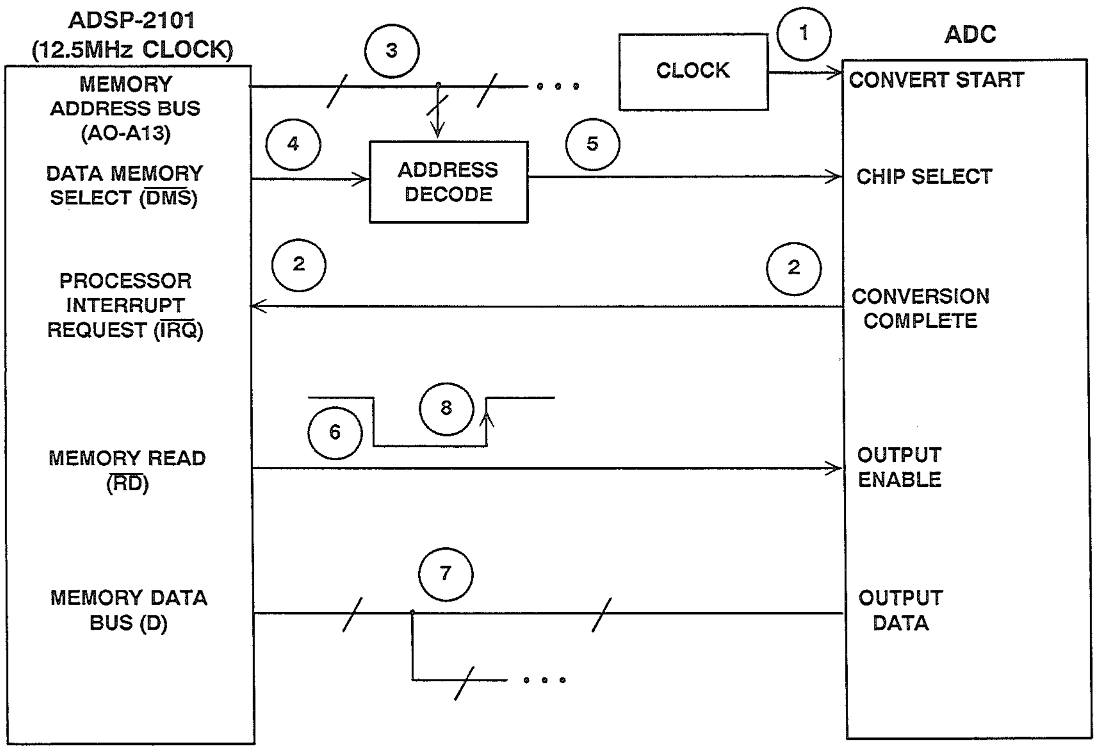


Figure 9.5

AD7871 14-BIT, 83 kSPS ADC FUNCTIONAL DIAGRAM

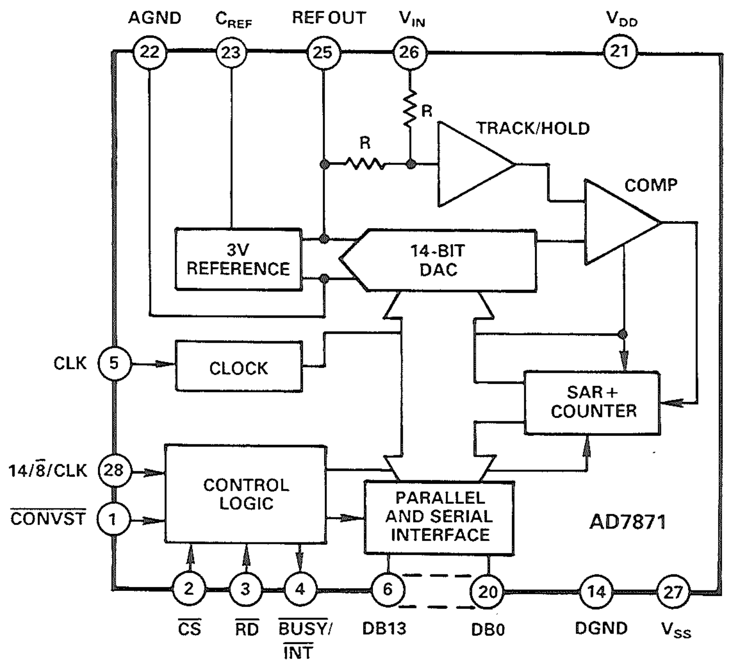


Figure 9.6

ADSP-2101 AND AD7871 PARALLEL *READ* INTERFACE TIMING SPECIFICATIONS

ADSP-2101 PROCESSOR (12.5MHz)	AD7871 ADC
t_{ASR} (Data Address, Data Memory Select Setup Time Before $\overline{RD}$ Low) = 5ns min	t_2 ($\overline{CS}$ to $\overline{RD}$ Setup Time) = 0ns min (Must Add Address Decode Time to This Value)
t_{RP} ($\overline{RD}$ Pulse Width) = 30ns + # wait states * 80ns min	t_3 ($\overline{RD}$ Pulse Width) = 60ns min
t_{RDD} ($\overline{RD}$ Low to Data Valid) = 25ns + # wait states * 80ns min	t_6 (Data Access Time After $\overline{RD}$) = 57ns max
t_{RDH} (Data Hold from $\overline{RD}$ High) = 0ns min	t_7 (Bus Relinquish Time after $\overline{RD}$) = 5ns min

Figure 9.7

Examining the timing specifications shown in Figure 9.7 reveals that for the timing between the devices to be compatible, one software wait state must be programmed into the ADSP-2101. A simplified interface diagram for the two devices is shown in Figure 9.8. The conversion complete signal from the AD7871 is designated BUSY/INT.

Parallel interfaces with other DSP processors can be designed in a similar manner by carefully examining the timing specifications for all appropriate signals for each device. The interface between the ADSP-2100 microprocessor (J-Grade, 6.144MHz Clock) is shown in Figure 9.9. Interfacing the AD7871 ADC to faster versions of the ADSP-2100 series requires the addition of wait states using the *Data Memory Acknowledge* (DMACK) signal. The DMACK signal indicates that the memory-mapped peripheral is ready for data transfer. If DMACK is not asserted when checked by the processor, wait states are automatically generated until DMACK is asserted. A detailed description wait state generation in the ADSP-2100 using the DMACK signal and the external hardware required is given in Reference 1 (also included at the end of this section).

The parallel interface between the AD7871 and the TMS32020/C25 is shown in Figure 9.10.

AD7871 ADC PARALLEL INTERFACE TO ADSP-2101

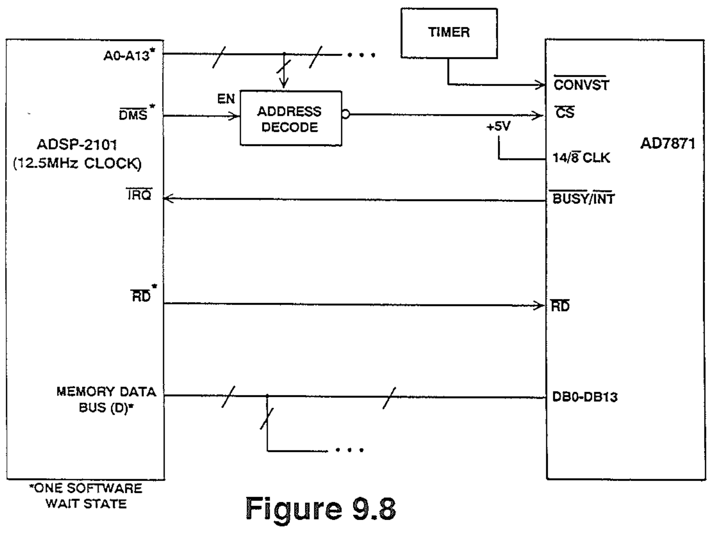


Figure 9.8

AD7871 PARALLEL INTERFACE TO ADSP-2100

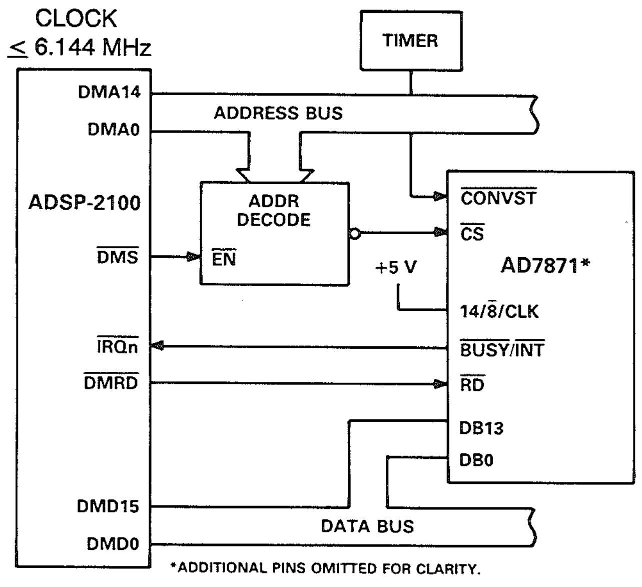


Figure 9.9

AD7871 PARALLEL INTERFACE TO TMS320/C25

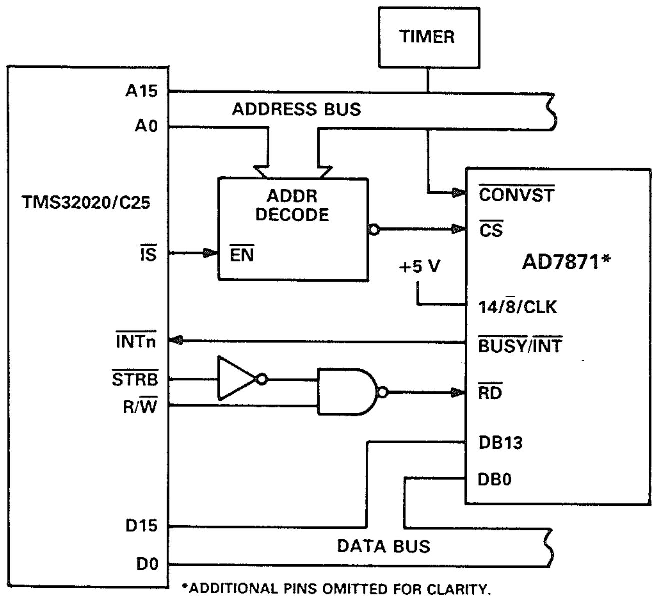


Figure 9.10

PARALLEL INTERFACING TO DSP PROCESSORS: *WRITING* DATA TO MEMORY-MAPPED DACS

A simplified block diagram of a typical DSP interface to a peripheral device showing write-mode signals is shown in Figure 9.11. The memory-write cycle timing diagram for the ADSP-2101 is shown in Figure 9.12. The write process may be initiated by the peripheral device by asserting the DSP *interrupt request* line indicating that the peripheral is ready to accept a new parallel data word. The DSP then places the address of the peripheral device on the *address bus* and asserts the *data memory select* ($\overline{\text{DMS}}$) line. This causes the output of the address decoder to assert the *chip select* input to the peripheral. The *write* ($\overline{\text{WR}}$) output of the DSP is asserted t_{ASW} ns after the negative-going edge of the $\overline{\text{DMS}}$ signal. The width of the $\overline{\text{WR}}$ pulse is t_{WP} ns. Data is placed on the data bus (D) and is valid t_{DW} ns before the $\overline{\text{WR}}$ line goes high. The positive-going transition of the $\overline{\text{WR}}$ line is used to clock the data on the data bus (D) into the external parallel memory. The data on the data bus remains valid for t_{DH} ns after the positive-going edge of the $\overline{\text{WR}}$ signal. The key timing requirements for the peripheral device are shown in Figure 9.13.

DAC/ADSP-2101 PARALLEL INTERFACE

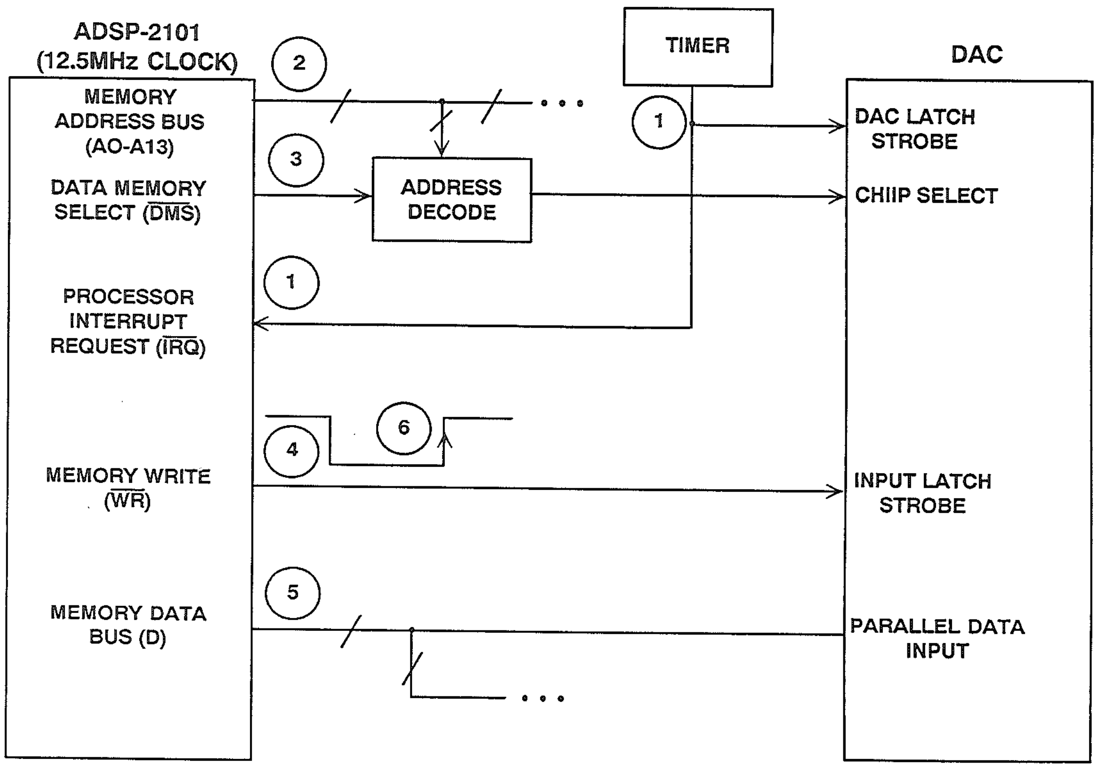


Figure 9.11

ADSP-2101 MEMORY WRITE TIMING

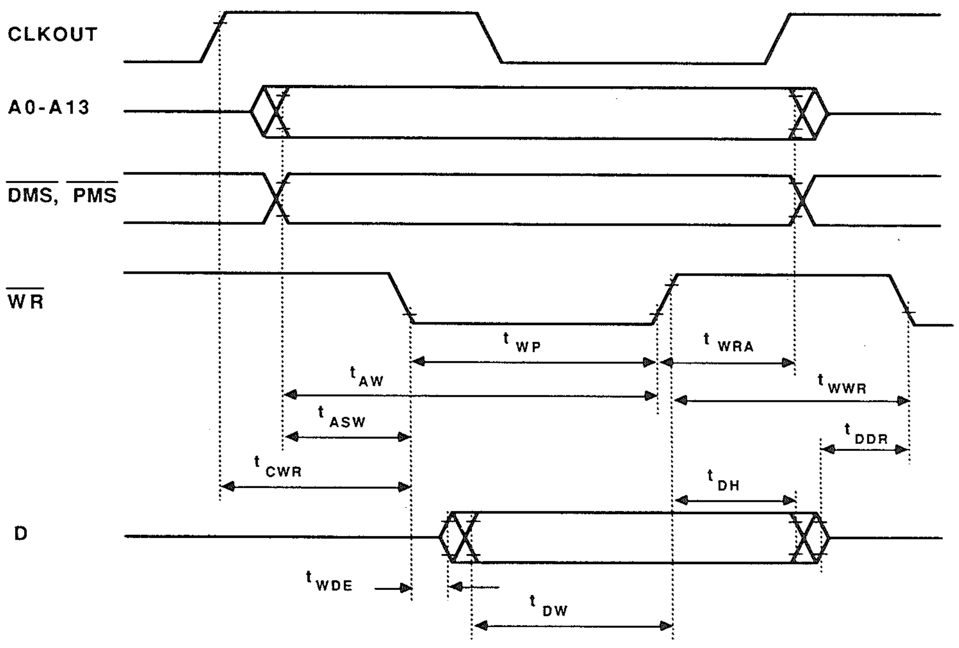


Figure 9.12

PARALLEL PERIPHERAL DEVICE WRITE INTERFACE KEY REQUIREMENTS

- Address Decode Time Plus Peripheral Chip Select Setup Time Must be Less Than Address and Data Memory Select Setup Time t_{ASW} (5ns for ADSP-2101 Operating at 12.5MHz)
- Input Data *Setup* Time Must be Less Than t_{DW} (20ns for ADSP-2101 Operating at 12.5MHz)
- Input Data *Hold* Time Must be Less Than t_{DH} (10ns for ADSP-2101 Operating at 12.5MHz)
- Peripheral Device Must Accept Input Write Clock Pulse of Width t_{WP} (30ns min for ADSP-2101 Operating at 12.5MHz)

Figure 9.13

If any of the timing constraints shown in Figure 9.13 are violated by the peripheral device, wait states must be added or the processor speed reduced. The relationship between these parameters for the ADSP-2101 are shown in Figure 9.14.

ADSP-2101 PARALLEL WRITE TIMING

- t_{CK} = Processor Clock Period (80ns Minimum)
- t_{ASW} = Address and Data Memory Select Time Before $\overline{WR}$ Low = $0.25t_{CK}$ -15ns Minimum
- t_{DW} = Data Setup Before $\overline{WR}$ High = $0.5t_{CK}$ - 20ns + #Wait States $*$ t_{CK}
- t_{DH} = Data Hold After $\overline{WR}$ High = $0.25t_{CK}$ - 10ns
- t_{WP} = $\overline{WR}$ Pulse Width = $0.5t_{CK}$ -10ns + #Wait States$*$ t_{CK}

Figure 9.14

The ADSP-2101 can easily be interfaced to slow peripheral devices using its programmable wait state generation capability which causes the Memory Address, $\overline{DMS}$, $\overline{WR}$, and Data Output lines to remain stable for an amount of additional time equal to the duration of the wait states.

PARALLEL DAC TO DSP INTERFACE

A typical parallel interface between a DSP and a DAC is shown in Figure 9.15. In most DSP applications the DAC is operated continuously from a stable clock source which is external to the DSP processor. The DAC should have double buffering: an input latch to handle the asynchronous DSP interface, and a second latch which drives the DAC current switches. The DAC latch strobe is derived from the external stable clock. In addition to clocking the DAC latch, the DAC latch strobe is also used to generate a processor interrupt which indicates the DAC is ready for new input data. The processor then asserts the *data memory select* line and places the DAC address on the memory address bus. The DAC *chip select* is then asserted, and the *data memory write* line loads the next data word on the data memory data bus into the DAC input latch. This completes the write cycle, and the DAC is now ready to receive the next DAC latch strobe from the external source. In order for the circuit shown in Figure 9.15 to operate properly, the timing between the two devices must be made compatible. This will be illustrated by considering a representative example of the ADSP-2101 processor interfaced to the AD7840 DAC.

The AD7840 is a 14 bit 100kSPS DAC which has both parallel and serial interface capability. A block diagram of the device is shown in Figure 9.16. The key interface timing specifications for the two devices are compared in Figure 9.17. Specifications for the ADSP-2101 are given for a clock frequency of 12.5MHz.

DAC/ADSP-2101 PARALLEL INTERFACE

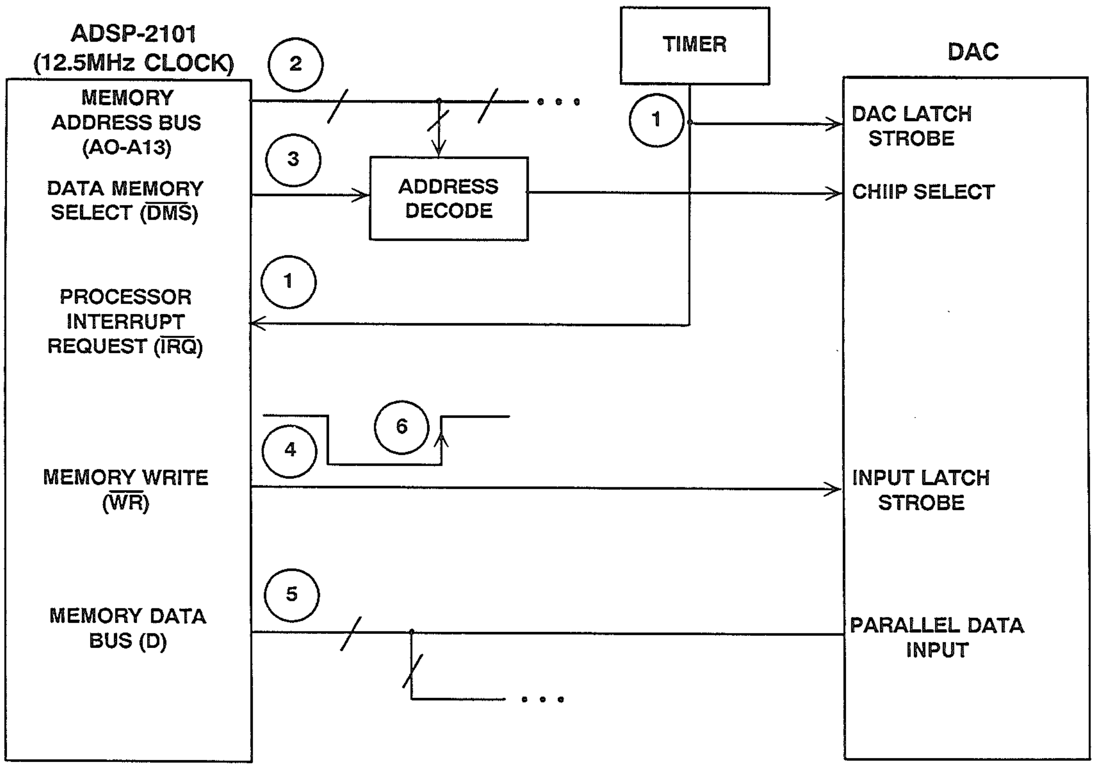

ADSP-2101
(12.5MHz CLOCK)
MEMORY ADDRESS BUS (AO-A13)
DATA MEMORY SELECT ($\overline{DMS}$)
PROCESSOR INTERRUPT REQUEST ($\overline{IRQ}$)
MEMORY WRITE ($\overline{WR}$)
MEMORY DATA BUS (D)
TIMER
ADDRESS DECODE
DAC
DAC LATCH STROBE
CHIIP SELECT
INPUT LATCH STROBE
PARALLEL DATA INPUT
1
2
3
4
5
6

Figure 9.15

AD7840 14-BIT, 100 kSPS DAC FUNCTIONAL DIAGRAM

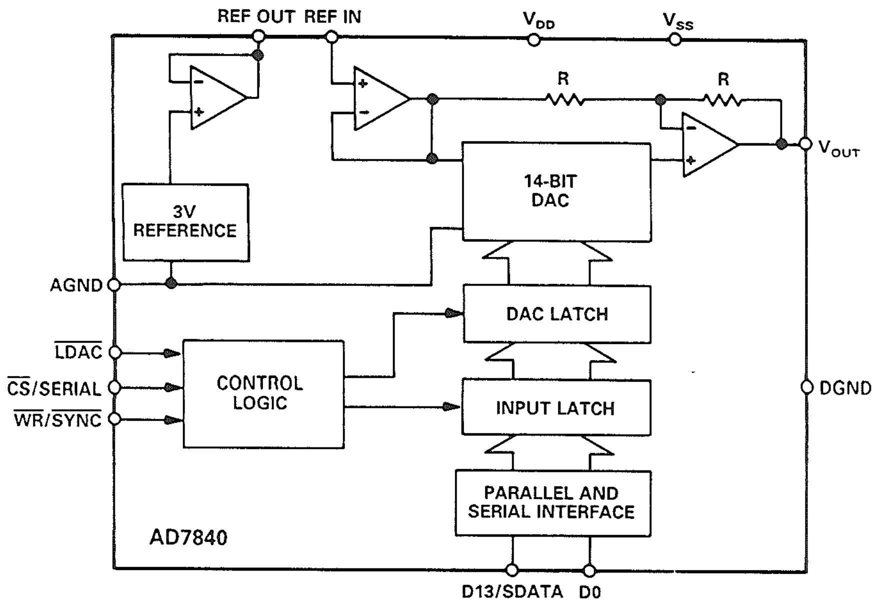


Figure 9.16

ADSP-2101 AND AD7840 PARALLEL *WRITE* INTERFACE TIMING SPECIFICATIONS

ADSP-2101 PROCESSOR (12.5MHz)	AD7840 DAC
t_{ASW} (Address and Data Memory Select Setup Before $\overline{WR}$ Low = 5ns min	t_1 = $\overline{CS}$ to $\overline{WR}$ Setup Time = 0ns min (Must Add Address Decode Time)
t_{WP} ($\overline{WR}$ Pulse Width) = 30ns + # wait states * 80ns min	t_3 ($\overline{WR}$ Pulse Width) = 45ns min
t_{DW} (Data Setup Before $\overline{WR}$ High) = 20ns + # wait states * 80ns min	t_4 (Data Valid to $\overline{WR}$ Setup Time) = 21ns min
t_{DH} (Data Hold After $\overline{WR}$ High) = 10ns	t_5 (Data Valid to $\overline{WR}$ Hold Time) = 10ns min

Figure 9.17

Examining the timing specifications shown in Figure 9.17 reveals that for the timing between the devices to be compatible, at least one software wait state must be programmed into the ADSP-2101. A simplified interface diagram for the two devices is shown in Figure 9.18.

Parallel interfaces with other DSP processors can be designed in a similar manner by carefully examining the timing specifications for all appropriate signals for each device.

The interface between the ADSP-2100 microprocessor (clock speeds up to 8.192MHz) is shown in Figure 9.19. Interfacing the AD7840 to the ADSP-2100A at clock speeds of greater than 8.144MHz requires the addition of wait states using the ADSP-2100 DMACK signal as described in Reference 1.

The parallel interface between the AD7840 DAC and the TMS32020/C25 is shown in Figure 9.20.

AD7840 DAC PARALLEL INTERFACE TO ADSP-2101

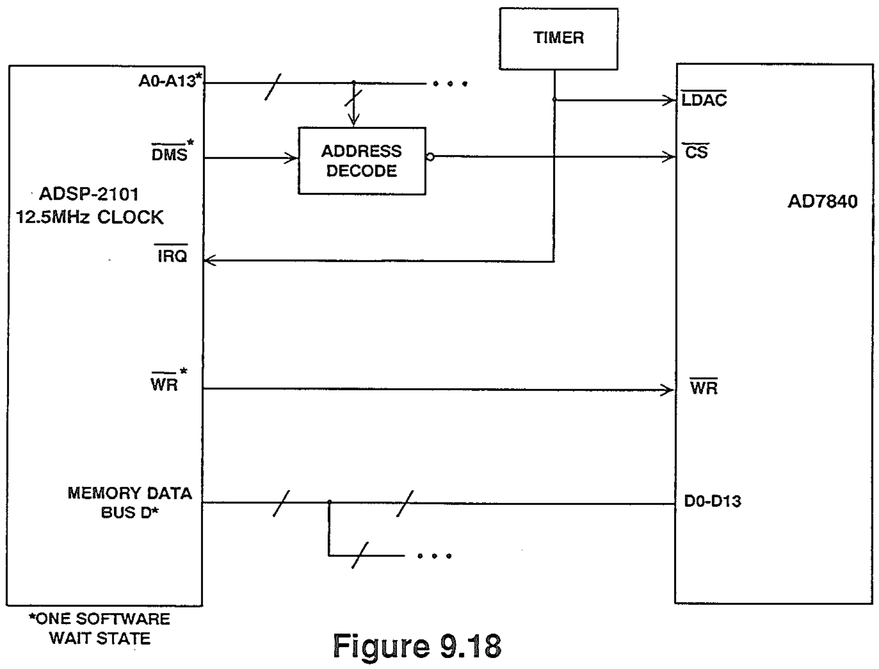


Figure 9.18

AD7840 PARALLEL INTERFACE TO ADSP-2100

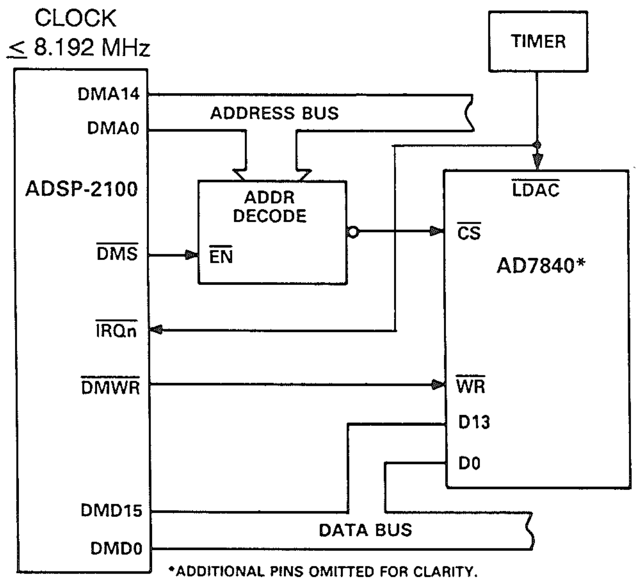


Figure 9.19

AD7840 PARALLEL INTERFACE TO TMS32020

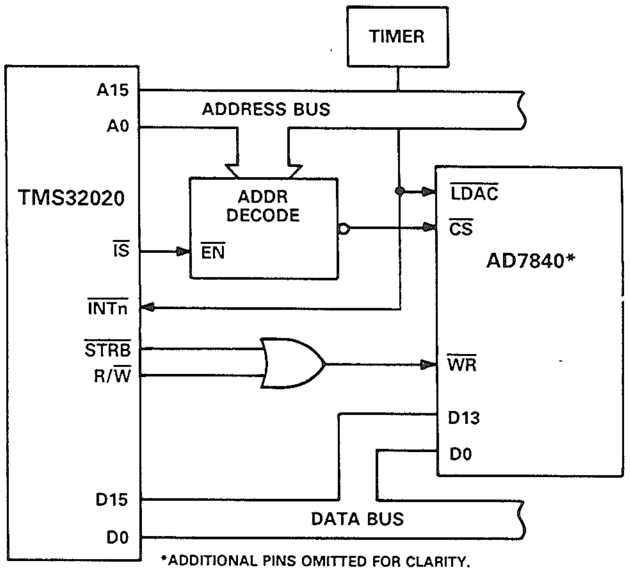


Figure 9.20

SERIAL INTERFACING TO DSP PROCESSORS

DSP processors which have serial ports (such as the ADSP-2101, DSP56000, and the TMS32020/C25) provide a simple interface to peripheral ADCs and DACs. Use of the serial port eliminates the need for using large parallel buses to connect the ADCs and DACs to the DSP. In order to understand serial data transfer better, we will first examine the serial port operation of the ADSP-2101.

A block diagram of one of the two serial ports of the ADSP-2101 is shown in Figure 9.21. The *Transmit* (TX) and *Receive* (RX) registers are identified by name in the ADSP-2101 assembly language, not memory mapped.

In the receiving portion of the serial port, the *Receive Frame Synchronization* (RFS) signal initiates reception. The serial *Receive Data* (DR) from the external device (ADC) is transferred into the *Receive Shift Register* one bit at a time. The negative-going edge of the *Serial Clock* (SCLK) is used to clock the serial data from the external device into the *Receive Shift Register*. When a complete word has been received, it is written to the *Receive Register* (RX), and the receive interrupt for that serial port is generated. The *Receive Register* is then read by the processor.

Writing to the *Transmit Register* readies the serial port for transmission. The *Transmit Frame Synchronization* (TFS) signal initiates transmission. The value in the Transmit Register (TX) is then written to the internal *Transmit Shift Register.* The data in the *Transmit Shift Register* is sent to the peripheral device (ADC) one bit at a time, and the positive-going edge of the *Serial Clock* (SCLK) is used to clock the serial *Transmit Data* (DT) into the external device. When the first bit has been transferred, the Serial Port generates the transmit interrupt. The *Transmit Register* can then be written with new data, even though the transmission of the previous data is not complete.

In the *normal* framing mode, the frame sync signal (RFS or TFS) is checked at the falling edge of SCLK. If the framing signal is asserted, data is available (transmit mode) or latched (receive mode) on the *next* falling edge of SCLK. The framing signal is not checked again until the word has been transmitted or received. In the *alternate* framing mode, the framing signal is asserted in the *same* SCLK cycle as the first bit of a word. The data bits are latched on the falling edge of SCLK, but the framing signal is checked only on the first bit. Internally-generated framing signals remain asserted for the length of the serial word. The *alternate* framing mode of the serial port in the ADSP-2101 is normally used to receive data from ADCs and transmit data to DACs.

The key features of the ADSP-2101 serial ports are summarized in Figure 9.22.

ADSP-2101 SERIAL PORT BLOCK DIAGRAM

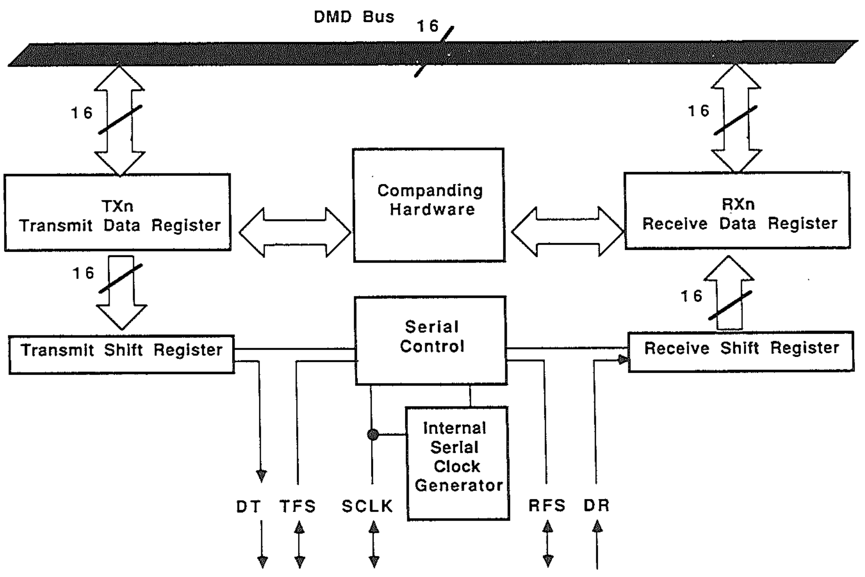


Figure 9.21

ADSP-2101 SERIAL PORTS KEY FEATURES

- Separate Transmit and Receive Sections for Each Port
- Double-Buffered Transmit and Receive Registers
- Serial Clock Can Be Internally (up to 6.25MHZ) or Externally (up to 12.5MHz) Generated
- Transmit and Receive Frame Sync Signals Can be Externally or Internally Generated
- Serial Data Words of 3 to 16 Bits Supported
- Automatically Generated Processor Interrupts
- Hardware Companding Capability

Figure 9.22

SERIAL ADC TO DSP INTERFACE

A timing diagram of the ADSP-2101 serial port operating in the receive mode (alternate framing) is shown in Figure 9.23. The first negative-going edge of the SCLK to occur after the rising edge of the RFS input clocks the MSB data from the ADC into the serial input latch. The process continues until all serial bits have been transferred into the serial input latch. The key timing specifications of concern are the serial data setup

(t_{SCS}) and hold times (t_{SCH}) with respect to the negative-going edge of the SCLK. In the case of the ADSP-2101, these values are both 10ns minimum. The RFS setup and hold times are also 10ns, respectively. Most peripheral ADCs will have no trouble meeting these specifications, even at the maximum serial data transfer rate of 12.5MHz.

The AD7872 ADC is a 14 bit, 83kSPS serial-only version of the AD7871. A block diagram of the device is shown in Figure 9.24. The device operates on a 2MHz external or internal clock. Figure 9.25 shows the AD7872 interfaced to the ADSP-2101. The timing diagram of the AD7872 is shown in Figure 9.26. The SSTRBbar signal is active-low, so the ADSP-2101 must be programmed to accept an inverted RFS input. The serial clock operates at a frequency of 2MHz (500ns period). The serial clock can be programmed for either continuous or gated operation. In this example, it operates in the continuous mode. The data bits are valid t_{12} ns (155ns max) after the positive-going edges of SCLK. This allows a setup time of 250 - 155 = 95ns minimum before the negative-going edges of SCLK. The hold-time after the negative-going edge of SCLK is therefore at least equal to one-half the clock period, or 250ns. The positive-going edge of the $\overline{\text{SSTRB}}$ signal occurs t_{13} ns (140ns max) after the positive-going edge of SCLK after the last data bit is transferred. This allows 250 - 140 = 110ns minimum before the next negative-going edge of SCLK. These simple calculations show that the data and RFS setup and hold requirements of the ADSP-2101 (10ns) are met with considerable margin.

The ADSP-2101 can be easily programmed to generate the 2MHz serial clock for the AD7872 if desired. Details can be found in the ADSP-2101 User's Manual/ Architecture. The Convert Start ($\overline{\text{CONVST}}$) signal is generated externally to the AD7872 from a stable clock source which is asynchronous to the Serial Clock.

The serial interface between the AD7872 and the DSP56000 is shown in Figure 9.27, and the interface with the TMS32020/C25 is shown in Figure 9.28. The simple interfaces shown to the three DSP processors are referred to as *zero-chip interfaces* because no additional *glue* logic is required.

ADSP-2101 SERIAL PORT RECEIVE TIMING

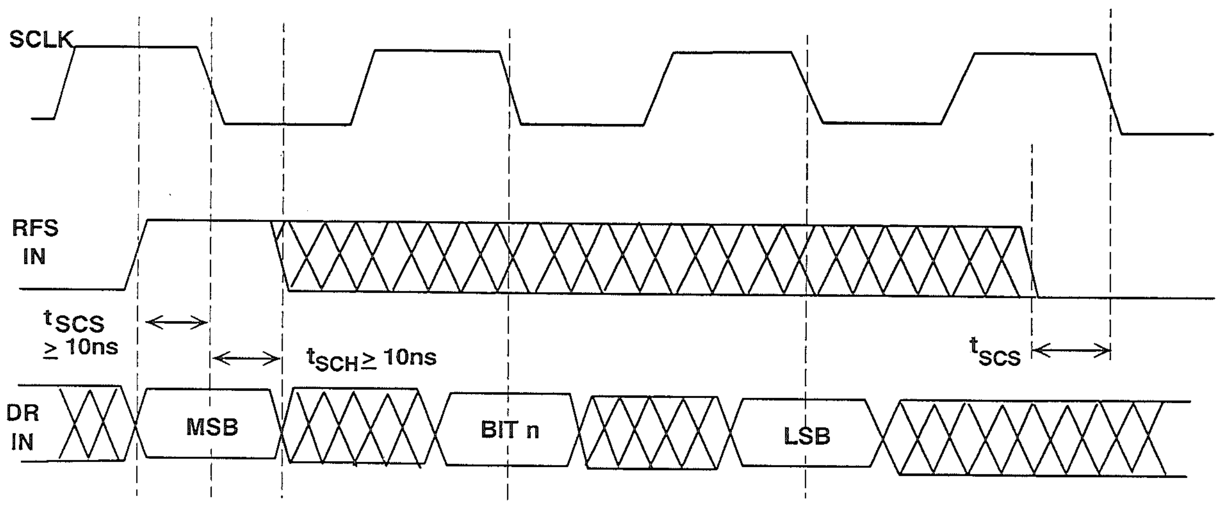


Figure 9.23

AD7872 14-BIT, 83kSPS SERIAL OUTPUT ADC BLOCK DIAGRAM

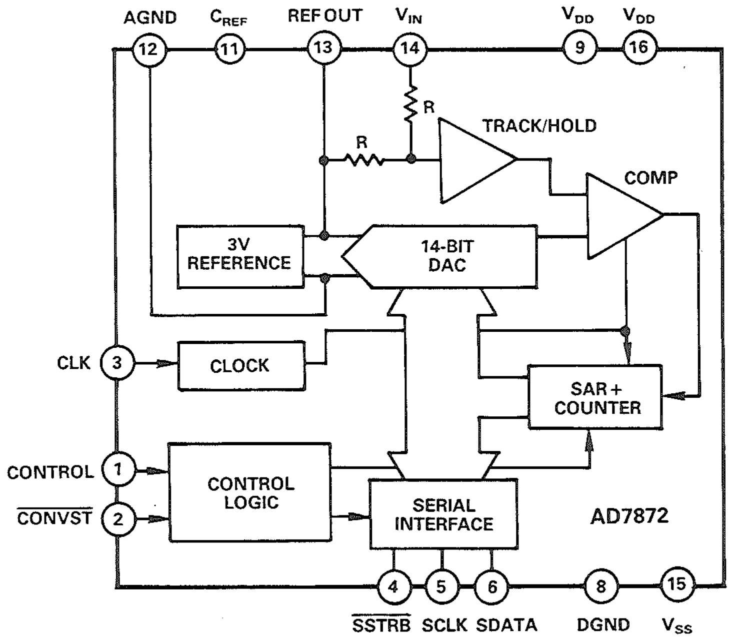
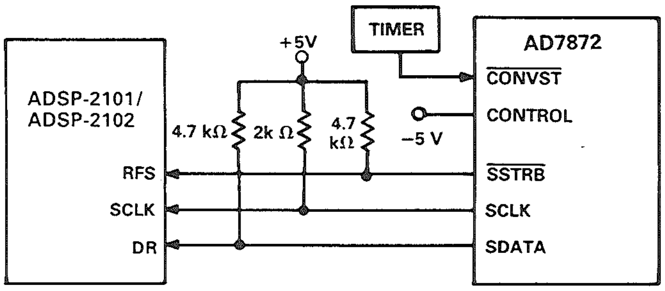


Figure 9.24

AD7872 SERIAL INTERFACE TO ADSP-2101



*ADDITIONAL PINS OMITTED FOR CLARITY.

Figure 9.25

AD7872 ADC SERIAL INTERFACE TIMING

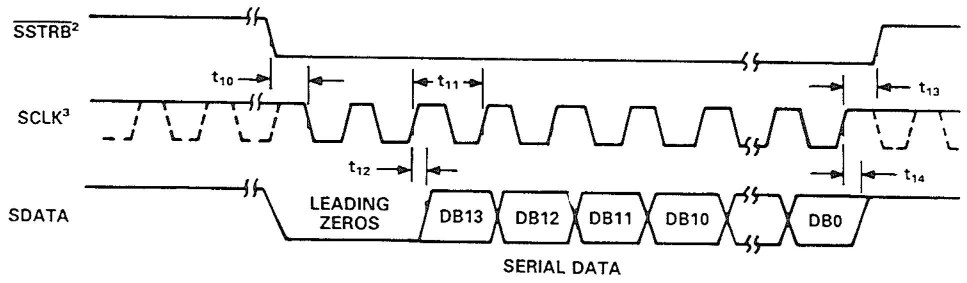


Figure 9.26

AD7872 SERIAL INTERFACE TO DSP56000

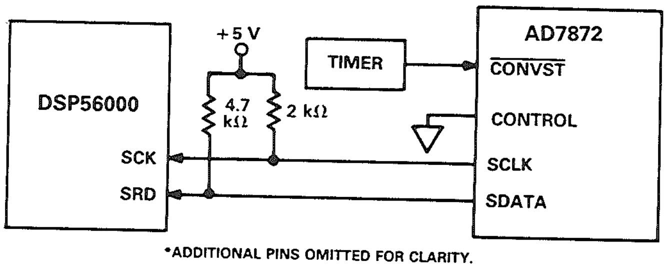


Figure 9.27

AD7872 SERIAL INTERFACE TO TMS32020/C25

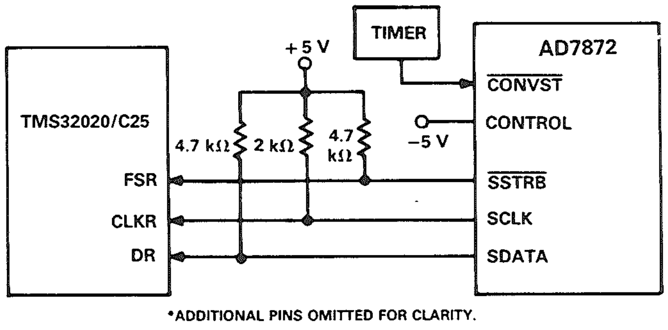


Figure 9.28

SERIAL DAC TO DSP INTERFACE

A timing diagram of the ADSP-2101 serial port operating in the alternate framing transmit mode (with internally generated Transmit Frame Sync) is shown in Figure 9.29. The first negative-going edge of the SCLK to occur after the rising edge of the TFS output clocks the MSB data from the serial port into the DAC serial input latch. The process continues until all serial bits have been transferred into the DAC serial input latch. The key timing specifications of concern are the data output setup and hold times with respect to the negative-going edge of the SCLK. The ADSP-2101 specifies that the TFS output will be a valid high t_{RH} ns (15ns max) after the positive-going edge of SCLK. The serial transmit data is valid t_{SCDV} ns (25ns max) after the positive-going edge of SCLK. Due to the high speed of the serial port interface of the ADSP-2101, data setup and hold times are therefore approximately equal to one-half the period of the serial clock for clock rates up to 12.5MHz.

The AD766 is a 16 bit serial DAC which can operate at sample rates up to 500kSPS and is fully specified in terms of both dc and ac parameters such as THD and SNR. A block diagram of the device is shown in Figure 9.30. Data is transmitted to the AD766 in a bit stream composed of 16 bit words with a serial, MSB first format. Three signals must be present to achieve proper operation: the *data*, *clock*, and *latch enable* signals. Input data bits are clocked into the input register on the falling edge of the clock signal. The LSB is clocked in on the 16th clock pulse. When all data bits are loaded, a low-going *latch enable* pulse updates the DAC input. Figure 9.31 illustrates the general signal requirements for data transfer for the AD766. Data setup and hold-times (with respect to the negative-going SCLK edge) are each 15ns. The negative-going edge of the latch enable must occur at least 15ns before the negative-going edge of SCLK. These detailed timing requirements are

ADSP-2101 SERIAL PORT TRANSMIT TIMING

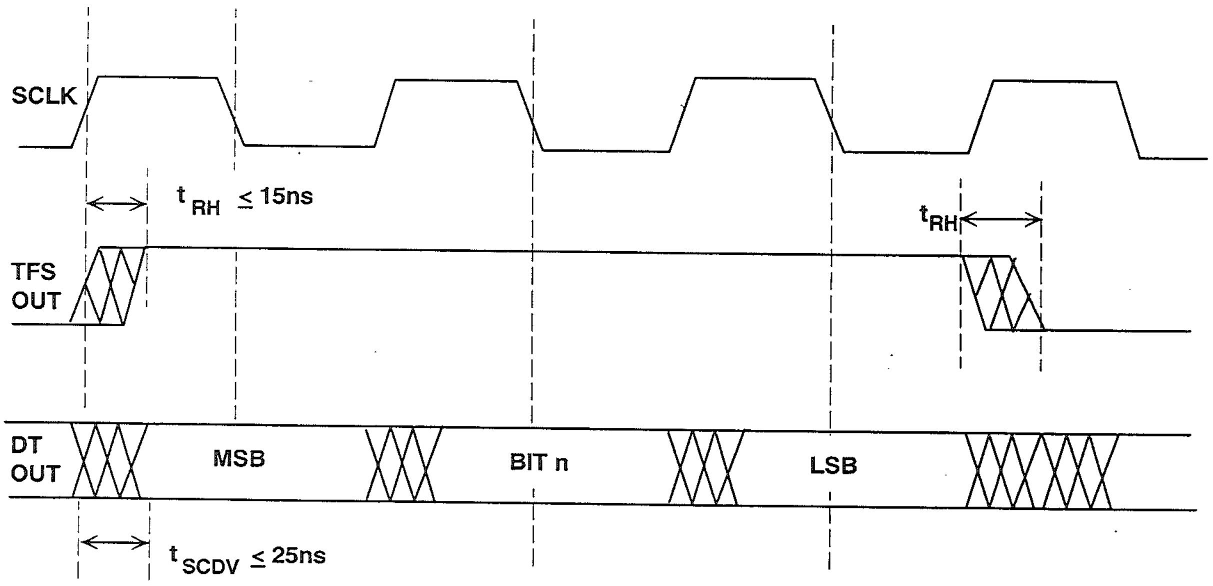


Figure 9.29

AD766 16-BIT, 500kSPS DSP DAC FUNCTIONAL DIAGRAM

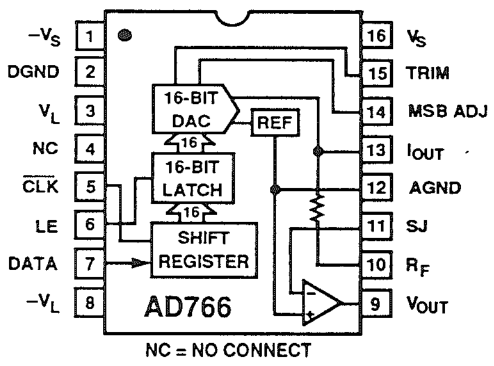


Figure 9.30

AD766 DAC SIGNAL REQUIREMENTS

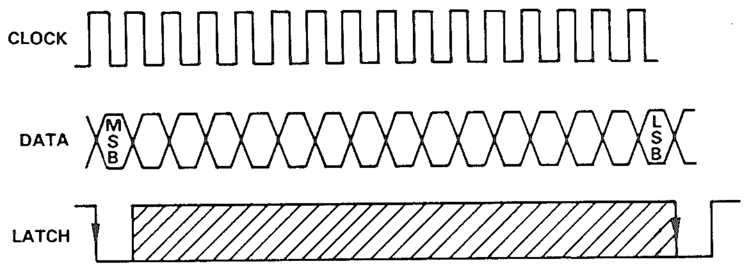


Figure 9.31

AD766 TIMING REQUIREMENTS

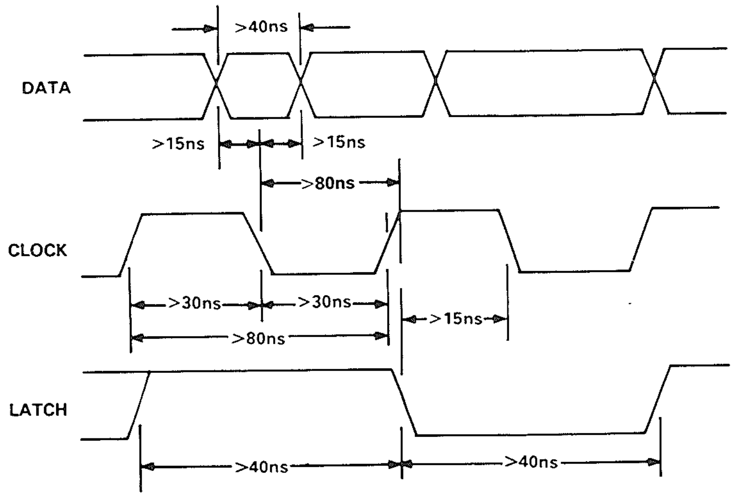


Figure 9.32

illustrated in Figure 9.32. These timing requirements are compatible with the serial ports of popular DSP processors. The AD766 input clock can run at a 12.5MHz rate. This clock rate will allow sampling rates up to 500kSPS.

The ADSP-2101 incorporates two complete serial ports which can be directly interfaced to the AD766 as shown in Figure 9.33. Using both serial ports, two AD766's can be directly interfaced with no additional hardware. The zero-chip interface to the TMS320C25 is shown in Figure 9.34. The maximum serial clock rate for the TMS320C25 is 5MHz. Figure 9.35 shows the serial interface to the DSP56000/56001.

AD766 SERIAL INTERFACE TO ADSP-2101

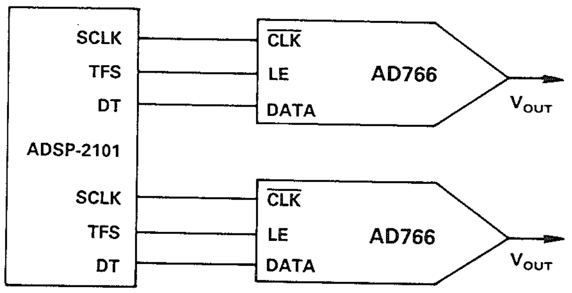


Figure 9.33

AD766 SERIAL INTERFACE TO TMS320C25

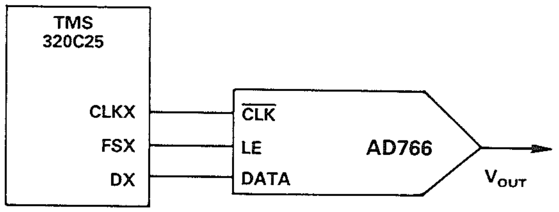


Figure 9.34

AD766 SERIAL INTERFACE TO DSP56000

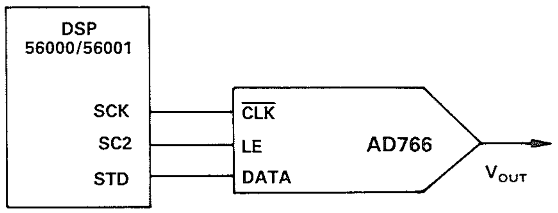


Figure 9.35

INTERFACING I/O PORTS AND CODECS TO DSPS

Since most DSP applications require both an ADC and a DAC, I/O Ports and Codecs have been developed which integrate the two functions on a single chip as well as provide easy-to-use interfaces to standard DSPs.

A functional block diagram of the AD7868 12 bit, 83kSPS I/O Port is shown in Figure 9.36. The AD7869 is a 14 bit I/O Port which is functionally equivalent to the AD7868. These devices are fully specified in terms of ac and dc performance. The SNR (including distortion) of the AD7868 is 72dB, while the AD7869 is 82dB. Both devices provide simple interfaces to the serial ports of standard DSP microcomputers such as the ADSP-2101 (see Figure 9.37), TMS3020/C25, and the DSP56000.

The ADSP-28msp02 is a complete voiceband codec (ADC and DAC) based on sigma-delta technology. A block diagram is shown in Figure 9.38. The device provides a complete analog front end for high performance voiceband DSP applications. Key features of the device are given in Figure 9.39.

AD7868 12-BIT, 83kSPS I/O PORT

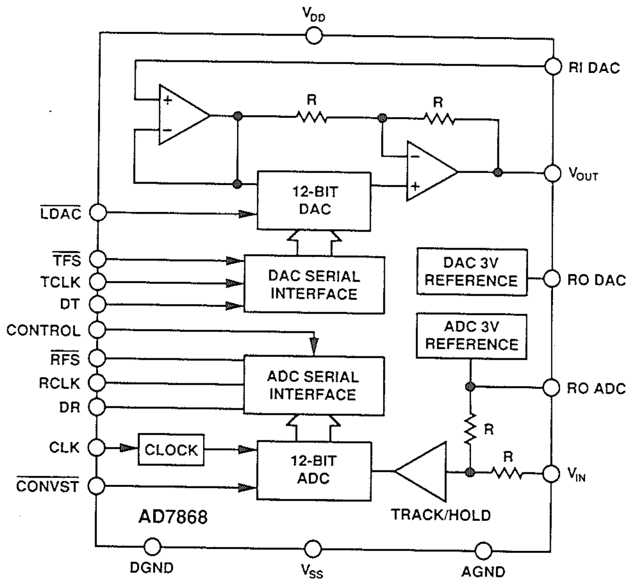


Figure 9.36

AD7868 I/O PORT INTERFACE TO ADSP-2101

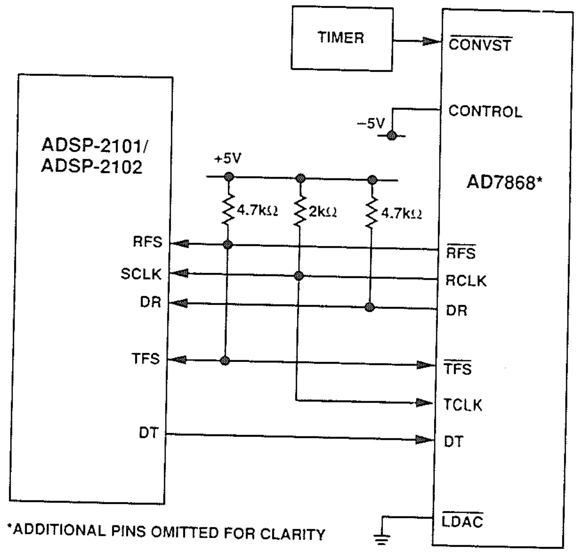


Figure 9.37

ADSP-28msp02 SIGMA-DELTA VOICEBAND CODEC

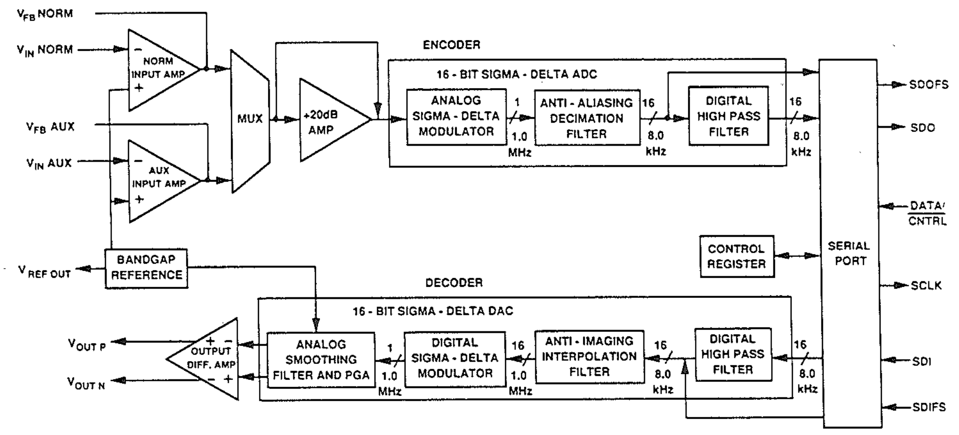


Figure 9.38

KEY FEATURES OF THE ADSP-28msp02 SIGMA-DELTA CODEC

- 16 bit 128x Oversampling Sigma-Delta ADC and DAC
- On-Chip Antialiasing and Smoothing Filters
- On-Chip Input and Output PGAs
- 8kSPS Sampling Rate
- 65dB SNR and THD
- Easy to Interface to Serial Port of Standard DSP Chips
- 24-pin DIP/SOIC Package
- Single +5V Supply, 100mW Power Dissipation
- Ideal for Voiceband Applications

Figure 9.39

Compared to traditional m-law and A-law codecs, the ADSP-28msp02's linear coded ADC and DAC maintain wide dynamic range throughout the transfer function. The encoder side of the device consists of two selectable analog input amplifiers and a sigma-delta ADC. The gain of the input amplifiers can be adjusted with the use of external resistors from -12dB to +26dB. A optional 20dB preamplifier can be inserted before the ADC. The preamplifier and the multiplexer are configured by bits in the control register. The decoder consists of a sigma-delta DAC and a differential amplifier. The output of the DAC drives an analog smoothing filter which converts the data into an analog voltage. The gain of the smoothing filter and PGA can be adjusted via the control register from -15dB to +6dB in 3dB steps. The ADSP-28msp02 easily interfaces to the serial ports of popular DSP microcomputers such as the ADSP-2101 as shown in Figure 9.40.

ADSP-28msp02 CODEC SERIAL PORT DSP INTERFACE

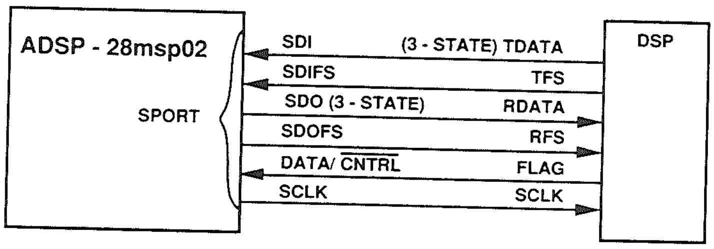


Figure 9.40

SERIAL VERSUS PARALLEL DSP INTERFACE SUMMARY

Some DSP processors such as the ADSP-2100 support only memory-mapped peripherals and have no serial port. A large number of peripheral devices can be connected to the parallel address and data bus. Each device is treated as a single location in the data memory. A number of high performance ADCs and DACs are available with parallel interfaces. Data setup and hold specifications, write and read pulse widths, etc., must be examined carefully to insure that there are no interface timing violations. Conflicts frequently occur because DSP processors are designed to operate at clock frequencies often exceeding 10MHz, while ADCs and DACs used in most DSP applications rarely exceed sampling rates of 500kSPS. These interfacing timing conflicts can usually be resolved with the addition of software or hardware wait states.

ADCs which interface to parallel DSPs must have tri-state outputs so that the data bus can be shared among other peripherals. The convert-start signal for the ADC is generated externally to minimize jitter. The conversion-complete signal is typically used to generate an interrupt request to the DSP processor. Care must be taken in the routing of the ADC parallel digital outputs to prevent digital switching noise from coupling into the ADC analog input.

DACs which interface to parallel DSPs must have double-buffered digital inputs. The latch which drives the DAC switches is updated continuously with a stable external clock source. The external clock is also used to generate an interrupt request to the processor. The input latch is loaded asynchronously by the DSP processor. Care must be taken in the routing of the digital input signals so that they don't couple into the DAC analog output.

The serial ports provided on modern DSP processors provide several advantages when interfacing to peripheral devices. The interface is simple (three wires) and requires little or no external *glue* logic components. Pin counts are minimized as well as logic switching noise. The serial port handles data transmission, reception, and also generates processor interrupt requests automatically. Serial clock and frame synchronizing signals may be generated either internally or externally.

Serial ADCs and DACs are available which are compatible with the timing of most DSP serial ports, and timing conflicts are rare. The Sigma-Delta and Successive Approximation ADC architecture are popular in DSP applications, and also have a serial output data format.

PARALLEL DSP INTERFACE CHARACTERISTICS

- Peripherals are Memory-Mapped
- Timing Conflicts May Require Software or Hardware Wait States
- ADCs Must Have Tri-State Outputs
- DACs Must Have Double-Buffered Inputs
- Routing of Digital Signals is Critical for Low Noise

Figure 9.41

SERIAL DSP INTERFACE CHARACTERISTICS

- Three-Wire, Zero-Chip Interface Typical
- Data Transmission, Reception, Processor Interrupts Handled by Serial Port
- Serial Clock and Frame Synchronization Signals Generated Internally or Externally
- Sigma-Delta and Successive Approximation ADCs are Naturally Serial Output Devices

Figure 9.42

References

1. Kapriel Karagozyan, *Wait State Generation on the ADSP-2100 and the ADSP-2100A*, Analog Devices Application Note E1317-8-8/89. (Included at end of this section)

2. *ADSP-2100 User's Manual / Architecture*, Analog Devices

3. *ADSP-2101/2102 User's Manual / Architecture*, Analog Devices

ANALOG DEVICES

APPLICATION NOTE

ONE TECHNOLOGY WAY • P. O. BOX 9106 • NORWOOD, MASSACHUSETTS 02062-9106 • 617/329-4700

Wait State Generation on the ADSP-2100 and ADSP-2100A

by Kapriel Karagozyan

INTRODUCTION

The ADSP-2100 microprocessor can interface to a large number of peripherals (A/D and D/A converters, FIFOs, etc.) by mapping them onto its data memory address space. Some of these peripherals may not be able to be accessed within a single processor cycle. The data memory interface may require the insertion of hardware wait states during read and/or write operations to such peripherals. The data memory acknowledge (DMACK) input is provided on the ADSP-2100 for this interface.

DMACK TIMING

The ADSP-2100 checks the status of the DMACK signal towards the end of each data memory access cycle. If the DMACK signal is not asserted high, the processor extends the current cycle by another full cycle. This extension occurs until the DMACK signal is sampled high, in which case the access is completed. When no wait states are needed, it is recommended that the DMACK input be tied to a logic 1 (always high).

All of the processor address and control lines are held steady during an extended memory access. The only active output is CLKOUT. The DMACK input is not checked during instruction cycles that do not access the data memory.

DMACK should be held high during normal processor operation. If wait states are desired, external wait state generation logic must return the DMACK input valid low within a required time range to ensure that the DMACK low level is recognized by the processor, which in turn extends the data memory access. The timing requirements relevant to the generation of DMACK are provided with the data memory read and write specifications in the *ADSP-2100 Data Sheet.* They are: DMA valid to DMACK valid (#75), $\overline{\text{DMRD}}$ low to DMACK valid (#74), $\overline{\text{DMWR}}$ low to DMACK valid (#99) and CLKOUT high to DMACK invalid (#103).

The DMACK input to the processor is internally sampled shortly before the rising edge of CLKOUT. The minimum requirement of specification #103 forces the wait state generation logic to keep the valid DMACK level until the rising edge of CLKOUT (which occurs once towards the end of every processor cycle) and ensures that the DMACK input is acknowledged by the processor. On the other hand, the maximum specification on #103 ensures that the low level DMACK does not run into the next cycle and cause another wait state. In order to complete the data memory access, DMACK must be brought high within the maximum specification in #103 and kept high until the next rising edge of CLKOUT. This ensures that DMACK is high when it is sampled again on the next cycle and thus causes the extended access to finish. To generate any number of wait states, the wait state generation logic should use CLKOUT as a counter to determine when the appropriate number of wait states have elapsed. The requirement #103 implies that CLKOUT must be used to clear the wait state generator, since it is the only actively switching output during an extended memory access.

WAIT STATE GENERATION CIRCUITS

The circuits illustrated in Figures 1, 2 and 3 are recommended implementations of external wait state generation circuitry for the ADSP-2100. The logic components are shown without part numbers because their speed requirements vary with processor speed grades. It is also possible to implement most of the logic using high speed PALs. The circuit shown in Figure 1 extends the memory access by one cycle, whereas the one in Figure 2 extends by two cycles. The circuit in Figure 3 shows how multiple external peripherals and their wait state signals can be gated to generate DMACK.

The circuit in Figure 1 normally keeps the DMACK signal at a logic high until an address requiring wait states is issued by the processor. The output of the D-flop is normally low due to the high level of $\overline{\text{DMWR}}$ or $\overline{\text{DMRD}}$ or the existence of an invalid address. When an address requiring wait states is decoded, DMACK is pulled low. The rising edge of CLKOUT causes the output of the D-flop and consequently the DMACK signal to rise high before the next cycle. Thus, a valid low level DMACK signal is generated, held low and brought back high within the processor's timing specifications. The processor samples DMACK low during the first cycle and extends the data

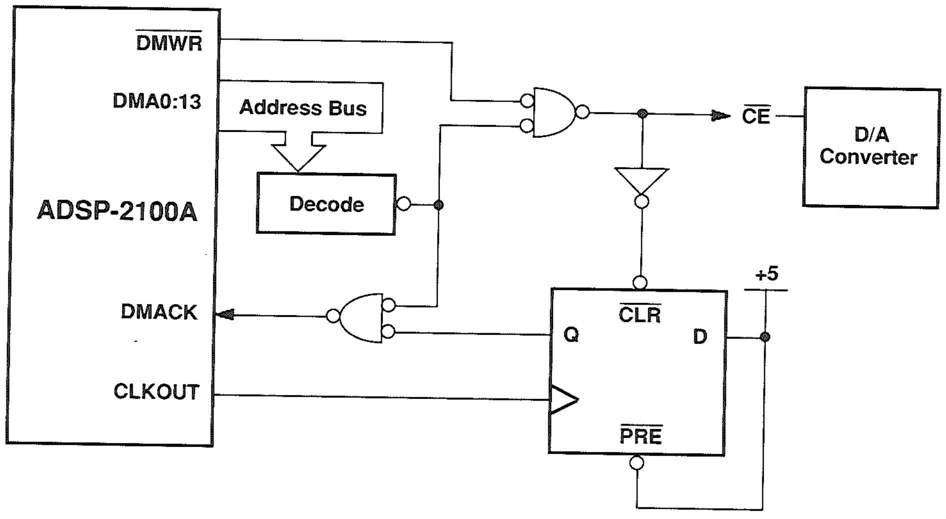


Figure 1. Wait State Generation Logic for One Wait State

memory access. DMACK is high on the second cycle, causing the processor to finish the memory access and resume its normal operation. The DMACK line may have glitches in this circuit configuration, but the circuit guarantees by design that DMACK is valid when needed. Glitches occur only whenever DMACK is not being sampled by the processor and thus are not recognized by the processor.

The operation of the circuit shown in Figure 2 is very similar to that of the one in Figure 1, with the difference that an extra D-flop is added to the DMACK path. This extra D-flop allows the circuit to bring the DMACK signal high after the second rising edge of CLKOUT, hence generating two wait states. More wait states can be generated by adding more D-flops to the DMACK path.

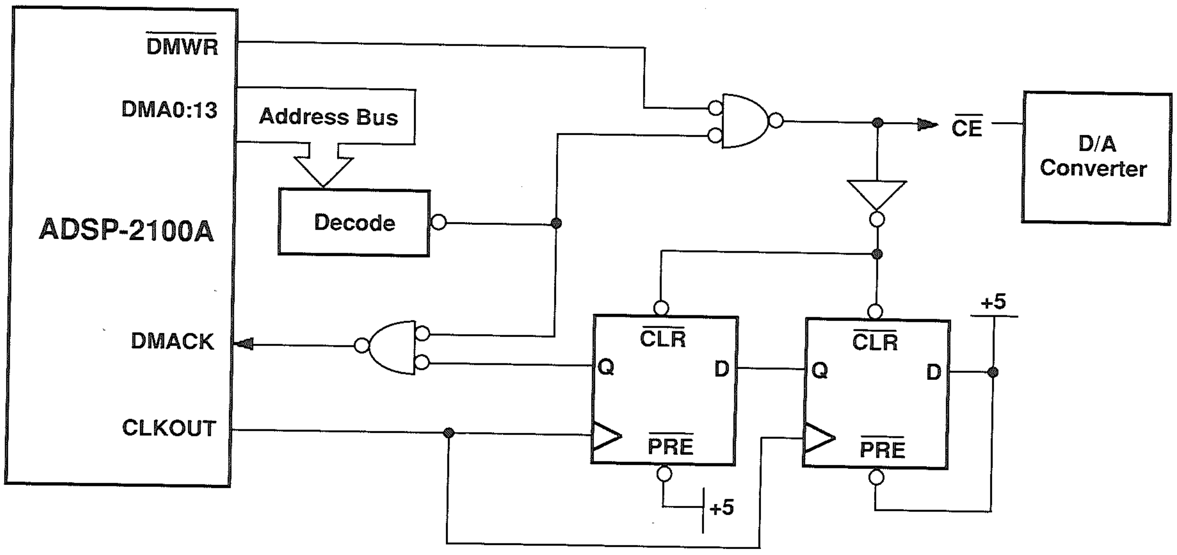


Figure 2. Wait State Generation Logic for Two Wait States

Figure 3 shows how peripherals requiring different numbers of wait states each have their own wait state generation logic. In this example, reading the A/D converter requires one wait state and writing the D/A converter requires two.

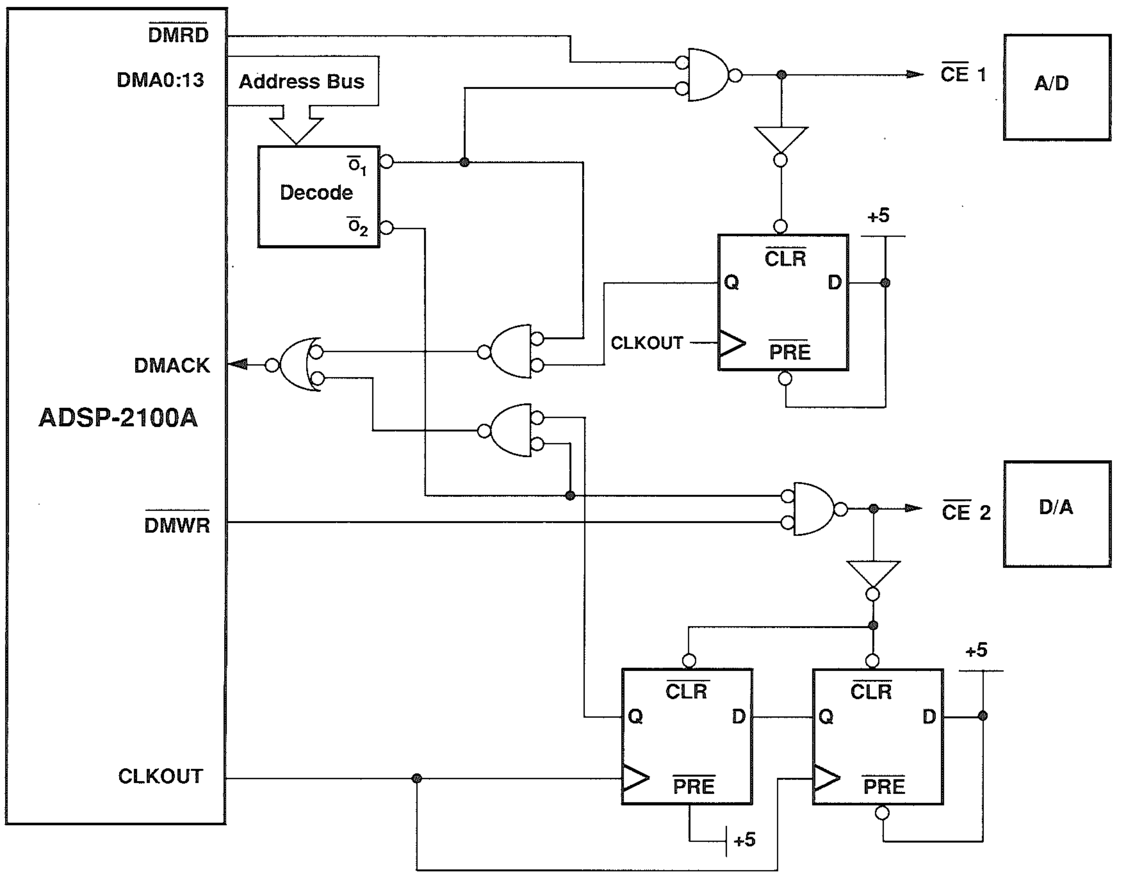
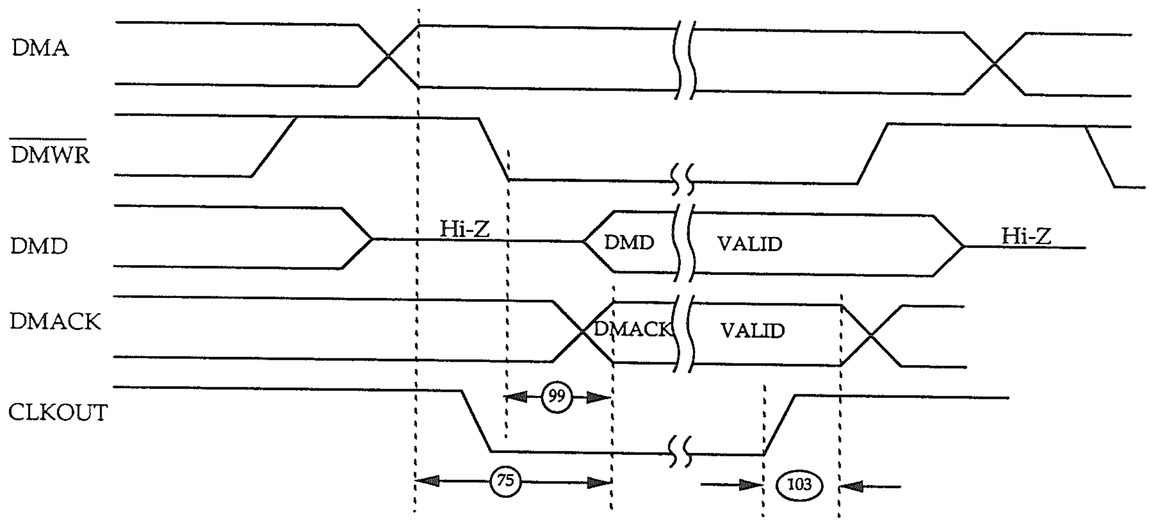


Figure 3. Wait State Generation Logic for Multiple Peripherals

ADSP-2100 DATA MEMORY WRITE CYCLE WITH WAIT STATES



Timing Requirements

74	$\overline{\text{DMRD}}$ low to DMACK valid
75	DMA valid to DMACK valid
99	$\overline{\text{DMWR}}$ low to DMACK valid
103	CLKOUT high to DMACK invalid