MAX17320 Software Implementation Guide

UG7161; Rev 0; 2/20

Abstract

This document covers the design details of the host software for the MAX17320 fuel gauge.

Table of Contents

List of Figures

List of Tables

Introduction

This application note describes the basic operations that the host software must implement to interface with the MAX17320. There are many additional features that are covered in the data sheet.

Table 1. Device Types

DEVICE TYPE	CELL COUNT	COMMUNICATION INTERFACE
MAX17320X20+/ MAX17320X22+/ MAX17320G20+/ MAX17320G22+	2–4 cell	I²C
MAX17320X10+/ MAX17320X12+/ MAX17320G10+/ MAX17320G12+	2–4 cell	1-Wire®

To distinguish the device type, read the DevName register (0x21). The 1-Wire/I²C distinction is made by a device present on one interface and not the other.

Initialization

At IC power-up, the fuel gauge retrieves the battery model parameters from the nonvolatile memory without any interaction from the host software.

Optionally, the host can change some of the auxiliary functions, such as alerts on voltage, current, state of charge (SOC), or temperature. If the nonvolatile programmed values are acceptable, then the host only needs to take action on the alert outputs. If the host desires different thresholds, the host can write to the VAlrtTh, TAlrtTh, SAlrtTh, or IAlrtTh registers as necessary. Refer to the MAX17320 data sheet for register details.

Useful Registers to Read

Table 2 lists the registers that the host finds useful in most applications.

Table 2. Useful Registers in Most Applications

ADDRESS	REGISTER NAME	PURPOSE/CONTENTS
0x000	Status	Alert status and chip status
0x005	RepCap	Reported remaining capacity
0x006	RepSOC	Reported state of charge
0x01A	VCell	Cell voltage for a single cell
0x01B	Temp	Temperature
0x01C	Current	Battery current
0x011	TTE	Time to empty
0x020	TTF	Time to full
0x0D9	ProtStatus	Fault status of the protection functionality
0x0AF	ProtAlrt	History of previous fault status of the protection functionality
0x1A8	nBattStatus	Permanent battery status information

Other applications might have more requirements and might want to read additional registers such as the ones listed in Table 3.

Table 3. Additional Registers

ADDRESS	REGISTER NAME	PURPOSE/CONTENTS
0x0D8, 0x0D7, 0x0D6, 0x0D5	Cell1, Cell2, Cell3, Cell4	Direct cell measurements of selected number of cells
0x0DA, 0xDBh	Batt, PCKP	The Batt register contains the total pack voltage measured inside the protector, and the PCKP register contains the voltage between PACK+ and GND.
0x13A, 0x139, 0x138, 0x137, 0x034	Temp1, Temp2, Temp3, Temp4, DieTemp	Individual temperature measurements from the thermistor and internal die temperature
0x0B9	AgeForecast	Projected total cycles the battery lasts until full capacity is below the target “dead” battery capacity
0x007	Age	Percentage of full capacity compared to the design capacity
0x017	Cycles	Number of cycles the battery has been used
0x0DD (AtRate=0x004)	AtTTE	After writing AtRate, hypothetical time to empty based on the AtRate value (refer to the data sheet for details)
0x010	FullCapRep	Full capacity based on present discharge conditions
0x0BE	TimerH	Time since first power-up
0x008, 0x009, 0x00A	MaxMinVolt, MaxMinTemp, MaxMinCurrent	Maximum and minimum values for each measurement since the last life log
0x014	RCell	Calculated battery resistance
0x019, 0x0D4, 0x0D3, 0x0D2, 0x0D1	AvgVCell, AvgCell1, AvgCell2, AvgCell3, AvgCell4	Average of voltages
0x01D	AvgCurr	Average current
0x016, 0x136, 0x135, 0x134, 0x133	AvgTA, AvgTemp1, AvgTemp2, AvgTemp3, AvgTemp4	Average of temperatures
0x0B2	VRipple	Filtered difference of average and instantaneous values of VCell

Write Protection

The MAX17320 includes write protection and a permanent locking function. The write protection prevents accidental overwrites of protection parameters and is enabled by default. This protection must be cleared before updating any registers or sending any commands and should be set after configuration changes are made.

If the host wants to write to a register, the global write protection must be disabled as well as the write protection for the specific register page by writing to the CommStat register (address 061h) as shown in Table 4. To prevent accidental unlocking of the write protection, the CommStat register must be written with the desired value two times in a row without accessing any other registers to set or clear any of the write-protection bits. All bits can be set or cleared in the same write sequence.

For example, writing 0x0000 to the CommStat register twice in a row clears the WPGlobal bit and all write-protect bits (WP1–WP5) at the same time.

Table 4. CommStat Register (061h) Format

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	X	DISOff	CHGOff	WP5	WP4	WP3	WP2	WP1	NVError	NVBusy	WPGlobal

WPGlobal: Write-Protection Global Enable Bit. Set to 1 to write-protect all register pages. Clear to 0 to allow individual write-protect bits (WP1-WP5) to be disabled.

WP1–WP5: Write-Protection Enable Bits. Set any bit to 1 to write protect the pages specified below. Clear any bit to 0 to allow pages to be writable. To update any of these bits, the WPGlobal bit must be 0.

- **WP1**: Write-protects register pages 1A, 1B, 1E
- **WP2**: Write-protects register pages 01, 02, 03, 04, 0B, 0D
- **WP3**: Write-protects register pages 18, 19
- **WP4**: Write-protects register page 1C
- **WP5**: Write-protects register page 1D

Alerts Management

The default configuration of this fuel gauge is to automatically clear the alert after the alert condition is no longer present. The host can set the alerts to be “sticky,” and make them persist until the host clears the alert. This setting can be changed in nonvolatile memory to permanently alter the behavior to “sticky” mode by changing the nConfig register. This change can be made by the pack maker or the end application if the nonvolatile memory is not locked. Alternatively, it can be changed by the host software in the Config register to only change it until the fuel gauge resets or when the host wants to disable this feature. The Config register details to make this change are provided in Table 5.

Table 5. Config Register (01Dh) Format

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	SS	TS	VS	0	PBen	0	0	SHDN	COMMSH	0	ETHRM	FTHRM	Aen	0	0

The SS, TS, and VS bits set the corresponding alerts to be “sticky.” The alerts are cleared by clearing the appropriate alert bit in the Status register.

An example to set the low SOC alert to 5% is as follows:

1. Set SAlrtTh register to 0xFF05. 0xFF__ indicates that the high alert is disabled. 0x__05 sets the low alert to 5%.
2. Set the Config.Aen bit to 1.
3. When the SOC decreases below 0x0500, the ALRT pin asserts, and the Status register indicates that the SOC alert has occurred.
4. To disable further alerts on the low SOC until the battery is charged, set the SAlrtTh register to 0xFF00 to disable low SOC alerts.

Authentication

This IC has an authentication feature to prevent clone batteries. The host writes a challenge to the IC and reads out a response. The host validates this response by either doing a parallel calculation with the same secret or using predefined challenge and response pairs.

Method 1: The Host Knows the Secret and Calculates a Valid Output to Check Against the Battery Pack Response

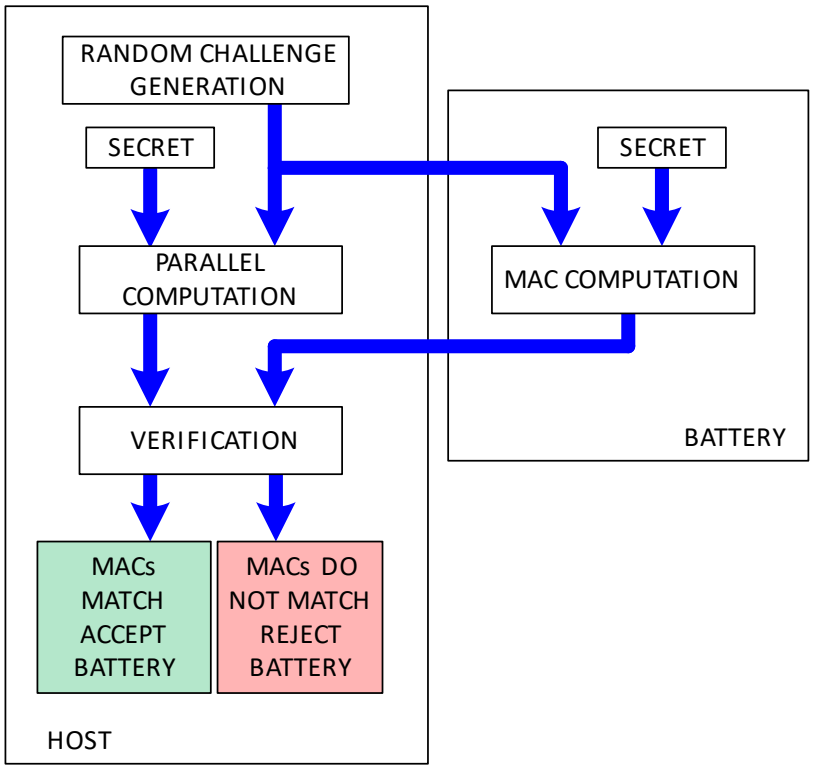

Figure 1. Parallel calculation with the same secret.

Method 2: The Host Does Not Know the Secret and the Challenge and Response Pairs Are Predefined for the Host

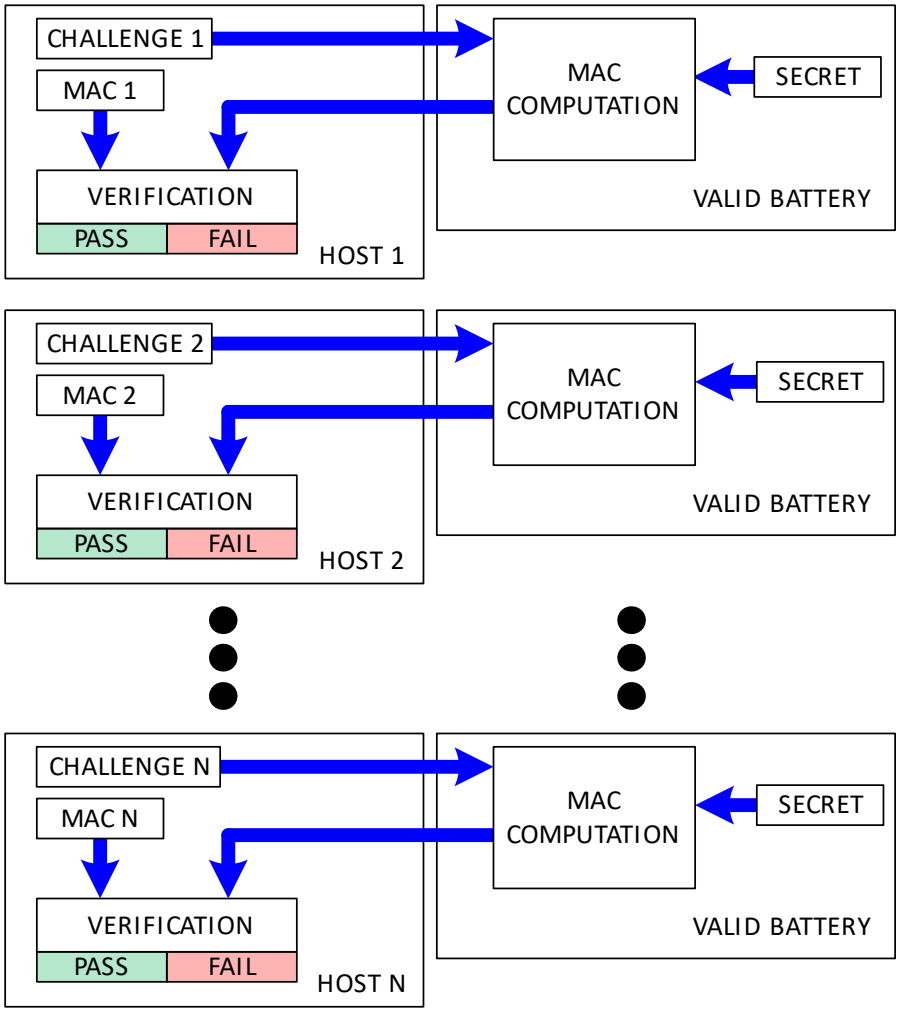

Figure 2. Predefined challenge-and-response pairs.

To authenticate the battery, the host writes the challenge to memory addresses 0x0C0 to 0x0C9, and then sends the Compute MAC without ROM ID command by writing to register 0x060 with the value 0x3600. The fuel gauge calculates a response after t_{SHA}, and the host can read it from spaces 0x0C0 to 0x0CF. The returned MAC computation is compared against the calculated or stored response, and the authenticity is verified.

If the host knows the secret, the Compute MAC with ROM ID command can be used for more security. Write to register 0x060 with the value 0x3500 to send the Compute MAC with ROM ID command. The returned MAC computation is available after t_{SHA}.

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	2/20	Initial release	—