Application Note

BQ769x2 Control Based on MSPM0 Through I2C

Adrian Liu, Eason Zhou, Helic Chi

ABSTRACT

The MSPM0 series microcontroller (MCU) portfolio offers a wide variety of 32-bit MCUs with ultra-low power and integrated analog and digital peripherals for sensing, measurement, and control applications. Reducing power consumption while performing complex real-time applications presents a major challenge for the recent embedded applications. This article was created to build a simple framework to help developers quickly setup the communication between MSPM0 and BQ769x2. Currently, the MSPM0 L series and G series are supported, and sample code is provided based on LP-MSPM0G3507 and LP-MSPM0L1306 LaunchPad™.

Table of Contents

List of Figures

List of Tables

Trademarks

LaunchPad™ and Code Composer Studio™ are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction

The BQ769x2 is a highly-integrated and accurate battery-monitor device for 3-series to 16-series battery packs. The MSPM0L1306 and MSPM0G3507 devices are highly-integrated, ultra-low-power 32-bit MCU of the MSPM0 family. In practice, the BQ769x2 monitor can measure battery cell voltage, temperature, and current, and reports this information to the MCU(MSPM0) through an I2C interface. The MCU can then make decisions based on the information provided by the monitor. The device can enable and disable the FETs, control the cell balancing feature, and do various responses according to specific user needs.

This application note provides sample code for MSPM0 and BQ769x2. Online SDK demonstration code can be accessed from MSPM0L1306 and MSPM0G3507. The offline SDK file paths are: `{MSPM0_SDK_INSTALL_DIR}\examples\nortos\LP_MSPM0L1306\demos\bq769x2_control_i2c` and `{MSPM0_SDK_INSTALL_DIR}\examples\nortos\LP_MSPM0G3507\demos\bq769x2_control_i2c`.

2 Hardware Connection

The I2C communication sample code implemented in this application note is based on the BQ76952EVM and MSPM0 LaunchPad™ (LP-MSPM0L1306 and LP-MSPM0G3507). MSPM0 acts as the I2C controller and BQ76952 acts as the peripheral. Figure 2-1 shows the simple system block diagram.

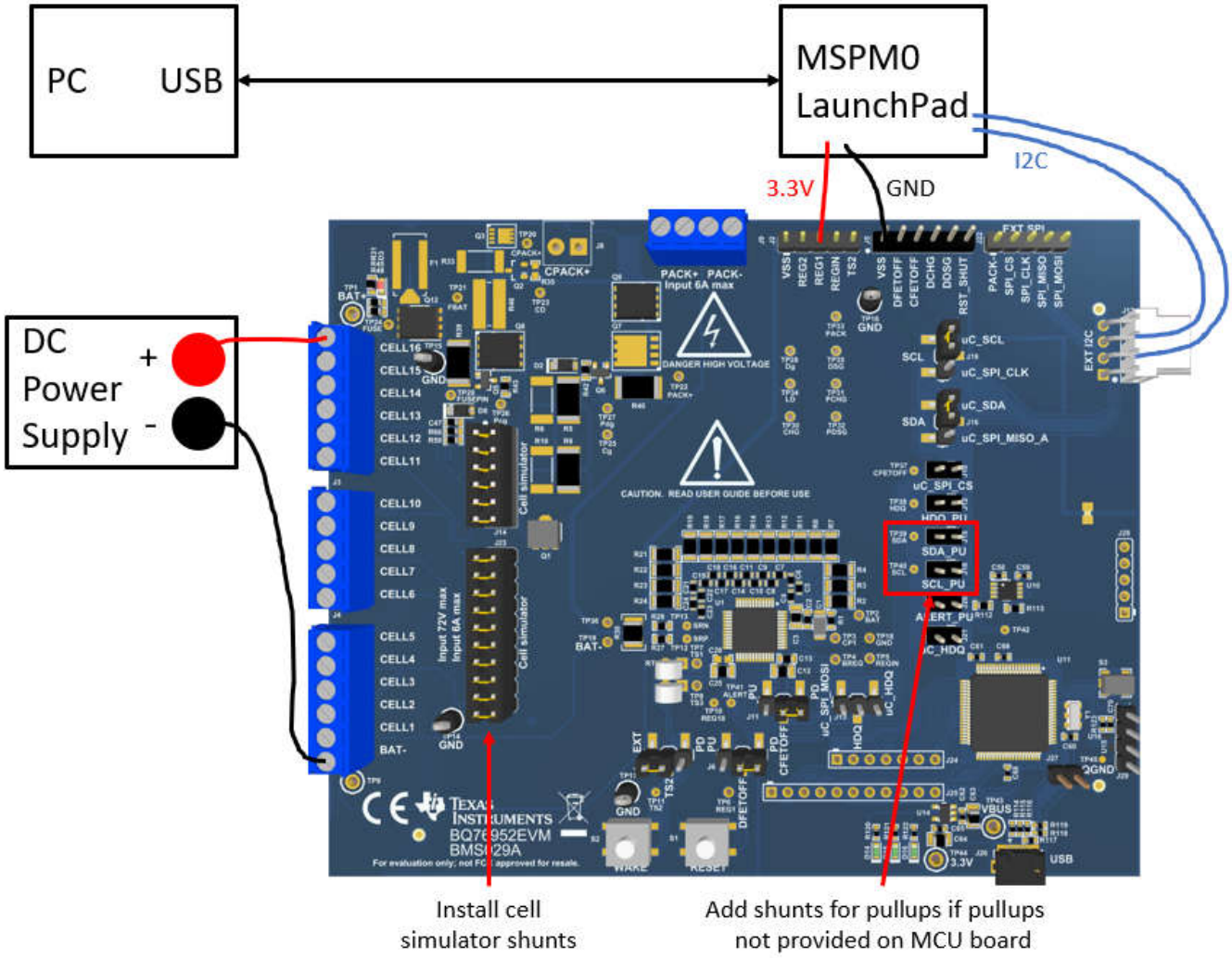


Figure 2-1. System Block Diagram

For BQ76952EVM, check the BQ76952 chip by bringing the chip up with the onboard MCU (EV2400). Download the latest version of BQSTUDIO (the BQSTUDIO-TEST version) and follow the steps in the quick start section of EVM User Guide.

1. Connect 10-72V DC power supply capable of 250mA minimum between the *BAT-* and *CELL16* terminals of EVM to power the device.
2. Connect the USB cable from micro-USB connector on the EVM to the PC. When this is connected, three green LEDs light up.
3. Check the BQStudio Dashboard on the left side of BQStudio window. The dashboard indicates whether the onboard MCU is connected and which firmware version is being used. The dashboard also indicates whether the BQ76952 device is communicating successfully.

After confirming the BQ76952EVM board works successfully, start setting up the I2C bridge between MSPM0 and BQ769x2. If the USB on BQ769x2 EVM is not powered, remember to remove the connectors on uC_SCL(J19) and uC_SDA(J16).

For MSPM0L1306 and MSPM0G3507 LaunchPad, PA1 is configured as the SCL pin, and PA0 is configured as the SDA pin. The LaunchPad is powered using the USB port on the host computer. Table 2-1 shows the signal and power connections between the two EVMs.

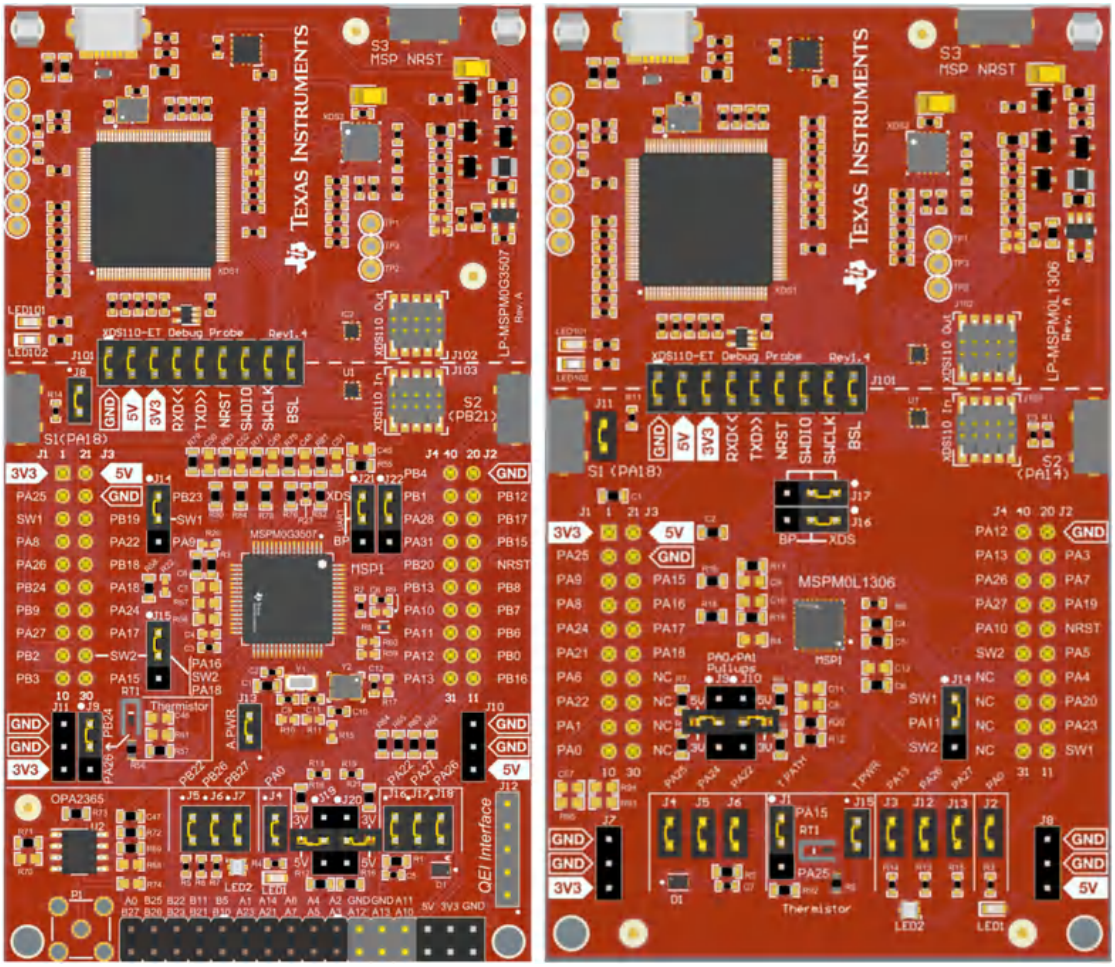


Figure 2-2. LP-MSPM0G3507 and LP-MSPM0L1306 Hardware Board

Table 2-1. Connecting the EVMs

Connection Type	Connection Name	LP-MSPM0L1306 Pin Number : Pin Name	LP-MSPM0G3507 Pin Number : Pin Name	BQ76952EVM Pin Number : Pin Name
I2C Interface	I2C : SCL	PA.1 : I2C0_SCL	PA.1 : I2C0_SCL	J17-2 : P26
	I2C : SDA	PA.0 : I2C0_SDA	PA.0 : I2C0_SDA	J17-3 : P27
Power connections	Power : 3.3V	J1-1	J1-1	J2-3 : REG1
	Power : Ground	J1-22	J1-22	J5-1 : VSS

In the I2C interface, besides the connection of SCL and SDA, there are pullup methods on both boards. Select one EVM to pull up the SDA and SCL based on the actual situation.

Short J15 and J18 can pull up the SDA and SCL in the BQ76952EVM. The default value of the pullup resistor is 10kΩ. Adjust the resistor according the I2C bus speed. Figure 2-3 shows the pullup jumper diagram on BQ76952EVM.

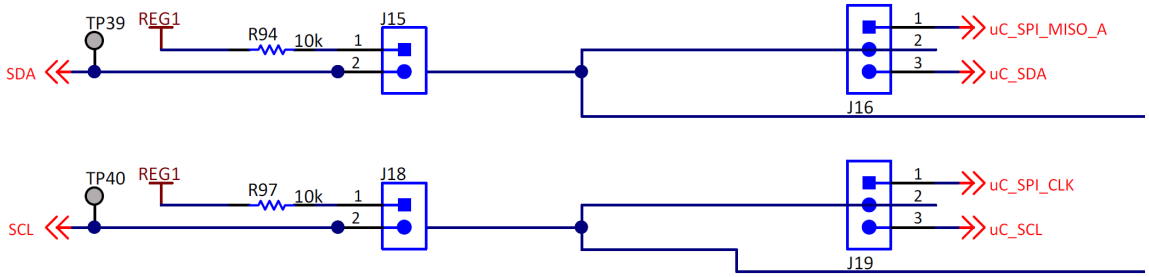


Figure 2-3. 3.3V Pullup for SDA and SCL on BQ76952EVM

For the LaunchPad, shorting J9-2-3 and J10-2-3 on LP-MSPM0L1306 or J19-1-2 and J20-1-2 on LP-MSPM0G3507 can pullup the SDA and SCL. The default value of the pullup resistor is 2.2kΩ. The value can also be adjusted according the I2C bus speed. Figure 2-4 shows the pullup jumper diagram on MSPM0L1306 LaunchPad.

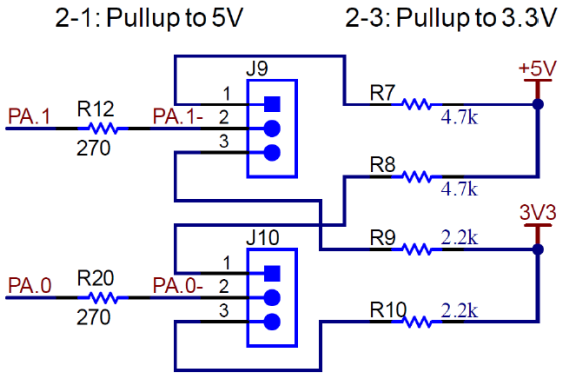


Figure 2-4. 3.3V Pullup for SDA and SCL on MSPM0L1306

Refer to the *BQ76952EVM User's Guide* and *LP-MSPM0L1306 LaunchPad User's Guide* or *LP-MSPM0G3507 LaunchPad User's Guide* for instructions to the other parts that are not mentioned.

3 Software Structure and Important Functions

Figure 3-1 shows the software project developed in Code Composer Studio™ (CCS). Obtain the code from MSPM0 SDK or from online MSPM0SDK, MSPM0G3507, or MSPM0L1306. The software project mainly consists of three parts. For other files, the files are the default files for the MSPM0 project.

The BQ769x2_protocol declares all of the memory registers, direct commands, sub-commands, and command only sub-commands' definitions in the BQ769x2 TRM. BQ769x2_protocol also has the functions of getting related status, faults, and measurement result.

The I2C_Communication mainly includes the write and read registers function based on I2C protocol for M0.

The main includes the highest system function code. After pressing the button (PA14 of LP-MSPM0L1306 or PB21 of MSPM0G3507), the MSPM0 starts communicating with BQ769x2. More details about the software are shown in the following section.

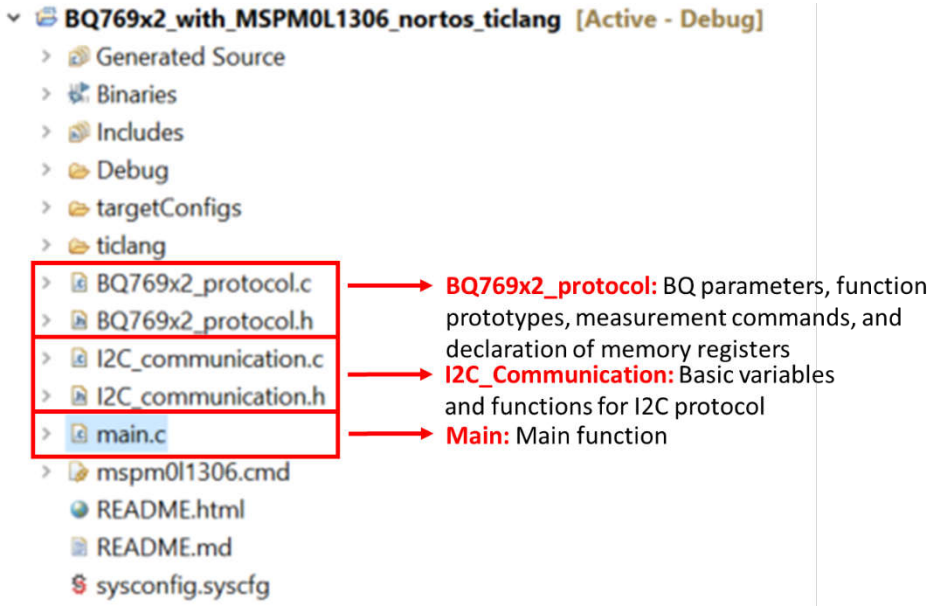


Figure 3-1. Software Project View

3.1 System Initialization

The sample code described in this application note is created by using MSPM0 I2C interface to control BQ76952 and realize necessary commands. The final code is composed of three types of functions: MSPM0 initializing functions, I2C communication functions and BQ76952 functions. The code shows a demonstration of how to communicate MSPM0 with BQ756952 by I2C interface and implement specific command.

The MSPM0 initialization is for system initialization of MCU power, system controller, system clock, and I2C peripherals, shown in Figure 3-2 and Figure 3-3. All the configuration is done by Sysconfig (Graphic code generation tool). The I2C peripheral in MSPM0 is configured as a Controller device to communicate with BQ769x2.

```
SYSCONFIG_WEAK void SYSCFG_DL_init(void)
{
    SYSCFG_DL_initPower();
    SYSCFG_DL_GPIO_init();
    /* Module-Specific Initializations*/
    SYSCFG_DL_SYSCTL_init();
    SYSCFG_DL_I2C_0_init();
}
```

Figure 3-2. SYSCFG_DL_init () Function

```
SYSCONFIG_WEAK void SYSCFG_DL_I2C_0_init(void) {

    DL_I2C_setClockConfig(I2C_0_INST,
        (DL_I2C_ClockConfig *) &gI2C_0ClockConfig);
    DL_I2C_disableAnalogGlitchFilter(I2C_0_INST);

    /* Configure Controller Mode */
    DL_I2C_resetControllerTransfer(I2C_0_INST);
    /* Set frequency to 400000 Hz*/
    DL_I2C_setTimerPeriod(I2C_0_INST, 7);
    DL_I2C_setControllerTXFIFOThreshold(I2C_0_INST, DL_I2C_TX_FIFO_LEVEL_BYTES_1);
    DL_I2C_setControllerRXFIFOThreshold(I2C_0_INST, DL_I2C_RX_FIFO_LEVEL_BYTES_1);
    DL_I2C_enableControllerClockStretching(I2C_0_INST);


    /* Enable module */
    DL_I2C_enableController(I2C_0_INST);

}
```

Figure 3-3. SYSCFG_DL_I2C_0_init () Function

The I2C module is initialized by the SYSCFG_DL_I2C_0_init() function. The clock source for I2C module is BUSCLK which depends on the power domain of MSPM0L. The I2C standard bus speed can be configured to Standard Mode(100k), Fast Mode(400k) and Fast Mode Plus(1M), three modes. This code is set to 100kHz. All details of these settings can be found in CCS Sysconfig page.

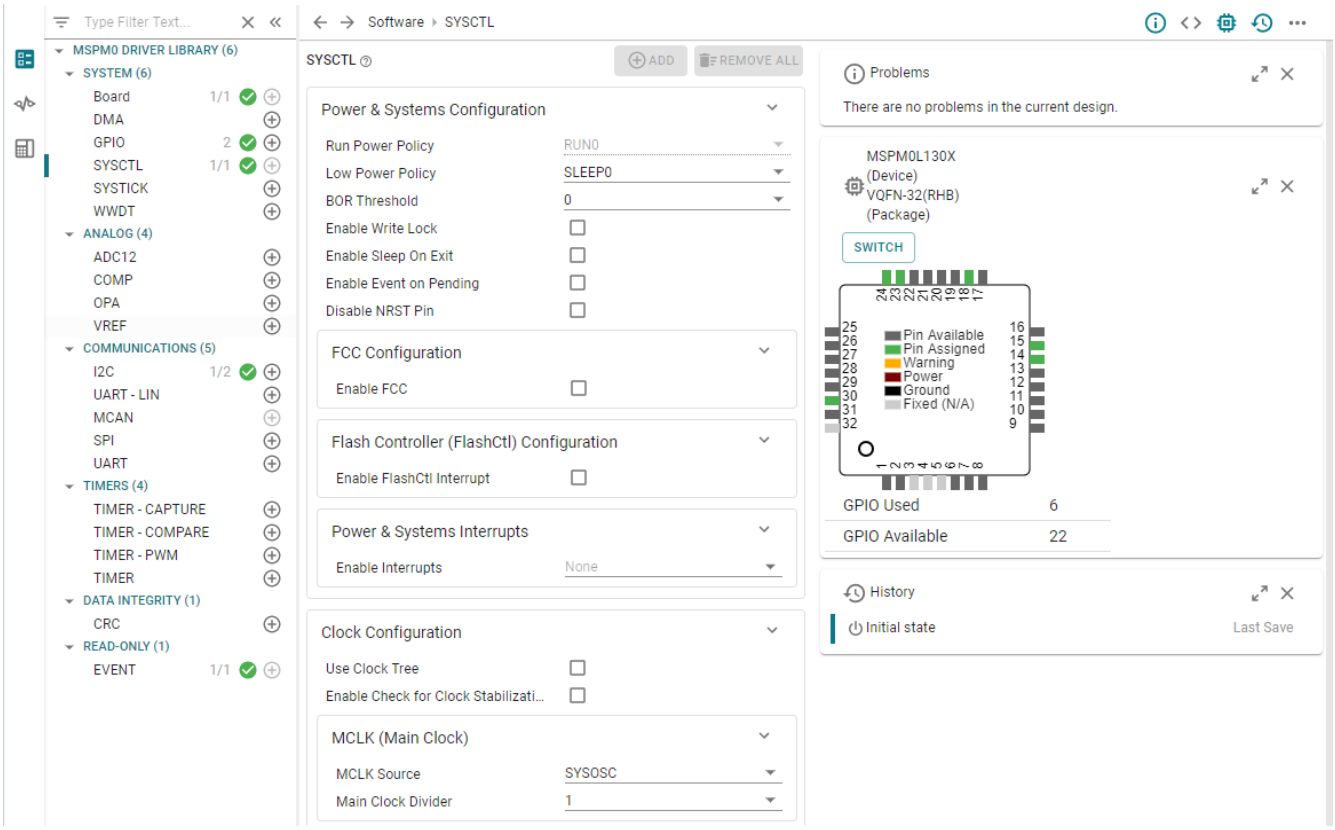


Figure 3-4. Sysconfig Setting Interface

3.2 Low-Level Command Control

Figure 3-5 lists the basic BQ769x2 register control functions based on the Inter-Integrated Circuit (I2C) Driver Library (`I2C_communication.h`) for M0.

```
//************I2C write register *********************
void I2C_WriteReg(uint8_t reg_addr, uint8_t *reg_data, uint8_t count)

//************I2C read register **********************
void I2C_ReadReg(uint8_t reg_addr, uint8_t *reg_data, uint8_t count)
```

Figure 3-5. Basic Communication Functions

These functions are used for MSPM0 to write and read the registers of the BQ device in byte. Referring to the BQ769x2 Technical Reference Manual and data sheet, direct commands, subcommands, and reads and writes to RAM registers can be realized on the register control functions. Figure 3-6 lists the prototype of functions.

```
//*********************************Function prototypes of BQ *********************************
void DirectCommands(uint8_t command, uint16_t data, uint8_t type)

void CommandSubcommands(uint16_t command)

void CommandSubcommands(uint16_t command)

void BQ769x2_SetRegister(uint16_t reg_addr, uint32_t reg_data, uint8_t datalen)
```

Figure 3-6. Basic Functions of BQ769x2

Direct Commands, Command-Only Subcommands, and Subcommands with data are predefined functions available to simplify communication with the battery monitor.

The input parameters for the Direct Commands function are the command, the data, and the type. There are two types: read and write. The user can read data from the command address and store data in the Rx state of the global variable to be read for the read type. The user can write data to the command address for the write type.

For Command-Only Subcommands functions, the input parameter is the subcommand such as shut down and reset. This function formats the transfer array then writes the array to hex 3E, where the monitor then operates based on the command.

The Subcommands with data differs from the command-only subcommands because there is data associated with each command, whether functioning to read the data or write data. The input parameters are similar to the BQ769x2 `SetRegister` function, but the last input is *typed* instead. This input can either be the defined macros R for read, W for write, or W2 for write two bytes.

The input parameters for the BQ769x2 `SetRegister` function are the data memory register address, the data memory intended to write, and the data length in bytes. First write hex 3E for the initial write for subcommands in direct memory, and then write to register hex 60 for the checksum to enter the data transmitted was correct. Finally, there are different cases for the three varying data lengths.

A complete list of commands is found in the BQ76952 Technical Reference Manual.

3.3 High-Level Function

Combined with the predefined commands integrated in BQ76952, this sample code provides several functions to operate BQ76952 to help the customer read the voltage, current, temperature directly, and the status. The major functions are listed in Figure 3-7.

```
// ******************************** BQ769x2 Measurement Commands   ************************************
void BQ769x2_ReadAlarmStatus()

void BQ769x2_ReadSafetyStatus()

void BQ769x2_ReadPFStatus()

void BQ769x2_ReadCurrent()

void BQ769x2_ReadAllTemperatures()

void BQ769x2_ReadPassQ()

void BQ769x2_ReadAllVoltages()
```

Figure 3-7. Measurement Commands of BQ769x2

These APIs use some of the commands mentioned to complete specific functions based on BQ76952 over I2C.

The `ReadAlarmStatus` function can be used to retrieve the alarm bits. The Alarm status command is 0x62 and can be read and write. When a masked flag transitions from low to high, a corresponding bit is latched in 0x62. The host can read the status and clear those latched bits by writing the 0x62 with a *1*.

In the `ReadSafetyStatus` function, the BQ76952 device integrates a broad suite of protections for battery management and provides the capability to enable individual protections, as well as to select which protections result in autonomous control of the FETs. Flags showing which safety faults can be present are available through the 0x03 Safety Status A, 0x05 Safety Status B, and 0x07 Safety Status C commands, and the presence of a fault can generate an interrupt to a host processor on the ALERT pin. The `ReadSafetyStatus` function reads the 0x03, 0x05, and 0x07 registers in sequence, and triggers the protection bit if there is any flag in the read back value of three registers. For more details of the bit description, see the device-related TRM.

In the `ReadRFStatus` function, the BQ76952 device integrates a suite of checks on battery operation and status that can trigger a Permanent Fail (PF) if conditions are considered so serious that the pack must be permanently disabled. When a Permanent Fail occurs, the BQ76952 device can be configured to provide a flag in related PF Status registers.

The `ReadAllTemperatures` function can report the most recent temperature measurement corresponding to the pins TS1, TS2 and TS3, and update them to an array of float type data. On the EVM there are two different thermistors. One is connected to the TS1 pin. The other is connected to the TS3 pin.

The `ReadPassQ` function can report a 64-bit value from the 0x0076 DASTATUS6() subcommand, which includes the integer and fractional portion of accumulated passed charge in user Amp-hours, and the number of seconds over which passed charge.

Finally, the `ReadAllVoltages` function is included to iteratively read all of the voltages. The function uses the direct command to read various voltage information taking in the measurement command shown in Table 4-6, then placing them into the `CellVoltage` array. Once the function iterates through all of the cell voltages, the `Stack_Voltage`, `Pack_Voltage`, and `LD_Voltage` are read and placed in the respective variables.

The bit-transaction details of these examples are documented in the next sections.

All of the read results are saved in these parameters in the protocol file (Figure 3-8).

```
// Global Variables for cell voltages, temperatures, Stack voltage, PACK Pin voltage, LD Pin voltage, CC2 current
uint16_t CellVoltage [16] = {0x01,0x02,0x03,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
float Temperature [3] = {0,0,0};
uint16_t Stack_Voltage = 0x00;
uint16_t Pack_Voltage = 0x00;
uint16_t LD_Voltage = 0x00;
uint16_t Pack_Current = 0x00;
uint16_t AlarmBits = 0x00;
```

Figure 3-8. Read Results of BQ769x2

4 Test Result of Important Functions

4.1 Read Alarm Status

Table 4-1 shows how to read the alarm status of BQ76952. The BQ769x2 `ReadAlarmStatus` function can be used to retrieve the alarm bits. As Figure 4-1 shows, the read back value is zero before configuring the BQ76952, and related bits change after BQ769x2_Init. The data is returned in little byte order format, in the following example, the 16-bit Alarm Status read 0x5082, which corresponds to a bit is set in Safety Status A (), full voltage scan complete, and voltage ADC scan complete. More description of register can be found in the technical reference manual.

Table 4-1. Alarm Status Command Description

Command	Name	Unit	Type	Description
0x62	Alarm Status	Hex	H2	Latched signal used to assert the ALERT pin. Write a bit high to clear the latch. Bit descriptions are found in the Alarm Status Register.

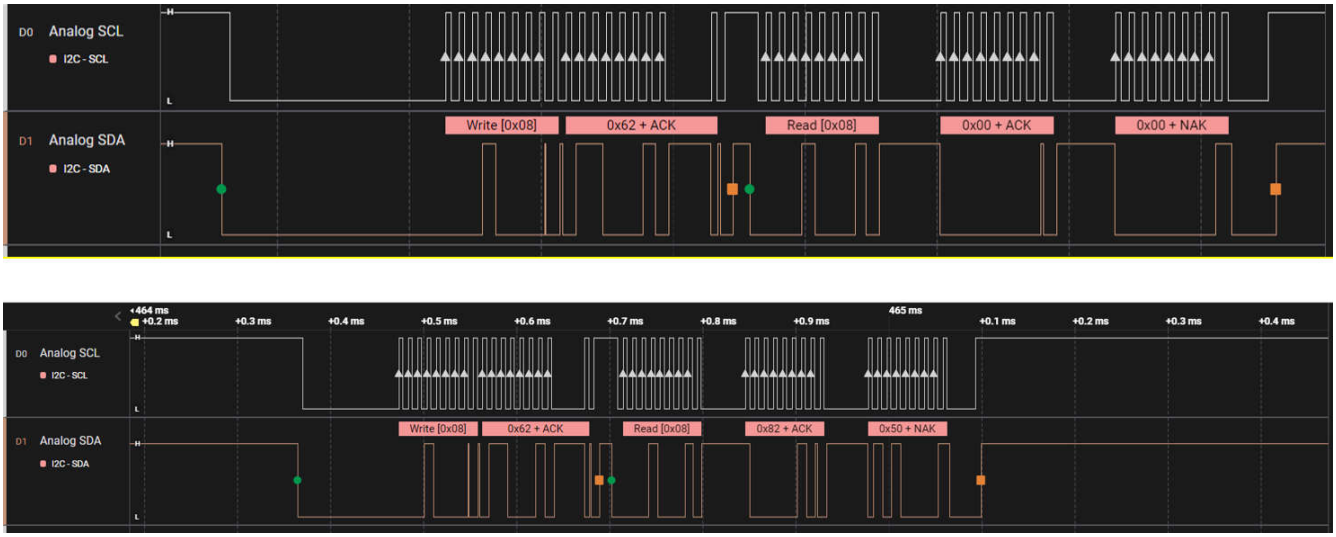


Figure 4-1. Captured I2C Waveform for Alarm Status Reading

4.2 Read Safety Status

Table 4-2 shows how to read the safety status of BQ76952. The following example read 0x03, 0x05 and 0x07 in sequence, and the value return in 0x03 is 0x0004, which means the Cell Undervoltage safety fault is present. More status description can be found in TRM.

Table 4-2. Safety Status Command Description

Command	Name	Unit	Type	Description
0x03	Safety Status A	Hex	H1	Provides individual fault signals when enabled safety faults have triggered. Bit descriptions can be found in Safety Status A Register.
0x05	Safety Status B	Hex	H1	Provides individual fault signals when enabled safety faults have triggered. Bit descriptions can be found in Safety Status B Register.
0x07	Safety Status C	Hex	H1	Provides individual fault signals when enabled safety faults have triggered. Bit descriptions can be found in Safety Status C Register.

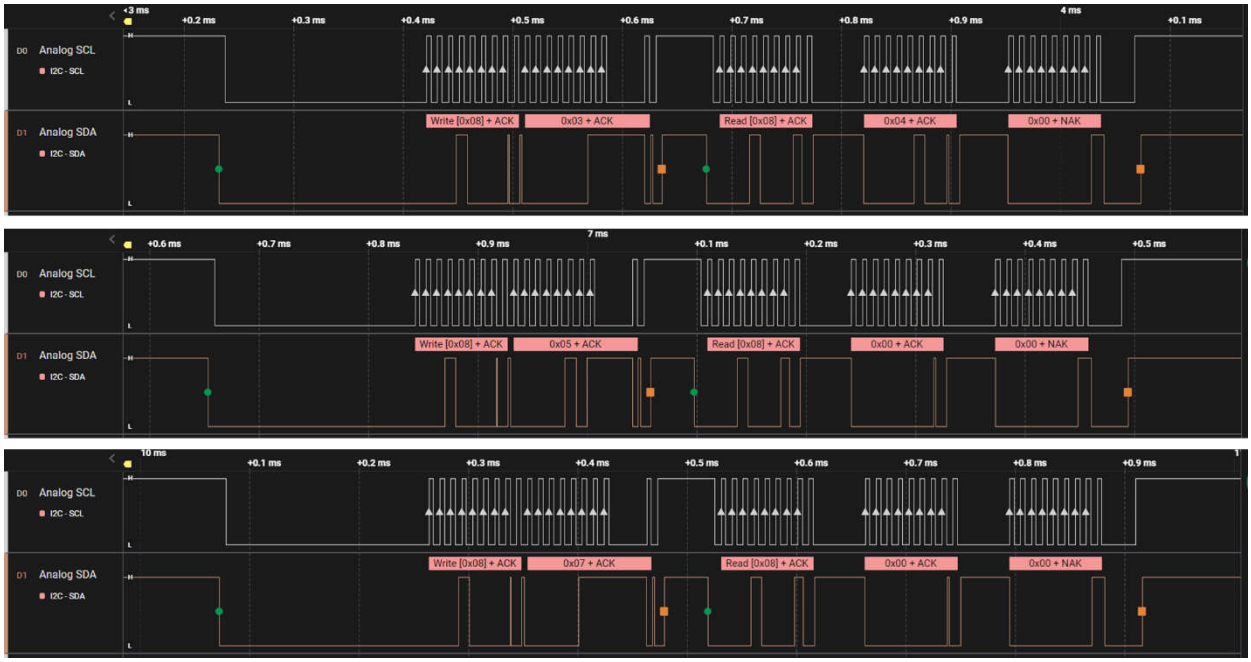


Figure 4-2. Captured I2C Waveform for Safety Status Reading

4.3 Read PF Status

Table 4-3 shows how to read the PF status of BQ76952. In the following example, the read value from PF Status A, B, and C are 0x0000, indicating that no serious faults have occurred.

Table 4-3. PF Status Command Description

Command	Name	Unit	Type	Description
0x0B	PF Status A	Hex	H1	Provides individual fault signals when enabled Permanent Fail faults have triggered. Bit descriptions can be found in PF Status A Register.
0x0D	PF Status B	Hex	H1	Provides individual fault signals when enabled Permanent Fail faults have triggered. Bit descriptions can be found in PF Status B Register.
0x0F	PF Status C	Hex	H1	Provides individual fault signals when enabled Permanent Fail faults have triggered. Bit descriptions can be found in PF Status C Register.

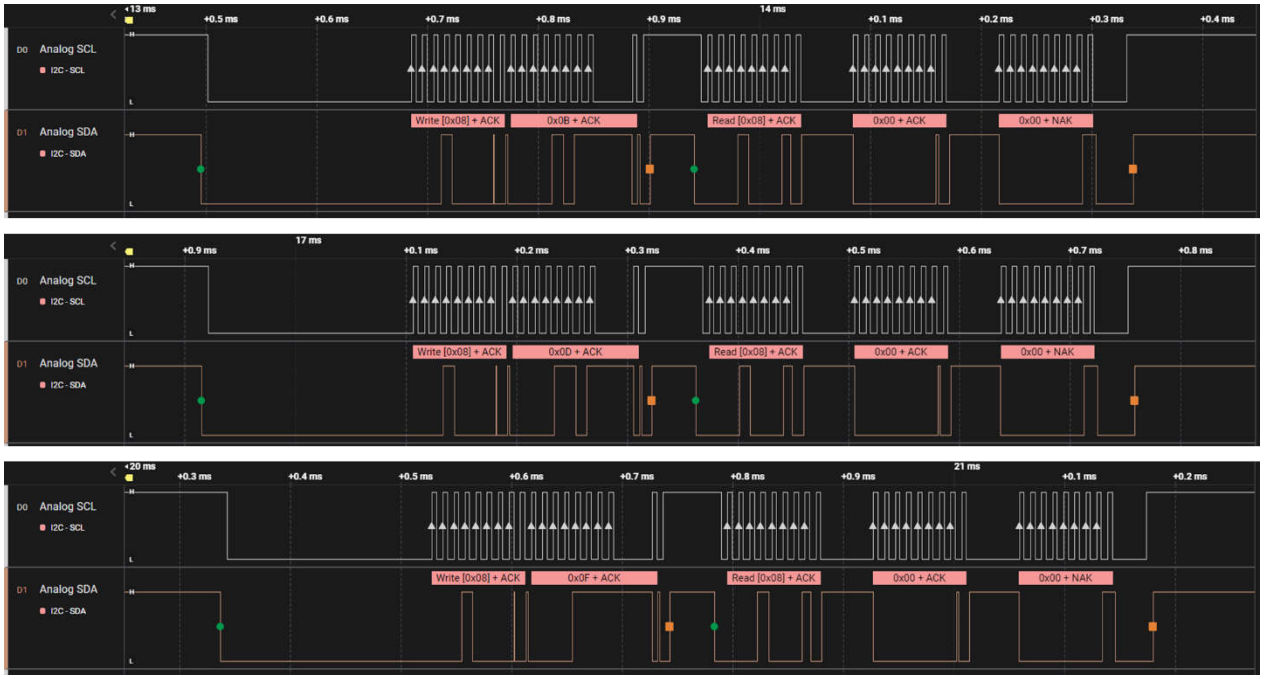


Figure 4-3. Captured I2C Waveform for PF Status Reading

4.4 Read Current

Read Current uses the `DirectCommand` function to read the CC2 current value from 0x3A and return the value. The read-back value is reported in mA.

Table 4-4 shows how to read the 16-bit current measurement from CC2. The current value in the following waveform is 0x00F, which converts to a decimal value of 15mA.

Table 4-4. Read Current Command Description

Command	Name	Unit	Type	Description
0x3A	CC2 Current	userA	I2	16-bit CC2 current

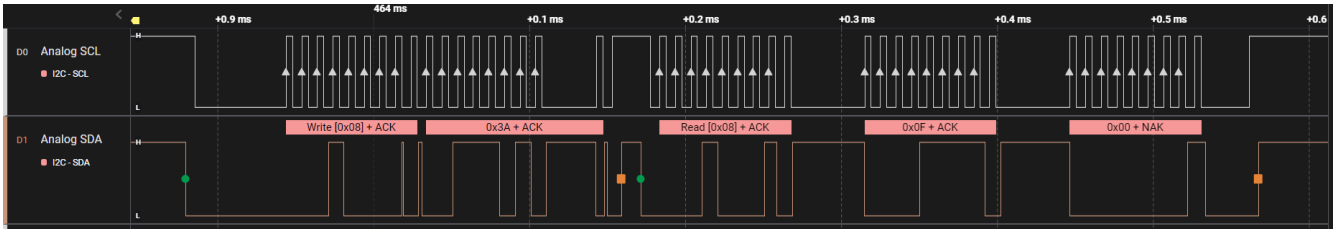


Figure 4-4. Captured I2C Waveform for Current Reading

4.5 Read All Temperatures

Read All Temperature takes in three commands as shown in Table 4-5 and reads them with the `DirectCommands` function, finally returning the float value. Table 4-5 shows how to read the TS Temperature of BQ76952. In this sample code, only TS1 and TS3 are configured, so only those two have returned values. In the following example, the reading of TS1 is 0x0B9D, representing a decimal value of 2973 (297.3K), then converts the value to about 24.15℃. The reading of TS3 is 0x0B9A, representing a decimal value of 2970 (297.0K), and then converts the value to about 23.85℃.

Table 4-5. TS Temperature Command Description

Command	Name	Unit	Type	Description
0x70	TS1 Temperature	0.1K	I2	When the TS1 pin is configured as a thermistor input, this reports the most recent temperature measurement. When configured as ADCIN, this instead reports the measured voltage at the TS1 pin in millivolts.
0x72	TS2 Temperature	0.1K	I2	When the TS2 pin is configured as a thermistor input, this reports the most recent temperature measurement. When configured as ADCIN, this instead reports the measured voltage at the TS2 pin in millivolts.
0x74	TS3 Temperature	0.1K	I2	When the TS3 pin is configured as a thermistor input, this reports the most recent temperature measurement. When configured as ADCIN, this instead reports the measured voltage at the TS3 pin in millivolts.

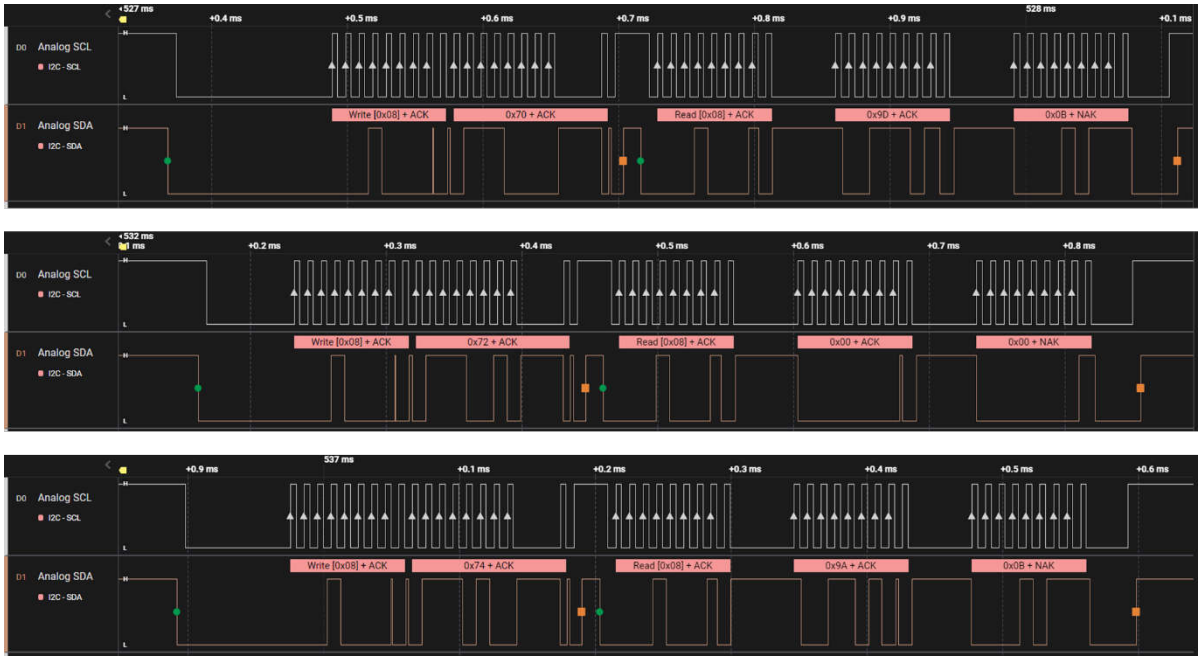


Figure 4-5. I2C Waveform for TS Temperature Reading

Temperature	float[3]	[24.1499996,-273.149994,23.8500004]	0x20000050
[0]	float	24.1499996	0x20000050
[1]	float	-273.149994	0x20000054
[2]	float	23.8500004	0x20000058

Figure 4-6. Register Values for TS Temperature Reading

4.6 Read All Voltages

Table 4-6 shows how to read the PF status of BQ76952. The Cell Voltages commands cover from 0x14 to 0x38, including all cell voltages, Stack voltage, PACK pin voltage, and LD pin voltage. In the following example, the 24.5V DC power supply is connected to the BQ76952EVM, and the Cell 1 voltage is read by writing the I2C command 0x14 followed by a 2-byte read. The data is returned in little format. In the following example, the 16-bit Cell 1 voltage reads 0x05FC, which corresponds to 1532mV and the Stack voltage reads 0x0988, which corresponds to 24.4V.

Table 4-6. Cell Voltage Command Description

Command	Name	Unit	Type	Description
0x14	Cell 1 Voltage	mV	I1	16-bit voltage on cell 1
0x16	Cell 2 Voltage	mV	I1	16-bit voltage on cell 2
0x18	Cell 3 Voltage	mV	I1	16-bit voltage on cell 3
0x1A	Cell 4 Voltage	mV	I1	16-bit voltage on cell 4
0x1C	Cell 5 Voltage	mV	I1	16-bit voltage on cell 5
0x1E	Cell 6 Voltage	mV	I1	16-bit voltage on cell 6
0x20	Cell 7 Voltage	mV	I1	16-bit voltage on cell 7
0x22	Cell 8 Voltage	mV	I1	16-bit voltage on cell 8
0x24	Cell 9 Voltage	mV	I1	16-bit voltage on cell 9
0x26	Cell 10 Voltage	mV	I1	16-bit voltage on cell 10
0x28	Cell 11 Voltage	mV	I1	16-bit voltage on cell 11
0x2A	Cell 12 Voltage	mV	I1	16-bit voltage on cell 12
0x2C	Cell 13 Voltage	mV	I1	16-bit voltage on cell 13
0x2E	Cell 14 Voltage	mV	I1	16-bit voltage on cell 14
0x30	Cell 15 Voltage	mV	I1	16-bit voltage on cell 15
0x32	Cell 16 Voltage	mV	I1	16-bit voltage on cell 16
0x34	Stack Voltage	mV	I1	16-bit voltage on top of stack
0x36	PACK Pin Voltage	mV	I1	16-bit voltage on PACK pin
0x38	LD Pin Voltage	mV	I1	16-bit voltage on LD pin

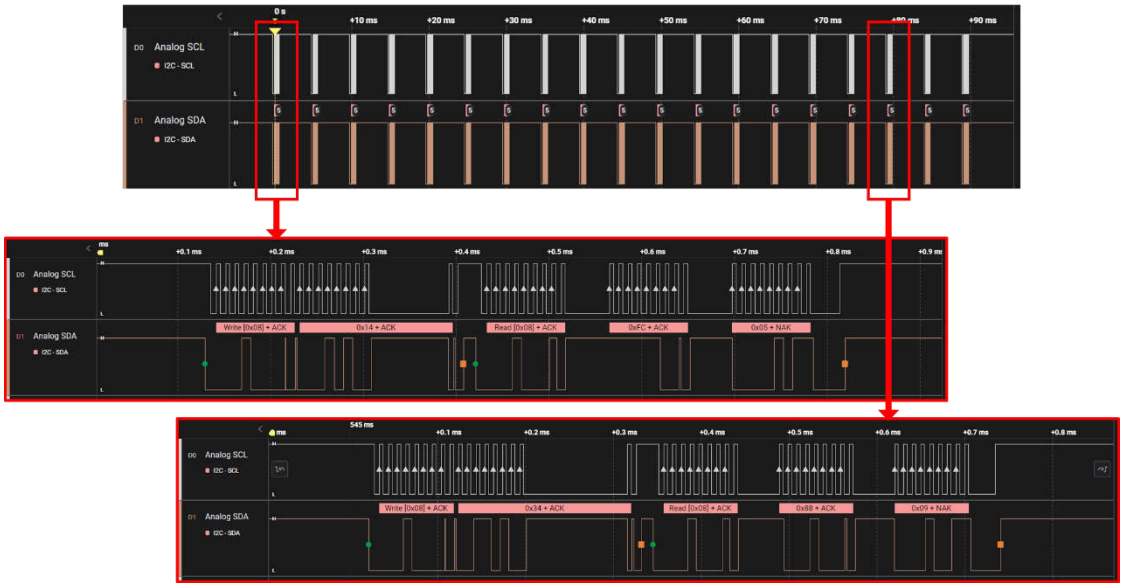
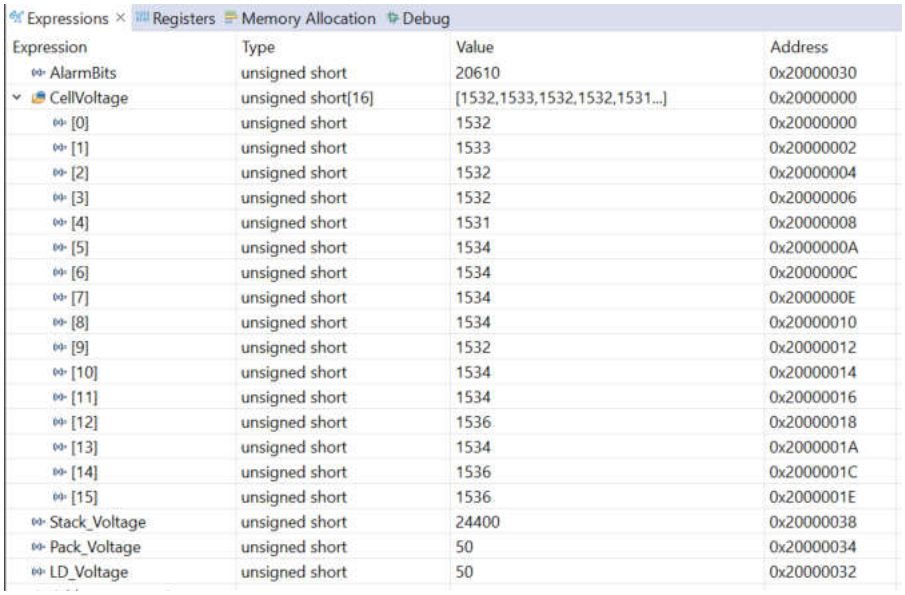


Figure 4-7. Captured I2C Waveform for Cell Voltage Reading



Expressions × Registers Memory Allocation Debug

Expression	Type	Value	Address
AlarmBits	unsigned short	20610	0x20000030
CellVoltage	unsigned short[16]	[1532,1533,1532,1532,1531...]	0x20000000
[0]	unsigned short	1532	0x20000000
[1]	unsigned short	1533	0x20000002
[2]	unsigned short	1532	0x20000004
[3]	unsigned short	1532	0x20000006
[4]	unsigned short	1531	0x20000008
[5]	unsigned short	1534	0x2000000A
[6]	unsigned short	1534	0x2000000C
[7]	unsigned short	1534	0x2000000E
[8]	unsigned short	1534	0x20000010
[9]	unsigned short	1532	0x20000012
[10]	unsigned short	1534	0x20000014
[11]	unsigned short	1534	0x20000016
[12]	unsigned short	1536	0x20000018
[13]	unsigned short	1534	0x2000001A
[14]	unsigned short	1536	0x2000001C
[15]	unsigned short	1536	0x2000001E
Stack_Voltage	unsigned short	24400	0x20000038
Pack_Voltage	unsigned short	50	0x20000034
LD_Voltage	unsigned short	50	0x20000032

Figure 4-8. Register Values for Cell Voltage Reading

5 References

- Texas Instruments, *BQ76942 Technical Reference Manual*
- Texas Instruments, *BQ76952 Technical Reference Manual*
- Texas Instruments, *MSPM0L1306 LaunchPad Development Kit User's Guide*
- Texas Instruments, *MSPM0G3507 LaunchPad Development Kit User's Guide*

6 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (May 2023) to Revision A (July 2024)	Page
• Added support for additional devices: MSPM0L1306 and MSPM0G3507	1
• Changed Figure 2-1, added the LP-MSPM0G3507 device to Figure 2-2, and added the LP-MSPM0G3507 connection to Table 2-1	3
• Changed the names of *Command-Only Subcommands* and *Subcommands* according to data in the *BQ76942 TRM* and the *BQ76952 TRM*	8
• Added reference documentation	15

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265