Application Note

BQ79600-Q1 Software Design Reference

Leslie Marquez Arroyo

ABSTRACT

This application note provides an outline for the basic communication between a host system and the BQ79600-Q1 bridge device with BQ79616-Q1 stacked battery monitor devices. This includes communication using either SPI interface or UART interface to communicate to the host. Examples, such as auto-addressing and reverse-addressing, are included to provide the user with simple demonstrations of the basic communication of the device. The information is meant to provide an overview of the communication information outlined in the *BQ79600-Q1 Automotive SPI/UART Communication Interface Functional-Safety Compliant With Automatic Host Wakeup* data sheet.

The communication examples used in this document are presented in a series of hexadecimal byte values. The actual device communication is sent in standard UART (universal asynchronous receiver-transmitter) format or SPI (Serial Peripheral Interface) format.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

Copyright © 2023 Texas Instruments Incorporated

1 Command Frames

Reading and writing registers using command frames underlies nearly all basic communication with the BQ79600-Q1. Command frames on BQ79600-Q1 follow the same structure as command frames on BQ79616-Q1. All read and write commands are provided in hexadecimal format, in command frame order.

1.1 Structure

1.1.1 Initialization Byte

Type	Value
Single device read	0x80
Single device write	0x90
Stack read	0xA0
Stack write	0xB0
Broadcast read	0xC0
Broadcast write	0xD0
Broadcast write reverse direction	0xE0

1.1.2 Device ID Address

Device ID address is only used for single device read/write commands. One byte is used to represent the device ID address, for example: 0x02 for device address 0x02.

1.1.3 Register Address

Two bytes are used to represent the register address, for example: 0x0306 to read/write register address 0x306.

1.1.4 Data

For read commands, one byte is used. This number represents the number of bytes to read - 1, with a maximum of 128 bytes requested, for example: 0x00 to read one byte of data.

For write commands, this represents the data bytes to be written, with a maximum of eight bytes of data to send, for example: 0xA500 to write two bytes with data 0xA500.

1.1.5 CRC

Two bytes are used for the CRC, calculated using the CRC-16-IBM generator polynomial.

1.2 Command Frame Template Tables

Command frame format templates are provided below for single device read/write, stack read/write, and broadcast read/write. For bit-level detail on the command frames, see the "Data Communication Protocol" section of the *BQ79600-Q1 Automotive SPI/UART Communication Interface Functional-Safety Compliant With Automatic Host Wakeup* data sheet.

Table 1-1. Single Device Read Command Frame

	Data	Comments
Initialization Byte	0x80	Always 0x80
Device ID Address	0x01	Device address 0x01 is addressed in this case
Register Address	0x0215	Start with address 0x215
Data	0x0B	Send 12 bytes worth of data back (register contents from 0x215 to 0x220)
CRC	0xCAB5	

Copyright © 2023 Texas Instruments Incorporated

Table 1-2. Single Device Write Command Frame

	Data	Comments
Initialization Byte	0x93	Writing four data bytes to a single device (0x90 for 1 byte of data)
Device ID Address	0x01	Device address 0x01 is addressed in this case
Register Address	0x0100	Start with address 0x100
Data	0x02B778BC	Write 4 bytes to registers 0x100-0x103
CRC	0x8A4C	

Table 1-3. Stack Read Command Frame

	Data	Comments
Initialization Byte	0xA0	Always 0xA0
Device ID Address	--	No address byte is sent in stack read
Register Address	0x0215	Start with address 0x215
Data	0x0B	Send 12 bytes worth of data back (register contents from 0x215 to 0x220) from each device in the stack
CRC	0xCCB3	

Table 1-4. Stack Write Command Frame

	Data	Comments
Initialization Byte	0xB3	Writing 4 bytes to the stack devices
Device ID Address	--	No address byte is sent in stack write
Register Address	0x0100	Start with address 0x100
Data	0x02B778BC	Write 4 bytes to registers 0x100-0x103 to all devices in stack
CRC	0x0A35	

Table 1-5. Broadcast Read Command Frame

	Data	Comments
Initialization Byte	0xC0	Always 0xC0
Device ID Address	--	No address byte is sent in broadcast mode
Register Address	0x0215	Start with address 0x215
Data	0x0B	Send 12 bytes worth of data back (register contents from 0x215 to 0x220).If register address is not a valid register address on BQ79600-Q1, the device will attach zeros to the response frame in the place where its response would have been
CRC	0xD2B3	

Table 1-6. Broadcast Write Command Frame

	Data	Comments
Initialization Byte	0xD3	Writing four bytes to all the devices
Device ID Address	--	No address byte is sent in broadcast mode
Register Address	0x0100	Start with address 0x100
Data	0x02B778BC	Write four bytes to registers 0x100-0x103 to all devices
CRC	0x6A33	

Copyright © 2023 Texas Instruments Incorporated

1.3 Read Register and Write Register Functions

When using the BQ79600 UART sample code, *ReadReg* function is used to generate a read command and receive a response, and *WriteReg* function is used to generate a write command. When using the BQ79600 SPI sample code, the names of these functions are *SpiReadReg* and *SpiWriteReg*, respectively. These act as the primary communication wrapper functions between the TMS570 LaunchPad and the BQ79600-Q1. The CRC is automatically calculated and appended by these functions.

1.3.1 ReadReg/SpiReadReg

The basic structure for the *ReadReg* and *SpiReadReg* functions is as follows:

```
UART sample code:
#_of_Read_Bytes = ReadReg(Device_Address, Register_Address, Incoming_Data_Byte_Array, #_Data_Bytes,
ms_Before_Time_Out, Packet_Type)
SPI sample code:
#_of_Read_Bytes = SpiReadReg(Device_Address, Register_Address, Incoming_Data_Byte_Array,
#_Data_Bytes, ms_Before_Time_Out, Packet_Type)
```

Device_Address, *#_Data_Bytes*, *ms_Before_Time_Out,* and *Packet_Type* are integers. *Register_Address* is a hex value (with the prefix "0x"). *Incoming_Data_Byte_Array* is an array of 1-byte hex values in the UART sample code and an array of 2-byte hex values in the SPI sample code.

Device_Address is ignored for broadcast and stack read commands.

For example:

```
UART sample code:
nRead = ReadReg(nDev_ID, 0x0306, bFrame, 12, 0, FRMWRT_SGL_R);
SPI sample code:
nRead = SpiReadReg(nDev_ID, 0x0306, bFrame, 12, 0, FRMWRT_SGL_R);
```

This line reads 12 bytes of data from register 0x0306 of the device *nDev_ID* and stores it in a local byte array (on the microcontroller) called *bFrame*. The packet type is a single device read.

1.3.2 WriteReg/SPIWriteReg

The basic structure for the *WriteReg* and *SpiWriteReg* function is as follows:

```
UART sample code:
#_of_Sent_Bytes = WriteReg(Device_Address, Register_Address, Data, #_Data_Bytes, Packet_Type)
SPI sample code:
#_of_Sent_Bytes = SpiWriteReg(Device_Address, Register_Address, Data, #_Data_Bytes, Packet_Type)
```

Device_Address, *#_Data_Bytes* and *Packet_Type* are integers, while *Register_Address* and *Data* are hex values (with the prefix "0x"). *Device_Address* is ignored for broadcast and stack writes.

For example:

```
UART sample code:
nSent = WriteReg(nDev_ID, 0x0306, 0x01, 1, FRMWRT_SGL_NR);
SPI sample code:
nSent = SpiWriteReg(nDev_ID, 0x0306, 0x01, 1, FRMWRT_SGL_NR);
```

This line writes to register 0x0306 of the device *nDev_ID* with one byte of data. The data sent is 0x01. The type of packet is a single device write.

Copyright © 2023 Texas Instruments Incorporated

1.3.3 Packet Types Available in Sample Code

The following table provides the various packet types available for use in the *ReadReg*/*SpiReadReg* and *WriteReg*/*SpiWriteReg* functions:

Frame Signifier	Packet Type
FRMWRT_SGL_W	Single Device Write
FRMWRT_SGL_R	Single Device Read
FRMWRT_STK_W	Stack Write
FRMWRT_STK_R	Stack Read
FRMWRT_ALL_W	Broadcast Write
FRMWRT_ALL_R	Broadcast Read

2 Quick Start Guide

To get started with measurements quickly, all that is required is the "Wake Sequence", "Auto-Addressing", and "Read Cell Voltages" sections of this guide.

2.1 Wake Sequence

When using UART, the wake ping is applied by the microcontroller to the MOSI/RX pin of the BQ79600-Q1 device by pulling the line low for 2.75 ms.

When using SPI, the microcontroller must pull the nCS line low, wait 2 µs, pull the MOSI/RX line low for a duration of 2.75 ms and pull it back high, wait 2 µs, and finally pull nCS pin high again.

To wake the device:

1. Send a wake ping (as described above).
2. Wait at least 3.5 ms.
3. Send a single device write to BQ79600-Q1 to set CONTROL1[SEND_WAKE]=1, which wakes up all stacked devices.

```
90 00 03 09 20 13 95       //Step 3 (wake up stacked devices)
```

4. Wait appropriate time to allow all devices to receive the WAKE tone and enter ACTIVE mode. To calculate the total time to wait, add the WAKE tone durantion (~1.6 ms) plus the time to enter ACTIVE mode (~10 ms) and multiply the result by the number of stacked BQ7961X-Q1 devices.

Note

If BQ79600-Q1 device is shut down through SHUTDOWN ping, the COMH RX and COML RX are disabled at next wake up. In such case, on step 1, host needs to first send a WAKE ping, wait at least 3.5 ms and then send a second WAKE ping. COMH RX and COML RX will be enabled after the second WAKE ping. Then proceed with steps 2 to 4.

2.2 Auto-Addressing

The following is the standard direction auto-address procedure.

2.2.1 Steps

1. Dummy stack write registers OTP_ECC_DATAIN1 to OTP_ECC_DATAIN8 = 0x00 to sync the DLL (delay-locked loop). These are 8 stack write commands.
2. Broadcast write to enable auto-addressing mode (CONTROL1=0x01).
3. Broadcast write consecutively to DIR0_ADDR = 0, 1, 2, 3 (register address 0x306).
4. Broadcast write to set all devices as stack device first (COMM_CTRL=0x02).
5. Single device write to the highest device in the stack to configure it as both stack and top of stack (COMM_CTRL=0x03).
6. Dummy stack read registers OTP_ECC_DATAIN1 to OTP_ECC_DATAIN8 to sync the DLL. These are 8 stack read commands.

Copyright © 2023 Texas Instruments Incorporated

2.2.2 Example Commands for a Stack of 3 Devices

```
B0 03 43 00 E7 D4       //Step 1 (dummy write OTP_ECC_DATAIN1 to sync DLL)
B0 03 44 00 E5 E4       //Step 1 (dummy write OTP_ECC_DATAIN2 to sync DLL)
B0 03 45 00 E4 74       //Step 1 (dummy write OTP_ECC_DATAIN3 to sync DLL)
B0 03 46 00 E4 84       //Step 1 (dummy write OTP_ECC_DATAIN4 to sync DLL)
B0 03 47 00 E5 14       //Step 1 (dummy write OTP_ECC_DATAIN5 to sync DLL)
B0 03 48 00 E0 E4       //Step 1 (dummy write OTP_ECC_DATAIN6 to sync DLL)
B0 03 49 00 E1 74       //Step 1 (dummy write OTP_ECC_DATAIN7 to sync DLL)
B0 03 4A 00 E1 84       //Step 1 (dummy write OTP_ECC_DATAIN8 to sync DLL)
D0 03 09 01 0F 74       //Step 2 (enable auto-addressing mode)
D0 03 06 00 CB 44       //Step 3 (set bridge device address DIR0_ADDR = 0)
D0 03 06 01 0A 84       //Step 3 (set stack 1 device address DIR0_ADDR = 1)
D0 03 06 02 4A 85       //Step 3 (set stack 2 device address DIR0_ADDR = 2)
D0 03 06 03 8B 45       //Step 3 (set stack 3 device address DIR0_ADDR = 3)
D0 03 08 02 4E E5       //Step 4 (set all stacked devices as stack)
90 03 03 08 03 53 98    //Step 5 (set stack 3 as both stack and top of stack)
A0 03 43 00 E3 14       //Step 6 (dummy read OTP_ECC_DATAIN1 to sync DLL)
A0 03 44 00 E1 24       //Step 6 (dummy read OTP_ECC_DATAIN2 to sync DLL)
A0 03 45 00 E0 B4       //Step 6 (dummy read OTP_ECC_DATAIN3 to sync DLL)
A0 03 46 00 E0 44       //Step 6 (dummy read OTP_ECC_DATAIN4 to sync DLL)
A0 03 47 00 E1 D4       //Step 6 (dummy read OTP_ECC_DATAIN5 to sync DLL)
A0 03 48 00 E4 24       //Step 6 (dummy read OTP_ECC_DATAIN6 to sync DLL)
A0 03 49 00 E5 B4       //Step 6 (dummy read OTP_ECC_DATAIN7 to sync DLL)
A0 03 4A 00 E5 44       //Step 6 (dummy read OTP_ECC_DATAIN8 to sync DLL)
```

Explanation of the first stack write command frame (B0 03 43 00 E7 D4):

- B0 = stack write of one byte
- 0343 = register address
- 00 = write value 0x00
- E7D4 = CRC

Explanation of the first broadcast write command frame (D0 03 09 01 0F 74):

- D0 = broadcast write of one byte
- 0309 = register address
- 01 = write value 0x01
- 0F74 = CRC

Explanation of first single device write command frame (90 03 03 08 03 53 98):

- 90 = single device write of one byte
- 03 = device address
- 0308 = register address
- 03 = write value 0x03
- 5398 = CRC

Explanation of first stack read command frame (A0 03 43 00 E3 14):

- A0 = stack read
- 0343 = register address
- 00 = read one byte of data
- E314 = CRC

2.3 Read Cell Voltages

2.3.1 Steps

1. Set all used cells to active. For example, for 16 cells ACTIVE_CELL=0x0A.
2. Set the desired run mode and start the ADC. For example, for continuous run ADC_CTRL1=0x06.
3. Wait the required round-robin time (192us per round robin, plus any reclocking delays from writing the ADC_CTRL1 register).
4. Loop read the appropriate cell measurement registers. For example, VCELL16_HI to VCELL1_LO.

Copyright © 2023 Texas Instruments Incorporated

2.3.2 Example Commands for a Stack of 3 Devices

```
B0 00 03 0A A6 13                  //Step 1 (16 active cells)
B0 03 0D 06 52 76                  //Step 2 (set continuous run and start ADC)
delay [192us + (5us x TOTALBOARDS)] //Step 3 (delay)
A0 05 68 1F 5C 2D                  //Step 4 (read ADC measurements)
```

2.3.3 Convert to Voltages

To convert 16-bit ADC values to actual voltages:

1. Convert the 16 bit value from two's complement form to a 16 bit decimal value.
2. Multiply by the ADC resolution (190.73 µV/LSB).

2.4 Reverse Addressing

This example provides details for reverse addressing the entire daisy chain. Note that once addressing in both directions is complete, the host can skip auto-addressing when changing directions.

2.4.1 Steps

1. Single device write to BQ79600-Q1 to set CONTROL1[DIR_SEL]=1 (CONTROL1=0x80).
2. Single device write to BQ79600-Q1 to set CONTROL1[SEND_WAKE]=1 without overwritting DIR_SEL bit (CONTROL1=0xA0).
3. Dummy stack write registers OTP_ECC_DATAIN1 to OTP_ECC_DATAIN8 = 0x00 to sync the DLL (delay-locked loop). These are 8 stack write commands.
4. Send a **broadcast write reverse** command to change direction on the stacked devices (CONTROL1=0x80). This command type should only ever be used for this one scenario, where the user is changing the direction of the daisy chain communications. Do not use this for other commands.
5. Broacast write to set all devices as stack device (COMM_CTRL=0x02). This clears the TOP_STACK bit if previously set in the North direction.
6. Broadcast write to enable auto-addressing mode (CONTROL1=0x81).
7. Broadcast write consecutively to DIR1_ADDR = 0, 1, 2, 3 (register address 0x307).
8. Broadcast write to set all devices as stack device first (COMM_CTRL=0x02).
9. Single device write to top device to configure it as both stack and top of stack (COMM_CTRL=0x03).
10. Dummy stack read registers OTP_ECC_DATAIN1 to OTP_ECC_DATAIN8 to sync the DLL. These are 8 stack read commands.

Copyright © 2023 Texas Instruments Incorporated

2.4.2 Example Commands for a Stack of Three Devices

```
90 00 03 09 80 13 ED   //Step 1 (change BQ79600-Q1 direction)
90 00 03 09 A0 12 35   //Step 2 (wake up stack devices)
B0 03 43 00 E7 D4      //Step 3 (dummy write OTP_ECC_DATAIN1 to sync DLL)
B0 03 44 00 E5 E4      //Step 3 (dummy write OTP_ECC_DATAIN2 to sync DLL)
B0 03 45 00 E4 74      //Step 3 (dummy write OTP_ECC_DATAIN3 to sync DLL)
B0 03 46 00 E4 84      //Step 3 (dummy write OTP_ECC_DATAIN4 to sync DLL)
B0 03 47 00 E5 14      //Step 3 (dummy write OTP_ECC_DATAIN5 to sync DLL)
B0 03 48 00 E0 E4      //Step 3 (dummy write OTP_ECC_DATAIN6 to sync DLL)
B0 03 49 00 E1 74      //Step 3 (dummy write OTP_ECC_DATAIN7 to sync DLL)
B0 03 4A 00 E1 84      //Step 3 (dummy write OTP_ECC_DATAIN8 to sync DLL)
E0 03 09 80 C0 14      //Step 4 (broadcast write reverse command)
D0 03 08 02 4E E5      //Step 5 (set all devices as stack)
D0 03 09 81 0E D4      //Step 6 (enable auto-addressing mode)
D0 03 07 00 CA D4      //Step 7 (set bridge device address DIR1_ADDR = 0)
D0 03 07 01 0B 14      //Step 7 (set stack 1 device address DIR1_ADDR = 1)
D0 03 07 02 4B 15      //Step 7 (set stack 2 device address DIR1_ADDR = 2)
D0 03 07 03 8A D5      //Step 7 (set stack 3 device address DIR1_ADDR = 3)
D0 03 08 02 4E E5      //Step 8 (set all devices as stack)
90 03 03 08 03 53 98   //Step 9 (set stack 3 as both stack and top of stack)
A0 03 43 00 E3 14      //Step 10 (dummy read OTP_ECC_DATAIN1 to sync DLL)
A0 03 44 00 E1 24      //Step 10 (dummy read OTP_ECC_DATAIN2 to sync DLL)
A0 03 45 00 E0 B4      //Step 10 (dummy read OTP_ECC_DATAIN3 to sync DLL)
A0 03 46 00 E0 44      //Step 10 (dummy read OTP_ECC_DATAIN4 to sync DLL)
A0 03 47 00 E1 D4      //Step 10 (dummy read OTP_ECC_DATAIN5 to sync DLL)
A0 03 48 00 E4 24      //Step 10 (dummy read OTP_ECC_DATAIN6 to sync DLL)
A0 03 49 00 E5 B4      //Step 10 (dummy read OTP_ECC_DATAIN7 to sync DLL)
A0 03 4A 00 E5 44      //Step 10 (dummy read OTP_ECC_DATAIN8 to sync DLL)
```

3 Revision History

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

Changes from Revision * (August 2020) to Revision A (October 2023)	Page
• Updated the numbering format for tables, figures, and cross-references throughout the document	1
• Initial release	2

Copyright © 2023 Texas Instruments Incorporated

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
Copyright © 2023, Texas Instruments Incorporated