Application Report
SLUA790–November 2016

Using I²C Communications With the bq34110, bq35100, and bq34z100-G1 Series of Gas Gauges

Battery Management Solutions

ABSTRACT

This application report provides examples that illustrate the bit-transaction details of I²C commands used with TI's bq34xxx series and bq35xxx series battery fuel gauges. The first two examples provide detailed explanations of using I²C commands to read standard parameters and *Control Subcommand* parameters. The first example shows how to implement a standard fuel gauge command that interrogates the gauge for reading cell voltage. The second example shows the added bit transitions required to access a *Control Subcommand*, specifically requesting firmware version information. Other I²C commands can be executed in the same manner, using the methodology of these two examples. The final sections provide step-by-step examples using the *bqStudio I²C Master Control Panel* to read and write parameters from these gauges.

Trademarks

1 Example 1: Reading Cell Voltage

I²C commands are always initiated by the host with a START (S) bit sequence, immediately followed by a 7-bit I²C address with the most-significant bit (MSB) sent first. An eighth bit of 0 is sent by the host, indicating that the next byte to be sent will be a write to the gauge. For the bq34xxx series and bq35xxx series of parts, these 8 bits form the byte 0xAA. Once the start bit and address byte have been successfully received by the gauge, the gauge responds with an ACKNOWLEDGE (A) bit sequence. The gauge is now ready for the subsequent command directive from the host. Further descriptions of the control bit sequences are presented in the Glossary: Control-Bit-Sequence Definitions.

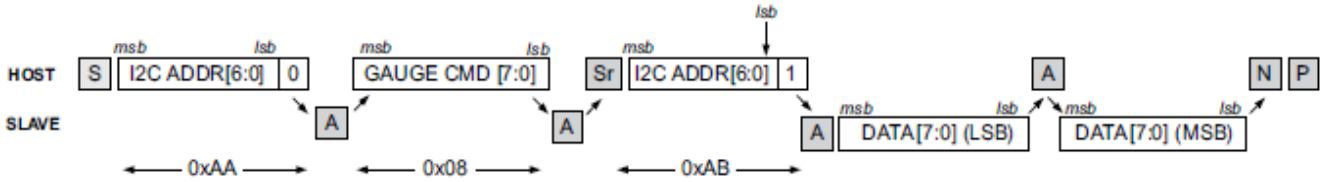


Figure 1. Simple I²C Command Transaction Sequence

After the successful transmission of the I²C address and read or write bit, the gauge command code can be sent by the host, in this case 0x08 for the *Voltage( )* command. The command code is actually a base address location within the gauge and must not be confused with the I²C address for the gauge. Once this location is sent by the host, the gauge responds by sending an ACKNOWLEDGE bit sequence and then executing the corresponding command subroutine. Even though two-byte locations are used for many gauge commands, writing to only the single byte is required to start gauge command processing. In this case, only 0x08 was written to the gauge, even though the command consists of the two consecutive command bytes 0x08 and 0x09. Most commands consist of two bytes, because the data is returned to these command locations and are also two bytes – the least-significant byte (LSB) is stored in the lower address (0x08), whereas the MSB is stored in the higher address (0x09). Like the I²C address data, the gauge command is sent MSB-first.

The host initiates the reading of the command data by sending a REPEAT START (Sr) bit sequence. This is immediately followed by the 7-bit I²C address of the gauge plus the read-bit directive (1), which together create the byte 0xAB. The gauge responds with an ACKNOWLEDGE bit sequence, then takes control of the data bus. The first data byte (LSB) is stored at the 0x08 location in the gauge and is strobed out by the gauge MSB-first. If the host responds with an ACKNOWLEDGE bit sequence, the gauge automatically increments the command location to 0x09, then strobes out the MSB stored there. If the host responds with a NO ACKNOWLEDGE (P or NACK) bit transmission, no further data is spooled to the host. The host terminates the present command packet by sending a STOP bit.

2 Example 2: Reading the Firmware Version

Reading the firmware version is an example of using the bq34z100-G1 subcommands. Subcommands are unique, as they represent another level of depth into the gauge command structure. All subcommands are accessed through the paired command locations at 0x00 and 0x01 in the gauge. The subcommands are written LSB-first. Hence, to send the FW_VERSION subcommand (0x0002), the host writes 0x02 to command location 0x00 and 0x00 to 0x01. Again, I²C always writes the MSB first. The format for address/command/data exchange between host and gauge is similar to the previous example and is shown in Figure 2. As in Example 1, the host initiates transmission with a START bit, followed by the gauge I²C address and a WRITE bit of 0 (0xAA). The gauge responds with an ACKNOWLEDGE, then the host specifies the command address of 0x00. Again, the gauge responds with an ACKNOWLEDGE. At this stage, the host must make additional writes to the gauge to set the subcommand code of 0x0002. Hence, the host sends the low byte of the subcommand (0x02). The gauge acknowledges. Then the host sends the high byte of the subcommand (0x00). The gauge issues an ACKNOWLEDGE. The host completes the writing process by issuing the STOP bit sequence. Now the gauge is prepared to return firmware information to the host.

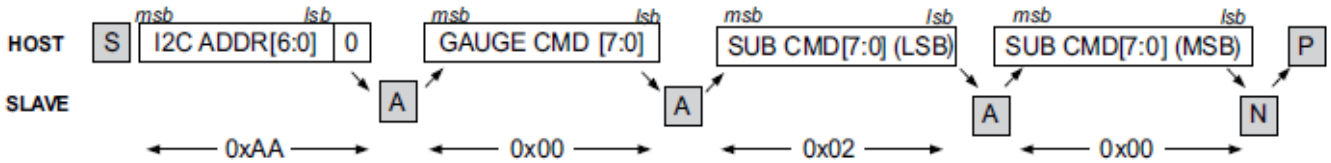


Figure 2. I²C Subcommand Initiation Sequence

To start the reading process, the host proceeds in a manner similar to Example 1, by issuing a START (S) bit sequence, immediately followed by the 7-bit I²C address of the gauge and the eighth bit of 0 (altogether, 0xAA for the bq34z100-G1 device). The gauge responds with an ACKNOWLEDGE bit sequence. The host sends the *Control( )* command of 0x00, and the gauge acknowledges. The gauge address location has now been set. To retrieve the data at 0x00 and 0x01, the host proceeds as before. It initiates the reading of the command data by sending a REPEATED START bit sequence. This is immediately followed by the 7-bit I²C address of the gauge plus the read-bit directive (1), which together create the byte 0xAB. The gauge responds with an ACKNOWLEDGE bit sequence, then takes control of the data bus. The first data byte (LSB) is stored at the 0x00 location in the gauge and is strobed out by the gauge MSB-first. If the host responds with an ACKNOWLEDGE bit sequence, the gauge automatically increments the command location to 0x01, then strobes out the MSB stored there. The host terminates the entire command process by sending a STOP bit.

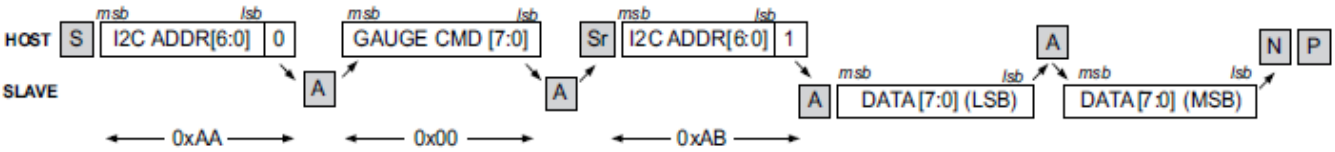


Figure 3. I²C Read Subcommand Sequence

2.1 Standard I²C Commands

We will use the *Voltage* as an example. Its command code is 08/09. Use the *I²C Master Control Panel* section to read the flash.

ENTER: Start Register *08*, Number of Bytes to Read *2* and PRESS the **Read** button.

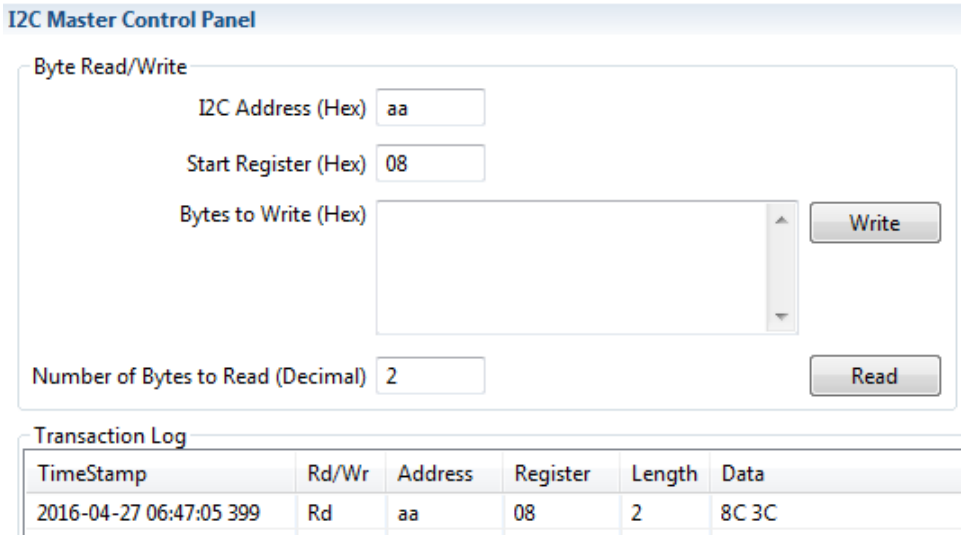


Figure 4. Standard I²C Command Reading Voltage

Byte swap the data and convert it to decimal. 3C8C is 15500 mV.

2.2 Extended I²C Commands

Read these like *Standard Commands*. As an example, use the *State-of-Health* (SOH), its command code is 2E/2F.

Use the *I²C Master Control Panel* section to read the flash.

ENTER: Start Register *2E*, Number of Bytes to Read *2*, and PRESS the **Read** button.

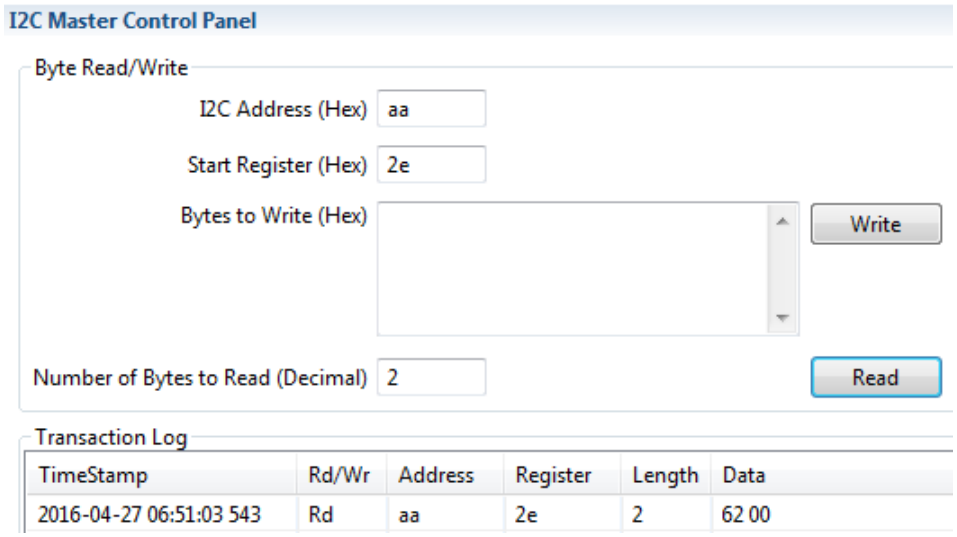


Figure 5. Extended I²C Command Reading State-of-Health

Byte swap the data and convert it to decimal. 0062 is 98% SOH.

2.3 Control Subcommands

Example 1:

DEVICE_TYPE Control() Subcommand 0001 and the correct answer is 0100 for the bq34z100-G1.

Use the I²C Master Control Panel section to read the flash.

ENTER: Start Register *00*, Bytes to Write *0100* and PRESS the **Write** button.
The I²C words occur in the following order in the data stream: AA, 00, 01, 00.

ENTER: Start Register *00*, Number of Bytes to Read *2* and PRESS the **Read** button.
The I²C words occur in the following order in the data stream: AA, 00, AB, 00, 01.

The device returns 0001, which is Little Endian for 0100. This is the DEVICE_TYPE for the bq34z100-G1.

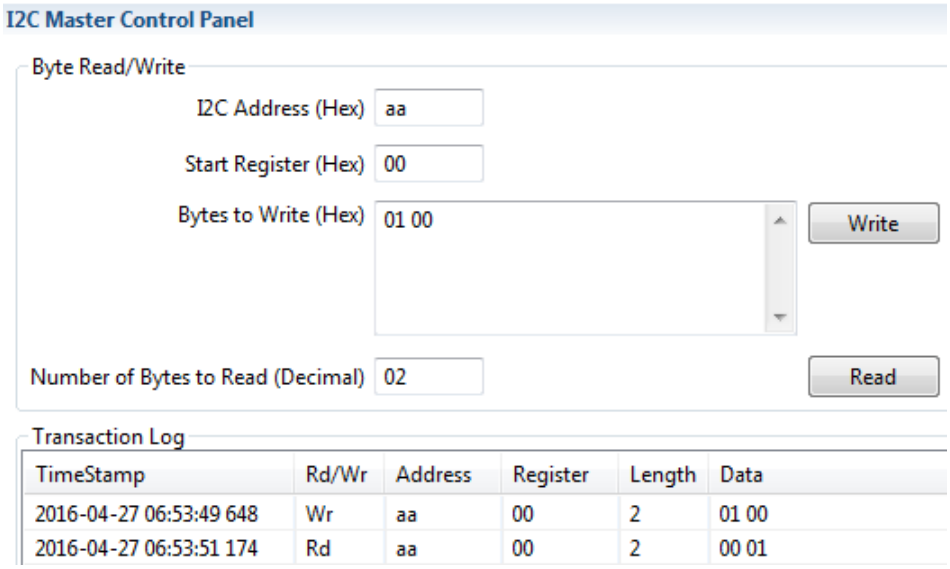


Figure 6. I²C Control Subcommand Reading DEVICE_TYPE

Example 2:

CHEM_ID Control() Subcommand 0008 and the correct answer is 0107 for the bq34z100-G1.

Use the *I²C Master Control Panel* section to read the flash.

ENTER: Start Register *00*, Bytes to Write *0800* and PRESS the **Write** button.
The I²C words occur in the following order in the data stream: AA, 00, 08, 00.

ENTER: Start Register *00*, Number of Bytes to Read *2* and PRESS the **Read** button.
The I²C words occur in the following order in the data stream: AA, 00, AB, 07, 01.

The device returns 0701, which is Little Endian for 0107. This is the default ChemID for the bq34z100-G1.

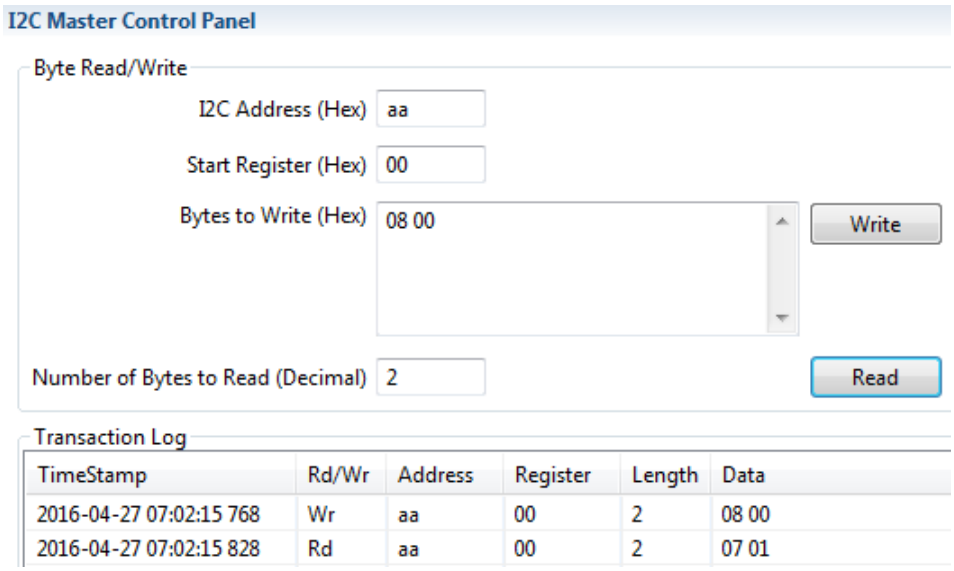


Figure 7. I²C Control Subcommand Reading CHEM_ID

Example 3:

FW_VERSION Control() Subcommand 0002 returns both the DEVICE_NUMBER and the FW_VERSION on the bq34110 and bq35100. You must read 4 bytes of data.

Use the *I²C Master Control Panel* section to read the flash.

ENTER: Start Register *3E*, Bytes to Write *0200* and PRESS the **Write** button.

The I²C words occur in the following order in the data stream: AA, 3E, 02, 00.

ENTER: Start Register *40*, Number of Bytes to Read *4* and PRESS the **Read** button.

The I²C words occur in the following order in the data stream: AA, 40, AB, 01, 00, 01, 02.

The device returns 0100 0102 and it is not returned Little Endian. The DEVICE_NUMBER is 0100 for the bq35100 and the FW_VERSION is 0102.

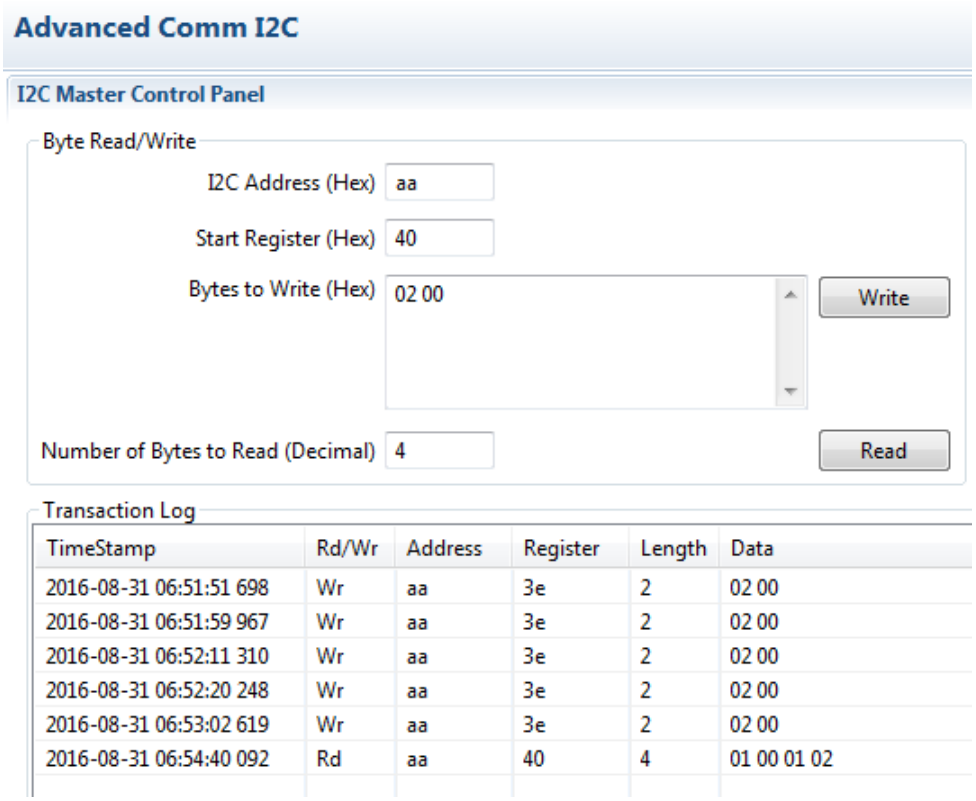


Figure 8. I²C Control Subcommand Reading FW_VERSION

2.4 Data Flash Access for the bq34z100-G1

Example 1:

Find the SubClass and Offset for the data that you want to read. We will use *Serial Number* for this example. SubClass 48, Offset 04 and it occupies 2 bytes.

Convert the SubClass HEX. 48 = 30H

Use the *I²C Master Control Panel* section to read the flash.

Start Register *61*, Bytes to Write *00* (Enable Flash x'fer command)

Start Register *3E*, Bytes to Write *30* (SubClass address)

Start Register *3F*, Bytes to Write *00* (Enable General Purpose Block)

Start Register *40*, Number of Bytes to Read *20*

The Serial number starts in the 5th byte. (0 is the first byte) The Serial Number is 0001 in this example.

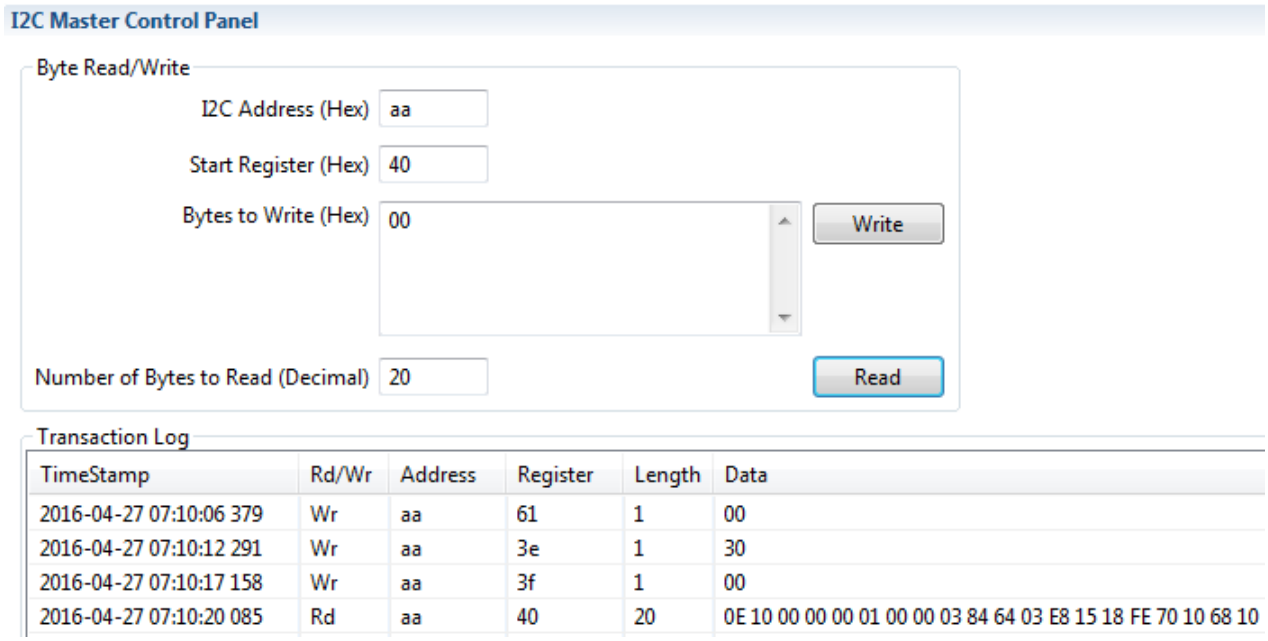


Figure 9. Data Flash Access Commands to Read the Serial Number

To read offset greater than 31, go to the next page; for example, I²C Command 3F, Byte 01

Example 2:

Find the SubClass and Offset for the data you want to read. Using *Device Chemistry* for this example; SubClass 30, Offset 55, and it occupies 5 bytes. The Offset exceeds 32 bytes, so read the data from the 2nd page of the SubClass. The *Device Chemistry* will be located in bytes 24-29 of the 2nd page. The first byte is the number of bytes in the string.

Convert the SubClass HEX. 48 = 30H

Use the I²C Master Control Panel section to read the flash.

Start Register *61*, Bytes to Write *00* (Enable Flash x'fer command)

Start Register *3E*, Bytes to Write *30* (SubClass address)

Start Register *3F*, Bytes to Write *01* (Enable General Purpose Block, 2nd page)

Start Register *40*, Number of Bytes to Read *32*

This is the data that was returned. The Device Chemistry starts in the 24th byte. (0 is the first byte) The 04 is the number of bytes currently programmed and 4C 49 4F 4E are ASCII for LION

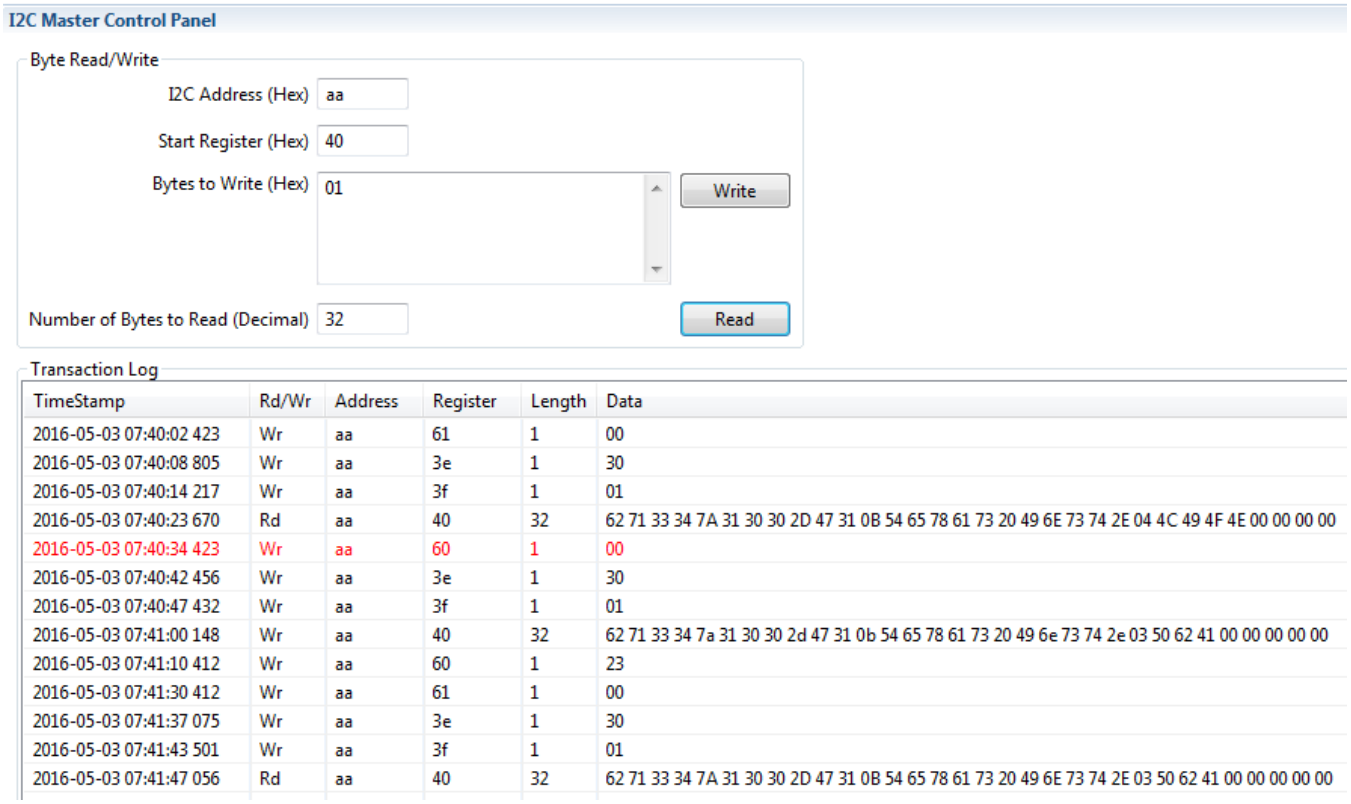
62 71 33 34 7A 31 30 30 2D 47 31 0B 54 65 78 61 73 20 49 6E 73 74 2E 04 4C 49 4F 4E 00 00 00 00

We want to change LION to PbA and PbA is 50 62 41 in ASCII. The new string will be:

62 71 33 34 7a 31 30 30 2d 47 31 0b 54 65 78 61 73 20 49 6e 73 74 2e 03 50 62 41 00 00 00 00 00

Where 03 is the number of bytes.

You will also need to calculate the checksum. Add the 32 bytes of data in hexadecimal and inverse the bits on the last byte to find the checksum. The string adds up to 7DC. The inverse of the last byte is 23, so this will be entered with the *60* command. Here is the full process to read the initial data flash contents, enter the new data and read the data flash to verify that it is correct.



TimeStamp	Rd/Wr	Address	Register	Length	Data
2016-05-03 07:40:02 423	Wr	aa	61	1	00
2016-05-03 07:40:08 805	Wr	aa	3e	1	30
2016-05-03 07:40:14 217	Wr	aa	3f	1	01
2016-05-03 07:40:23 670	Rd	aa	40	32	62 71 33 34 7A 31 30 30 2D 47 31 0B 54 65 78 61 73 20 49 6E 73 74 2E 04 4C 49 4F 4E 00 00 00 00
2016-05-03 07:40:34 423	Wr	aa	60	1	00
2016-05-03 07:40:42 456	Wr	aa	3e	1	30
2016-05-03 07:40:47 432	Wr	aa	3f	1	01
2016-05-03 07:41:00 148	Wr	aa	40	32	62 71 33 34 7a 31 30 30 2d 47 31 0b 54 65 78 61 73 20 49 6e 73 74 2e 03 50 62 41 00 00 00 00 00
2016-05-03 07:41:10 412	Wr	aa	60	1	23
2016-05-03 07:41:30 412	Wr	aa	61	1	00
2016-05-03 07:41:37 075	Wr	aa	3e	1	30
2016-05-03 07:41:43 501	Wr	aa	3f	1	01
2016-05-03 07:41:47 056	Rd	aa	40	32	62 71 33 34 7A 31 30 30 2D 47 31 0B 54 65 78 61 73 20 49 6E 73 74 2E 03 50 62 41 00 00 00 00 00

Figure 10. Data Flash Access Commands to Change the Device Chemistry Value

3 Data Flash Access for the bq34110 and bq35100

Find the data flash address for the data that you want to read or change. Use the *Operation Config A* register on the bq35100 for this example. The parameter is located at 0z41b1 and it occupies 1 byte and can be found in the Technical Reference Manual (TRM) or on the *bqStudio Data Memory* screen.

Use the *Advanced Comm* section to read the data flash.

Start Register *3E*, Bytes to Write *b1 41* (data flash address)

Start Register *40*, Number of Bytes to Read *01*

The *Operation Config A* register is 80 in this example.

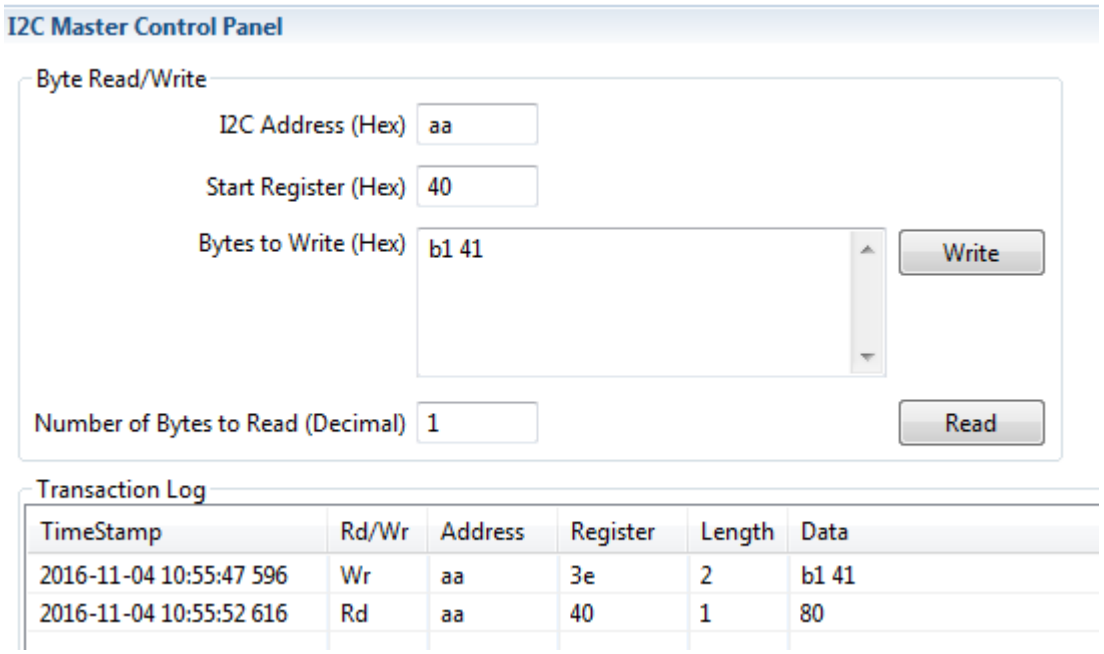


Figure 11. Data Flash Access Commands to Read the Operation Config A

Change the *Operation Config A* register to 0x82 to change to *End-of Service* mode.

Use the Advanced Comm section to write the data flash.

Start Register *3E*, Bytes to Write *b1 41* (data flash address)

Start Register *40*, Byte to Write *82* (new data)

Start Register *60*, Byte to Write *8b* (checksum)

Start Register *61*, Byte to Write *05* (length of data written)

The checksum is calculated by adding the starting address and the data in hexadecimal and the taking the inverse of the last byte. In the example, the checksum calculation will be 41 + b1 + 82 = 174. Inverse the last byte. 74 => 8b

The *Length of Data Written* is set to the number of bytes + 4. In this example it is set to 1 + 4 = 5.

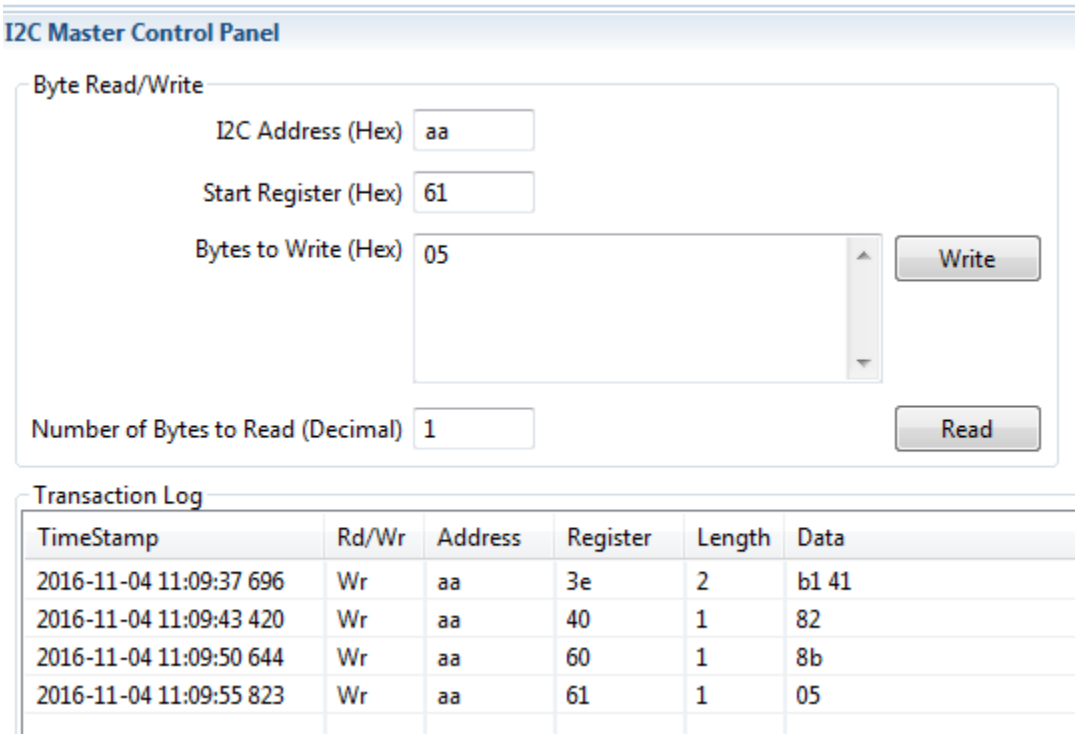


Figure 12. Data Flash Access Commands to Change the Operation Config A

4 Summary

In summary, remember the following critical aspects when implementing I^2C communication between host and bq34xxx and bq35xxx gauges:

1. Handshaking between the host and gauge is performed by five bit sequences: START, REPEATSTART, ACKNOWLEDGE, NO ACKNOWLEDGE, and STOP.
2. The I^2C address and read/write bit is the first data item sent at the beginning of an I^2C packet transmission. The I^2C address must not be mistaken for the command address in the gauge, the latter being the second data field to be transmitted by the host.
3. The host initiates all communication to the gauge and uses the WRITE directive at the end of the I^2C address (total byte is 0xAA). When followed by the gauge command, this sets the command address from which the host writes or reads gauge data.
4. When reading or writing multiple bytes, the host should use the base-address auto-increment feature of the gauge, rather than specifying the gauge address location each time a byte is transferred.
5. Whether command address or gauge data, all I^2C data is transferred between host and gauge with the least-significant byte first.
6. All bytes transferred between host and gauge is transferred most-significant bit first.

5 Glossary: Control-Bit-Sequence Definitions

START: The START-bit sequence is a host-generated bit that begins the transmission of every packet. It is defined by a high-to-low transition on the SDA line, while the SCL line is high.

REPEAT START: The REPEAT-START-bit sequence is also a host-generated bit. It has the same characteristics as the START bit, but appears in the middle of a packet transmission. It tells the slave that the gauge command (address) has been specified and data is ready for transfer to or from that command address.

STOP: The STOP-bit sequence indicates the end of a transmission packet. It is consists of a low-to-high transition of the SDA line, while the SCL line is high.

ACKNOWLEDGE: The ACKNOWLEDGE-bit sequence follows every (successful) data field sent between the host and gauge. The device receiving the data field is responsible for sending the bit sequence. The sequence consists of the SDA line being maintained in a low-status, while the SCL line is pulsed high.

NO ACKNOWLEDGE: The NO-ACKNOWLEDGE-bit sequence is frequently used by the host or gauge to indicate the last data byte has been transmitted/received, and that the host should terminate the packet with a STOP-bit sequence. It can also be used to indicate that an I^2C device is not listening or capable of responding to the host at a given time. The sequence consists of the SDA line floating in a high state, while the SCL line is pulsed high.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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