# Updating the bq275xx Firmware at Production

**Abstract**

The bq275xx devices are gas gauges that are flash based and if needed can be updated. There may be a case in which the gas gauge firmware has been updated but the system-side gas gauges such as bq27500 or bq27510 may already be embedded in an enclosed system that does not have external communication access to the gas gauge device to allow updating with external tools. At this point it may be useful that the application system main processor perform the actual firmware updating rather than having an external tool to do so. Also you may have the case in a pack side device implementation that you may need to update the data flash image but no longer have access to I2C and would have to use HDQ communication to make the update. This application note will provide all the information needed to implement a system firmware that will send the necessary I2C or HDQ commands to have the firmware or data flash updated in the bq275xx devices. This application note must be used along with a file of .bqfs type for firmware plus data flash updates or a file of .dffs type for dataflash only updates. There is also a tool that allows creating a bqfs and a dffs file with an encrypted srecord (senc) and optional file containing the specific Data Flash Image (DFI).

## 1. Flash Stream File Format

This section describes the file format for the TI BMS FlashStream (.bqfs) and Data Flash FlashStream (.dffs) files. These files contain instructions for I2C or HDQ operations required for updating a device's instruction flash (IF) and/or data flash (DF).

### *1.1 File Structure*

The .bqfs file is an ASCII text file which contains both commands and data.  Each line of the file represents one command and potentially 96 bytes of data, as described below.  No row will contain more than 96 data bytes.  The first two characters of each row represent the command, followed by a ":".

"W:" – indicates the row is a command to write one or more bytes of data.
"R:" – indicates the row is a command to read one or more bytes of data.
"C:" – indicates the row is a command to read and compare one or more bytes of data.
"X:" – indicates the row is a command to wait a given number of milliseconds before proceeding.

White space is used to separate fields within the .bqfs and .dffs files.  Each row contains one and only one of the above commands.

The commands discussed in this section can be implemented by a system that can perform multi-byte operations for I2C or single-byte operations for I2C and HDQ.

## *1.2 Write Command*

**<u>I2C</u>**
The write command "W:" instructs the I2C master to write one or more bytes to a given I2C address and given register address. The I2C address format used throughout this document is based on an 8-bit representation of the address. The format of this sequence is:
"W: I2CAddr RegAddr Byte0 Byte1 Byte2…".

For example, the following:

W: AA 55 AB CD EF 00

indicates that the I2C master should write the byte sequence 0xAB 0xCD 0xEF 0x00 to register 0x55 of the device addressed at 0xAA.

Or more precisely, it indicates to write the following data to the device address 0xAA:
0xAB to register 0x55
0xCD to register 0x56
0xEF to register 0x57
0x00 to register 0x58

**<u>HDQ</u>**
The write command "W:" instructs the HDQ master to write one byte to a given register address. The format of this sequence is:
"W: RegAddr Byte".

For example, the following:

W: 55 AB

indicates that the HDQ master should write the data 0xAB into register 0x55.

## *1.3 Read Command*

**<u>I2C</u>**
The read command "R:" instructs the I2C master to read a given number of bytes from a given I2C address and given register address. The format of this sequence is:
"R: I2CAddr RegAddr NumBytes"

For example, the following:

R: AA 55 100

indicates the I2C master should read 100 (decimal) bytes of data from register address 0x55 of device address 0xAA.

**HDQ**
The read command "R:" instructs the HDQ master to read a byte from a given register address. The read command for HDQ will always expect only one data byte in return. The format of this sequence is:
"R: RegAddr NumBytes"

For example, the following:

```
R: 55 1
```

indicates the HDQ master should read 1 byte from register address 0x55.

### *1.4 Read and Compare Command*

**I2C**
The read and compare command is formatted identically to the write command. The data presented with this command should match the data read exactly, or the operation should cease with an error indication to the user. The format of this sequence is:
"C: i2cAddr RegAddr Byte0 Byte1 Byte2…".

An example of this command is as follows:

```
C: AA 55 AB CD EF 00
```

This example expects the master to read back 4 bytes from the register address 0x55 of the device addressed at 0xAA and then compare the data to the values given on the line command in this same order as 0xAB, 0xCD, 0xEF and 0x00.

**HDQ**
The read and compare consists of making an HDQ read to a given register address and verify that the data received is same as expected by FlashStream command. The format of this sequence is:
"C: RegAddr Byte"

An example of this command is as follows:

```
C: 55 AB
```

This example expects the master to read back the data from register address 0x55 and compare to data 0xAB. An error indication should be provided to the user if data is not 0xAB.

*1.5 Wait Command*

The wait command indicates that the host should wait a minimum of the given number of milliseconds before continuing to the next row of the flash stream. For example, the following:

X: 200

indicates that the I2C or HDQ master must wait at least 200ms before continuing.

## 2. Firmware Updating Flow

The basic programming flow expected for customers to update the firmware and/or data flash of a bq275xx device is summarized in figure 1. To call upon the commands given in the bqfs of dffs file the user must ensure that the target device is in ROM mode. While in ROM mode, the target device responds to the I2C address of 0x16 (8 bit) or 0x0B (7 bit) if using I2C. From here on out the 8 bit I2C address reference will be used. To enter ROM mode, 0x0F00 must be written to register address 0x00 of the target device if in I2C mode, or 0x00 into register 0x00 and 0x0F into register 0x01 if in HDQ mode. Remember that the I2C address of the device is 0xAA while it is in normal gas gauge mode (default). This step of the process is not part of the bqfs or dffs command sequence. It is up to the customer to provide the Enter ROM Mode command before using the bqfs or dffs file (figure 2 and figure 3).

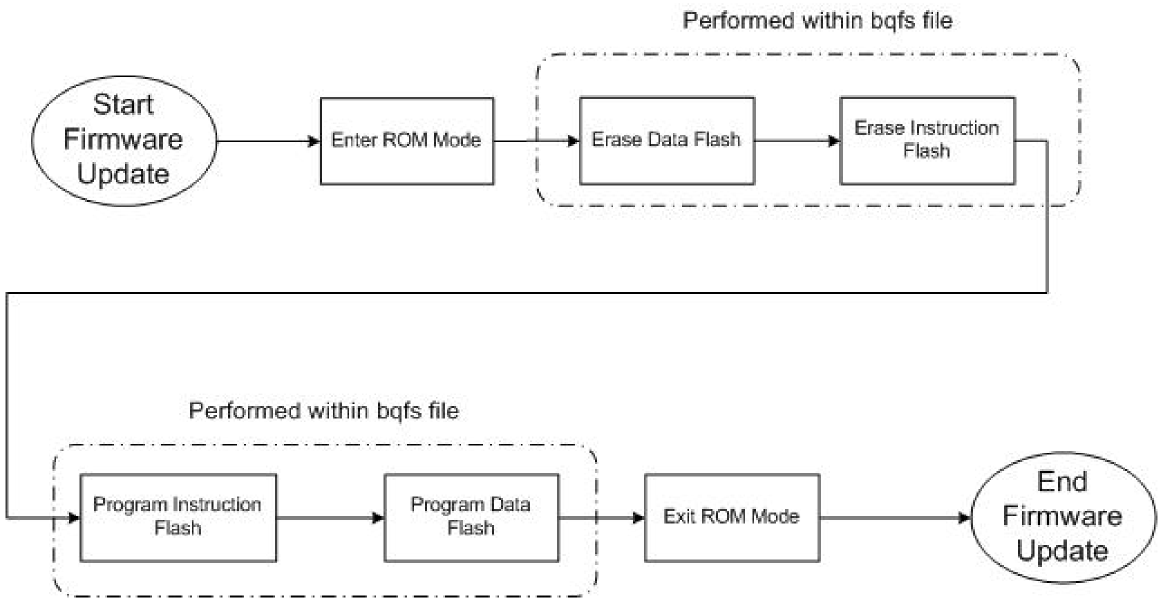


Figure 1. Basic Firmware Programming Flow

Before entering ROM mode it must be confirmed that the device is not sealed from a firmware perspective. To verify that the device is not sealed read Control Status by first writing 0x00 into register 0x00 and 0x01 and then read back register 0x01. If bit 5 is set then it will require sending the Unseal keys. If bit 6 is set then it will require sending the Full Access Unseal key. These keys are sent by writing the respective keys into the Control register. The keys are 32-bit each. If both keys are required then they must be entered in the order of Unseal key first and then Full Access Unseal key. For any 32-bit key there must not be any intermittent communication other than the actual communication to write the keys.

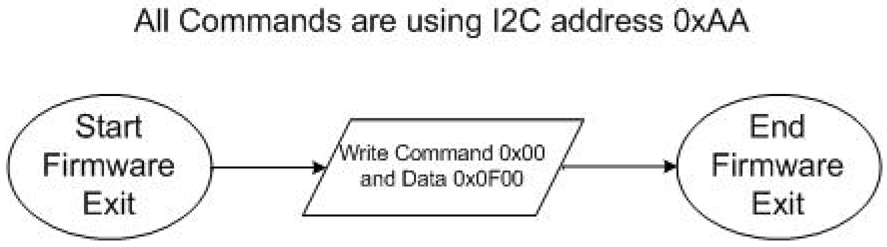

All Commands are using I2C address 0xAA
Start Firmware Exit
Write Command 0x00 and Data 0x0F00
End Firmware Exit

Figure 2. Command to enter ROM mode in I2C

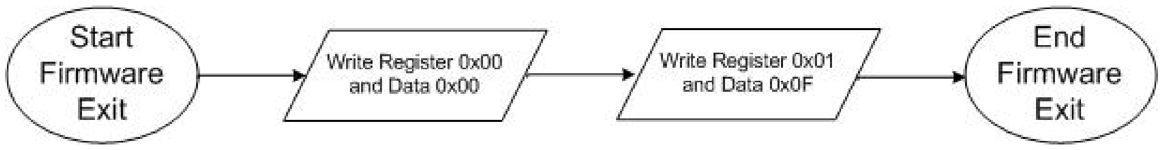
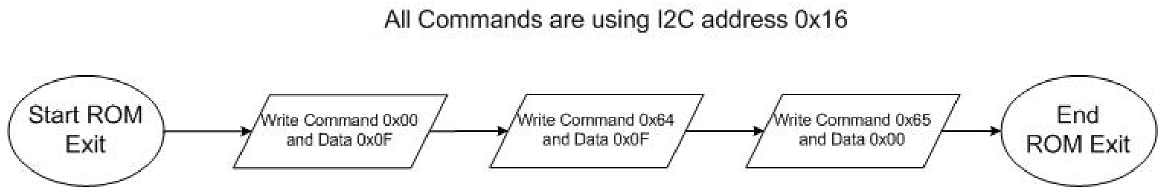

Start Firmware Exit
Write Register 0x00 and Data 0x00
Write Register 0x01 and Data 0x0F
End Firmware Exit

Figure 3. Commands to enter ROM mode in HDQ

In I2C it is evident when operating in firmware or ROM mode based on the I2C slave address. If the device sends an Acknowledge signal to the address 0xAA it indicates that it is in firmware mode and if it acknowledges the address 0x16 then it is in ROM mode. Unfortunately HDQ does not support slave addressing. The following process can be indicative as for in what mode the gas gauge IC is operating in:

1) Try to modify register address 0x04. It should be “read only” in firmware mode.
    a. x = ReadByte( 0x04 )
    b. WriteByte( 0x04, ~x ) // Write back different data
    c. y = ReadByte( 0x04 )
    d. if ( x == y ) then ‘not in ROM mode’
2) If step 1 says we should be in ROM mode, then proceed with bqfs or dffs sequence.

Once in ROM mode, the customer test setup should be able to open the bqfs or dffs file and perform each one of the I2C or HDQ transactions in strictly the same order as given with the FlashStream file. Upon completing all the commands within the FlashStream file, the customer test system should send the Exit ROM Mode procedure (figure 4). There must be at least a 250ms delay before attempting to proceed with I2C or HDQ communication to the target device using the I2C address 0xAA after exiting ROM Mode.


All Commands are using I2C address 0x16
Start ROM Exit
Write Command 0x00 and Data 0x0F
Write Command 0x64 and Data 0x0F
Write Command 0x65 and Data 0x00
End ROM Exit
Figure 4. ROM Mode Exit Sequence in I2C or HDQ

## 3. Debugging bqfs Reader and Programmer

It will take some development effort for any given customer to implement a program that can open a bqfs and give out I2C or HDQ commands based on bqfs format. During the development phase it would be recommended to have a dedicated prototype board that would allow to easily replacing target devices given that during debug of program it is likely that devices become useless.

The bqfs sequence is created such to minimize the chances of hindering any devices in the program debug process and also during actual production run. The strategy taken by the program developer should be that during debug that once placing a target device in ROM Mode that you do not allow exiting ROM Mode unless the full process is completed successfully. Exiting ROM Mode while firmware or data flash not fully programmed may put the device in an unrecoverable mode.

The bqfs file contains some sequences defined as Read and Compare. When implementing these commands within the customer’s program they should be used as checkpoints at which it is decided to continue with the process or actually start from beginning of bqfs sequence. It is up to the customer to decide how many attempts to do before considering a device unprogrammable.

## 4. Creating a bqfs and dffs that Contains Specific Customer DFI

The bqfs file provided with this document will be based on the latest firmware revision for a given bq275xx device and its default data flash configuration. A customer may be in one of two situations in a production environment: a) requires programming a specific DFI and retain current version of firmware or b) needs to program firmware and also specific DFI. It would not be time effective to expect customers to use a bqfs that programs the default firmware and data flash and to then reprogram the data flash to a custom DFI.

There is a tool that allows converting a senc file into a version that will program custom DFI from the beginning. It is required having a DFI file to use this tool and an original senc file. A senc file is a file that actually is used to reprogram the firmware of a bq275xx device by using the evaluation software. These files are input to the bqfs Update Tool and after executing creates a bqfs and dffs file that will contain the desired firmware with the data flash configuration specific to a customer’s application (Figure5).

The user has the option to create a bqfs and dffs based on I2C (default) or HDQ. The Update Tool is called from a command screen (DOS) by running the FlashStream.exe file. The command structured for the tool is displayed when calling the FlashStrem.exe. The associated files used with the Update Tool must be within the same directory as the tool.

The application note "Going to Production with the bq2750x" (slua449a) instructs on how to create a DFI for the first time. The application note "Updating Firmware with the bq2750x and EVM" (slua453) provides instructions on how to update a DFI that was created with an older version of firmware and is now required to have a DFI that is compatible with the later version firmware without requiring going through the whole process described in slua449a.

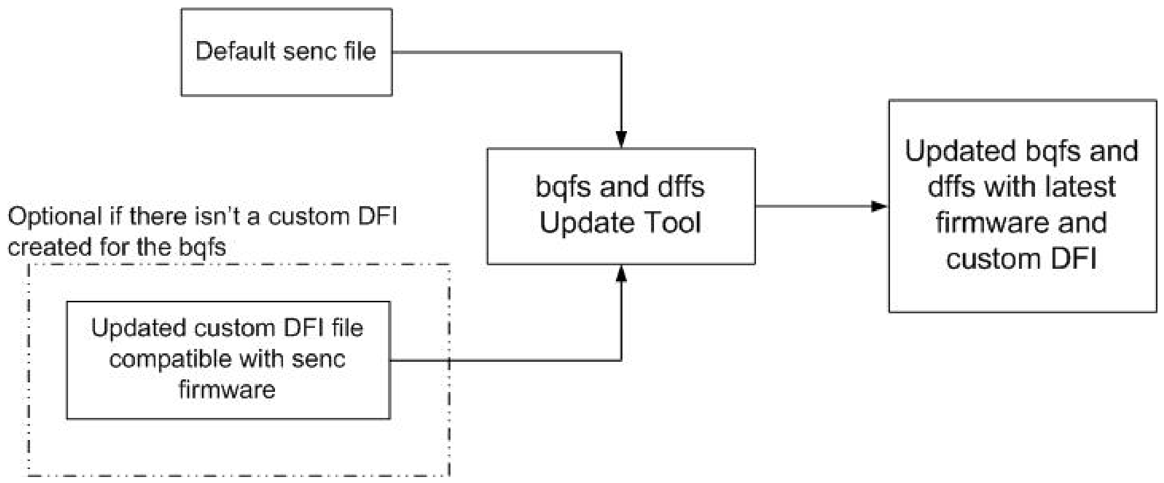

Default senc file
bqfs and dffs Update Tool
Updated bqfs and dffs with latest firmware and custom DFI
Optional if there isn't a custom DFI created for the bqfs
Updated custom DFI file compatible with senc firmware

Figure 5. bqfs and dffs Update Tool Process