Application Report
SLVA148A–October 2003–Revised September 2004

bq2083, bq2084 and bq2085 Calibration Procedure

Doug Williams *PMP Portable Power*

ABSTRACT

This application report describes the optional calibration of the offsets, voltage, temperature, current, and coulomb counter in the bq2083, bq2084, and bq2085 advanced gas gauge devices. Additional programming details and examples are provided in two appendixes.

1 Introduction

The bq2083, bq2084, and bq2085 advanced gas gauges have two separate analog to digital converters, which can be calibrated for gain and offset to improve measurement accuracy. One converter (generally referred to as *ADC*) is used for cell voltage, temperature, and high speed current measurements. The other (generally referred to as *Coulomb Counter)* is for normal current measurement and coulomb counting.

The gas gauge firmware has a routine, which automatically measures and compensates for the offset in each converter. However, this function is only performed at the beginning of the sleep mode, which is initiated when both the SMB data and clock lines are held low and no charge or discharge current is flowing. This is a thorough offset calibration, requiring approximately 20 seconds to complete.

For a calibration strategy that minimizes overall cycle time, the automatic offset routine has both advantages and disadvantages. The advantage is that the procedure is done without the use of any capital equipment, and totally unattended. The disadvantage is that since the module must first be assembled and programmed, it is undesirable to wait 20 seconds for an auto-offset calibration before proceeding with the gain calibrations.

The calibration strategy uses a quick (less than 2 seconds) tester-based offset calibration, adequate enough to insure accuracy in the subsequent gain calibrations. Then the device is free to complete its auto-offset calibration in sleep mode when the part is not using precious calibration resources.

2 Calibration Constants in Data Flash

CC Offset is the measured offset for the Coulomb Counter. This value is automatically measured each time the gas gauge enters the sleep mode. The typical value for the bq2083/5 is 1500 (units are 584 nV). Typical for the bq2084 is 15000 (units are 146 nV).

ADC Offset is the measured offset for the ADC. This value is automatically measured each time the gas gauge enters the sleep mode. The typical value for the bq2083/5 is 5(units are 584 nV). Typical for the bq2084 is 15 (units are 146 nV).

DSC Offset is the measured offset for the ADC, when it is used to measure current in the fast mode. This value is automatically measured each time the gas gauge enters the sleep mode. The typical value for the bq2083/5 is 5 (units are 584 nV). Typical for the bq2084 is 15(units are 146 nV).

AFE Ref is the full scale adjustment for the AFE and ADC combination. Typical value is 9750 (units are 0.1 mV)

Temperature Offset is simply the difference between the measured temperature and the actual temperature. It is, therefore, not strictly an offset, but rather a value that compensates for the thermistor and associated resistor divider errors. Typical value is 0 (units are 0.1°).

Sense Resistor Gain is the full scale adjustment for the current measurement. Typical value is 15312 for a 0.02-Ω resistor. The value can be expressed in calibrated ohms. Units are 0.0032653 Ω.

Coulomb Counter Delta is the charge that is counted for each bit of the coulomb counter. Typical for the bq2083/5 is 5.575e+005 (units are mAh / 2^{32}). Typical for the bq2084 is 1.414e+005 (units are mAh/2^{32})

Table 1. Calibration Constants

CONSTANT NAME	bq2083/85 ADDRESS	bq2084 ADDRESS	DATA TYPE	USED FOR
CC Offset	0xc1, 0xc2	0xd2, 0xd3	Signed integer	Current and Coulomb Counter offset
ADC Offset	0xc4	0xd5	Signed byte	Voltage measurement offset
DSC Offset	0xc3	0xd4	Signed byte	Fast current measurement offset
AFE Ref	0xb8, 0xb9	0xc9	Unsigned integer	Voltage and temperature measurement gain
Temperature Offset	0xc5	0xd6	Signed byte	Temperature offset
Sense Resistor Gain	0xbA, 0xbb	0xcb, 0xcc	Unsigned integer	Current measurement gain
Coulomb Counter Delta	0xbcC ~ 0xbfF	0xcd ~ 0xd0	Xemics floating point	Coulomb Counter gain

3 Initial Current/Voltage Offset Calibration

The offset of the Coulomb Counter converter can be determined by reading its value in calibration mode, with its input internally shorted. The following steps should be followed for initial offset calibration of both converters:

1. Put the part into calibration mode by sending the word 0x0653 to command 0x00. Note that once the part is in calibration mode, none of the standard SBS commands work.
2. Send calibration command 0x54 to place the part in internal calibration mode. This places an internal short across the current input.
3. This is an oversampling integrating converter with greater than 20-bit resolution where multiple samples are averaged. However, a new current conversion only occurs every 250 ms. To assure an accurate reading of a contiguous block of conversions; conversions must be read from the part in sequence after they become available.

In a tight loop, use calibration command 0x50 to read a counter that is incremented each time a new 250-ms sample is available. When an increment of this counter is detected, read the value of the latest conversion using calibration command 0x56. Sum four such successive readings, and then multiply that value by 16. Save this value as *CC Offset.*

4. Obtaining the voltage *ADC Offset* is slightly more difficult to perform, although it only needs to be read once. This step may not be necessary to meet your system voltage accuracy specifications. After detecting a new 250-ms sample with command 0x50, issue the SET_FLAG_OFFSET calibration command 0x63. This starts the process of measuring the ADC offset. After issuing this command, then is necessary to wait at least 250 milliseconds, and then use command 0x53 to read a block of voltage values. *The ADC Offset* is in the tenth data byte that is returned.
5. Use command 0x5e to force the part to reset. This causes it to exit calibration mode and return to normal. Be sure to wait at least 200 milliseconds after issuing the command to insure a complete reset.
6. Write the *CC Offset* unsigned integer to data flash (use address in table). Write the ADC Offset signed byte to data flash. In addition, the *ADC Offset* value should be written to DSC Offset signed byte in data flash. The DSC offset is the offset value used when the ADC is employed as a fast current measurement device for certain alarm functions. See Appendix B for flash memory write commands.

4 Voltage Gain Calibration

The following sequence provides the voltage gain calibration:

1. For maximum gain accuracy, insure that a valid *ADC Offset* value has been written to the part, either from the procedure above or from the automatic offset calibration routine.
2. Read the old full-scale value *AFE Ref* from its data flash location (recommended). Or, use the default value for first time calibration.
3. Apply a known set of cell voltages to the AFE. The absolute value is not critical, but an accurate actual value should be known and traceable to the local standards lab.
4. Read the SBS reported voltage, using Command 0x09. For maximum accuracy, the voltage should be read several times and averaged.
5. Calculate the new full scale value as:
   AFE Ref (new) = (Actual Voltage/Reported Voltage) x *AFE Ref* (old).
6. Write the *AFE Ref* unsigned integer to data flash.

5 Temperature Calibration

The temperature calibration is actually just an offset between the gas gauge reported temperature and the correct temperature. Follow the steps below to calibrate the temperature measurement function of the gas gauge.

1. Set the *Temperature Offset* to 0 by writing 0 to this location in data flash
2. Read the measured temperature from the gas gauge using the SBS command 0x08. The value is in units of 0.1°K.
3. Calculate the *Temperature Offset* as: $-10 \times$ [(MeasuredTemperature/10) – Actual Temperature in °K]
4. Write the resulting *Temperature Offset* signed byte value to data flash.

6 Current Gain Calibration

Current gain calibration requires the application of approximately 2 amps of discharge current through the device being calibrated. The absolute test current value is not critical, but an accurate actual value must be known and traceable to the local standards lab.

The *Coulomb Counter Delta* value is stored in a floating point format known as Xemics Floating Point. This is not the standard IEEE floating point that you may be familiar with. See Appendix A for Visual Basic code examples that convert this format to and from decimal. You will not be required to read this value; to verify correct flash writing, you can just compare the actual 4 bytes read with those previously written without worrying about the number that they represent.

Follow the steps below to calibrate the current gain:

1. To insure gain accuracy, valid *DSC Offset* and *CC Offset* values must have been written to the part, either from the offset procedure above or from the automatic offset calibration routine.
2. With the 2 amp discharge load applied, read the reported SBS current using command 0x0a. For maximum accuracy, read it at least three times and calculate the average.
3. Calculate the new sense resistor gain value as:
   Sense Resistor Gain(new) = (Actual Current/Reported Current) $\times$*Sense Resistor Gain*(old).
4. Write the Sense Resistor Gain (unsigned integer) to data flash.
5. Calculate the new Coulomb Counter Delta for the bq2083 and bq2085 as:
   Coulomb Counter Delta = *Sense Resistor Gain*/32767/3600 x 2^{32}. For the bq2084, the formula is:
   Coulomb Counter Delta = *Sense Resistor Gain* / 32767 / 3600 / 4 x 2^{32}.
6. Write *Coulomb Counter Delta* to data flash, using Xemics Floating Point format. Use the equivalent of the Xemics_Dec2Storage function in Appendix A. It returns four bytes. Byte 1 is written to the lowest data flash location shown in the Table, Byte 2 to the next location, etc.

Appendix A Xemics Floating Point

The following VB function can be used to convert a decimal value to the Xemics Floating Point format. This, or similar technique is required to store the *Coulomb Counter Delta* factor. The supporting HexByte function is also included. The user may prefer to create a function that just returns the four bytes in the arguments, rather than converting to an 8-character hex string as this one does.

The reverse function is also included here for informational purposes, but is not required for calibration since the old factor is not required.

A.1 Function Xemics_Dec2Storage(ByVal X As Double) As String

```
Dim iByte1 As Integer
Dim iByte2 As Integer
Dim iByte3 As Integer
Dim iByte4 As Integer
Dim iExp As Integer
Dim bNegative As Boolean
Dim fMantissa As Double


'// Don't blow up with logs of zero
If X = 0 Then X = 0.0000001


If X < 0 Then
bNegative = True
X = -X
End If


'// find the correct exponent
iExp = Int((LOG(X) / LOG(2)) + 1) '// remember - log of any base is ln(x)/ln(base)


'// MS byte is the exponent + 0x80
iByte1 = iExp + 128


'// Divide input by this exponent to get mantissa
fMantissa = X / 2 ^ iExp


'// Scale it up
fMantissa = fMantissa / 2 ^ -24


'// Split the mantissa into 3 bytes
iByte2 = fMantissa \ 2 ^ 16
iByte3 = (fMantissa - (iByte2 * 2 ^ 16)) \ 2 ^ 8
iByte4 = fMantissa - (iByte2 * 2 ^ 16) - (iByte3 * 2 ^ 8)


'// subtract the sign bit if number is positive
If False = bNegative Then iByte2 = iByte2 And &H7F


Xemics_Dec2Storage = HexByte(iByte1) & HexByte(iByte2) & HexByte(iByte3) & HexByte(iByte4)
End Function
```

A.2 Function HexByte(vDecimalByte) As String

```
Dim s As String


s = Hex$(vDecimalByte)
If Len(s) = 1 Then s = "0" & s
HexByte = s


End Function
```

A.3 Function Xemics_Storage2Dec(vMSByte, vMidHiByte, vMidLoByte, vLSByte) As Double

```
Dim bIsPositive As Boolean
Dim fExponent As Double
Dim fResult As Double


'// Get the sign, its in the 0x00 80 00 00 bit
If (vMidHiByte And 128) = 0 Then bIsPositive = True


'// Get the exponent, it's 2^(MSbyte - 0x80)
fExponent = 2 ^ (vMSByte - 128)


'// Or in 0x80 to the MidHiByte
vMidHiByte = vMidHiByte Or 128


'// get value out of midhi byte
fResult = (vMidHiByte) * 2 ^ 16


'// add in midlow byte
fResult = fResult + (vMidLoByte * 2 ^ 8)


'// add in LS byte
fResult = fResult + vLSByte


'// multiply by 2^-24 to get the actual fraction
fResult = fResult * 2 ^ -24


'// multiply fraction by the ‘exponent’ part
fResult = fResult * fExponent


'// Make negative if necessary
If False = bIsPositive Then fResult = -fResult


Xemics_Storage2Dec = fResult


End Function
```

Appendix B Writing and Reading Data Flash

There are two methods to write and read flash. The simplest method uses commands 0x50, 0x51 and 0x52 as described below. This covers the complete range of the data flash for the bq2083 and bq2085. However, the bq2084 contains many data flash locations outside of this range (above address 0xff). The preferred technique, for both types of devices is to read and write entire blocks. This is generally faster if there are several locations to read and/or write. This technique also allows full programming of the bq2084 data flash constants.

B.1 Writing to Data Flash With Command 0x50

The user may use Command 0x50 to write individual bytes up through 0xff only. The first parameter is the data byte, the second is the address.

For example, writing a 16-bit integer requires two writes:
lError = WriteSMBusWord(&H50, yDataMS, yAddress)
lError = WriteSMBusWord(&H50, yDataLS, yAddress + 1)

B.2 Reading from Data Flash With Commands 0x51 and 0x52

The user may use Command 0x51 to set the address (up to 0xff only) of the flash byte to read. Then use Command 0x52 to read the byte. When setting the address, the first byte is a dummy. For example, to read an integer:

```
‘set flash address for ms data byte
lError = WriteSMBusWord(&H51, yDummy, yAddress)


'read ms byte
lError = ReadSMBusWord(&H52, yDummy, yDataMS)


'set flash address for ls data byte
lError = WriteSMBusWord(&H51, yDummy, yAddress + 1)


'read ls byte
lError = ReadSMBusWord(&H52, yDummy, yDataLS)


lDataWord = (256 * CLng(yDataMS)) + yDataLS
```

B.3 Reading/Writing Data Flash With the SMB Block Protocol

The flash data can be read or written with the following read/write page commands which use the SMB Block protocol. In order to write, however, the entire block must first be read, then edited and written back.

B.4 For the bq2083 and bq2085

```
0x58 64 bytes Reads/Writes data flash locations 0x0000-0x003f
0x59 64 bytes Reads/Writes data flash locations 0x0040-0x007f
0x5a 64 bytes Reads/Writes data flash locations 0x0080-0x00bf
0x5b 42 bytes Reads/Writes data flash locations 0x00c0-0x00e9
```

Or use:

```
0x60 16 bytes Reads/Writes data flash locations 0x0000-0x000f
0x61 16 bytes Reads/Writes data flash locations 0x0010-0x001f
0x62 16 bytes Reads/Writes data flash locations 0x0020-0x002f
0x63 16 bytes Reads/Writes data flash locations 0x0030-0x003f
0x64 16 bytes Reads/Writes data flash locations 0x0040-0x004f
0x65 16 bytes Reads/Writes data flash locations 0x0050-0x005f
0x66 16 bytes Reads/Writes data flash locations 0x0060-0x006f
0x67 16 bytes Reads/Writes data flash locations 0x0070-0x007f
0x68 16 bytes Reads/Writes data flash locations 0x0080-0x008f
0x69 16 bytes Reads/Writes data flash locations 0x0090-0x009f
0x6a 16 bytes Reads/Writes data flash locations 0x00a0-0x00af
0x6b 16 bytes Reads/Writes data flash locations 0x00b0-0x00bf
0x6c 16 bytes Reads/Writes data flash locations 0x00c0-0x00cf
0x6d 16 bytes Reads/Writes data flash locations 0x00d0-0x00df
0x6e 10 bytes Reads/Writes data flash locations 0x00e0-0x00e9
```

B.5 For the bq2084

For the bq2084

```
0x58 64 bytes Reads/Writes data flash locations 0x0000-0x003f
0x59 64 bytes Reads/Writes data flash locations 0x0040-0x007f
0x5a 64 bytes Reads/Writes data flash locations 0x0080-0x00bf
0x5b 64 bytes Reads/Writes data flash locations 0x00c0-0x00ff
0x5c 52 bytes Reads/Writes data flash locations 0x0100-0x133
```

Or use:

```
0x60 16 bytes Reads/Writes data flash locations 0x0000-0x000f
0x61 16 bytes Reads/Writes data flash locations 0x0010-0x001f
0x62 16 bytes Reads/Writes data flash locations 0x0020-0x002f
0x63 16 bytes Reads/Writes data flash locations 0x0030-0x003f
0x64 16 bytes Reads/Writes data flash locations 0x0040-0x004f
0x65 16 bytes Reads/Writes data flash locations 0x0050-0x005f
0x66 16 bytes Reads/Writes data flash locations 0x0060-0x006f
0x67 16 bytes Reads/Writes data flash locations 0x0070-0x007f
0x68 16 bytes Reads/Writes data flash locations 0x0080-0x008f
0x69 16 bytes Reads/Writes data flash locations 0x0090-0x009f
0x6a 16 bytes Reads/Writes data flash locations 0x00a0-0x00af
0x6b 16 bytes Reads/Writes data flash locations 0x00b0-0x00bf
0x6c 16 bytes Reads/Writes data flash locations 0x00c0-0x00cf
0x6d 16 bytes Reads/Writes data flash locations 0x00d0-0x00df
0x6e 16 bytes Reads/Writes data flash locations 0x00e0-0x00ef
0x6f 16 bytes Reads/Writes data flash locations 0x00f0-0x00ff
0x70 16 bytes Reads/Writes data flash locations 0x0100-0x10f
0x71 16 bytes Reads/Writes data flash locations 0x0110-0x011f
0x72 16 bytes Reads/Writes data flash locations 0x0120-0x012f
0x73 4 bytes Reads/Writes data flash locations 0x0130-0x133
```

B.6 Some Example VB Code

In the function below, it is demonstrated how to write a word to flash using the 16 byte SMB block commands above. ReadSMBusByteArray( ) and WriteSMBusByte Array are low level communication functions that implement the SMBus block commands.

g_fWRITE_DELAY is a delay value of 0.12 seconds to insure that the block write has completed.

```
Function WriteFlashWord(iAddress As Integer, yDataMS As Byte, yDataLS As Byte) As Long
Dim lerror As Long
Dim yDtaRow(16) As Byte
Dim yNextDtaRow(16) As Byte
Dim bWrapAround As Boolean
Dim iLen As Integer
Dim iLenNext As Integer
iLen = 16
iLenNext = 16


'//Is this a wrap-around write?
If (iAddress Mod 16) = 15 Then bWrapAround = True


'//Read the data flash row
lerror = ReadSMBusByteArray(&h60 + (iAddress \ 16), yDtaRow(), iLen)


'//Read the following row if needed
If bWrapAround Then
lerror = ReadSMBusByteArray(&h60 + (iAddress \ 16) + 1, yNextDtaRow(), iLenNext)
End If


'//Modify elements in first data row
yDtaRow(iAddress Mod 16) = yDataMS


If (False = bWrapAround) Then
yDtaRow((iAddress Mod 16) + 1) = yDataLS
Else
yNextDtaRow(0) = yDataLS
End If


'//Write the row(s) back
lerror = WriteSMBusByteArray(&h60 + (iAddress \ 16), yDtaRow(), iLen)
If (True = bWrapAround) Then
'
// wait for block write to finish
DoDelay g_fWRITE_DELAY
lerror = WriteSMBusByteArray(&h60 + (iAddress \ 16) + 1, yNextDtaRow(), iLenNext)


Else
'//wait for block write to finish
DoDelay g_fWRITE_DELAY
End If


WriteFlashWord = lerror


End Function
```

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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