

Data Flash Programming and Calibrating the bq3060 Gas Gauge

PMP - Battery Monitoring Solutions

ABSTRACT

This application report presents a strategy for high-speed, economical calibration and data flash programming of the bq3060 advanced gas gauge. VB6 code examples are provided.

Contents

1	Introduction	2
2	Writing the Data Flash Image to Each Target Device	3
2.1	Preparing the Data Flash Image Pack	3
2.2	Reading and Saving the Data Flash Image	3
2.3	Writing the Data Flash Image to Each Target Device	4
3	Calibrating the bq3060	5
3.1	Characterization of Board Offset	5
3.2	Calibration of CC Offset, Temperature, Current, and Pack Voltage	6
3.3	Calibration of Voltage	7
4	Writing Pack-Specific Data Flash Locations	10

List of Figures

1	Software Board Offset Calibration Interface for Characterizing Board Offset.....	5
2	bq3060 Voltage Translation Circuit Diagram.....	8
3	Typical Voltage Error Band After Software Voltage Calibration; Data Measured at 3800 mV Each Cell and Across Temperature.....	8
4	Voltage Error for Post-Calibrated bq803x-Based Devices.....	9

List of Tables

1	Voltage Accuracy of the Factory-Calibrated Devices	8
---	--	---

1 Introduction

The bq3060 is the latest CEDV (Compensated End of Discharge Voltage) advanced gas gauge, built with new silicon hardware technology and an Impedance Track™-like architecture for both data flash access and calibration. The bq3060 targets to reduce unit production cost and capital equipment investment. The calibration method is quick and simple because most of the calibration routines, such as calibration of current, offset, and temperature, are built into the firmware of the target device. The voltage calibration is done in the factory, which provides a level of accuracy that meets the needs of most applications (See [Table 1](#)). However, if a higher level of voltage accuracy is desired, this document also provides a method of post-factory voltage calibration.

The methods in this document are presented as VB6 (Visual Basic 6) functions. These functions were copied directly from working code. In order to read from and write to the data flash, they use five types of SMBus read and write functions. These can be duplicated in any software environment that has SMBus communication capabilities. As used herein, each read/write function is designed for communication with a gas gauge, so the device address (0x16) is omitted for clarity.

1. WriteSMBusInteger() has two arguments – the SMBus command and a signed integer. Internally, this function separates the integer into two bytes for transmission by the SMBus write-word protocol.
2. WriteSMBusByteArray() has three arguments – the SMBus command, the array of bytes, and an integer specifying the length of the byte array. Internally, this function separates the byte array into separate bytes for transmission by the SMBus write-block protocol.
3. WriteSMBusCommand() has only one argument – the SMBus command.
4. ReadSMBusUnsignedInteger has two arguments – the SMBus command and the returned integer.
5. ReadSMBusByteArray() has three arguments– the SMBus command, the returned array of bytes, and the returned length of the byte array. It is internally implemented with the SMBus read-block protocol.

Also used in these functions is a simple delay routine called DoDelay. VB6 code for this procedure is provided at the end of the document.

Error handling is not implemented in this sample code, because requirements are unique and varied. Also, constants are hard-coded into the functions to improve clarity rather than documenting them in code elsewhere as would normally be good coding practice.

A good strategy for bq3060 production is an eight-step process flow:

1. Write the data flash image to each device. This image was created using bqEASY.
2. Calibrate the device.
3. Update any individual flash locations, such as serial number, lot code, and date.
4. Perform any desired protection tests.
5. Connect the cells.
6. Perform additional desired protection tests.
7. Send 0x0021 to Manufacturer Access 0x00 command, to enable Lifetime and Permanent Fail functions.
8. Seal the pack.

In this document, the first three steps are examined in detail.

2 Writing the Data Flash Image to Each Target Device

2.1 Preparing the Data Flash Image Pack

The bq3060 ICs are shipped preprogrammed with default parameter values. To create the data flash image that is used for every production pack, assemble a battery pack with the default firmware, and set the data flash constants for the application. This includes number of serial cells, design capacity, CEDV gauging parameters, to name a few. Alternatively, use bqEASY software to set the desired data flash constants. In addition, Board Offset needs to be characterized and to be used when creating bq3060 data flash image with bqEASY. Refer to Section 3.1 for more details of Board Offset.

2.2 Reading and Saving the Data Flash Image

Note that this step only needs to be done once for a given project.

```
Function SaveDataFlashImageToFile(sFileName As String) As Long
    Dim iNumberOfRows As Integer
    Dim lError As Long
    Dim yRowData(32) As Byte
    Dim yDataFlashImage(&H400) As Byte
    Dim iRow As Integer Dim iIndex As Integer
    Dim iLen As Integer
    Dim iFileNumber As Integer

    '// FOR CLARITY, WITHOUT USING CONSTANTS
    '// 0x400 is the data flash size.
    '0x400 \ 32 = 32 rows
    iNumberOfRows = &H400 \ 32
    '// PUT DEVICE INTO ROM MODE
    lError = WriteSMBusInteger(&H0, &HF00)
    DoDelay 0.01
    '// READ THE DATA FLASH, ROW BY ROW
    For iRow = 0 To iNumberOfRows -1
        '// Set the address for the row. &H9 (0x09) is the ROM mode command.
        '// 0x200 is the row number where data flash starts.
        '// Multiplication by 32 gives the actual physical address where each row starts
        lError = WriteSMBusInteger(&H9, (&H200 + iRow) * 32)
        '// Read the row. &HC (0x0c) is the ROM mode command.
        lError = ReadSMBusByteArray(&HC, yRowData, iLen)
    '//Copy this row into its place in a big byte array
    For iIndex = 0 To 32 -1
        yDataFlashImage((iRow * 32) + iIndex) = yRowData(iIndex)
    Next iIndex
    Next iRow
    '// WRITE DATA FLASH IMAGE TO FILE
    iFileNumber = FreeFile
    Open sFileName For Binary Access Write As #iFileNumber
    Put #iFileNumber, , yDataFlashImage
    Close #iFileNumber
    '// EXECUTE GAS GAUGE PROGRAM
    lError = WriteSMBusCommand(&H8)
End Function
```

2.3 Writing the Data Flash Image to Each Target Device

The following method is fast. It only takes about 2 seconds to write the entire data flash in this manner.

CAUTION

If power is interrupted during the process, the device may become unusable.

```

Function WriteDataFlashImageFromFile(sFileName As String) As Long Dim lError As Long
  Dim iFileNumber As Integer
  Dim iNumberOfRows As Integer
  Dim iRow As Integer
  Dim iIndex As Integer
  Dim yRowData(32) As Byte
  Dim yDataFlashImage(&H400) As Byte

  '// READ THE FLASH IMAGE FROM THE FILE INTO A GLOBAL BYTE ARRAY
  iFileNumber = FreeFile
  Open sFileName For Binary Access Read As #iFileNumber
  Get #iFileNumber, , yDataFlashImage
  Close #iFileNumber

  '// FOR CLARITY, WITHOUT USING CONSTANTS
  iNumberOfRows = &H400 \ 32 '32 Rows

  '// PUT DEVICE INTO ROM MODE
  lError = WriteSMBusInteger(&H0, &HF00)
  DoDelay 0.01

  '// ERASE DATA FLASH, ROWS ARE ERASED IN PAIRS
  For iRow = 0 To iNumberOfRows -1 Step 2
    lError = WriteSMBusInteger(&H11, iRow)
    DoDelay 0.04
  Next iRow

  '// WRITE EACH ROW
  For iRow = 0 To iNumberOfRows -1
    '// Set the row to program into the first element of the 33 byte array
    yRowData(0) = iRow

    '// Copy data from the full array to the row array
    For iIndex = 0 To 31
      yRowData(iIndex + 1) = yDataFlashImage((iRow * 32) + iIndex)
    Next iIndex

    '// Write the row. Length is 33 because first byte is row number
    lError = WriteSMBusByteArray(&H10, yRowData, 32 + 1)
    DoDelay 0.02
  Next iRow

  '// EXECUTE GAS GAUGE PROGRAM
  lError = WriteSMBusCommand(&H8)
End Function

```

3 Calibrating the bq3060

In this application report, calibration refers to an action before battery cells are attached. Power supplies should be used to simulate series cells and to supply power to the bq3060. For current calibration, apply the current source directly across the sense resistor.

Before calibration, Board Offset needs to be characterized and to be used when creating bq3060 data flash image with bqEASY. Calibration of bq3060 in a step-by-step manner should in general obey the following sequence:

- CC Offset
- Voltage (if desired)
- Temperature
- Current
- Pack Voltage (if desired)

In production, however, CC Offset, Temperature, Current, and Pack Voltage calibration can be combined in a single step as detailed in Section 3.2; and software Voltage calibration (described in Section 3.3), if desired, should be done subsequently.

3.1 Characterization of Board Offset

Board offset is a system-level offset, often caused by component mismatch and noise coupling. Like any other offset, board offset varies from system to system and has dependency on temperature. For the bq3060, there is no need for individual board offset calibration. Instead, board offset needs to be characterized in the product development phase. The number of boards and measurement samples should be of sufficient quantity to adequately represent the distribution of board offset. Data analysis yields the programming value of board offset.

The bq3060 evaluation software, bqEVSW, provides board offset measurement on the *Calibration* screen with the *Software Board Offset Calibration* button. Before board offset calibration, be sure that the device is powered from the cell inputs and no charger is connected. This ensures that absolutely no current flows through the sense resistor during offset measurement. The default setting of sampling time, located right above the software board offset calibration button, is 2 seconds. Use the default setting to characterize the board offset.



Figure 1. Software Board Offset Calibration Interface for Characterizing Board Offset

It is also recommended to take at least 10 boards for board-offset characterization and make at least 5 measurement samples on each board. Of course, more samples always yield better data quality. The average of all the numbers is calculated and programmed into the Board Offset data flash location. This should be done before using bqEASY to create the production data flash image.

3.2 Calibration of CC Offset, Temperature, Current, and Pack Voltage

It only takes about 5 seconds to accurately calibrate CC offset, temperature, and current. In the bq3060, most calibration routines have been incorporated into firmware algorithms, which can be initiated with SMBus commands. The hardware for calibration is also simple. One current source, one voltage source (if using resistor divider to simulate the cells), and one temperature sensor are all that is required. The accuracy of the sources is not important, only their **stability**. However, accurately calibrated reference measurement equipment should be used for determining the actual arguments to the function.

The elapsed time for calibration can be changed by modifying values in the data flash, but this is not recommended. Use the default values for the times in DF.Calibration.Config

In the CalibrateAll() function, command 0x51 is used to setup a current offset, current, and temperature calibration of the device. Pack voltage calibration is generally not performed because it is only used to detect the presence of a charger, and its accuracy is not required for standard applications. In this case, Pack Voltage refers to a separate measurement of the voltage at the pack terminal through the PACK pin, and is unrelated to the SBS.Voltage() readout. Note that a successful Pack Voltage calibration requires that the pack+ terminal be connected to a stable reference voltage

The definition of the bits in command 0x51 are:

Bit 0	Coulomb Counter Offset	Bit 8	Pack Gain
Bit 1	Reserved	Bit 9	Pack Voltage
Bit 2	ADC Offset	Bit 10	AFE Error
Bit 3	Temperature, Internal	Bit 11	Reserved
Bit 4	Temperature, External 1	Bit 12	Reserved
Bit 5	Temperature, External 2	Bit 13	Reserved
Bit 6	Current	Bit 14	Run ADC Task Continuously
Bit 7	Reserved	Bit 15	Run CC Task Continuously

Bits 14 and 15 should always be set. These cause the Coulomb Counter and ADC tasks to run continuously, just as they do in normal operation. This has been found to increase the accuracy of the calibration.

After command 0x51 is issued, the calibration sequence is started in the firmware of the gas gauge. The calibrations are run in sequence starting from the least significant bit. Then, command 0x52 is used to poll these bits, which change from high to low as the tasks are completed. However, bits 14 and 15 do not change; hence, the masking of them in the polling loop.

It can be seen from this code that a simple modification to command 0x51 would allow it to work as a single function calibration. For example, to only calibrate current, only bit 6 could be set.

```
Function CalibrateAll(iCurrent As Integer, iTemperature As Integer, iCells As Integer) As Long
    '// iCurrent is in milliamps (normally negative, such as -2000)
    '// iTemperature is in Kelvin/10 units, so the argument is: 10 * (Celsius + 273.15)

    Dim lError As Long
    Dim bDoingCal As Boolean
    Dim iValue As Long

    '// GO TO CALIB MODE
    lError = WriteSMBusInteger(&H0, &H40)
    '// WRITE THE NUMBER OF CELLS lError = WriteSMBusInteger(&H63, iCells)

    '// WRITE THE ACTUAL CURRENT & TEMPERATURE
    lError = WriteSMBusInteger(&H60, iCurrent)
    lError = WriteSMBusInteger(&H62, iTemperature)

    '// START CALIBRATION
    '// Useful cal 10 byte &HD5 -External temperature sensor 1
    '// &HF5 -External temperature sensor 1 and 2
    '// &HCD -Internal temperature sensor
    lError = WriteSMBusInteger(&H51, &HC0C5)

    '// POLL CALIBRATION STATUS -WAIT FOR LOWER 14 BITS TO ALL CLEAR
    bDoingCal = True
    While bDoingCal
        lError = ReadSMBusUnsignedInteger(&H52, iValue)
        bDoingCal = iValue And &H3FFF
        DoDelay 0.2 '// check every 200 millisecond
    Wend

    '// TRANSFER RESULTS TO DATAFLASH
    lError = WriteSMBusCommand(&H72)
    DoDelay 0.1
    '// Ensure write process is finished

    '// EXIT CALIB MODE
    lError = WriteSMBusCommand(&H73)
End Function
```

3.3 Calibration of Voltage

3.3.1 bq3060 Voltage Translation Overview

The bq3060 voltage sensing and translation circuit was designed with a new architecture. Four translation factors, or k factors, characterize the four high voltage translation resistor-dividers from the VC1~VC4 pins to VSS. **Note: The filter resistor value for bq3060 must be 1kΩ.** Unlike TI's earlier generations of devices, the bq3060 is calibrated in the factory, and the four k factors are stored in the device (hidden), which are used for generating voltage readings. The accuracy of the factory-calibrated devices is given in [Table 1](#).

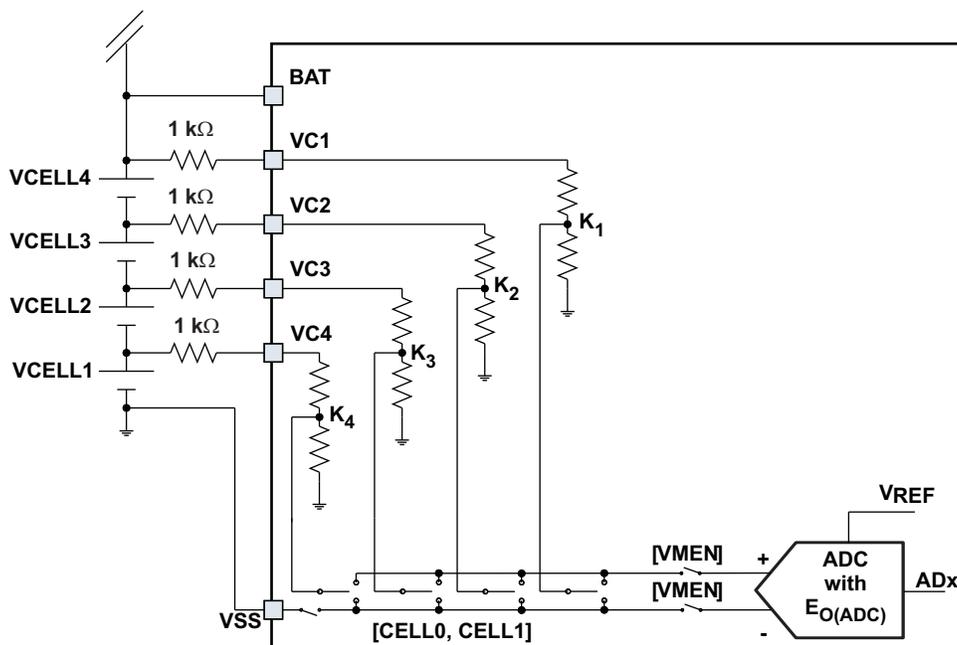


Figure 2. bq3060 Voltage Translation Circuit Diagram

Table 1. Voltage Accuracy of the Factory-Calibrated Devices

Temperature (°C)		CELL VOLTAGE MEASUREMENT ACCURACY (mV)		
		-10 ~ +60		±10
	-40 ~ +85		±10	±35

In the bq3060, the factory calibration should meet the need for most applications using a CEDV gauging algorithm. However, if a better voltage accuracy is desired, a software voltage calibration should be performed. Typical voltage accuracy performance after the software voltage calibration (calibrated at cell voltage = 4000mV and measured at cell voltage = 3800mV) is illustrated in Figure 3.

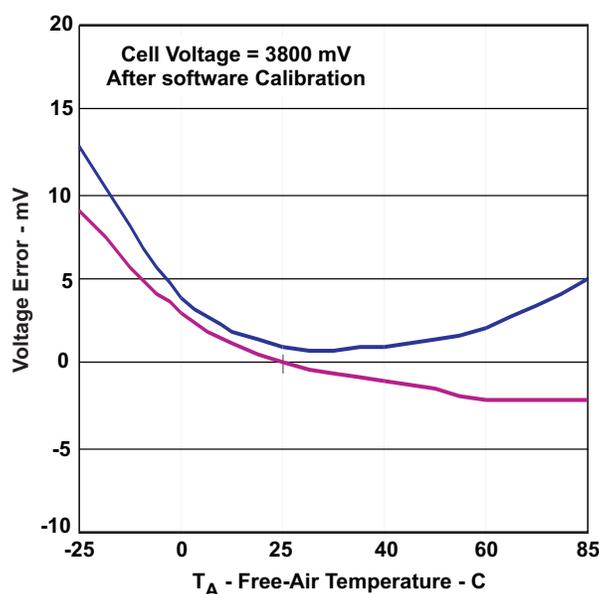


Figure 3. Typical Voltage Error Band After Software Voltage Calibration; Data Measured at 3800 mV Each Cell and Across Temperature

As a comparison, Figure 4 shows the voltage accuracy of bq803x(hardware platform)-based devices after calibration. The bq803x-based devices include the bq20z70, bq20z90, bq20z75, and bq20z95.

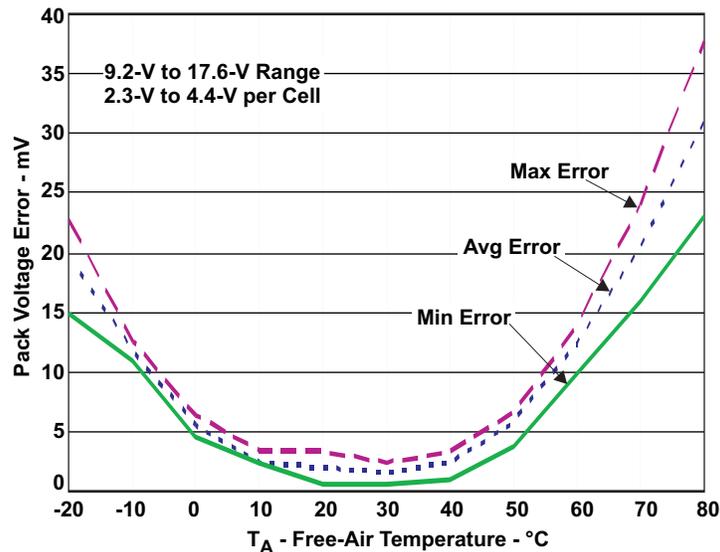


Figure 4. Voltage Error for Post-Calibrated bq803x-Based Devices

3.3.2 Hardware Requirement for Software Voltage Calibration

Software voltage calibration requires low impedance voltage sources to minimize loading to the internal voltage translation circuit. Ideally, four stable power supplies should be used to simulate the four cells.

Alternatively, a resistor divider and only one stable power supply can be used at the cost of a very small calibration error. If the resistance for each cell is less than 250 Ω, the additional voltage error caused by the loading effect is about 1mV per cell.

The accuracy of the voltage sources is not important, only their stability. However, accurately calibrated reference measurement equipment should be used for determining the actual arguments to the function. For periodic voltage measurement, a DVM with better than 1-mV accuracy is required.

3.3.3 VB6 Code for the Software Voltage Calibration

```
Function SoftwareVoltageCalibration(VoltageAtVCxPins() As Integer, iCells As Integer) As Long
'//Inputs:
'//VoltageAtVCxPins(1 To 4) are the voltages at VC1 to VC4 pins of bq3060 referencing VSS
'//iCells is the number of serial cells
'//Local Variables:
'//VCKFactor() are the old K-Factors
'//NewVCKFactor() are the new K-Factors
'//AD_Count() are the calculated AD conversion readings
'//CellVoltages() are the individual cell voltages read from SBS cmds

Dim lError As Long
Dim yLocalArray(32) As Byte
Dim VCKFactor(1 To 4) As Integer
Dim NewVCKFactor(1 To 4) As Integer
Dim AD_Count(1 To 4) As Double
Dim CellVoltages(1 To 4) As Integer

'//Get the old K-factors from the Data Flash
'//Set SubClassID to 104 (Calibration Data)
lError = WriteSMBusInteger(&H77, 104)
lError = ReadSMBusByteArray(&H78, yLocalArray, 32)

'//Set the K-Factor override flag to 0x9669 to enable the old K-Factors in Data Flash
yLocalArray(16) = &H96
'// MS byte yLocalArray(17) = &H69
'// LS byte
'//Write the K-Factor override flag to data flash
```

```

lError = WriteSMBusInteger(&H77, 104)
lError = WriteSMBusByteArray(&H78, yLocalArray, 32)
'//Wait at least 2.5 seconds for voltage to settle completely
DoDelay (2.5)

'//Save the old K-factors to VCKFactor(), call BytesToWord Function
BytesToWord yLocalArray(8), yLocalArray(9), VCKFactor(4) '//bottom cell, for VC4 pin
BytesToWord yLocalArray(10), yLocalArray(11), VCKFactor(3)
BytesToWord yLocalArray(12), yLocalArray(13), VCKFactor(2)
BytesToWord yLocalArray(14), yLocalArray(15), VCKFactor(1) '//top cell, for VC1 pin

'//Read individual cell voltages as translated by the old K-Factors
lError = ReadSMBusInteger(&H3F, CellVoltages(1)) '//bottom cell
lError = ReadSMBusInteger(&H3E, CellVoltages(2))
lError = ReadSMBusInteger(&H3D, CellVoltages(3))
lError = ReadSMBusInteger(&H3C, CellVoltages(4)) '//top cell

'//Calculate AD count, note that CellVoltages() needs to be converted to Double type
AD_Count(4) = CDb1(CellVoltages(1)) / CDb1(VCKFactor(4)) '//bottom cell
AD_Count(3) = CDb1(CellVoltages(1) + CellVoltages(2)) / CDb1(VCKFactor(3))
If iCells > 2 Then AD_Count(2) = CDb1(CellVoltages(1) + CellVoltages(2) + CellVoltages(3))
/ CDb1(VCKFactor(2))
If iCells > 3 Then AD_Count(1) = CDb1(CellVoltages(1) + CellVoltages(2) + CellVoltages(3)
+ CellVoltages(4)) / CDb1(VCKFactor(1))

'//Calculate the new K-Factors
NewVCKFactor(4) = VoltageAtVCxPins(4) / AD_Count(4) '//bottom cell
NewVCKFactor(3) = VoltageAtVCxPins(3) / AD_Count(3)
If iCells > 2 Then NewVCKFactor(2) = VoltageAtVCxPins(2) / AD_Count(2)
If iCells > 3 Then NewVCKFactor(1) = VoltageAtVCxPins(1) / AD_Count(1)

'//Write the new K-factors back to the data flash
WordToBytes NewVCKFactor(4), yLocalArray(8), yLocalArray(9) '//bottom cell
WordToBytes NewVCKFactor(3), yLocalArray(10), yLocalArray(11)
If iCells > 2 Then WordToBytes NewVCKFactor(2), yLocalArray(12), yLocalArray(13)
If iCells > 3 Then WordToBytes NewVCKFactor(1), yLocalArray(14), yLocalArray(15)

lError = WriteSMBusInteger(&H77, 104)
lError = WriteSMBusByteArray(&H78, yLocalArray, 32) '//Write page back to flash
DoDelay (0.5) '//Ensure flash write is finished

End Function
Public Sub BytesToWord(ByVal msb As Byte, ByVal lsb As Byte, ByRef nWord As Integer)
'//K-Factors are unsigned integer
nWord = msb * 256 + lsb
End Sub
Public Sub WordToBytes(ByVal nWord As Integer, ByRef msb As Byte, ByRef lsb As Byte)
'//K-Factors are unsigned integer
msb = (nWord And &HFF00) \ 256
lsb = nWord And &HFF
End Sub

```

4 Writing Pack-Specific Data Flash Locations

The third step is to fine tune the data flash a little for each pack, to give it a unique identity. In the following example, the pack Serial Number is written using subclass and offset information found in the gas gauge product Technical Reference manual. Modifications to single data flash locations normally require a block read of the 32-byte data flash page, then updating the desired element of the block, and writing it back to the device. This procedure is documented in the product Technical Reference manual.

```

Function WritePackSerialNumber(iSerialNumber As Integer) As Long
Dim lError As Long
Dim yData(32) As Byte
Dim iLen As Integer

'// SET THE SUBCLASS TO 48 (FOUND IN PRODUCT Technical Reference)
lError = WriteSMBusInteger(&H77, 48)

'// READ THE PAGE
lError = ReadSMBusByteArray(&H78, yData(), iLen)

'// REPLACE THE TWO BYTES AT OFFSET 14 (FOUND IN Technical Reference) WITH NEW S/N
yData(14) = (iSerialNumber And &HFF00) \ 256 '// modify MS byte yData(15) = iSerialNumber And
&HFF '// modify LS byte

'// WRITE THE PAGE BACK TO FLASH
lError = WriteSMBusByteArray(&H78, yData(), iLen)

'// FLASH WRITES ARE SLOW

```

```
DoDelay 0.1
End Function

Sub DoDelay(fWaitTime As Single)
Dim vTime As Variant

vTime = Timer
While Timer < (vTime + fWaitTime)
'// fix midnight problem
If Timer < vTime Then Exit Sub
'// Yield to various Windows events while the delay is in progress
DoEvents
Wend
End Sub
```

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 TI 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. Information of third parties may be subject to additional restrictions.

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.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

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
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2010, Texas Instruments Incorporated