TI Lithium Fuel Gauge Calibration

# Introduction

TI Lithium battery fuel gauges require lots of calibration. The process is described in their document [Going to Production with the bq275xx](http://www.ti.com/lit/an/slua449e/slua449e.pdf). Unfortunately, that document is written for an old version of TI calibration software and leaves some information out. This document was written to fill in the missing pieces. It is highly recommended that you read the TI document before continuing with this one.

# Setup

1. Download and install TI’s [Battery Management Studio](http://www.ti.com/tool/bqstudio) software. This will calibrate the fuel gauge using I2C.
2. Connect a battery to the fuel gauge. Must not be a dead battery, or the fuel gauge will not turn on and Battery Management Studio won’t be able to communicate with it. If you’re using one of TI’s fuel gauge evaluation modules, connect the battery-under-test to the module’s CELL+ and CELL- terminals.
3. Connect the fuel gauge (with attached battery) to a TI Evaluation Kit Interface. You can use TI’s [EV2400 Interface Board](http://www.ti.com/tool/EV2400), which provides a basic USB to I2C interface, or the [Gauge Development Kit (GDK)](http://www.ti.com/tool/bq27gdk000evm), which includes a programmable load and charger as well. Using the GDK will greatly simplify the current calibration and learning cycle process. The remainder of this guide assumes you’re using the GDK, while footnotes describe the steps for the EV2400.[[1]](#footnote-1)
4. Connect the GDK’s External EVM I2C jack to the fuel gauge.
5. Connect GDK’s EXT Load+ and EXT Load- terminals to fuel gauge’s PACK+ and PACK- terminals.
6. Apply 6V power to GDK’s 6V PLUG and PGND test points.
7. Connect the GDK’s USB port to PC. Always apply 6V power first.
8. Open Battery Management Studio.
9. The *Dashboard* plugin (left-most panel) should display fuel gauge status, battery mV, battery percent of charge, and GDK status[[2]](#footnote-2). The *Registers* plugin should display various metrics read from the fuel gauge, as well as the value of status register bits in the *Bit Registers* section.[[3]](#footnote-3)
   1. If not, check for dead battery or bad connections.

# Calibration

This process compensates for variations between each fuel gauge circuit. You’ll need to perform these steps for 20-30 production boards, and then average the results. The resulting coefficients (in *Data Memory* plugin 🡪 *Calibration*) must then be added to the “golden file” which is created later.

1. Open *Calibration* plugin and perform the following tests. Note that some tests may not be available, depending on your fuel gauge:
2. **Coulomb Counter Offset, Board Offset**
   1. Select *Calibrate CC Offset* and *Calibrate Board Offset*.
   2. Click *Calibrate Gas Gauge*. If successful, a green checkmark will be displayed next to the button.
3. **Voltage**
   1. Measure voltage across PACK+ and PACK- terminals of fuel gauge EVM, or use the Voltage reported by GDK in Battery Management Studio’s *Dashboard* plugin.
   2. Type the voltage in *Applied Voltage*.
   3. Check *Calibrate Voltage* and uncheck everything else.
   4. Click *Calibrate Gas Gauge*. If successful, a green checkmark will be displayed next to the button.
4. **Current**
   1. Apply a constant load to the battery pack. Ideally 1000 mA.[[4]](#footnote-4)
      1. Open GDK plugin.[[5]](#footnote-5)
      2. In *Discharge Cycle* panel, set Const Current to 1000 mA, and hit *Start*.
      3. In *Dashboard* tab, read current (should be something like -999 mA).
   2. Type the measured current in *Calibration* tab into *Applied Current*.
   3. Check *Calibrate Current* and uncheck everything else.
   4. Click *Calibrate Gas Gauge*. If successful, a green checkmark will be displayed next to the button.
5. **Temperature – Skip this; it’s not needed.**
6. **Battery Settings (must be set before running Learning Cycle or Rel-Dis-Rel cycle)**
   1. Open *Data Memory* plugin. Set the following settings per your battery’s parameters. This example is for Panasonic’s NCR1850B 3500 mAh battery.
      1. *Configuration 🡪 Data 🡪 Design Voltage* = 3600 mV
      2. *Configuration 🡪 Data 🡪 Design Capacity* = C =3350 mAh[[6]](#footnote-6)
      3. *Configuration 🡪 Data 🡪 Design Energy* = 12060 mWh
      4. *Configuration 🡪 Data 🡪 SOH Load I* = typical discharge current = -160 mA.
      5. *Configuration 🡪 Charge 🡪 Charge Voltage* = 4200 mV.
      6. *Configuration 🡪 Charge Termination 🡪 Taper Current* = C / 20 = 170 mA.

# Chemistry ID Selection

These actions are documented in TI’s [Simple Guide to Chemical ID Selection Tool (GPC)](http://www.ti.com/lit/an/slva725a/slva725a.pdf). The settings that follow are for Panasonic’s NCR1850B 3500 mAh battery. If you already know the Chemistry ID of your battery, you can skip Finding the Chemistry ID, and just Program the Chemistry ID.

## Finding the Chemistry ID (Rel-Dis-Rel Cycle)

1. Go to GDK plugin, and Auto Cycle tab. We want to charge fully, then relax for 2 hours, then discharge at C / 10, then relax for 5 hours. Once again, this example is for Panasonic’s NCR1850B 3500 mAh battery. Set the following settings:
   1. Cycle Setup 🡪 Begin each cycle with *Charge* task.
   2. Loop Setup 🡪 Repeat cycle exactly 1 time.
   3. Charge Setup 🡪 Charge Voltage = 4200 mV.
   4. Charge Setup 🡪 Charge Current = C/2 = 1625 mA.
   5. Charge Termination 🡪 Taper Current = C/20 = 167 mA.
   6. Relax After Charge 🡪 Relax Time = 120 min (2 hours).
   7. Discharge Setup 🡪 Const Current = C / 10 = 335 mA.
   8. Discharge Termination 🡪 Term Voltage = 3000 mV.
   9. Relax After Discharge 🡪 Relax Time = 300 min (5 hours).
2. In the *Registers* plugin, click *Start Log*. Save it somewhere you’ll remember.
3. In *GDK* plugin, click *Start*. (If it fails, you may have to click the *Connect Battery Cmd* in the Dashboard’s GDK section first.)
4. When the test finishes, go to *Registers* plugin and click *Stop Log*.
5. Find the log file created during the Rel-Dis-Rel cycle.
6. Change the file extension to *csv*.
7. Format and upload it according to the “Simple Guide” mentioned at the start of this section. In addition to the listed requirements, you should probably remove the charging portion of the log file, format the log file to have only Elapsed Time, Temperature, Voltage, and Average Current columns, and if the test was run too long, remove all but the first five hours of the last relax stage.
8. You should receive an email (to the account registered with TI) with a zip file containing a text file. The text file has several recommended Chem IDs, including one it chooses as the best.

## Program the Chemistry ID

1. In Battery Management Studio’s Chemistry plugin, select the battery that matches the recommended Chemistry ID.[[7]](#footnote-7)
2. Click *Update Chemistry from Database*.

# Learning Cycle

TI has two application notes, [SLUA597](https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/180/7418.Achieving-The-Successful-Learning-Cycle.pdf) and [SLUA903](http://www.ti.com/lit/an/slua903/slua903.pdf), both titled “Achieving The Successful Learning Cycle,” which are the definitive guides to learning cycles. This is just notes.

**Applicable RAM Control/Flag Bits:**

These status bits are found in the *Registers* plugin 🡪 *Bit Registers*.

* **VOK** – “Voltage OK for Omax Update.” Set when charging/discharging starts.
* **QEN** – “Qmax Update Enable.” Automatically set when learning cycle starts.
* **RUP\_DIS** – “Resistance Update Disable.” Set when gauge is unsure about battery state (for many reasons).
* **FC** – “Full Charge.” Set when the gauge detects the battery is fully charged.

## Applicable Data Flash:

* *Data Memory 🡪 Gas Gauging 🡪 State 🡪 Qmax Cell 0* – this is the maximum battery capacity [mAh]. It is updated as part of the learning cycle process.
* *Data Memory 🡪 Gas Gauging 🡪 State 🡪 Update Status* – the state of the learning cycle.
  + For system-side gauges it can be 0x00 (not calibrated), 0x01 (Qmax updated), or 0x02 (Resistance Table updated; learning cycle finished).
  + For pack-side gauges it can be 0x04 (not calibrated), 0x05 (Qmax updated), or 0x06 (Resistance Table updated; learning cycle finished).

## How a Learning Cycle Works

1. Start with a discharged, fully relaxed battery (3.0 – 3.3V). Update Status should be 0 or 4.
2. Enable IT\_ENABLE.
   1. **RUP\_DIS** bit should clear.
   2. **VOK** bit should set.
   3. **QEN** bit should set.
3. Charge battery fully, then allow battery to relax (4.1 – 4.2V) (relax time is at least two hours).
   1. **FC** bit should set.
   2. After relaxation, gauge takes voltage measurement and updates “Qmax Cell 0.”
   3. **VOK** bit should clear.
   4. Update Status should be set to 1 or 5.
4. Discharge battery fully at C/5 rate (can be between C/10 and C/3), then relax (3.0 – 3.3V). If you’re not using the C/5 rate, make sure to set the fuel gauge’s discharge current threshold to a value that will work with your setup (*Data Memory* plugin 🡪 *Configuration 🡪 Gas Gauging 🡪 Dsg Current Threshold*).
   1. VOK should set when discharge begins.
   2. Resistance table is updated every 11.1% state of charge, then more rapidly below 20%.
   3. VOK bit should clear.
   4. Qmax Cell 0 is updated.
   5. Update Status should be set to 2 or 6.

## Steps to Perform a Learning Cycle

The settings that follow are for Panasonic’s NCR1850B 3350 mAh battery, and are mostly the same as for the [Rel-Dis-Rel Cycle](#_Finding_Chemistry_ID).

1. Go to *Learning Cycle* plugin.
2. If using the GDK, select GDK Mode. Make sure the GDK is connected and works first.
3. Set configuration tab settings.
   1. Charge Voltage = 4200 mV.
   2. Charge Current = 1625 mA.
   3. Discharge Current = 670 mA = C / 5.
   4. Charge is terminated when Taper Current drops below 167 mA.
   5. Termination voltage = 3000 mV.
4. In the *Registers* plugin, click *Start Log*. Save it somewhere you’ll remember.
5. Go to *Control* tab. Click *Start*. May give an error about a bit not being in correct state at first. If that happens, try this:
   1. Go to *Registers* plugin. In *Bit Registers*, we need to set *RUP\_DIS* bit high.
   2. On *Commands* plugin, click UNSEAL.
   3. On *Commands* plugin, click IT\_ENABLE.
   4. On *Commands* plugin, click RESET.
6. When the test finishes, go to *Registers* plugin and click *Stop Log*.

# Create Golden Image

A “golden image” is a firmware image that must be programmed on production gauges. It contains calibration data and desired settings.

**Part 1: Set Required Settings**

Average the calibration data obtained during [Calibration](#_Calibration) of 20-30 production PCBs. Program them into the fuel gauge using the *Data Memory* plugin 🡪 *Calibration* 🡪 *Data*.

The following settings configure the SE pin to go low when the battery is below 3.0 V:

**Data Memory 🡪 Configuration 🡪 Registers 🡪 Pack Config = 0x195F**

|  |  |
| --- | --- |
| 0 | INTPOL = pin polarity (1 = active high) |
| 0 | INTSEL = pin select (0 = SE pin, 1 = HDQ pin) |
| 1 | HOST\_IE = interrupt enable (0 = disabled) |
| ? | SLEEP = Enable/Disable sleep mode (0 = disabled) |

**Data Memory 🡪 Configuration 🡪 Registers 🡪 Pack B = 0xD7**

**Data Memory 🡪 Configuration 🡪 Registers 🡪 Pack C = 0xB8**

|  |  |
| --- | --- |
| 0 | BTP\_EN = Battery Trip Point, an advanced feature (0 = disabled) |

**Data Memory 🡪 Configuration 🡪 Registers 🡪 Pack D = 0x64**

|  |  |
| --- | --- |
| 1 | SE\_POL = polarity of SE pin (1 = active high) |
| 1 | SE\_PU = pullup enable for SE pin (0 = disabled) |
| 0 | SE\_EN = enable SE pin (alternate?) function (0 = disabled) |
| 0 | IMAXRESRVEN = ? |
| 0 | IMAXEN = max current interrupt enable (0 = disabled) |

**Data Memory 🡪 Configuration 🡪 Discharge**

SOC1 Set Threshold = 65535 mAh

BL Set Volt Threshold = 3000 mV

BL Set Volt Time = 2 s

BL Clear Volt Threshold = 3100 mV

**Part 2: Create the Image**

1. Reset the CycleCount to 0 via the Data Memory plugin (Configuration 🡪 Data 🡪 Cycle Count).
2. Double-check all battery settings (described above in [**Battery Settings**](#_Battery_Settings) section). Don’t forget to set SOH Load I, if you haven’t already.
3. Change any additional settings you need the production gauges to have. For the air flow hood,
4. Go to *Registers* plugin. Under *Bit Registers*, check Control Status. FAS and SS bits must be low (green) or the Golden Image cannot be created.
   1. If they aren’t, send UNSEAL command to set SS bit low, then send UNSEAL\_FULL\_ACCESS command to set FAS bit low.
5. In Battery Management Studio, open *Golden Image* plugin.
6. Select an output directory.
7. Click *Create Image Files*.

# Production Programming

Production programming can be accomplished using the GDK, the EV2400, or a microcontroller running special firmware. As part of this effort, two open-sourced programs were developed and tested to program TI parts. The first is a Python script which converts the Golden *Image Files* created in the previous section to a data structure in an h-file. The second is a C++ program, compatible with Arduino and Energia, which combines with the h-file to create a microcontroller firmware which will program a TI fuel gauge. Both programs were developed with information found in TI’s guide [Updating the bq275xx Firmware at Production](http://www.ti.com/lit/an/slua541a/slua541a.pdf).

Both programs are released with MIT license. Instructions for use are in the README, and further details are included as comments in the code. The link follows: <https://github.com/DwyerInstruments/TI_FuelGaugeProgrammer>

# Bibliography

[Going to Production with the bq275xx](http://www.ti.com/lit/an/slua449e/slua449e.pdf)

[Quickstart Guide for Gauge Development Kit](http://www.ti.com/lit/ml/sluub24/sluub24.pdf)

[Simple Guide to Chemical ID Selection Tool (GPC)](http://www.ti.com/lit/an/slva725a/slva725a.pdf)

[Achieving The Successful Learning Cycle](https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/180/7418.Achieving-The-Successful-Learning-Cycle.pdf) (SLUA597)

[Achieving The Successful Learning Cycle (SLUA903)](http://www.ti.com/lit/an/slua903/slua903.pdf)

[Updating the BQ275xx Firmware at Production](http://www.ti.com/litv/pdf/slua541a)

# Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **By** | **Description** |
| 3/29/2016 | Adam Johnson | Finally remembered to add a revision block. |
| 3/29/2016 | Adam Johnson | Added settings that must be added to golden image.  Reformatted document for clarity. |
| 3/29/2016 | Adam Johnson | Added step to add average calibration data to golden image. |
| 9/7/2018 | Adam Johnson | Added Production Programming, Bibliography sections.  Updated links. |
| 11/19/2018 | Adam Johnson | Added note regarding *Design Energy Scale*. |
| 11/20/2018 | Adam Johnson | Removed steps no longer necessary in Battery Management Studio. |

1. If you’re using the EV2400, connect the EV2400’s I2C port to the fuel gauge I2C, and connect the EV2400’s USB port to the PC. No other connections or external power are necessary. [↑](#footnote-ref-1)
2. If you’re using the EV2400, the EV2400’s firmware version will be displayed instead. [↑](#footnote-ref-2)
3. red = high; green = low; grey = reserved/not used [↑](#footnote-ref-3)
4. If the current is less than 100 mA, Battery Management Studio will not allow calibration to be performed. [↑](#footnote-ref-4)
5. If you’re using the EV2400, apply a current load using your own circuitry, and then measure the applied load. [↑](#footnote-ref-5)
6. If Design Energy can’t be set, the number may be too large. Adjust Design Energy Scale and try again. [↑](#footnote-ref-6)
7. If the recommended Chemistry ID is not listed, update the list. Download chemistry update from <http://www.ti.com/tool/gasgaugechem-sw> (this link can also be found by an internet search for “TI chemistry updater”). The download is a zip file. Unzip it to find a second zip file. In Battery Management Studio, go to Help  Update Chemistry. There will be an error saying automatic update failed. That’s OK. Browse to the second zip file. After the update, you should see an expanded list of batteries. [↑](#footnote-ref-7)