# bq27411-G1

# Technical Reference

Literature Number: SLUUAS7B
January 2014–Revised June 2015

# Contents

# Preface

This document is a detailed Technical Reference Manual (TRM) for using and configuring the bq27411-G1 battery fuel gauge. This TRM document is intended to complement but not supersede information in the *bq27411-G1 Single Cell Li-Ion Battery Fuel Gauge for Battery Pack Integration Data Sheet* (SLUSBN7).

## Formatting Conventions Used in This Document:

| Information Type | Formatting Convention | Example |
|---|---|---|
| Commands | *Italics* with *parentheses* and no breaking spaces | *RemainingCapacity()* command |
| Data Memory | *Italics*, **bold**, and *breaking spaces* | ***Design Capacity*** data |
| Register bits and flags | Brackets and *italics* | *[SOC1]* bit |
| Data Memory bits | Brackets, *italics*, and **bold** | ***[TEMPS]*** bit |
| Modes and states | ALL CAPITALS | UNSEALED mode |

## Related Documentation from Texas Instruments

To obtain a copy of any of the following TI documents, go to the TI Web site at www.ti.com.

1. *bq27411-G1 Single Cell Li-Ion Battery Fuel Gauge for Battery Pack Integration* (SLUSBN7)
2. *Quick Start Guide for bq27411-G1* (SLUUAP6)
3. *bq27411 EVM: Single-Cell Technology User's Guide* SLUUAP3)
4. *SmartFlash OTP Programming Application Report* (SLUA703)

## Trademarks

Impedance Track is a trademark of Texas Instruments. All other trademarks are the property of their respective owners.

# General Description

The bq27411-G1 battery fuel gauge accurately predicts the battery capacity and other operational characteristics of a single, Li-based, rechargeable cell. It can be interrogated by a system processor to provide cell information, such as state-of-charge (SOC). The device is orderable in three predefined standard configurations:

- bq27411-G1A is predefined for $LiCoO_2$-based batteries for 4.2-V maximum charge voltage.
- bq27411-G1B is predefined for $LiCoO_2$-based batteries for 4.3-V maximum charge voltage.
- bq27411-G1C is predefined for $LiCoO_2$-based batteries for 4.35-V maximum charge voltage.

Unlike some other Impedance Track™ fuel gauges, the bq27411-G1 cannot be programmed with specific battery chemistry profiles. For many battery types and applications, the predefined standard chemistry profiles available in the bq27411-G1A, bq27411-G1B, or bq27411-G1C fuel gauge are sufficient matches from a gauging perspective.

Information is accessed through a series of commands, called *Standard Commands*. Further capabilities are provided by the additional *Extended Commands* set. Both sets of commands, indicated by the general format *Command()*, are used to read and write information contained within the control and status registers, as well as its data locations. Commands are sent from system to gauge using an $I^2C$ serial communications engine, and can be executed during application development, system manufacture, or end-equipment operation.

The key to the high-accuracy, fuel gauging prediction is Texas Instruments proprietary Impedance Track™ algorithm. This algorithm uses cell measurements, characteristics, and properties to create SOC predictions that can achieve high accuracy across a wide variety of operating conditions and over the lifetime of the battery.

The fuel gauge measures the charging and discharging of the battery by monitoring the voltage across a small-value, external sense resistor. Cell impedance is computed based on current, open-circuit voltage (OCV), and cell voltage under loading conditions.

The fuel gauge uses an integrated temperature sensor for estimating cell temperature. Alternatively, the system processor can provide temperature data for the fuel gauge.

To minimize power consumption, the fuel gauge has several power modes: INITIALIZATION, NORMAL, SLEEP, and HIBERNATE. The fuel gauge passes automatically between these modes, depending upon the occurrence of specific events, though a system processor can initiate some of these modes directly.

# Functional Description

## 2.1 Fuel Gauging

The bq27411-G1 battery fuel gauge measures the cell voltage, temperature, and current to determine battery SOC. The fuel gauge monitors the charging and discharging of the battery by sensing the voltage across a small-value, external shunt resistor (10 mΩ) between the SRP and SRN pins. By integrating the charge passing through the battery, the battery SOC is adjusted during the charging or discharging of the battery.

The total battery capacity is found by comparing states of charge before and after applying the load with the amount of charge passed. When an application load is applied, the impedance of the cell is measured by comparing the OCV obtained from a predefined function for present SOC with the measured voltage under load. Measurements of OCV and charge integration determine chemical SOC and chemical capacity (Qmax). The initial value for Qmax is defined by ***Design Capacity*** and should match the cell manufacturers' data sheet. The fuel gauge acquires and updates the battery-impedance profile during normal battery usage. The impedance profile along with SOC and the Qmax value are used to determine *FullChargeCapacity()* and *StateOfCharge()*, specifically for the present load and temperature. *FullChargeCapacity()* is reported as capacity available from a fully-charged battery under the present load and temperature until *Voltage()* reaches the ***Terminate Voltage***. *NominalAvailableCapacity()* and *FullAvailableCapacity()* are the uncompensated (no or light load) versions of *RemainingCapacity()* and *FullChargeCapacity()*, respectively.

The fuel gauge has two flags, *[SOC1]* and *[SOCF]*, accessed by the *Flags()* command that warn when the battery SOC has fallen to critical levels. When *StateOfCharge()* falls below the first capacity threshold, as specified in ***SOC1 Set Threshold***, the *[SOC1]* (state-of-charge initial) flag is set. The flag is cleared once *StateOfCharge()* rises above ***SOC1 Set Threshold***. All units are in mAh.

When *StateOfCharge()* falls below the second capacity threshold, ***SOCF Set Threshold***, the *[SOCF]* (state-of-charge final) flag is set, serving as a final discharge warning. If ***SOCF Set Threshold*** = –1, the flag is inoperative during discharge. Similarly, when *StateOfCharge()* rises above ***SOCF Clear Threshold*** and the *[SOCF]* flag has already been set, the *[SOCF]* flag is cleared. All units are in %.

## 2.2 Temperature Measurement

The fuel gauge measures temperature via its internal on-chip sensor. This internal temperature data will be used by the Impedance Track™ algorithm if the ***OpConfig [TEMPS]*** bit is cleared. Alternatively, if ***OpConfig [TEMPS]*** bit is set, the system processor can set the temperature for the fuel gauging algorithm by writing to the *Temperature()* register.

Regardless of which sensor is used for measurement, the system processor can request the current battery temperature being used by the algorithm by calling the *Temperature()* command. Internal temperature data is always available via the *InternalTemperature()* command.

## 2.3 Current Measurement

The fuel gauge measures current by sensing the voltage across a small-value, external shunt resistor (10 mΩ) between the SRP and SRN pins. Internally, voltage passes through a gain stage before conversion by the coulomb counter. The current measurement data is available through the *AverageCurrent()* command.

## 2.4 Operating Modes

The fuel gauge has different operating modes: POR, INITIALIZATION, NORMAL, CONFIG UPDATE, SLEEP, and HIBERNATE. Upon powering up from OFF or SHUTDOWN mode, a power-on reset (POR) occurs and the fuel gauge begins INITIALIZATION. In NORMAL mode, the fuel gauge is fully powered and can execute any allowable task. Configuration data in RAM can be updated by the host using the CONFIG UPDATE mode. In SLEEP mode, the fuel gauge turns off the high-frequency oscillator clock to enter a reduced-power state, periodically taking measurements and performing calculations. In HIBERNATE mode, the fuel gauge is in a very-low-power state, but can be woken up by communication.

### *2.4.1 SHUTDOWN Mode*

In SHUTDOWN mode, the LDO is disabled so internal power and all RAM-based volatile data are lost. The host can command the gauge to immediately enter SHUTDOWN mode by first unsealing the gauge and then enabling the mode with a *SHUTDOWN_ENABLE* subcommand (Section 4.1.12) followed by the *SHUTDOWN* subcommand (Section 4.1.13). To exit SHUTDOWN mode, the GPOUT pin must be raised from logic low to logic high for at least 200 µs.

### *2.4.2 POR and INITIALIZATION Modes*

Upon POR, the fuel gauge copies ROM-based configuration defaults to RAM and begins the INITIALIZATION mode where essential data is initialized. It will remain in INITIALIZATION mode as a halted-CPU state when an adapter or other power source is present to power the fuel gauge (and system), yet no battery has been detected. The occurrence of a POR or a *Control() RESET* subcommand will set the *Flags() [ITPOR]* status bit to indicate that RAM has returned to ROM default data. When battery insertion is detected, a series of initialization activities begin including an OCV measurement. In addition, *CONTROL_STATUS [QMAX_UP]* and *[RES_UP]* bits are cleared to allow unfiltered learning of Qmax and impedance. Completion of the INITIALIZATION mode is indicated by the *CONTROL_STATUS [INITCOMP]* bit.

Some commands, issued by a system processor, can be processed while the fuel gauge is halted in this mode. The gauge will wake up to process the command and then return to the halted state awaiting battery insertion. The current consumption of the INITIALIZATION mode is similar to that of the NORMAL mode.

### *2.4.3 CONFIG UPDATE Mode*

If the application requires different configuration data for the fuel gauge, the system processor can update RAM-based Data Memory parameters using the *Control() SET_CFGUPDATE* subcommand to enter the CONFIG UPDATE mode. Operation in this mode is indicated by the *Flags() [CFGUPMODE]* status bit. In this mode, fuel gauging is suspended while the host uses the extended data commands to modify the configuration data blocks. To resume fuel gauging, the host sends a *Control() SOFT_RESET*, *EXIT_CFGUPMODE*, or *EXIT_RESIM* subcommand to exit the CONFIG UPDATE mode which clears both *Flags() [ITPOR]* and *[CFGUPMODE]* bits. After a timeout of approximately 240 seconds (4 minutes), the gauge will automatically exit the CONFIG UPDATE mode if it has not received a *SOFT_RESET*, *EXIT_CFGUPMODE*, or *EXIT_RESIM* subcommand from the host.

### *2.4.4 NORMAL Mode*

The fuel gauge is in NORMAL mode when not in any other power mode. During this mode, *AverageCurrent()*, *Voltage()*, and *Temperature()* measurements are taken once per second, and the interface data set is updated. Decisions to change states are also made. This mode is exited by activating a different power mode.

Because the gauge consumes the most power in NORMAL mode, the Impedance Track™ algorithm minimizes the time the fuel gauge remains in this mode.

### 2.4.5 SLEEP Mode

SLEEP mode is entered automatically if the feature is enabled (***OpConfig [SLEEP]***) = 1) and *AverageCurrent()* is below the programmable level ***Sleep Current*** (default = 10 mA). When entry into SLEEP mode has been qualified, but before entering it, the fuel gauge performs an ADC autocalibration to minimize the offset.

During SLEEP mode, the fuel gauge remains in a very-low-power idle state and automatically wakes up briefly every 20 seconds to take data measurements.

After making the measurements on the 20-second interval, the fuel gauge will exit SLEEP mode when *AverageCurrent()* rises above ***Sleep Current*** (default = 10 mA). Alternatively, an early wake-up before the 20-second internal is possible if the instantaneous current detected by an internal hardware comparator is above an approximate threshold of ±30 mA.

### 2.4.6 HIBERNATE Mode

HIBERNATE mode could be used when the system equipment needs to enter a very-low-power state and minimal gauge power consumption is required. This mode is ideal when a system equipment is set to its own HIBERNATE, SHUTDOWN, or OFF mode.

Before the fuel gauge can enter HIBERNATE mode, the system must use the *SET_HIBERNATE* subcommand to set the *[HIBERNATE]* bit of the *CONTROL_STATUS* register. The fuel gauge waits to enter HIBERNATE mode until it has taken a valid OCV measurement and the magnitude of the average cell current has fallen below ***Hibernate Current***. The fuel gauge can also enter HIBERNATE mode if the cell voltage falls below the ***Hibernate Voltage***. The fuel gauge will remain in HIBERNATE mode until the system issues a direct I$^2$C command to the fuel gauge. I$^2$C communication that is not directed to the fuel gauge will only briefly wake it up and the fuel gauge immediately returns to HIBERNATE mode.

It is the responsibility of the system to wake the fuel gauge after it has gone into HIBERNATE mode and to prevent a charger from charging the battery before the *Flags() [OCVTAKEN]* bit is set which signals an initial OCV reading has been taken. For maximum initialization accuracy, any significant charge or discharge current should be postponed until the *CONTROL_STATUS [INITCOMP]* bit is set. This could take up to 10 seconds. After waking, the fuel gauge can proceed with the initialization of the battery information. During HIBERNATE mode, RAM-based data values are maintained, but gauging status is lost. Upon exit from HIBERNATE mode, the fuel gauge will immediately reacquire measurements and reinitialize all gauging predictions.

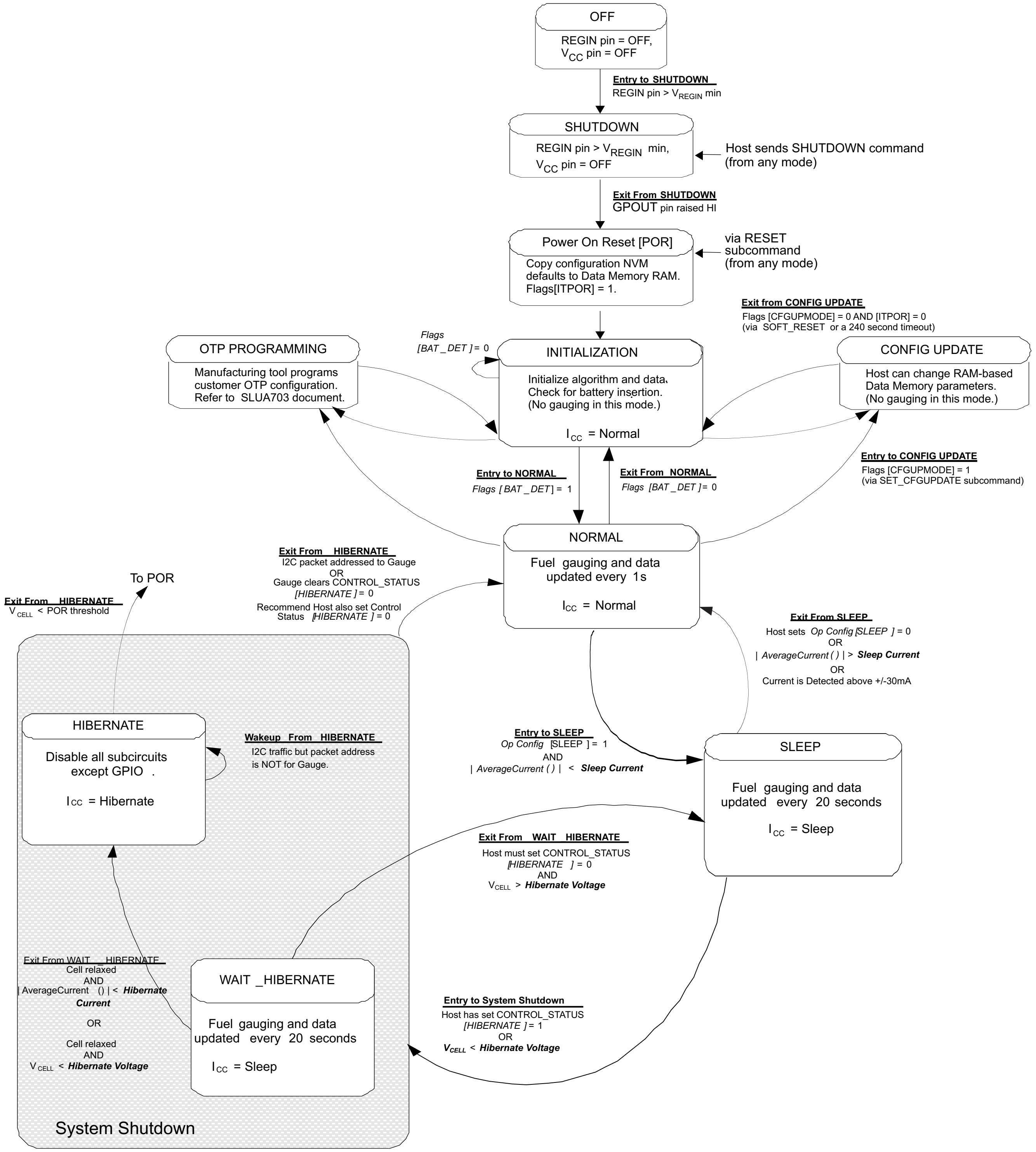

**Figure 2-1. Operating Modes Diagram**

### 2.4.7 One-time Programmable Programming Mode

The fuel gauge is shipped from the Texas Instruments factory with partially preprogrammed one-time programmable (OTP) nonvolatile memory (NVM) containing parameters related to the CHEM_ID for the specific -G1A, -G1B, or -G1C ordering option (see the bq27411-G1 data sheet (SLUSBN7)). The remaining OTP parameters are required to be programmed at the customer's factory for a fully-functional gauging solution.

## 2.5 Pin Descriptions

### 2.5.1 PROG Pin

This pin is used for OTP NVM programming during customer factory configuration. During the programming sequence, an external power supply is required to source the PROG pin at 7.4 V with 5 mA maximum. To avoid a floating input condition on the PROG pin, TI recommends terminating it with a 1.0-MΩ (typical) pulldown resistor to $V_{SS}$. More details are available in the SmartFlash OTP Programming (SLUA703) document.

### 2.5.2 TEST Pin

This digital factory TEST pin must be pulled low for proper operation in the customer's application. TI recommends terminating it with a 1.0-MΩ (typical) pulldown resistor to $V_{SS}$ for reduced power consumption.

### 2.5.3 GPOUT Pin

The GPOUT pin is a multiplexed pin and the polarity of the pin output can be selected via the ***OpConfig [GPIO_POL]*** bit. The function is defined by the ***OpConfig [BATLOWEN]*** bit. If the bit is set, the Battery Low Indicator (BAT_LOW) function is selected for the GPOUT pin. If it is cleared, the SOC interrupt (SOC_INT) function is selected for the GPOUT pin.

When the BAT_LOW function is activated, the signaling on the multiplexed pin follows the status of the *[SOC1]* bit in the *Flags()* register. The fuel gauge has two flags accessed by the *Flags()* function that warn when the battery SOC has fallen to critical levels. When *StateOfCharge()* falls below the first capacity threshold, as specified in ***SOC1 Set Threshold***, the *[SOC1]* flag is set. The flag is cleared once *StateOfCharge()* rises above ***SOC1 Set Threshold***. The GPOUT pin automatically reflects the status of the *[SOC1]* flag when ***[BATLOWEN]*** = 1.

When *StateOfCharge()* falls below the second capacity threshold, ***SOCF Set Threshold***, the *[SOCF]* flag is set, serving as a final discharge warning. Similarly, when *StateOfCharge()* rises above ***SOCF Clear Threshold*** and the *[SOCF]* flag has already been set, the *[SOCF]* flag is cleared.

When the SOC_INT function is activated, the GPOUT pin generates a 1-ms pulse width under various conditions as described in Table 2-1.

**Table 2-1. SOC_INT Function Definition**

| | Enable Condition | Pulse Width | Description |
|---|---|---|---|
| Change in SOC | (***SOCI Delta***) ≠ 0 | 1 ms | During charge, when the SOC is greater than (>) the points:<br>100% – n × (***SOCI Delta***) and 100%;<br>During discharge, when the SOC reaches (≤) the points:<br>100% – n × (***SOCI Delta***) and 0%;<br>where n is an integer starting from 0 to the number generating SOC no less than 0%.<br>Examples:<br>For ***SOCI Delta*** = 1% (default), the SOC_INT intervals are 0%, 1%, 2%, …, 99%, and 100%.<br>For ***SOCI Delta*** = 10%, the SOC_INT intervals are 0%, 10%, 20%, …, 90%, and 100%. |
| State Change | (***SOCI Delta***) ≠ 0 | 1 ms | Upon detection of entry to a charge or a discharge state. Relaxation is not included. |
| Battery Removal | ***OpConfig [BIE]*** bit is set. | 1 ms | When battery removal is detected by the BIN pin. |

### 2.5.4 Battery Detection (BIN)

The function of ***OpConfig [BIE]*** bit is described in Table 2-2. When battery insertion is detected and INITIALIZATION mode is completed, the fuel gauge transitions to NORMAL mode to start Impedance Track™ fuel gauging. When battery insertion is not detected, the fuel gauge remains in INITIALIZATION mode.

**Table 2-2. Battery Detection**

| *OpConfig [BIE]* | Battery Insertion Requirement | Battery Removal Requirement |
|---|---|---|
| 1 | (1) Host drives BIN pin from logic high to low to signal battery insertion.<br>or<br>(2) A weak pullup resistor can be used (between BIN and $V_{CC}$ pins). When a battery pack with a pulldown resistor is connected, it can generate a logic low to signal battery insertion. | (1) Host drives the BIN pin from logic low to high to signal battery removal.<br>or<br>(2) When a battery pack with a pulldown resistor is removed, the weak pullup resistor can generate a logic high to signal battery removal. |
| 0 | Host sends *BAT_INSERT* subcommand to signal battery insertion. | Host sends *BAT_REMOVE* subcommand to signal battery removal. |

Chapter 3

# Application Examples

## 3.1 Data Memory Parameter Update Example

The following example shows the command sequence needed to modify a Data Memory parameter. For this example, the default ***Design Capacity*** is updated from 1000 mAh to 1200 mAh. All device writes (wr) and reads (rd) are implied to the I$^2$C 8-bit addresses 0xAA and 0xAB, respectively.

| Step | Step Description | Pseudo Code |
|---|---|---|
| 1 | If the device has been previously SEALED, UNSEAL it by sending the appropriate keys to *Control()* (0x00 and 0x01). Write the first 2 bytes of the UNSEAL key using the *Control(0x8000)* command. Without writing any other bytes to the device, write the second (identical) 2 bytes of the UNSEAL key using the *Control(0x8000)* command.<br>**Note**: The remaining steps in this table will use this single-packet method when writing multiple bytes. | `//Two-byte incremental Method`<br>`  wr 0x00 0x00 0x80;`<br>`  wr 0x00 0x00 0x80;`<br>`//Alternative single byte method`<br>`  wr 0x00 0x00;`<br>`  wr 0x01 0x80;`<br>`  wr 0x00 0x00;`<br>`  wr 0x01 0x80;` |
| 2 | Send *SET_CFGUPDATE* subcommand, *Control(0x0013)*. | `wr 0x00 0x13 0x00;` |
| 3 | Confirm CFGUPDATE mode by polling *Flags()* register until bit 4 is set. May take up to 1 second. | `rd 0x06 Flags_register;` |
| 4 | Write 0x00 using *BlockDataControl()* command (0x61) to enable block data memory control. | `wr 0x61 0x00;` |
| 5 | Write 0x52 using the *DataBlockClass()* command (0x3E) to access the State subclass (82 decimal, 0x52 hex) containing the ***Design Capacity*** parameter. | `wr 0x3E 0x52;` |
| 6 | Write the block offset location using *DataBlock()* command (0x3F).<br>**Note**: To access data located at offset 0 to 31, use offset = 0x00. To access data located at offset 32 to 41, use offset = 0x01. | `wr 0x3F 0x00;` |
| 7 | Read the 1-byte checksum using the *BlockDataChecksum()* command (0x60). Expect 0xE8 for -G1B checksum. | `rd 0x60 OLD_Csum;` |
| 8 | Read both ***Design Capacity*** bytes starting at 0x4A (offset = 10). Block data starts at 0x40, so to read the data of a specific offset, use address 0x40 + mod(offset, 32). Expect 0x03 0xE8 for -G1B for a 1000-mAh default value.<br>**Note**: LSB byte is coincidentally the same value as the checksum. | `rd 0x4A OLD_DesCap_MSB;`<br>`rd 0x4B OLD_DesCap_LSB;` |
| 9 | Write both ***Design Capacity*** bytes starting at 0x4A (offset = 10). For this example, the new value is 1200 mAh. (0x04B0 hex) | `wr 0x4A 0x04;`<br>`wr 0x4B 0xB0;` |
| 10 | Compute the new block checksum. The checksum is (255 – x) where x is the 8-bit summation of the *BlockData()* (0x40 to 0x5F) on a byte-by-byte basis. A quick way to calculate the new checksum uses a data replacement method with the old and new data summation bytes. Refer to the code for the indicated method. | `temp = mod(255 - OLD_Csum`<br>`  - OLD_DesCap_MSB`<br>`  - OLD_DesCap_LSB, 256);`<br>`NEW_Csum = 255 - mod(temp +`<br>`  + 0x04 + 0xB0, 256);` |
| 11 | Write new checksum. The data is actually transferred to the Data Memory when the correct checksum for the whole block (0x40 to 0x5F) is written to *BlockDataChecksum()* (0x60). For this example New_Csum is 0x1F. | `wr 0x60 New_Csum;`<br>`//Example: wr 0x60 0x1F` |
| 12 | Exit CFGUPDATE mode by sending *SOFT_RESET* subcommand, *Control(0x0042)* | `wr 0x00 0x42 0x00;` |
| 13 | Confirm CFGUPDATE has been exited by polling *Flags()* register until bit 4 is cleared. May take up to 1 second. | `rd 0x06 Flags_register;` |
| 14 | If the device was previously SEALED, return to SEALED mode by sending the *Control(0x0020)* subcommand. | `wr 0x00 0x20 0x00;` |

# *Chapter 4*

# *Standard Commands*

The fuel gauge uses a series of 2-byte standard commands to enable system reading and writing of battery information. Each standard command has an associated command-code pair, as indicated in Table 4-1. Because each command consists of two bytes of data, two consecutive I$^2$C transmissions must be executed both to initiate the command function and to read or write the corresponding two bytes of data.

**Table 4-1. Standard Commands**

| NAME | | COMMAND CODE | UNIT | SEALED ACCESS |
|---|---|---|---|---|
| *Control()* | CNTL | 0x00 and 0x01 | NA | RW |
| *Temperature()* | TEMP | 0x02 and 0x03 | 0.1°K | RW |
| *Voltage()* | VOLT | 0x04 and 0x05 | mV | R |
| *Flags()* | FLAGS | 0x06 and 0x07 | NA | R |
| *NominalAvailableCapacity()* | | 0x08 and 0x09 | mAh | R |
| *FullAvailableCapacity()* | | 0x0A and 0x0B | mAh | R |
| *RemainingCapacity()* | RM | 0x0C and 0x0D | mAh | R |
| *FullChargeCapacity()* | FCC | 0x0E and 0x0F | mAh | R |
| *AverageCurrent()* | | 0x10 and 0x11 | mA | R |
| *StandbyCurrent()* | | 0x12 and 0x13 | mA | R |
| *MaxLoadCurrent()* | | 0x14 and 0x15 | mA | R |
| *AveragePower()* | | 0x18 and 0x19 | mW | R |
| *StateOfCharge()* | SOC | 0x1C and 0x1D | % | R |
| *InternalTemperature()* | | 0x1E and 0x1F | 0.1°K | R |
| *StateOfHealth()* | SOH | 0x20 and 0x21 | num / % | R |
| *RemainingCapacityUnfiltered()* | | 0x28 and 0x29 | mAh | R |
| *RemainingCapacityFiltered()* | | 0x2A and 0x2B | mAh | R |
| *FullChargeCapacityUnfiltered()* | | 0x2C and 0x2D | mAh | R |
| *FullChargeCapacityFlitered()* | | 0x2E and 0x2F | mAh | R |
| *StateOfChargeUnfiltered()* | | 0x30 and 0x31 | % | R |

## 4.1 Control(): 0x00 and 0x01

Issuing a *Control()* command requires a subsequent 2-byte subcommand. These additional bytes specify the particular control function desired. The *Control()* command allows the system to control specific features of the fuel gauge during normal operation and additional features when the device is in different access modes, as described in Table 4-2.

**Table 4-2.** *Control()* **Subcommands**

| CNTL Function | CNTL Data | SEALED Access | Description |
|---|---|---|---|
| CONTROL_STATUS | 0x0000 | Yes | Reports the status of device. |
| DEVICE_TYPE | 0x0001 | Yes | Reports the device type (0x0421). |
| FW_VERSION | 0x0002 | Yes | Reports the firmware version of the device. |
| DM_CODE | 0x0004 | Yes | Reports the Data Memory Code number stored in OTP memory. |
| PREV_MACWRITE | 0x0007 | Yes | Returns previous MAC command code. |
| CHEM_ID | 0x0008 | Yes | Reports the chemical identifier of the battery profile used by the fuel gauge. |
| BAT_INSERT | 0x000C | Yes | Forces the *Flags() [BAT_DET]* bit set when the ***OpConfig [BIE]*** bit is 0. |
| BAT_REMOVE | 0x000D | Yes | Forces the *Flags() [BAT_DET]* bit clear when the ***OpConfig [BIE]*** bit is 0. |
| SET_HIBERNATE | 0x0011 | Yes | Forces *CONTROL_STATUS [HIBERNATE]* to 1. |
| CLEAR_HIBERNATE | 0x0012 | Yes | Forces *CONTROL_STATUS [HIBERNATE]* to 0. |
| SET_CFGUPDATE | 0x0013 | No | Forces *CONTROL_STATUS [CFGUPMODE]* to 1 and gauge enters the CONFIG UPDATE mode. |
| SHUTDOWN_ENABLE | 0x001B | No | Enables device SHUTDOWN mode. |
| SHUTDOWN | 0x001C | No | Commands the device to enter SHUTDOWN mode. |
| SEALED | 0x0020 | No | Places the device in SEALED access mode. |
| TOGGLE_GPOUT | 0x0023 | Yes | Commands the device to toggle the GPOUT pin for 1 ms. |
| RESET | 0x0041 | No | Performs a full device reset. |
| SOFT_RESET | 0x0042 | No | Gauge exits the CONFIG UPDATE mode. |
| EXIT_CFGUPDATE | 0x0043 | No | Exits the CONFIG UPDATE mode without an OCV measurement and without resimulating to update *StateOfCharge().* |
| EXIT_RESIM | 0x0044 | No | Exits the CONFIG UPDATE mode without an OCV measurement and resimulates with the updated configuration data to update *StateOfCharge().* |

### 4.1.1 CONTROL_STATUS: 0x0000

Instructs the fuel gauge to return status information to control addresses 0x00 and 0x01. The read-only status word contains status bits that are set or cleared either automatically as conditions warrant or through using specified subcommands.

**Table 4-3. CONTROL_STATUS Bit Definitions**

| | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|
| **High Byte** | SHUTDOWNEN | WDRESET | SS | CALMODE | CCA | BCA | QMAX_UP | RES_UP |
| **Low Byte** | INITCOMP | HIBERNATE | RSVD | SLEEP | LDMD | RUP_DIS | VOK | RSVD |

**High Byte**

SHUTDOWNEN = Indicates the fuel gauge has received the *SHUTDOWN_ENABLE* subcommand and is enabled for SHUTDOWN. Active when set.

WDRESET = Indicates the fuel gauge has performed a Watchdog Reset. Active when set.

SS = Indicates the fuel gauge is in the SEALED state. Active when set.

CALMODE = Indicates the fuel gauge is in calibration mode. Active when set.

CCA = Indicates the Coulomb Counter Auto-Calibration routine is active. The CCA routine will take place approximately 3 minutes and 45 seconds after the initialization as well as periodically as conditions permit. Active when set.

BCA = Indicates the fuel gauge board calibration routine is active. Active when set.

QMAX_UP = Indicates Qmax has updated. True when set. This bit is cleared after a POR or when the *Flags() [BAT_DET]* bit is set. When this bit is cleared, it enables fast learning of battery Qmax.

RES_UP = Indicates that resistance has been updated. True when set. This bit is cleared after POR or when the *Flags() [BAT_DET]* bit is set. Also this bit can only be set after Qmax is updated (*[QMAX_UP]* is set). When this bit is cleared, it enables fast learning of battery impedance.

**Low Byte**

INITCOMP = This bit is cleared upon POR, or receiving a *RESET* or *SOFT_RESET* subcommand. This bit is set after INITIALIZATION mode has completed and new command register data values are valid 1 second after this bit is set.

HIBERNATE = Indicates a request for entry into HIBERNATE from SLEEP mode has been issued. True when set.

RSVD = Reserved.

SLEEP = Indicates the fuel gauge is in SLEEP mode. True when set.

LDMD = Indicates the algorithm is using constant-power model. True when set. Default is 1.

RUP_DIS = Indicates the Ra table updates are disabled. Updates are disabled when set.

VOK = Indicates cell voltages are ok for Qmax updates. True when set.

RSVD = Reserved.

### 4.1.2 DEVICE_TYPE: 0x0001

Instructs the fuel gauge to return the device type to addresses 0x00 and 0x01. The value returned is 0x0421.

### 4.1.3 FW_VERSION: 0x0002

Instructs the fuel gauge to return the firmware version to addresses 0x00 and 0x01.

### 4.1.4 DM_CODE: 0x0004

Instructs the fuel gauge to return the 8-bit ***DM Code*** as the least significant byte of the 16-bit return value at addresses 0x00 and 0x01. The *DM_CODE* subcommand provides a simple method to determine the configuration code stored in Data Memory. (See Section 6.5.1.5.3.)

### 4.1.5 PREV_MACWRITE: 0x0007

Instructs the fuel gauge to return the previous command written to addresses 0x00 and 0x01. The value returned is limited to less than 0x0015.

### 4.1.6 CHEM_ID: 0x0008

Instructs the fuel gauge to return the chemical identifier for the Impedance Track™ configuration to addresses 0x00 and 0x01. The expected value for bq27411-G1A is 0x0128 and for bq27411-G1B is 0x0312.

### 4.1.7 BAT_INSERT: 0X000C

This subcommand forces the *Flags() [BAT_DET]* bit to set when the battery insertion detection is disabled via ***OpConfig [BIE]*** = 0. In this case, the gauge does not detect battery insertion from the BIN pin logic state, but relies on the *BAT_INSERT* host subcommand to indicate battery presence in the system. This subcommand also starts Impedance Track™ gauging.

### 4.1.8 BAT_REMOVE: 0X000D

This subcommand forces the *Flags() [BAT_DET]* bit to clear when the battery insertion detection is disabled via ***OpConfig [BIE]*** = 0. In this case, the gauge does not detect battery removal from the BIN pin logic state, but relies on the *BAT_REMOVE* host subcommand to indicate battery removal from the system.

### 4.1.9 SET_HIBERNATE: 0x0011

Instructs the fuel gauge to force the *CONTROL_STATUS [HIBERNATE]* bit to 1. If the necessary conditions are met, this enables the gauge to enter the HIBERNATE power mode after the transition to SLEEP power state is detected. The *[HIBERNATE]* bit is automatically cleared upon exiting from HIBERNATE mode.

### 4.1.10 CLEAR_HIBERNATE: 0x0012

Instructs the fuel gauge to force the *CONTROL_STATUS [HIBERNATE]* bit to 0. This prevents the gauge from entering the HIBERNATE power mode after the transition to SLEEP power state is detected. It can also be used to force the gauge out of HIBERNATE mode.

### 4.1.11 SET_CFGUPDATE: 0x0013

Instructs the fuel gauge to set the *Flags() [CFGUPMODE]* bit to 1 and enter the CONFIG UPDATE mode. This command is only available when the fuel gauge is UNSEALED.

**NOTE:** A *SOFT_RESET* subcommand is typically used to exit the CONFIG UPDATE mode to resume normal gauging.

### 4.1.12 SHUTDOWN_ENABLE: 0x001B

Instructs the fuel gauge to enable SHUTDOWN mode and to set the *CONTROL_STATUS [SHUTDOWNEN]* status bit.

### 4.1.13 SHUTDOWN: 0x001C

Instructs the fuel gauge to immediately enter SHUTDOWN mode after receiving this subcommand. The SHUTDOWN mode is effectively a power-down mode with only a small circuit biased by the BAT pin which is used for wake-up detection. To enter SHUTDOWN mode, the *SHUTDOWN_ENABLE* subcommand must have been previously received. To exit SHUTDOWN mode, the GPOUT pin must be raised from logic low to logic high for at least 200 µs.

### 4.1.14 SEALED: 0x0020

Instructs the fuel gauge to transition from UNSEALED state to SEALED state and will set bit 7 (0x80) in the ***Update Status*** register to 1. The fuel gauge should always be set to the SEALED state for use in end equipment. The SEALED state blocks accidental writes of specific subcommands (see Table 4-2) and most Standard and Extended Commands (see Table 4-1 and Table 5-1).

### *4.1.15 PULSE_SOC_INT: 0x0023*

This subcommand can be useful for system level debug or test purposes. It instructs the fuel gauge to pulse the GPOUT pin for approximately 1 ms within 1 second of receiving the command.

**NOTE:** The GPOUT pin must be configured for the SOC_INT output function with ***OpConfig [BATLOWEN]*** cleared.

### *4.1.16 RESET : 0x0041*

This command instructs the fuel gauge to perform a full device reset and reinitialize RAM data to the default values from ROM and is therefore not typically used in field operation. The gauge sets the *Flags() [ITPOR]* bit and enters the INITIALIZE mode. Refer to Figure 2-1. This command is only available when the fuel gauge is UNSEALED.

### *4.1.17 SOFT_RESET : 0x0042*

This subcommand instructs the fuel gauge to perform a partial (soft) reset from any mode with an OCV measurement. The *Flags() [ITPOR]* and *[CFGUPMODE]* bits are cleared and a resimulation occurs to update both *StateOfCharge()* and *StateOfChargeUnfiltered()*. Refer to Figure 2-1. Upon exit from CONFIG UPDATE mode, the fuel gauge will check bit 7 (0x80) in the ***Update Status*** register. If bit 7 (0x80) in the Update Status register is set the fuel gauge will be placed into the SEALED state. This subcommand is only available when the fuel gauge is UNSEALED.

### *4.1.18 EXIT_CFGUPDATE : 0x0043*

This subcommand exits CONFIG UPDATE mode without an OCV measurement and without resimulating to update *StateOfChargeUnfiltered()* or *StateOfCharge()*. The *Flags() [ITPOR]* and *[CFGUPMODE]* bits are cleared. Upon exit from CONFIG UPDATE mode, the fuel gauge will check bit 7 (0x80) in the ***Update Status*** register. If bit 7 (0x80) in the ***Update Status*** register is set the fuel gauge will be placed into the SEALED state.

If a new OCV measurement or resimulation is desired, either the *SOFT_RESET* or *EXIT_RESIM* subcommand should be used to exit CONFIG UPDATE mode. If EXIT_CFGUPDATE subcommand has been used to exit CONFIG UPDATE mode, the *SOFT_RESET* or *EXIT_RESIM* subcommand will not provide a new OCV measurement and/or resimulation. To get the new OCV measurement and/or resimulation the fuel gauge must first be placed into CONFIG UPDATE mode (*SET_CFGUPDATE* subcommand) and then the *SOFT_RESET* or *EXIT_RESIM* subcommand should be used to exit the CONFIG UPDATE mode. This subcommand is only available when the fuel gauge is UNSEALED.

### *4.1.19 EXIT_RESIM : 0x0044*

This subcommand exits CONFIG UPDATE mode without an OCV measurement. The *Flags() [ITPOR]* and *[CFGUPMODE]* bits are cleared and a resimulation occurs to update *StateOfChargeUnfiltered()*. Upon exit from CONFIG UPDATE mode, the fuel gauge will check bit 7 (0x80) in the ***Update Status*** register. If bit 7 (0x80) in the ***Update Status*** register is set, the fuel gauge will be placed into the SEALED state. This subcommand is only available when the fuel gauge is UNSEALED.

## 4.2 Temperature(): 0x02 and 0x03

This read- and write-word function returns an unsigned integer value of the temperature in units of 0.1°K measured by the fuel gauge. If ***OpConfig [TEMPS]*** bit = 0 (default), a read command will return the internal temperature sensor value and write command will be ignored. If ***OpConfig [TEMPS]*** bit = 1, a write command sets the temperature to be used for gauging calculations while a read command returns to the temperature previously written.

## 4.3 Voltage(): 0x04 and 0x05

This read-only function returns an unsigned integer value of the measured cell-pack voltage in mV with a range of 0 to 6000 mV.

## 4.4 Flags(): 0x06 and 0x07

This read-word function returns the contents of the fuel gauging status register, depicting the current operating status.

**Table 4-4. Flags Bit Definitions**

| | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|
| **High Byte** | OT | UT | RSVD | RSVD | RSVD | RSVD | FC | CHG |
| **Low Byte** | OCVTAKEN | RSVD | ITPOR | CFGUPMODE | BAT_DET | SOC1 | SOCF | DSG |

**High Byte**

OT = Over-temperature condition is detected. *[OT]* is set when *Temperature()* ≥ ***Over Temp*** (default = 55°C). *[OT]* is cleared when *Temperature()* < ***Over Temp*** – ***Temp Hys***.

UT = Under-temperature condition is detected. *[UT]* is set when *Temperature()* ≤ ***Under Temp*** (default = 0°C). *[UT]* is cleared when *Temperature()* > ***Under Temp*** + ***Temp Hys***.

RSVD = Bits 5:2 are reserved.

FC = Full charge is detected. If the ***FC Set%*** is a positive threshold, *[FC]* is set when SOC ≥ ***FC Set %*** and is cleared when SOC ≤ ***FC Clear %*** (default = 98%). By default, ***FC Set%*** = –1, therefore *[FC]* is set when the fuel gauge has detected charge termination.

CHG = Fast charging allowed. If SOC changes from 98% to 99% during charging, the *[CHG]* bit is cleared. The *[CHG]* bit will become set again when SOC ≤ 95%.

**Low Byte**

OCVTAKEN = Cleared on entry to RELAX mode and set to 1 when OCV measurement is performed in RELAX mode.

RSVD = Reserved

ITPOR = Indicates a POR or RESET subcommand has occurred. If set, this bit generally indicates that the RAM configuration registers have been reset to default values and the host should reload the configuration parameters using the CONFIG UPDATE mode. This bit is cleared after the *SOFT_RESET* subcommand is received.

CFGUPMODE = Fuel gauge is in CONFIG UPDATE mode. True when set. Default is 0. Refer to Section 2.4.3 for details.

BAT_DET = Battery insertion detected. True when set. When ***OpConfig [BIE]*** is set, *[BAT_DET]* is set by detecting a logic high-to-low transition at the BIN pin. When ***OpConfig [BIE]*** is low, *[BAT_DET]* is set when host issues the *BAT_INSERT* subcommand and is cleared when host issues the *BAT_REMOVE* subcommand. Gauge predictions are not valid unless *[BAT_DET]* is set.

SOC1 = If set, *StateOfCharge()* ≤ ***SOC1 Set Threshold***. The *[SOC1]* bit will remain set until *StateOfCharge()* ≥ ***SOC1 Clear Threshold***.

SOCF = If set, *StateOfCharge()* ≤ ***SOCF Set Threshold***. The *[SOCF]* bit will remain set until *StateOfCharge()* ≥ ***SOCF Clear Threshold***.

DSG = Discharging detected. True when set.

## 4.5 NominalAvailableCapacity(): 0x08 and 0x09

This read-only command pair returns the uncompensated (less than C/20 load) battery capacity remaining. Units are mAh.

## 4.6 FullAvailableCapacity(): 0x0A and 0x0B

This read-only command pair returns the uncompensated (less than C/20 load) capacity of the battery when fully charged. *FullAvailableCapacity()* is updated at regular intervals. Units are mAh.

## 4.7 RemainingCapacity(): 0x0C and 0x0D

This read-only command pair returns the remaining battery capacity compensated for load and temperature. If ***OpConfigB [SMOOTHEN]*** = 1, this register is equal to *RemainingCapacityFiltered()*; otherwise, it is equal to *RemainingCapacityUnfiltered()*. Units are mAh.

## 4.8 FullChargeCapacity(): 0x0E and 0x0F

This read-only command pair returns the compensated capacity of the battery when fully charged. *FullChargeCapacity()* is updated at regular intervals and is compensated for load and temperature. If ***OpConfigB [SMOOTHEN]*** = 1, this register is equal to *FullChargeCapacityFiltered()*; otherwise, it is equal to *FullChargeCapacityUnfiltered()*. Units are mAh.

## 4.9 AverageCurrent(): 0x10 and 0x11

This read-only command pair returns a signed integer value that is the average current flow through the sense resistor. In NORMAL mode, it is updated once per second and is calculated by dividing the 1-second change in coulomb counter data by 1 second. Large current spikes of short duration will be averaged out in this measurement. Units are mA.

## 4.10 StandbyCurrent(): 0x12 and 0x13

This read-only function returns a signed integer value of the measured standby current through the sense resistor. The *StandbyCurrent()* is an adaptive measurement. Initially it reports the standby current programmed in ***Initial Standby*** and, after spending several seconds in standby, reports the measured standby current.

The register value is updated every second when the measured current is above the ***Deadband*** and is less than or equal to 2 × ***Initial Standby***. The first and last values that meet these criteria are not averaged in, because they may not be stable values. To approximate a 1-minute time constant, each new *StandbyCurrent()* value is computed by taking approximately 93% weight of the last measured standby current and approximately 7% of the currently measured average current.

## 4.11 MaxLoadCurrent(): 0x14 and 0x15

This read-only function returns a signed integer value, in units of mA, of the maximum load conditions. The *MaxLoadCurrent()* is an adaptive measurement which is initially reported as the maximum load current programmed in ***Initial MaxLoad*** current. If the measured current is greater than ***Initial MaxLoad***, then *MaxLoadCurrent()* updates to the new current. *MaxLoadCurrent()* is reduced to the average of the previous value and ***Initial MaxLoad*** whenever the battery is charged to full after a previous discharge to an SOC less than 50%. This prevents the reported value from maintaining an unusually high value.

## 4.12 AveragePower(): 0x18 and 0x19

This read-only function returns a signed integer value of the average power during the charging or discharging of the battery. It is negative during discharge and positive during charge. A value of 0 indicates that the battery is not being discharged. The value is reported in units of mW.

## 4.13 StateOfCharge(): 0x1C and 0x1D

This read-only function returns an unsigned integer value of the predicted remaining battery capacity expressed as a percentage of *FullChargeCapacity()*, with a range of 0 to 100%.

## 4.14 InternalTemperature(): 0x1E and 0x1F

This read-only function returns an unsigned integer value of the internal temperature sensor in units of 0.1°K as measured by the fuel gauge. If ***OpConfig [TEMPS]*** = 0, this command will return the same value as *Temperature()*.

## 4.15 StateOfHealth(): 0x20 and 0x21

0x20 SOH percentage: this read-only function returns an unsigned integer value, expressed as a percentage of the ratio of predicted FCC(25°C, ***SOH LoadI***) over the *DesignCapacity()*. The FCC (25°C, ***SOH LoadI***) is the calculated FCC at 25°C and the ***SOH LoadI*** which is programmed in factory (default = –400 mA). The range of the returned SOH percentage is 0x00 to 0x64, indicating 0 to 100% correspondingly.

0x21 SOH Status: this read-only function returns an unsigned integer value, indicating the status of the SOH percentage:

- 0x00: SOH not valid (initialization)
- 0x01: Instant SOH value ready
- 0x02: Initial SOH value ready
  - Calculation based on default Qmax
  - May not reflect SOH for currently inserted pack
- 0x03: SOH value ready
  - Calculation based on learned Qmax
  - Most accurate SOH for currently inserted pack following a Qmax update
- 0x04 through 0xFF: Reserved

## 4.16 RemainingCapacityUnfiltered(): 0x28 and 0x29

This read-only command pair returns the true battery capacity remaining. This value can jump as the gauge updates its predictions dynamically. Units are mAh.

## 4.17 RemainingCapacityFiltered(): 0x2A and 0x2B

This read-only command pair returns the filtered battery capacity remaining. This value is not allowed to jump unless *RemainingCapacityUnfiltered()* reaches empty or full before *RemainingCapacityFiltered()* does. Units are mAh.

## 4.18 FullChargeCapacityUnfiltered(): 0x2C and 0x2D

This read-only command pair returns the compensated capacity of the battery when fully charged. Units are mAh. *FullChargeCapacityUnfiltered()* is updated at regular intervals.

## 4.19 FullChargeCapacityFiltered(): 0x2E and 0x2F

This read-only command pair returns the filtered compensated capacity of the battery when fully charged. Units are mAh. *FullChargeCapacityFiltered()* is updated at regular intervals. It has no physical meaning and is manipulated to ensure the *StateOfCharge()* register is smoothed if ***OpConfigB [SMOOTHEN]*** = 1.

## 4.20 StateOfChargeUnfiltered(): 0x30 and 0x31

This read-only command pair returns the true state-of-charge. Units are %. *StateOfChargeUnfiltered()* is updated at regular intervals, and may jump as the gauge updates its predictions dynamically. *StateOfChargeUnfiltered()* is always calculated as *RemainingCapacityUnfiltered() / FullChargeCapacityUnfiltered()*, rounded up to the nearest whole percentage point.

# Extended Data Commands

Extended data commands offer additional functionality beyond the standard set of commands. They are used in the same manner; however unlike standard commands, extended commands are not limited to 2-byte words. The number of command bytes for a given extended command ranges in size from single to multiple bytes, as specified in Table 5-1.

**Table 5-1. Extended Commands**

| Name | Command Code | Unit | SEALED Access[1] [2] | UNSEALED Access[1] [2] |
|---|---|---|---|---|
| *OpConfig()* | 0x3A and 0x3B | NA | R | R |
| *DesignCapacity()* | 0x3C and 0x3D | mAh | R | R |
| *DataClass()* [2] | 0x3E | NA | NA | RW |
| *DataBlock()* [2] | 0x3F | NA | RW | RW |
| *BlockData()* | 0x40 through 0x5F | NA | R | RW |
| *BlockDataCheckSum()* | 0x60 | NA | RW | RW |
| *BlockDataControl()* | 0x61 | NA | NA | RW |
| Reserved | 0x62 through 0x7F | NA | R | R |

[1] SEALED and UNSEALED states are entered via commands to *Control()* 0x00 and 0x01

[2] In sealed mode, data cannot be accessed through commands 0x3E and 0x3F.

## 5.1 OpConfig(): 0x3A and 0x3B

SEALED and UNSEALED Access: This command returns the ***OpConfig*** Data Memory register setting which is most useful for system level debug to quickly determine device configuration.

## 5.2 DesignCapacity(): 0x3C and 0x3D

SEALED and UNSEALED Access: This command returns the ***Design Capacity*** Data Memory value and is most useful for system level debug to quickly determine device configuration.

## 5.3 DataClass(): 0x3E

UNSEALED Access: This command sets the data class to be accessed. The class to be accessed should be entered in hexadecimal.

SEALED Access: This command is not available in SEALED mode.

## 5.4 DataBlock(): 0x3F

UNSEALED Access: This command sets the data block to be accessed. When 0x00 is written to *BlockDataControl()*, *DataBlock()* holds the block number of the data to be read or written. Example: writing a 0x00 to *DataBlock()* specifies access to the first 32-byte block and a 0x01 specifies access to the second 32-byte block, and so on.

SEALED Access: This command is not available in SEALED mode.

## 5.5 BlockData(): 0x40 through 0x5F

UNSEALED Access: This data block is the remainder of the 32-byte data block when accessing general block data.

## 5.6 BlockDataChecksum(): 0x60

UNSEALED Access: This byte contains the checksum on the 32 bytes of block data read or written. The least-significant byte of the sum of the data bytes written must be complemented ([255 – *x*], for *x* the least-significant byte) before being written to 0x60. For a block write, the correct complemented checksum must be written before the *BlockData()* will be transferred to RAM.

SEALED Access: This command is not available in SEALED mode.

## 5.7 BlockDataControl(): 0x61

UNSEALED Access: This command is used to control the data access mode. Writing 0x00 to this command enables *BlockData()* to access to RAM.

SEALED Access: This command is not available in SEALED mode.

## 5.8 Reserved – 0x62 through 0x7F

# Data Memory

## 6.1 Data Memory Interface

### *6.1.1 Accessing the Data Memory*

The Data Memory contains initialization, default, cell status, calibration, configuration, and user information. Most Data Memory parameters reside in volatile RAM that are initialized by associated parameters from ROM. However, some Data Memory parameters are directly accessed from ROM and do not have an associated RAM copy. The Data Memory can be accessed in several different ways, depending in which mode the fuel gauge is operating and what data is being accessed.

Commonly accessed Data Memory locations, frequently read by a system, are conveniently accessed through specific instructions, already described in Chapter 5, *Extended Data Commands*. These commands are available when the fuel gauge is either in UNSEALED or SEALED mode.

Most Data Memory locations, however, are only accessible in the UNSEALED mode by use of the evaluation software or by Data Memory block transfers. These locations should be optimized and/or fixed during the development and manufacturing processes. They become part of a golden image file and then can be written to multiple battery packs. Once established, the values generally remain unchanged during end-equipment operation.

To access Data Memory locations individually, the block containing the desired Data Memory location or locations must be transferred to the command register locations, where they can be read to the system or changed directly. This is accomplished by sending the set-up command *BlockDataControl()* (0x61) with data 0x00. Up to 32 bytes of data can be read directly from the *BlockData()* (0x40 through 0x5F), externally altered, then rewritten to the *BlockData()* command space. Alternatively, specific locations can be read, altered, and rewritten if their corresponding offsets index into the *BlockData()* command space. Finally, the data residing in the command space is transferred to Data Memory, once the correct checksum for the whole block is written to *BlockDataChecksum()* (0x60).

Occasionally, a Data Memory class is larger than the 32-byte block size. In this case, the *DataBlock()* command designates in which 32-byte block the desired locations reside. The correct command address is then given by 0x40 + offset modulo 32. For an example of this type of Data Memory access, refer to Section 3.1.

Reading and writing subclass data are block operations up to 32 bytes in length. During a write, if the data length exceeds the maximum block size, then the data is ignored.

None of the data written to memory are bounded by the fuel gauge — the values are not rejected by the fuel gauge. Writing an incorrect value may result in hardware failure due to firmware program interpretation of the invalid data. The written data is persistent, so a POR does resolve the fault.

### *6.1.2 Access Modes*

The fuel gauge provides two access modes, UNSEALED and SEALED, that control the Data Memory access permissions. The default access mode of the fuel gauge is UNSEALED, so the system processor must send a SEALED subcommand after a gauge reset to utilize the data protection feature.

### 6.1.3 SEALING and UNSEALING Data Memory Access

The fuel gauge implements a key-access scheme to transition from SEALED to UNSEALED mode. Once SEALED via the associated subcommand, a unique set of two keys must be sent to the fuel gauge via the *Control()* command to return to UNSEALED mode. The keys must be sent consecutively, with no other data being written to the *Control()* register in between.

When in the SEALED mode, the *CONTROL_STATUS [SS]* bit is set; but when the ***Sealed to Unsealed*** keys are correctly received by the fuel gauge, the *[SS]* bit is cleared. The ***Sealed to Unsealed*** key has two identical words stored in ROM with a value of 0x8000 8000. Therefore, *Control()* should supply 0x8000 and 0x8000 (again) to unseal the part.

## 6.2 Data Types Summary

**Table 6-1. Data Type Decoder**

| Type | Min Value | Max Value |
|---|---|---|
| F4 | ±9.8603 × $10^{-39}$ | ±5.707267 × $10^{37}$ |
| H1 | 0x00 | 0xFF |
| H2 | 0x00 | 0xFFFF |
| H4 | 0x00 | 0xFFFF FFFF |
| I1 | –128 | 127 |
| I2 | –32768 | 32767 |
| I4 | −2,147,483,648 | 2147483647 |
| Sx | 1-byte string | X-byte string |
| U1 | 0 | 255 |
| U2 | 0 | 65535 |
| U4 | 0 | 4,294,967,295 |

## 6.3 OTP Default Versus Typical Data Memory Values

As shipped from the Texas Instruments factory, many bq27411-G1 parameters in OTP NVM are left in the unprogrammed state (zero) while some parameters directly associated with the CHEMID are preprogrammed. This partially programmed configuration facilitates customization for each end application. Upon device reset, the contents of OTP are copied to associated volatile RAM-based Data Memory blocks. For proper operation, all parameters in RAM-based Data Memory require initialization -- either by updating Data Memory parameters in a lab or evaluation situation or by programming the OTP for customer production. In addition to the Data Type and expected minimum and maximum values for each RAM-based Data Memory parameter, the following tables show:

- A Default value for the partially programmed OTP NVM as shipped from the Texas Instruments factory.
- A Typical value appropriate for most applications or an Application Dependent (App Dep) value to be determined by the customer.

## 6.4 Data Memory Summary Tables

### Table 6-2. Data Memory Summary - Configuration

| Subclass ID | Subclass | Offset | Name | Data Type | Min Value | Max Value | (-G1A, -G1B, -G1C) | | Unit (EVSW Unit)* |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Default Value | Typical Value | |
| 2 | Safety | 0 | Over Temp | I2 | –1200 | 1200 | 0 | 550 | 0.1°C |
| | | 2 | Under Temp | I2 | –1200 | 1200 | 0 | 0 | 0.1°C |
| | | 4 | Temp Hys | U1 | 0 | 255 | 0 | 50 | s |
| 36 | Charge Termination | 0 | Min Taper Capacity | I2 | 0 | 1000 | 0 | 25 | mAh |
| | | 2 | Current Taper Window | U1 | 0 | 60 | 0 | 40 | s |
| | | 3 | TCA Set % | I1 | –1 | 100 | 0 | 99 | % |
| | | 4 | TCA Clear % | I1 | –1 | 100 | 0 | 95 | % |
| | | 5 | FC Set % | I1 | –1 | 100 | 0 | –1 | % |
| | | 6 | FC Clear % | I1 | –1 | 100 | 0 | 98 | % |
| | | 7 | DODatEOC Delta T | I2 | 0 | 1000 | 0 | 50 | 0.1°C |
| 48 | Data | 2 | Initial Standby | I1 | –256 | 0 | 0 | -3 | mA |
| | | 3 | Initial MaxLoad | I2 | –32767 | 0 | 0 | -200 | mA |
| 49 | Discharge | 0 | SOC1 Set Threshold | U1 | 0 | 255 | 0 | 10 | % |
| | | 1 | SOC1 Clear Threshold | U1 | 0 | 255 | 0 | 15 | % |
| | | 2 | SOCF Set Threshold | U1 | 0 | 255 | 0 | 2 | % |
| | | 3 | SOCF Clear Threshold | U1 | 0 | 255 | 0 | 5 | % |
| 64 | Registers | 0 | OpConfig | H2 | 0x0000 | 0xFFFF | 0x0000 | 0x25F8 | |
| | | 2 | OpConfigB | H1 | 0x00 | 0xFF | 0x00 | 0x0F | |
| | | 3 | DM Code | U1 | 0 | 255 | A = 0x80<br>B = 0x90<br>B = 0xA0 | | |
| 68 | Power | 7 | Hibernate I | I2 | 0 | 700 | 0 | 3 | mA |
| | | 9 | Hibernate V | I2 | 2450 | 3000 | 0 | App Dep[1] | mV |
| | | 11 | FS Wait | U1 | 0 | 255 | 0 | 1 | s |

[1] Application Dependent setting based on size and type battery.

## Table 6-3. Data Memory Summary - Gas (Fuel) Gauging

| | | | | | | | (-G1A, -G1B, -G1C) | | |
|---|---|---|---|---|---|---|---|---|---|
| **Subclass ID** | **Subclass** | **Offset** | **Name** | **Data Type** | **Min Value** | **Max Value** | **Default Value** | **Typical Value** | **Unit (EVSW Unit)*** |
| 80 | IT Cfg | 4 | Sec. Relax Time | U2 | 0 | 65535 | 0 | 30 | s |
| | | 10 | Average Time | U1 | 0 | 255 | 0 | 8 | Counts |
| | | 17 | Min DOD Res Update | U1 | 0 | 255 | 0 | 70 | % |
| | | 19 | Max Res Factor | U1 | 0 | 255 | 0 | 20 | — |
| | | 20 | Min Res Factor | U1 | 0 | 255 | 0 | 5 | — |
| | | 22 | Ra Filter | U2 | 0 | 1000 | 0 | 800 | — |
| | | 28 | Max V Delta | U1 | 0 | 255 | 0 | 10 | — |
| | | 35 | Fast Qmax Start DOD % | U1 | 0 | 255 | 0 | 92 | % |
| | | 36 | Fast Qmax End DOD % | U1 | 0 | 255 | 0 | 96 | % |
| | | 37 | Fast Qmax Start Volt Delta | I2 | 0 | 4200 | 0 | 125 | mV |
| | | 39 | Fast Qmax Current Threshold | I2 | 0 | 1000 | 0 | 4 | .1 Hr rate |
| | | 41 | Fast Qmax Min Points | U1 | 0 | 255 | 0 | 3 | |
| | | 42 | Fast Qmax Volt Buffer | U1 | 0 | 255 | 0 | 25 | mV |
| | | 44 | Qmax Capacity Err | U1 | 0 | 100 | 0 | 15 | 0.1% |
| | | 45 | Max Qmax Change | U1 | 0 | 255 | 0 | 20 | % |
| | | 48 | ResRelax Time | U2 | 0 | 65534 | 0 | 500 | s |
| | | 52 | User Rate-mW | I2 | –14000 | –3000 | 0 | 0 | cW |
| | | 54 | Reserve Cap-mWh | I2 | 0 | 14000 | 0 | 0 | 10 mW |
| | | 59 | Max Sim Rate | U1 | 0 | 255 | 0 | 1 | Hr rate |
| | | 60 | Min Sim Rate | U1 | 0 | 255 | 0 | 20 | Hr rate |
| | | 61 | Ra Max Delta | U2 | 0 | 100 | 0 | 11 | mΩ |
| | | 70 | Min Delta Voltage | I2 | –32000 | 32000 | 0 | 0 | |
| | | 72 | Max Delta Voltage | I2 | –32000 | 32000 | 0 | 200 | |
| | | 74 | DeltaV Max dV | I2 | 0 | 32000 | 0 | 100 | mV |
| | | 76 | TermV Valid t | U1 | 0 | 255 | 0 | 2 | sec |

### Table 6-3. Data Memory Summary - Gas (Fuel) Gauging (continued)

| Subclass ID | Subclass | Offset | Name | Data Type | Min Value | Max Value | (-G1A, -G1B, -G1C) | | Unit (EVSW Unit)* |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Default Value | Typical Value | |
| 81 | Current Thresholds | 0 | Dsg Current Threshold | I2 | 0 | 2000 | 0 | 167 | mA |
| | | 2 | Chg Current Threshold | I2 | 0 | 2000 | 0 | 100 | mA |
| | | 4 | Quit Current | I2 | 0 | 1000 | 0 | 250 | mA |
| | | 6 | Dsg Relax Time | U2 | 0 | 8191 | 0 | 60 | s |
| | | 8 | Chg Relax Time | U1 | 0 | 255 | 0 | 60 | s |
| | | 9 | Quit Relax Time | U1 | 0 | 63 | 0 | 1 | s |
| | | 10 | Transient Factor Charge | U1 | 0 | 255 | 0 | 179 | |
| | | 11 | Transient Factor Discharge | U1 | 0 | 255 | 0 | 179 | |
| | | 12 | Max IR Correct | U2 | 0 | 1000 | 0 | 400 | mV |
| 82 | State | 0 | Qmax Cell 0 | I2 | 0 | 32767 | 0 | 16384 | mAh |
| | | 2 | Update Status | H1 | 0x00 | 0x36 | 0x00 | 0x00 | |
| | | 3 | Reserve Cap-mAh | I2 | 0 | 9000 | 0 | App Dep[1] | mA |
| | | 5 | Load Select/Mode | H1 | 0x00 | 0xFF | 0 | 0x81 | |
| | | 6 | Q Invalid Max V | I2 | 0 | 32767 | A = 3803<br>B = 3814<br>C = 3808 | | mV |
| | | 8 | Q Invalid Min V | I2 | 0 | 32767 | A = 3752<br>B = 3748<br>C = 3752 | | mV |
| | | 10 | Design Capacity | I2 | 0 | 32767 | 0 | App Dep[1] | mA |
| | | 12 | Design Energy | I2 | 0 | 32767 | 0 | App Dep[1] | mWh |
| | | 14 | Default Design Cap | I2 | 0 | 32767 | A = 1340<br>B = 5580<br>C = 7361 | | mWh |
| | | 16 | Terminate Voltage | I2 | 2800 | 3700 | 0 | App Dep[1] | mV |
| | | 26 | SOCI Delta | U1 | 0 | 100 | 0 | 1 | % |
| | | 27 | Taper Rate | I2 | 0 | 2000 | 0 | 100 | .1 Hr rate |
| | | 29 | Taper Voltage | I2 | 0 | 5000 | A = 4100<br>B = 4200<br>C = 4250 | | mV |
| | | 31 | Sleep Current | I2 | 0 | 100 | 0 | 10 | mA |
| | | 33 | V at Chg Term | I2 | 0 | 5000 | A = 4190<br>B = 4290<br>C = 4340 | | mV |
| | | 35 | Avg I Last Run | I2 | –32768 | 32767 | 0 | -50 | mA |
| | | 37 | Avg P Last Run | I2 | –32768 | 32767 | 0 | -50 | mW |
| | | 39 | Delta Voltage | I2 | –32768 | 32767 | 0 | 1 | mV |

[1] Application Dependent setting based on size and type battery.

### Table 6-4. Data Memory Summary - Resistance Tables

| Subclass ID | Subclass | Offset | Name | Data Type | Min Value | Max Value | Default Value | Typical Value | Unit (EVSW Unit)* |
|---|---|---|---|---|---|---|---|---|---|
| 88 | R_a NVM | 0 - 28 | Cell0 R_a 0 - 14 | I2 | | | 0 | App Dep[1] | $2^{-10}\Omega$ |
| 89 | R_a RAM | 0 - 28 | Cell0 R_a 0 - 14 | I2 | | | [R_a RVM] | Dynamic[2] | $2^{-10}\Omega$ |

[1] Application Dependent setting based on size and type battery.

[2] R_a RAM table initialized by R_a NVM and dynamic updated during gauging.

### Table 6-5. Data Memory Summary - Calibration and Security

| Subclass ID | Subclass | Offset | Name | Data Type | Min Value | Max Value | Default Value | Typical Value | Unit (EVSW Unit)* |
|---|---|---|---|---|---|---|---|---|---|
| 104 | Data | 0 | Board Offset | I1 | –128 | 127 | 0 | 0 | |
| | | 1 | Int Temp Offset | I1 | –128 | 127 | 0 | 0 | |
| | | 2 | Pack V Offset | I1 | –128 | 127 | 0 | 0 | |
| 105 | CC Cal | 0 | CC Offset | I2 | –32768 | 32767 | 0 | | |
| | | 2 | CC Cal Temp | I2 | 0 | 32767 | 303.0 | | 0.1K |
| | | 4 | CC Gain | F4 | 1.0e–1 | 4.0e+1 | 20.400 | | |
| | | 8 | CC Delta | F4 | 2.9826e+4 | 1.193046e+6 | 20.316 | | |
| 107 | Current | 1 | Deadband | U1 | 0 | 255 | 0 | 5 | mA |
| | | 19 | RDL Tempco | F4 | 1.0e–20 | 4.0e+1 | 0 | | |
| 112 | Codes | 0 | Sealed to Unsealed | H4 | 0x0000/0x0000 | 0xFFFF/0xFFFF | 0x8000/0x8000 | | – |

## 6.5 Data Memory Parameter Descriptions

### 6.5.1 *Configuration Class*

#### 6.5.1.1 Safety Subclass

#### *6.5.1.1.1 Over Temp, Under Temp, Temp Hys*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 2 | Safety | Over Temp | I2 | –1200 | 1200 | 0 | 550 | 0.1°C |
| | | Under Temp | I2 | –1200 | 1200 | 0 | 0 | 0.1°C |
| | | Temp Hys | U1 | 0 | 255 | 0 | 50 | 0.1°C |

An Over-Temperature condition is detected if *Temperature()* ≥ ***Over Temp*** (default = 55°C) and indicated by setting the *Flags() [OT]* bit. The *[OT]* bit is cleared when *Temperature()* < ***Over Temp*** – ***Temp Hys*** (default = 50°C).

An Under-Temperature condition is detected if *Temperature()* ≤ ***Under Temp*** (default = 0°C) and indicated by setting the *Flags() [UT]* bit. The *[UT]* bit is cleared when *Temperature()* > ***Under Temp*** + ***Temp Hys*** (default = 5°C).

#### 6.5.1.2 Charge Termination Subclass

#### *6.5.1.2.1 Min Taper Capacity, Current Taper Window*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 36 | Charge Termination | Min Taper Capacity | I2 | 0 | 1000 | 0 | 25 | mAh |
| | | Current Taper Window | U1 | 0 | 60 | 0 | 40 | s |

Refer to description of ***Taper Rate*** and ***Taper Voltage*** for details on ***Min Taper Capacity*** and ***Current Taper Window***. See Section 6.5.2.3.8.

#### *6.5.1.2.2 Terminate Charge Alarm Set % (TCA Set %), Terminate Charge Alarm Clear % (TCA Clear %)*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 36 | Charge Termination | TCA Set % | I1 | –1% | 100% | 0 | 99 | % |
| | | TCA Clear % | I1 | –1% | 100% | 0 | 95 | % |

The *Flags() [CHG]* bit is set when SOC reaches ***TCA Set*** and is cleared when it drops below ***TCA Clear***.

The *Flags() [CHG]* bit is set when Primary Charge Termination conditions are met and ***TCA Set*** is set to –1%.

#### 6.5.1.2.3 Full Charge Set % (FC Set %), Full Charge Clear % (FC Clear %)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 36 | Charge Termination | FC Set % | I1 | –1% | 100 | 0 | –1 | % |
| | | FC Clear % | I1 | –1% | 100 | 0 | 98 | % |

The *Flags() [FC]* bit is set when SOC reaches ***FC Set*** and is cleared when it drops below ***FC Clear***.

The *Flags() [FC]* bit is set when Primary Charge Termination conditions are met and ***FC Set*** is set to –1%.

#### 6.5.1.2.4 DOD at EOC Delta Temperature

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 36 | Charge Termination | DODatEOC Delta T | I2 | 0 | 1000 | 0 | 50 | 0.1°C |

During relaxation and charge start, *RemainingCapacity()* = *FullChargeCapacity()* – *Qstart()*. But with temperature decreases, *Qstart()* can become much smaller than old *FullChargeCapacity()* resulting in over-estimation of *RemainingCapacity()*. To improve accuracy, *FullChargeCapacity()* is updated when temperature change from previous *FullChargeCapacity()* update is more than ***DODatEOC Delta T*** since the last simulation.

### 6.5.1.3 Data Subclass

#### 6.5.1.3.1 Initial Standby Current

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 48 | Data | Initial Standby | I1 | –256 | 0 | 0 | –3 | mA |

This is the initial value that is reported in *StandbyCurrent()*. The *StandbyCurrent()* value is updated every 1 second when the measured current meets the following criteria:
|Current| > ***Deadband*** and Current ≤ 2 × ***Initial Standby.***

**NOTE:** Current is negative during discharge.

This value depends on the system. The initial standby current is the current load drawn by the system when in low-power mode.

#### 6.5.1.3.2 Initial Maximum Load Current

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 48 | Data | Initial MaxLoad | I2 | –32767 | 0 | 0 | –200 | mA |

This is the initial value that is reported in *MaxLoadCurrent()*. The *MaxLoadCurrent()* is updated to the new current when Current > ***Initial MaxLoad***. *MaxLoadCurrent()* is reduced to the average of the previous value and ***Initial MaxLoad*** whenever the battery is charged to full after a previous discharge to an SOC less than 50%. This prevents the reported value from maintaining an unusually high value. Default value depends on the system.

### 6.5.1.4 Discharge Subclass

#### 6.5.1.4.1 State of Charge 1 Set or Clear Threshold (SOC1 Set Threshold, SOC1 Clear Threshold)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 49 | Discharge | SOC1 Set Threshold | U1 | 0 | 255 | 0 | 10 | % |
| | | SOC1 Clear Threshold | U1 | 0 | 255 | 0 | 15 | % |

When *StateOfCharge()* falls to or below the first capacity threshold, specified in ***SOC1 Set Threshold***, the *Flags() [SOC1]* bit is set. This bit is cleared once *StateOfCharge()* rises to or above ***SOC1 Clear Threshold***.

These values are up to the user's preference.

#### 6.5.1.4.2 State of Charge Final Set or Clear Threshold (SOCF Set Threshold, SOCF Clear Threshold)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 49 | Discharge | SOCF Set Threshold | U1 | 0 | 255 | 0 | 2 | % |
| | | SOCF Clear Threshold | U1 | 0 | 255 | 0 | 5 | % |

When *RemainingCapacity()* falls to or below the final capacity threshold, specified in ***SOCF Set Threshold***, the *Flags() [SOCF]* bit is set. This bit is cleared once *RemainingCapacity()* rises to or above ***SOCF Clear Threshold***. The *[SOCF]* bit serves as the final discharge warning.

These values are up to the user's preference.

### 6.5.1.5 Registers Subclass

#### *6.5.1.5.1 OpConfig*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 64 | Registers | OpConfig | H2 | 0x0000 | 0xFFFF | 0x0000 | 0x25F8 | hex |

**Table 6-6. Op Config Register Definition**

| | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|
| **High Byte** | RSVD0 | RSVD0 | RSVD1 | RSVD0 | RSVD0 | RSVD1 | RSVD0 | RSVD1 |
| **Default =** | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| | **0x25** | | | | | | | |
| **Low Byte** | RSVD1 | RSVD1 | SLEEP | RMFCC | RSVD1 | RSVD0 | RSVD0 | TEMPS |
| **Default =** | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | **0xF8** | | | | | | | |

SLEEP = The fuel gauge can enter sleep, if operating conditions allow. True when set.

RMFCC = RM is updated with the value from FCC on valid charge termination. True when set.

TEMPS = Selects the temperature source. Enables the host to write *Temperature()* if set. If cleared, the internal temperature sensor is used for *Temperature()*.

RSVD0 = Reserved. Default is 0. (Set to 0 for proper operation)

RSVD1 = Reserved. Default is 1. (Set to 1 for proper operation)

#### *6.5.1.5.2 OpConfigB*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 64 | Registers | OpConfigB | H1 | 0x00 | 0xFF | 0x00 | 0x0F | hex |

**Table 6-7. Op ConfigB Register Definition**

| | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|
| | RSVD0 | RSVD0 | RSVD0 | RSVD0 | RSVD1 | SMOOTHEN | RSVD1 | RSVD1 |
| **Default =** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | **0x0F** | | | | | | | |

SMOOTHEN = Enables the SOC smoothing feature. True when set.

RSVD0 = Reserved. Default is 0. (Set to 0 for proper operation)

RSVD1 = Reserved. Default is 1. (Set to 1 for proper operation)

#### 6.5.1.5.3 DM Code

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 64 | Registers | DM Code | H1 | 0x00 | 0xFF | A = 0x80<br>B = 0x90<br>B = 0xA0 | | hex |

This register contains the value to be returned by the *DM_CODE* subcommand. In addition to the *CHEM ID* subcommand, the *DM_CODE* subcommand can be used to distinguish each type of bq27411-G1A, B, or C device.

### 6.5.1.6 Power Subclass

#### 6.5.1.6.1 Hibernate Current

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 68 | Power | Hibernate I | U2 | 0 | 700 | 0 | 3 | mA |

When |*AverageCurrent()*| < ***Hibernate I***, the gauge enters HIBERNATE mode if *CONTROL_STATUS [HIBERNATE]* bit is set and cell is relaxed. This setting should be below any normal application currents.

#### 6.5.1.6.2 Hibernate Voltage

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 68 | Power | Hibernate V | U2 | 2400 | 3000 | 0 | App Dep | mV |

When |Voltage| < ***Hibernate V***, the gauge enters HIBERNATE mode. The *CONTROL_STATUS [HIBERNATE]* bit has no impact for the gauge to enter HIBERNATE mode. This setting should be below any normal application voltage.

#### 6.5.1.6.3 Full Sleep Wait Time

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 68 | Power | FS Wait | U1 | 0 | 255 | 0 | 1 | s |

FULLSLEEP mode can be entered by setting the ***FS Wait*** to a number larger than 0. The FULLSLEEP mode is entered when the timer counts down to 0.

This value is set to 0 to disable the ***FS Wait*** function.

## *6.5.2 Gas (Fuel) Gauging Class*

### 6.5.2.1 IT Cfg Subclass

#### *6.5.2.1.1 Sample Secondary Relax Time*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Sec. Relax Time | U2 | 0 | 65535 | 0 | 30 | s |

***Sec. Relax Time*** sets the measurement frequency of the Ra values that calculate the Ra grid point value. At the same time, it also sets the frequency of Fast Ra Scaling calculation.

#### *6.5.2.1.2 Average Time (Sample Average Relax Time)*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Average Time | U1 | 0 | 255 | 0 | 8 | Counts |

Sample ***Average Time*** specifies time used for voltage, temperature, and current averages used for resistance calculation.

#### *6.5.2.1.3 Minimum DOD Resistance Update*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Min DOD Res Update | U1 | 0 | 255 | 0 | 70 | % |

***Min DOD Res Update*** specifies the delta DOD between 2 grid points. Resistance measurements are not used for the next grid calculation if they were made below ***Min DOD Res Update***.

#### *6.5.2.1.4 Maximum and Minimum Resistance Factors*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Max Res Factor | U1 | 0 | 255 | 0 | 20 | — |
| | | Min Res Factor | U1 | 0 | 255 | 0 | 5 | — |

The ***Max Res Factor*** and ***Min Res Factor*** parameters are cumulative filters which limit the change in Ra values to a scale on a per discharge cycle basis.

For Ra_new > Ra_old,
New Ra = min(Ra_new, Ra_old × ***Max Res Factor*** ÷ 10)

For Ra_new < Ra_old
New Ra = max(Ra_new, Ra_old × ***Min Res Factor*** ÷ 10)

#### 6.5.2.1.5 Ra Filter

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Ra Filter | U2 | 0 | 1000 | 0 | 800 | — |

***Ra Filter*** is a filter constant used to calculate the filtered Ra value that is stored into Data Memory from the old Ra value.

Ra = (Ra_old × ***Ra Filter*** + Ra_new × (1000 – ***Ra Filter***))÷ 1000

#### 6.5.2.1.6 Max V Delta

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Max V Delta | U1 | 0 | 255 | 0 | 10 | — |

***Max V Delta*** specifies dV/dt condition for Qmax qualification in RELAX mode. Value of 10 corresponds to 1 µV/s.

#### 6.5.2.1.7 Fast Qmax Start/End DOD%, Fast Qmax Start Voltage Delta, Fast Qmax Current Threshold

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Fast Qmax Start DOD% | U1 | 0 | 255 | 0 | 92 | % |
| | | Fast Qmax End DOD % | U1 | 0 | 255 | 0 | 96 | % |
| | | Fast Qmax Start Volt Delta | I2 | 0 | 4200 | 0 | 125 | mV |
| | | Fast Qmax Current Threshold | I2 | 0 | 1000 | 0 | 4 | 0.1 Hr rate |
| | | Fast Qmax Min Points | U1 | 0 | 255 | 0 | 3 | — |
| | | Fast Qmax Volt Buffer | U1 | 0 | 255 | 0 | 25 | mV |

Fast Qmax measurement starts when the following conditions are met,

- DOD > ***Fast Qmax Start DOD%*** or
  Voltage < ***Terminate Voltage*** + ***Fast Qmax Start Volt Delta***
- Current < C / ***Fast Qmax Current Threshold***

Fast Qmax measurement is calculated at the end of discharge when the following conditions are met:

- Number of Fast Qmax measurements > ***Fast Qmax Min Points***
- DOD > ***Fast Qmax End DOD%*** or
  Voltage < ***Terminate Voltage*** + ***Fast Qmax Volt Buffer***

### 6.5.2.1.8 Qmax Capacity Error

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Qmax Capacity Err | U1 | 0 | 100 | 0 | 15 | 0.1% |

***Qmax Capacity Err*** specifies maximum capacity error allowed during Qmax update. Capacity error is estimated based on the time spent for Qmax measurement.

### 6.5.2.1.9 Maximum Qmax Change

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Max Qmax Change | U1 | 0 | 255 | 0 | 20 | % |

***Max Qmax Change*** specifies maximum allowed change in Qmax value during Qmax update. Qmax update is disqualified if change from previous Qmax value is greater than ***Max Qmax Change***.

### 6.5.2.1.10 Simulation Res Relax Time (ResRelax Time)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | ResRelax Time | U2 | 0 | 65534 | 0 | 500 | s |

This value is used for Impedance Track™ transient modeling of effective resistance. The resistance increases from 0 to final value determined by the Ra table as defined by the exponent with time constant ***Res Relax Time*** during discharge simulation. Default value has been optimized for typical cell behavior.

### 6.5.2.1.11 User Rate-Pwr

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | User Rate-Pwr | I2 | -14000 | -3000 | 0 | 0 | cW |

This is the discharge rate in mW used for Impedance Track™ simulation of voltage profile to determine discharge capacity. It is only used when ***Load Mode*** = 1 (constant power) and ***Load Select*** = 6 (user-defined rate).

### 6.5.2.1.12 Reserve Energy (Reserve Cap-mWh)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Reserve Cap-mWh | I2 | 0 | 14000 | 0 | 0 | cWh |

***Reserve Cap-mWh*** determines how much actual remaining capacity exists after reaching 0 *RemainingCapacity()* before ***Terminate Voltage*** is reached. This register is only used if ***Load Mode*** = 1 (constant power). A no-load rate of compensation is applied to this reserve capacity. This is a specialized function to allow time for a controlled shutdown after 0 *RemainingCapacity()* is reached.

#### *6.5.2.1.13 Max Simulation Rate, Min Simulation Rate*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Max Sim Rate | U1 | 0 | 255 | 0 | 1 | Hr rate |
| | | Min Sim Rate | U1 | 0 | 255 | 0 | 20 | Hr rate |

Maximum and minimum limits for current used in simulation runs. The parameters are functions of *DesignCapacity()* (that is, C/***Max Sim Rate*** or C/***Min Sim Rate***).

#### *6.5.2.1.14 Ra Max Delta*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Ra Max Delta | U2 | 0 | 100 | 0 | 11 | mΩ |

During the update of Ra values a filtering process is performed to eliminate unexpected fluctuations in the updated Ra values. ***Ra Max Delta*** limits the change in Ra values to an absolute magnitude per Ra update.

### 6.5.2.2 Current Thresholds Subclass

#### *6.5.2.2.1 Discharge Detection Threshold*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 81 | Current Thresholds | Dsg Current Threshold | I2 | 0 | 2000 | 0 | 167 | .1 hr |

This register is used as a threshold in the gauge to determine if actual discharge current is flowing out of the battery. This is independent of the *Flags() [DSG]* bit, which indicates whether the gauge is in discharge mode or charge mode. If the gauge is charging, then the *[DSG]* bit is 0 and any other time, the *[DSG]* bit is set to 1. Impedance Track™ algorithm in the gauge requires more definitive information about whether current is flowing in either the charge or discharge direction. ***Dsg Current Threshold*** is used for this purpose. This default threshold should be sufficient for most applications. This threshold should be set low enough to be below any normal application load current but high enough to prevent noise or drift from affecting the measurement.

#### *6.5.2.2.2 Charge Detection Threshold*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 81 | Current Thresholds | Chg Current Threshold | I2 | 0 | 2000 | 0 | 100 | .1 hr |

This register is used as a threshold in the gauge to determine if actual charge current is flowing into the battery. This is independent of the *Flags() [DSG]* bit, which indicates whether the gauge is in discharge mode or charge mode. If the gauge is charging, then the *[DSG]* bit is 0 and any other time, the *[DSG]* bit is set to 1. Impedance Track™ algorithm in the gauge requires more definitive information about whether current is flowing in either the charge or discharge direction. ***Chg Current Threshold*** is used for this purpose. This default threshold should be sufficient for most applications. This threshold should be set low enough to be below any normal charge current but high enough to prevent noise or drift from affecting the measurement.

#### 6.5.2.2.3 Quit Current, Discharge Relax Time, Charge Relax Time

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 81 | Current Thresholds | Quit Current | I2 | 0 | 1000 | 0 | 250 | .1 hr |
| | | Dsg Relax Time | U2 | 0 | 8191 | 0 | 60 | s |
| | | Chg Relax Time | U1 | 0 | 255 | 0 | 60 | s |

The ***Quit Current*** is used as part of the Impedance Track™ algorithm to determine when the fuel gauge enters relaxation mode from a current flowing mode in either the charge direction or the discharge direction. The value of ***Quit Current*** is set to a default value that should be above the standby current of the system. Either of the following criteria must be met to enter relaxation mode:

- |*AverageCurrent()*| < |***Quit Current***| for ***Dsg Relax Time***
- |*AverageCurrent()*| > |***Quit Current***| for ***Chg Relax Time***

After about 6 minutes in relaxation mode, the fuel gauge attempts to take accurate OCV readings. An additional requirement of dV/dt < 2 µV/s is required for the fuel gauge to perform Qmax updates. These updates are used in the Impedance Track™ algorithms. It is critical that the battery voltage be relaxed during OCV readings and that the current is not higher than C/20 when attempting to go into relaxation mode.

The ***Quit Current*** is used as part of the Impedance Track™ algorithm to determine when the gauge goes into relaxation mode from a current-flowing mode in either the charge direction or the discharge direction. Either of the following criteria must be met to enter relaxation mode:

- *AverageCurrent()* is greater than (–)***Quit Current*** and then goes within (±)***Quit Current*** for ***Dsg Relax Time***.
- *AverageCurrent()* is less than ***Quit Current*** and then goes within (±)***Quit Current*** for ***Chg Relax Time***.

After 30 minutes in relaxation mode, the gauge starts checking if the dV/dt < 4 µV/s requirement for OCV readings is satisfied. When the battery relaxes sufficiently to satisfy this criteria, the gauge takes an OCV reading for updating Qmax and for accounting for self-discharge. These updates are used in the Impedance Track™ algorithms.

It is critical that the battery voltage is relaxed during OCV readings to get the most accurate results. This current must not be higher than C/20 when attempting to go into relaxation mode; however, it should not be too low as to prevent going into relaxation mode due to noise. This should always be less than ***Chg Current Threshold*** or ***Dsg Current Threshold***.

#### 6.5.2.2.4 Quit Relax Time

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 81 | Current Thresholds | Quit Relax Time | U1 | 0 | 63 | 0 | 1 | s |

The ***Quit Relax Time*** is a delay time to exit relaxation. If current is great than ***Chg Current Threshold*** or less than ***Dsg Current Threshold*** and this condition is maintained during ***Quit Relax Time***, then exiting relaxation is permitted.

This setting is particular to handheld applications in which low duty cycle dynamic loads are possible.

#### 6.5.2.2.5 Transient Factor Charge, Transient Factor Discharge

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 81 | Current Thresholds | Transient Factor Charge | U1 | 0 | 255 | 0 | 179 | |
| | | Transient Factor Discharge | U1 | 0 | 255 | 0 | 179 | |

The ***Transient Factor Charge*** and ***Transient Factor Discharge*** parameters provide an adjustment of the computed resistance due to transient voltage readings upon pack insertion for either charge or discharge conditions. The values range from 0 to 255 (default 179) and used as a scaling factor / 256 to adjust resistance such that R_adj = R × Transient_Factor / 256.

#### 6.5.2.2.6 Max IR Correct

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 81 | Current Thresholds | Max IR Correct | U2 | 0 | 1000 | 0 | 400 | mV |

The ***Max IR Correct*** is a maximum IR correction applied to OCV lookup under load. It only applies to OCV lookup after wakeup with detected charge current when gauge needs to establish capacity baseline, but the current is already flowing.

### 6.5.2.3 State Subclass

#### 6.5.2.3.1 Qmax Cell 0

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Qmax Cell 0 | I2 | 0 | 32767 | 0 | 400 | mAh |

Qmax contains the maximum chemical capacity of the cell, and is determined by comparing states of charge before and after applying the load with the amount of charge passed. It corresponds to capacity at low rate (~C/20) of discharge. For high accuracy, this value is periodically updated by the gauge during operation. The Impedance Track™ algorithm updates this value and maintains it.

#### 6.5.2.3.2 Update Status

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Update Status | H1 | 0x00 | 0x06 | 0x00 | 0x00 | hex |

Bit 0 (0x01) of the ***Update Status*** register indicates that the gauge has learned new Qmax parameters and is accurate. The remaining bits are reserved. Bit 0 is updated as needed by the Impedance Track™ algorithm and should never be modified.

#### 6.5.2.3.3 Reserve Capacity (Reserve Cap-mAh)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Reserve Cap-mAh | I2 | 0 | 9000 | 0 | 0 | mAh |

***Reserve Cap-mAh*** determines how much actual remaining capacity exists after reaching 0 *RemainingCapacity()* before ***Terminate Voltage*** is reached. This register is only used if ***Load Mode*** = 0 (constant-current). A no-load rate of compensation is applied to this reserve capacity. This is a specialized function to allow time for a controlled shutdown after 0 *RemainingCapacity()* is reached.

#### 6.5.2.3.4 Load Select/Mode

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 80 | IT Cfg | Load Select/Mode | H1 | 0x00 | 0xFF | 0x00 | 0x81 | — |

***Load Mode*** configures the fuel gauge to use either a constant-current or constant-power model for the Impedance Track™ algorithm as. When ***Load Mode*** is 0, the Constant Current Model is used. When 1 (default), the Constant Power Model is used. The *CONTROL_STATUS [LDMD]* bit reflects the status of ***Load Mode***.

***Load Select*** is used in conjunction with ***Load Mode*** to define the type of load model that computes the load-compensated capacity in the Impedance Track™ algorithm.

If ***Load Mode*** = 0 (Constant current), then the following options are available:

**Table 6-8. Load Select/Mode Parameter Encoding**

|  | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|
|  | Load Mode | RSVD | RSVD | RSVD | RSVD | Load Select[2:0] |  |  |
| **Default =** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | **0x00** |  |  |  |  |  |  |  |
| **Typical =** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
|  | **0x81** |  |  |  |  |  |  |  |

Load Mode = Bit 7 contains the value for ***Load Mode***. Refer to tables below for operational details.

RSVD = Reserved. Set to 0 for proper operation.

Load Select[2:0] = Bits 2:0 contain the value for ***Load Select***. Refer to tables below for operational details. Default is 1.

**Table 6-9. Current Model Used When Load Mode = 0**

| Load Select Value | Current Model Used |
|---|---|
| 0 | Average discharge current from previous cycle: There is an internal register that records the average discharge current through each entire discharge cycle. The previous average is stored in this register. |
| 1 (default) | Present average discharge current: This is the average discharge current from the beginning of this discharge cycle until present time. |
| 2 | Average current: based off the *AverageCurrent()* |
| 3 | Current: based off of a low-pass-filtered version of *AverageCurrent()* (τ = 14 s) |
| 4 | Design capacity / 5: C Rate based off of ***Design Capacity*** / 5 or a C / 5 rate in mA. |
| All others | Reserved |

If ***Load Mode*** = 1 (Constant power) then the following options are available:

**Table 6-10. Power Model Used When Load Mode = 1**

| Load Select Value | Power Model Used |
|---|---|
| 0 | Average discharge power from previous cycle: There is an internal register that records the average discharge power through each entire discharge cycle. The previous average is stored in this register. |
| 1 (default) | Present average discharge power: This is the average discharge power from the beginning of this discharge cycle until present time. |
| 2 | Average current × voltage: based off the *AverageCurrent()* and *Voltage()*. |
| 3 | Current × voltage: based off of a low-pass-filtered version of *AverageCurrent()* (τ = 14 s) and *Voltage()* |
| 4 | Design energy / 5: C Rate based off of ***Design Energy*** /5 or a C / 5 rate in mA . |
| 5 | Reserved |
| 6 | User_Rate-mW: Use the value in ***User Rate-mW***. This provides a user-configurable load model. |
| All others | Reserved |

### 6.5.2.3.5 Q Invalid Maximum and Minimum Voltage

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Q Invalid MaxV | I2 | 0 | 32767 | A = 3803<br>B = 3814<br>C = 3808 | | mV |
| | | Q Invalid MinV | I2 | 0 | 32767 | A = 3752<br>B = 3748<br>C = 3752 | | mV |

***Q Invalid Max V*** and ***Q Invalid Min V*** specify the Qmax disqualification voltage region generally known as the "flat region" of the OCV vs DOD curve. OCV measurement for Qmax calculation is disallowed in this region.

### 6.5.2.3.6 Design Capacity, Design Energy, Default Design Cap

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Design Capacity | I2 | 0 | 32767 | 0 | App Dep | mAh |
| | | Design Energy | I2 | 0 | 32767 | 0 | App Dep | mWh |
| | | Default Design Cap | I2 | 0 | 32767 | A = 1340<br>B = 5580<br>C = 7361 | | mAh |

***Design Capacity*** is used for compensated battery capacity remaining and capacity when fully charged calculations are done by the gauge. It is also used for constant-current model for Impedance Track™ algorithm when ***Load Mode*** is 0 (constant-current) and ***Load Select*** is 4 (***Design Capacity*** / 5 for constant discharge). The *CONTROL_STATUS [LDMD]* bit indicates Impedance Track™ algorithm using constant-current model when cleared.

***Design Energy*** is used for compensated battery capacity remaining and capacity when fully charged calculations are done by the gauge. It is also used for constant-power model for Impedance Track™ algorithm when ***Load Mode*** is 1 (constant-power) and ***Load Select*** is 4 (***Design Energy*** / 5 for constant discharge). The *CONTROL_STATUS [LDMD]* bit indicates Impedance Track™ algorithm using constant-power model when set.

These values should be set based on the battery specification. See the data sheet from the battery manufacturer.

***Default Design Cap*** contains the capacity of the pack originally used to generate the CHEMID data and is used along with ***Design Capacity*** or ***Design Energy*** to scale data for the Impedance Track™ algorithm. ***Default Design Cap*** should not require modification for any specific application.

### *6.5.2.3.7 Terminate Voltage*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Terminate Voltage | I2 | 2800 | 3700 | 0 | App Dep | mV |

***Terminate Voltage*** is used in the Impedance Track™ algorithm to compute *RemainingCapacity()*. This is the absolute minimum voltage for end of discharge, where the remaining chemical capacity is assume to be zero.

This register is application dependent. It should be set based on battery cell specification to prevent damage to the cells or the absolute minimum system input voltage, taking into account impedance drop from the PCB traces, FETs, and wires.

### *6.5.2.3.8 Taper Rate, Taper Voltage*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Taper Rate | I2 | 0 | 2000 | 0 | 100 | .1 Hr rate |
| | | Taper Voltage | I2 | 0 | 5000 | A = 4100<br>B = 4200<br>C = 4250 | | mV |

***Taper Rate*** is used in the Primary Charge Termination Algorithm. *AverageCurrent()* is integrated over each of the two ***Current Taper Window*** periods separately and averaged separately to determine two averages (IRateAvg1, IRateAvg2).

The ***Taper Voltage*** threshold defines the minimum voltage necessary for as a qualifier for detection of charge termination.

Three requirements must be met to qualify for Primary Charge Termination:

- During two consecutive periods of ***Current Taper Windows***:
  IRateAvg1 < ***Taper Rate*** and IRateAvg2 < ***Taper Rate***
- During the same periods: Accumulated change in capacity < ***Min Taper Capacity*** per ***Current Taper Window***
- *Voltage()* > ***Taper Voltage***

When Primary Charge Termination conditions are met, the *Flags() [FC]* bit is set and *[CHG]* bit is cleared. Also, if the ***OpConfig [RMFCC]*** bit is set, then *RemainingCapacity()* is set equal to *FullChargeCapacity()*.

#### 6.5.2.3.9 Sleep Current

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Sleep Current | I2 | 0 | 100 | 0 | 10 | mA |

When *AverageCurrent()* is less than ***Sleep Current*** or greater than (–)***Sleep Current***, the gauge enters SLEEP mode if the feature is enable by setting the ***OpConfig [SLEEP]*** bit.

This setting should be below any normal application currents.

#### 6.5.2.3.10 Voltage at Charge Termination (V at Chg Term)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | V at Chg Term | I2 | 0 | 5000 | A = 4190<br>B = 4290<br>C = 4340 | | mV |

***V at Chg Term*** is a learned Data Memory parameter as the voltage reading at the full charge taper termination. This voltage is used to calculate DODatEOC.

#### 6.5.2.3.11 Average Current Last Run (Avg I Last Run)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Avg I Last Run | I2 | –32768 | 32767 | 0 | -50 | mA |

The gauge logs the current averaged from the beginning to the end of each discharge cycle. It stores this average current from the previous discharge cycle in this register. This register should never need to be modified. It is only updated by the gauge when required.

#### 6.5.2.3.12 Average Power Last Run (Avg P Last Run)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Avg P Last Run | I2 | –32768 | 32767 | 0 | -50 | mW |

The gauge logs the power averaged from the beginning to the end of each discharge cycle. It stores this average power from the previous discharge cycle in this register. To get a correct average power reading the gauge continuously multiplies instantaneous current times voltage to get power. It then logs this data to derive the average power. This register should never need to be modified. It is only updated by the gauge when required.

#### 6.5.2.3.13 Pulse Delta Voltage

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 82 | State | Delta Voltage | I2 | –32768 | 32767 | 0 | 1 | mV |

The gauge stores the maximum difference of Voltage during short load spikes and normal load, so the Impedance Track™ algorithm can calculate *RemainingCapacity()* for pulsed loads. It is added to ***Terminate Voltage*** for Impedance Track™ simulations.

### 6.5.3 Ra Table Class

#### 6.5.3.1 R_a NVM Subclass, R_a RAM Subclass

##### 6.5.3.1.1 Cell0 R_a NVM Table

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 88 | R_a NVM | Cell0 R_a 0 - 14 | I2 | | | 0 | App Dep | $2^{-10}$ Ω |
| 89 | R_a RAM | Cell0 R_a 0 - 14 | I2 | | | [R_a RVM] | Dynamic | $2^{-10}$ Ω |

The Ra Table class has 15 values. The R_a ROM is used to initialize R_a RAM upon gauge reset. Each of these values represents a resistance value normalized at 0°C for the associated Qmax Cell 0 -based SOC grid point as found by the following rules:

For Cell0 Ra M where:

- If 0 ≤ M ≤ 7: The data is the resistance normalized at 0° for:
  SOC = 100% – (M × 11.1%)
- If 8 ≤ M ≤ 14: The data is the resistance normalized at 0° for:
  SOC = 100% – [77.7% + (M – 7) × 3.3%]

This gives a profile of resistance throughout the entire SOC profile of the battery cells concentrating more on the values closer to 0%.

Normal Setting:

> These resistance profiles are used by the gauge for the Impedance Track™ algorithm. The only reason this data is displayed and accessible is to give the user the ability to update the resistance data on golden image files. This resistance profile description is for information purposes only. It is not intended to give a detailed functional description for the gauge resistance algorithms. It is important to note that this data is in mΩ and is normalized to 0°C. The following are useful observations to note with this data throughout the application development cycle:
>
> - Watch for negative values in the Ra Table class. Negative numbers in profiles should never be anywhere in this class.
> - Watch for smooth consistent transitions from one profile grid point value to the next throughout each profile. As the gauge does resistance profile updates, these values should be roughly consistent from one learned update to another without huge jumps in consecutive grid points.

## 6.5.4 Calibration Class

### 6.5.4.1 Data Subclass

#### 6.5.4.1.1 Board Offset

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 104 | Data | Board Offset | I1 | –128 | 127 | 0 | 0 | µV |

***Board Offset*** is the second offset register. Its calibrates all that the ***CC Offset*** does not calibrate out. This includes board layout, sense resistor and copper trace, and other potential offsets that are external to the fuel gauge.

#### 6.5.4.1.2 Internal Temperature Offset (Int Temp Offset)

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 104 | Data | Int Temp Offset | I1 | –128 | 127 | 0 | 0 | °C |

The gauge has a temperature sensor built into the fuel gauge. The ***Int Temp Offset*** is used for calibrating offset errors in the measurement of the reported *Temperature()* if a known temperature offset exists between the fuel gauge and the battery cell. The gain of the internal temperature sensor is accurate enough that a calibration for gain is not required.

#### 6.5.4.1.3 Pack V Offset

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 104 | Data | Pack V Offset | I1 | –128 | 127 | 0 | 0 | mV |

This is the offset to calibrate the gauge analog-to-digital converter for cell voltage measurement.

***Pack V Offset*** should not require modification by the user. It is modified by the Voltage Calibration function from Calibration mode.

### 6.5.4.2 CC Cal Subclass

#### *6.5.4.2.1 (Coulomb Counter Sense Resistor)CC Offset, CC Gain, CC Delta*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 104 | Data | CC Offset | I2 | –32768 | 32767 | 0 | | mV |
| | | CC Cal Temp | I2 | 0 | 32767 | 303.0 | | |
| | | CC Gain | F4 | 1.0e–1 | 4.0e+1 | 20.400 | | |
| | | CC Delta | F4 | 2.9826e+4 | 1.193046e+6 | 20.316 | | |

***CC Offset***, ***CC Gain***, and ***CC Delta*** are internal calibration parameters that require no customer changes and provided for debug purposes only.

Two offsets are used for calibrating the offset of the internal coulomb counter, board layout, sense resistor, copper traces, and other offsets from the coulomb counter readings. ***CC Offset*** is the calibration value that primarily corrects for the offset error of the fuel gauge coulomb counter circuitry. The other offset calibration is ***Board Offset*** and is described separately. ***CC Offset*** is a correction for small noise/errors; therefore, to maximize accuracy, it takes about 16 seconds to calibrate the offset. Since it is impractical to do 16-second offset during IC production, the fuel gauge in will periodically perform a ***CC Offset*** automatic calibration in SLEEP mode. During the automatic calibration the fuel gauge will set the *CONTROL_STATUS [CCA]* bit.

***CC Gain*** is the gain factor for calibrating sense resistor, trace, and internal coulomb counter errors. It is used in the algorithm that reports *AverageCurrent()*. ***CC Delta*** is a fixed constant based on ***CC Gain*** used to cancel out the time base error.

### 6.5.4.3 Current Subclass 107

#### *6.5.4.3.1 Deadband*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 107 | Current | Deadband | U1 | 0 | 255 | 0 | 5 | mA |

The ***Deadband*** creates a filter window to the reported *AverageCurrent()* register where the current is reported as 0. Any negative current above this value or any positive current below this value is displayed as 0.

Only a few reasons may require changing the default value:

1. If the PCB layout has issues that cause inconsistent board offsets from board to board.
2. An extra noisy environment.

#### *6.5.4.3.2 RDL Tempco*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 107 | Current | RDL Tempco | F4 | 1.0e–20 | 4.0e+1 | 0 | 0 | |

Set to zero for proper operation.

### 6.5.5 *Security Class*

#### 6.5.5.1 Codes Subclass

##### *6.5.5.1.1 Sealed to Unsealed, Unsealed to Full Access*

| Subclass ID | Subclass | Name | Type | Min | Max | Default | Typical | Unit |
|---|---|---|---|---|---|---|---|---|
| 112 | Codes | Sealed to Unsealed | H4 | 0x0000 0000 | 0xFFFF FFFF | 0x8000 8000 | | hex |

SEALED access mode provides some protection against data corruption due to data packet errors. The fuel gauge implements a key-access scheme to transition from SEALED to UNSEALED mode. Once SEALED via the associated subcommand, a unique set of two keys must be sent to the fuel gauge via the *Control()* command to return to UNSEALED mode. The keys must be sent consecutively, with no other data being written to the *Control()* register in between.

When in the SEALED mode, the *CONTROL_STATUS [SS]* bit is set; but when the ***Sealed to Unsealed*** keys are correctly received by the fuel gauge, the *[SS]* bit is cleared. The ***Sealed to Unsealed*** key has two identical words stored in ROM with a value of 0x8000 8000. Then *Control()* should supply 0x8000 and 0x8000 (again) to unseal the part.

# Revision History

**Changes from Original (January 2014) to B Revision** **Page**

- Changed minor documentation typos .......... 4
- Changed the low Bit3 .......... 22
- Changed the BAT_DET bit description .......... 22

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

| **Products** | | **Applications** | |
|---|---|---|---|
| Audio | www.ti.com/audio | Automotive and Transportation | www.ti.com/automotive |
| Amplifiers | amplifier.ti.com | Communications and Telecom | www.ti.com/communications |
| Data Converters | dataconverter.ti.com | Computers and Peripherals | www.ti.com/computers |
| DLP® Products | www.dlp.com | Consumer Electronics | www.ti.com/consumer-apps |
| DSP | dsp.ti.com | Energy and Lighting | www.ti.com/energy |
| Clocks and Timers | www.ti.com/clocks | Industrial | www.ti.com/industrial |
| Interface | interface.ti.com | Medical | www.ti.com/medical |
| Logic | logic.ti.com | Security | www.ti.com/security |
| Power Mgmt | power.ti.com | Space, Avionics and Defense | www.ti.com/space-avionics-defense |
| Microcontrollers | microcontroller.ti.com | Video and Imaging | www.ti.com/video |
| RFID | www.ti-rfid.com | | |
| OMAP Applications Processors | www.ti.com/omap | **TI E2E Community** | e2e.ti.com |
| Wireless Connectivity | www.ti.com/wirelessconnectivity | | |

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