# ModelGauge User's Guide

Edition 3 September 2012

maxim integrated™

## Table of Contents

## Table of Figures

# 1 Overview of ModelGauge

ModelGauge™ is a sophisticated Li+ battery-modeling scheme that tracks the battery's relative state-of-charge (SOC) continuously over a widely varying charge/discharge profile. ModelGauge uses only voltage measurements; no current sensing is required.

The model effectively simulates the internal dynamics of a Li+ battery and considers the time effects of a battery caused by the chemical reactions and impedance in the battery. This allows the relaxed open circuit voltage (OCV) of the cell to be determined even while the cell is being charged or discharged. Once the open circuit voltage of the cell is determined, a internal look up table converts OCV to an SOC output.

ModelGauge does not accumulate error with time. This is advantageous compared to traditional coulomb counters, which suffer from SOC drift caused by current-sense offset and cell self-discharge. There is no need to perform learn cycles or to correct error at end of a charge cycle or end of a discharge cycle.

## 1.1 Best Applications for ModelGauge

1. Cell Chemistry must have voltage slope. At its core the ModelGauge algorithm monitors changes in the cell's voltage. Cell chemistries such as LiFePO4 that have a very flat voltage curve during discharge provide a challenge to ModelGauge due to voltage measurement resolution.

2. Low Cost. ModelGauge products are excellent for cost sensitive applications. They require few external components and are offered in extremely small package options minimizing board space. They function equally well when designed into the cell pack or the host system.

3. Does not require current or accumulated current information. ModelGauge products convert voltage to state-of-charge as a percentage. They do not measure current or calculate mAh.

## 1.2 OCV to SOC Relationship

Unlike the loaded cell voltage during charging or discharging, a cell's relaxed open circuit voltage is a very good indicator of the state-of-charge. The relationship between the two changes very little over age and operating conditions. It is important to note that this relationship can vary greatly between different cell chemistries and manufactures and can also vary greatly depending on the application. One application may consider 3.3V as empty (0%) while another may consider 2.8V as empty.

To achieve the best accuracy, the internal initialization or INI table used by each ModelGauge IC should be custom generated based on the cells and the application parameters. This custom table information is then loaded into the IC during operation. The complete procedure for doing this is explained in this document.

## 1.3 ModelGauge Product Selection

The table in *Figure 1. ModelGauge Product Feature Table* details the main features available in the ModelGauge product line allowing the user to select the IC that best fits the application. The algorithm at the core of each IC is the same.

| | MAX17040 | MAX17041 | MAX17043 | MAX17044 | MAX17048 | MAX17049 | MAX17058 | MAX17059 |
|---|---|---|---|---|---|---|---|---|
| **Pack Configuration** | 1 Cell | 2 Cell (2S) | 1 Cell | 2 Cell (2S) | 1 Cell | 2 Cell (2S) | 1 Cell | 2 Cell (2S) |
| **A/D Measurements** | Voltage Only | Voltage Only | Voltage Only | Voltage Only | Voltage Only | Voltage Only | Voltage Only | Voltage Only |
| **Outputs** | SOC % | SOC % | SOC % | SOC % | SOC %<br>CRATE | SOC %<br>CRATE | SOC % | SOC % |
| **Active Current (µA)** | 50.0 | 50.0 | 50.0 | 50.0 | 23.0 | 23.0 | 23.0 | 23.0 |
| **Sleep Current (µA)** | 1.0 | 1.0 | 1.0 | 1.0 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Alerts** | | | Low SOC | Low SOC | Low SOC<br>Low Voltage<br>High Voltage<br>SOC Change | Low SOC<br>Low Voltage<br>High Voltage<br>SOC Change | Low SOC | Low SOC |
| **Special Features** | | | | | Programmable Battery Insertion /Removal Detection<br><br>4µA Hibernate Mode | Programmable Battery Insertion /Removal Detection<br><br>4µA Hibernate Mode | Programmable Battery Insertion /Removal Detection | Programmable Battery Insertion /Removal Detection |
| **TDFN Package** | 2mm x 3mm<br>8-PIN with EP | 2mm x 3mm<br>8-PIN with EP | 2mm x 3mm<br>8-PIN with EP | 2mm x 3mm<br>8-PIN with EP | 2mm x 2mm<br>8-PIN with EP | 2mm x 2mm<br>8-PIN with EP | 2mm x 2mm<br>8-PIN with EP | 2mm x 2mm<br>8-PIN with EP |
| **WLP Package** | 1.6mm x 1.6mm<br>3x3 0.4mm pitch | 1.6mm x 1.6mm<br>3x3 0.4mm pitch | 1.6mm x 1.6mm<br>3x3 0.4mm pitch | 1.6mm x 1.6mm<br>3x3 0.4mm pitch | 0.9mm x 1.7mm<br>2x4 0.4mm pitch | 0.9mm x 1.7mm<br>2x4 0.4mm pitch | 0.9mm x 1.7mm<br>2x4 0.4mm pitch | 0.9mm x 1.7mm<br>2x4 0.4mm pitch |
| **Required Components** | 2 Resistors<br>2 Capacitors | 2 Resistors<br>2 Capacitors<br>Voltage Regulator | 2 Resistors<br>2 Capacitors | 2 Resistors<br>2 Capacitors<br>Voltage Regulator | 1 Capacitor | 1 Capacitor<br>Voltage Regulator | 1 Capacitor | 1 Capacitor<br>Voltage Regulator |

*Figure 1. ModelGauge Product Feature Table*

## 2 Implementing ModelGauge into your Design

Implementing a ModelGauge fuel gauge IC in an application is a several step process that requires planning and interaction with the characterization team at Maxim. It is necessary to know the steps involved at the beginning of design in order to achieve the best possible performance out of the fuel gauge and to avoid product development delays. *Figure 2* gives an overview of the steps required to implement ModelGauge in an application.

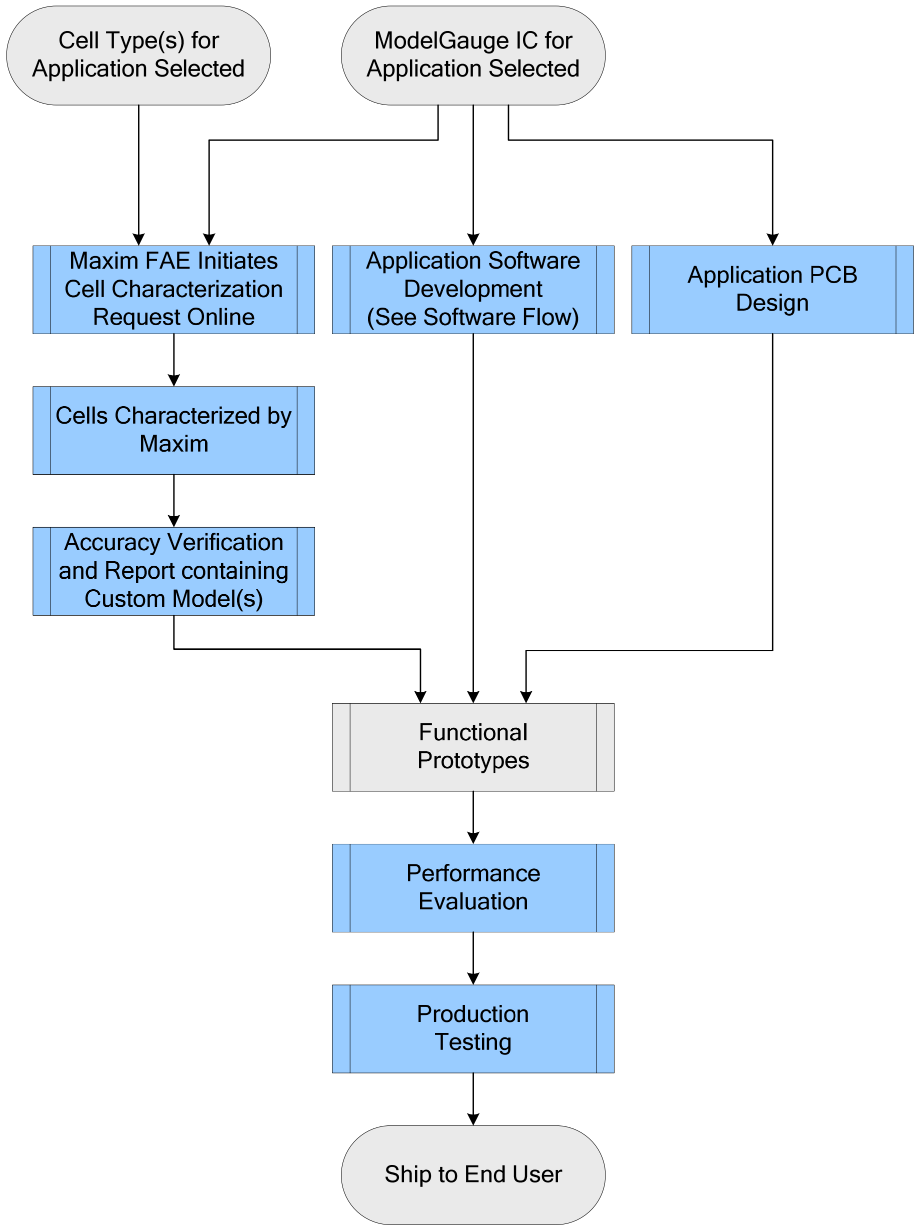

Cell Type(s) for Application Selected
ModelGauge IC for Application Selected
Maxim FAE Initiates Cell Characterization Request Online
Application Software Development (See Software Flow)
Application PCB Design
Cells Characterized by Maxim
Accuracy Verification and Report containing Custom Model(s)
Functional Prototypes
Performance Evaluation
Production Testing
Ship to End User

*Figure 2. ModelGauge Design Development*

# 3 Battery Characterization

Maxim's Fuel Gauge devices offer superior performance with competitive pricing and footprint size. The algorithm's excellent accuracy requires precise battery characterization. This process involves charging and discharging the battery at currents similar to what will be experienced in the application. Each test is customized to the application and results in a unique configuration, a custom model for that specific battery and application. Fuel gauge performance will be drastically improved if characterization is performed at load parameters of the actual application. ModelGauge may not perform well without the custom model, and the model may not perform well on other batteries.

## 3.1 Process Outline

1. Select a battery for use in your application.
2. Define the operating conditions of the application.
3. Contact a local Maxim Field Applications Engineer and fill out the Characterization Request Form.
4. Ship 3 or 4 battery packs to Maxim for characterization referencing the project number assigned by filling out the Characterization Request Form. **For safety reasons please ship battery packs not loose cells.** The characterization data needs to be collected from the perspective of the application. If characterization is performed on loose cells and the application is powered by a battery pack with protection circuitry then the additional resistance could affect ModelGauge performance.
5. After characterization is complete a performance report and custom model for the application will be emailed to you.
6. Begin your performance evaluation.

## 3.2 Application Operating Conditions

Prior to characterizing your cells it is important to understand how a battery is loaded dramatically affects how much it can discharge. Under heavy load, a battery supplies much less charge than under light load. Most batteries are recommended to be charged with constant current and then constant voltage in order to charge as much as possible. After the voltage reaches full, current continues to flow, so more energy is being stored in the battery even though the voltage remains the same. Conversely, a battery can be discharged with a constant current until it reaches its empty voltage. If the load is held to this constant voltage, current continues to flow, decreasing exponentially. There are many load currents which can correspond to the same empty voltage, depending on the preceding load current. The State Of Charge, on the other hand, does not depend on the preceding load current.

At high currents, the battery voltage for a given SOC is lower than at lower currents due to the low ion mobility of Li-Ion chemistry and the internal resistance of the battery.

### 3.2.1 Charge Termination Voltage

Typical Li-Ion cells are charged to 4.2V. Terminating charge at a lower voltage will reduce cell capacity by not allowing it to reach a “fully” charged state. Increasing the voltage above 4.2V will likely result in unsafe conditions. Please read the cell manufacturer specifications closely when making changes to Charge Voltage.

*Figure 3* shows an example of how charge voltage affects capacity.

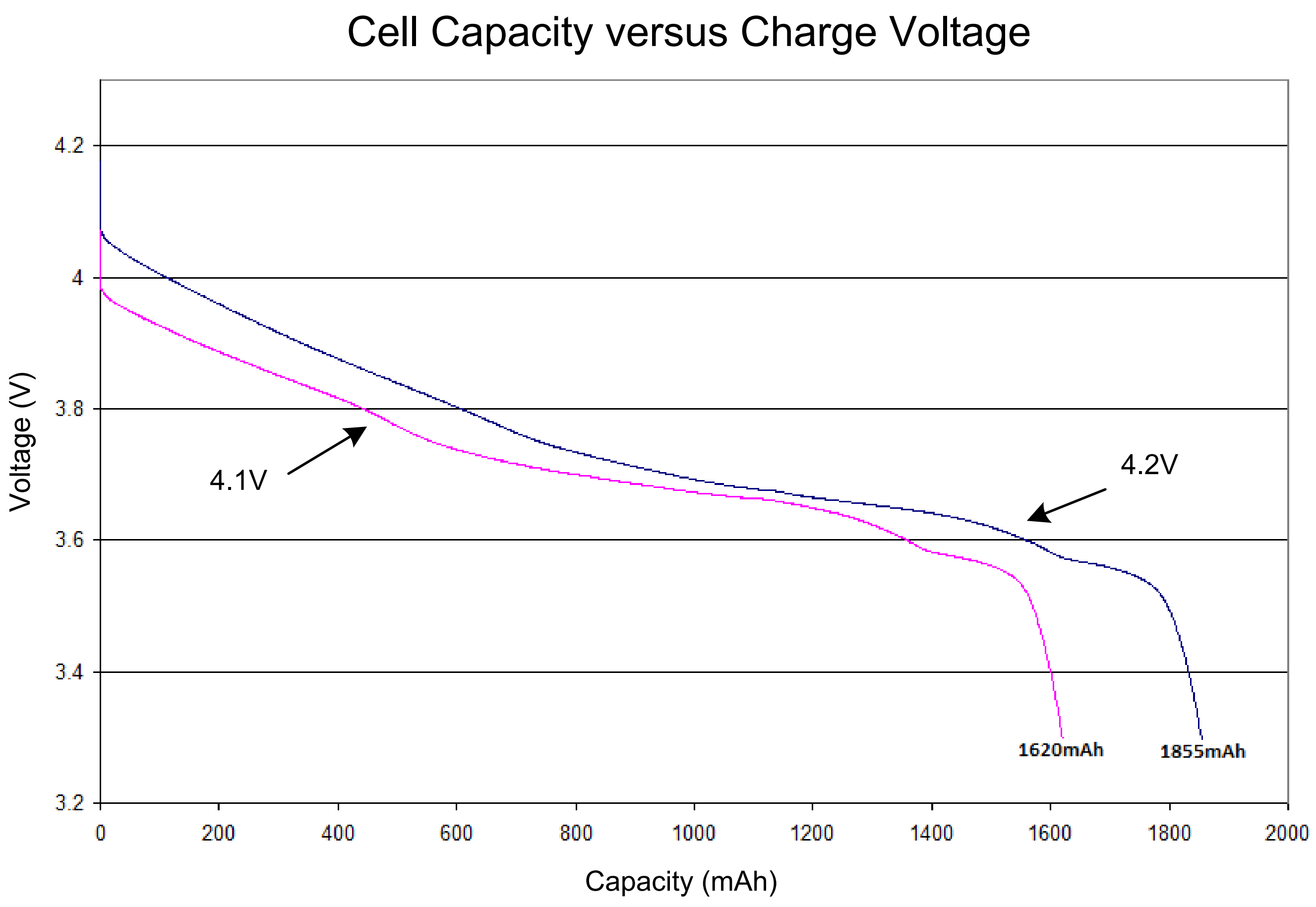


*Figure 3. Cell Capacity versus Charge Voltage*

### 3.2.2 Charge Termination Current

The lower the termination current the more “full” the cell will be when charge completes.

*Figure 4* shows how changes the termination current affect the capacity of the cell.

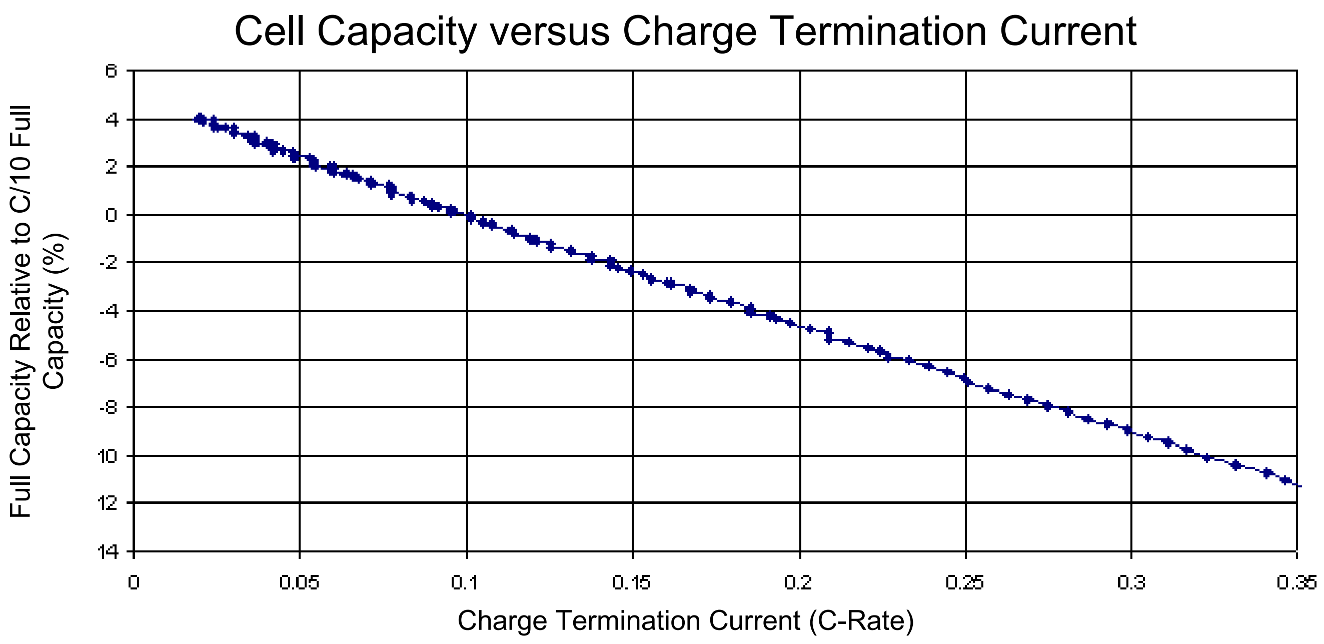


*Figure 4. Cell Capacity versus Charge Termination Current*

### 3.2.3 Application Loads

Characterization is performed at 3 different loads: standby, typical and heavy. When defining the loads for characterization the application current should be averaged over a period of 1-5 minutes. The discharge data is collected at a constant load from full to empty. Specifying too high of a current will cause a decrease in the characterized capacity – the fuel gauge will report empty early. In the chart below, if we assume the application has a 3.4V empty voltage, there is a 6.5% capacity difference between light and heavy load. The capacity difference will grow with a larger "heavy" discharge current.

The example shown in *Figure 5* has a 50mA "light" discharge, 650mA "medium" discharge and a 1300mA "heavy" discharge. The battery capacity is rated at 650mAhr. The discharge rates are C/13, 1C, and 2C for light, medium, heavy respectively.

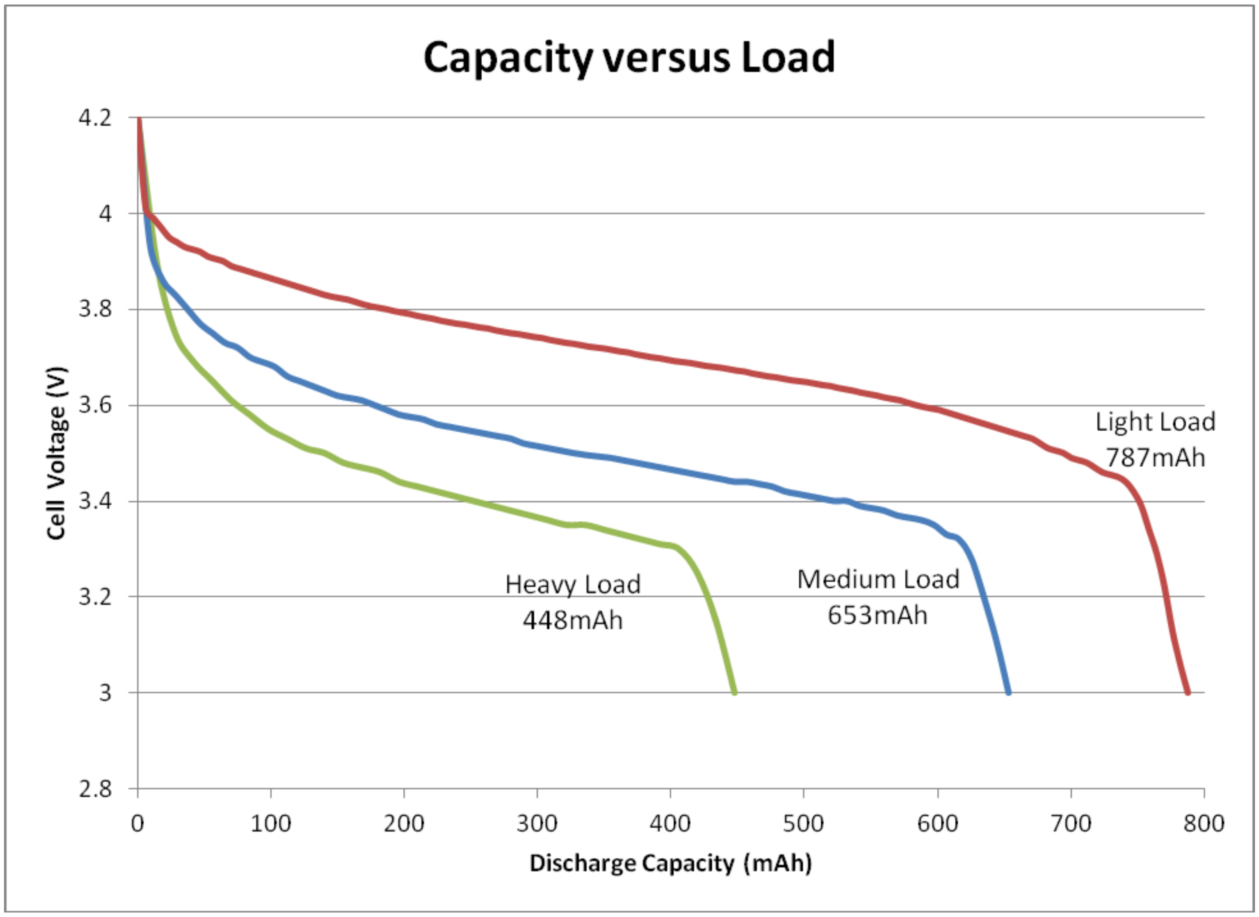


*Figure 5. Cell Capacity versus Discharge Rate*

### 3.2.3.1 Standby Load Current

This is the amount of current the device consumes when powered up, but idle. This current should have a value similar to the rated battery capacity divided by the expected standby time.

Examples:

- A cell phone in a pocket, not in active use, ready to receive a call.
- A tablet in sleep mode but ready to immediately awaken.

### 3.2.3.2 Typical Load Current

This is the amount of current the device consumes when used by a typical consumer. A very simple approximation is the Heavy Load (see below) times some duty cycle D, plus the Standby Load times (1-D). The duty cycle D is usually no more than 25% for most applications. For optimal, real-world performance the fuel gauge model should be focused on typical conditions. Examples:

- A cell phone where the user makes a ~20 minute call every 2 hours and browses the web for ~10 minutes every hour.
- A tablet where the user downloads a large file for 2 minutes (heavy radio use), then spends 10 minutes reading it (low radio/processor use), then streams video for 5 minutes (heavy radio/processor use) each hour.

#### 3.2.3.3 Heavy Load Current

The amount of current the device consumes when using all features simultaneously. The battery capacity divided by this current should be no less than the minimum usable device time. Heavy load is the maximum loading which could persist over many minutes. Fuel gauging should not focus on peak load conditions which only persist for milliseconds or seconds. Examples:

- A cell phone on a call while browsing the web at full brightness and playing music.
- A tablet streaming HD video with full brightness at 100% CPU usage. The battery capacity divided by this current should be no less than the minimum usable device time, e.g. nobody sells a new tablet which will use its whole battery in 1 hour.

### 3.2.4 Empty Voltage

Often it is desirable to design margin into the empty voltage to ensure that the application does not shut down unexpectedly. However, as empty voltage increases, the battery capacity decreases and it becomes more difficult to predict empty. Take another example discharge and vary the empty point from 3.0V up to 3.5V

*Figure 6* shows there is a large change in capacity at medium load. Also, notice that the capacity difference between heavy and light load at 3.0V is minimal as opposed to heavy load losing a large portion of the cell’s capacity at higher empty voltages. This decrease in capacity difference will present less of a challenge for ModelGauge when predicting empty.

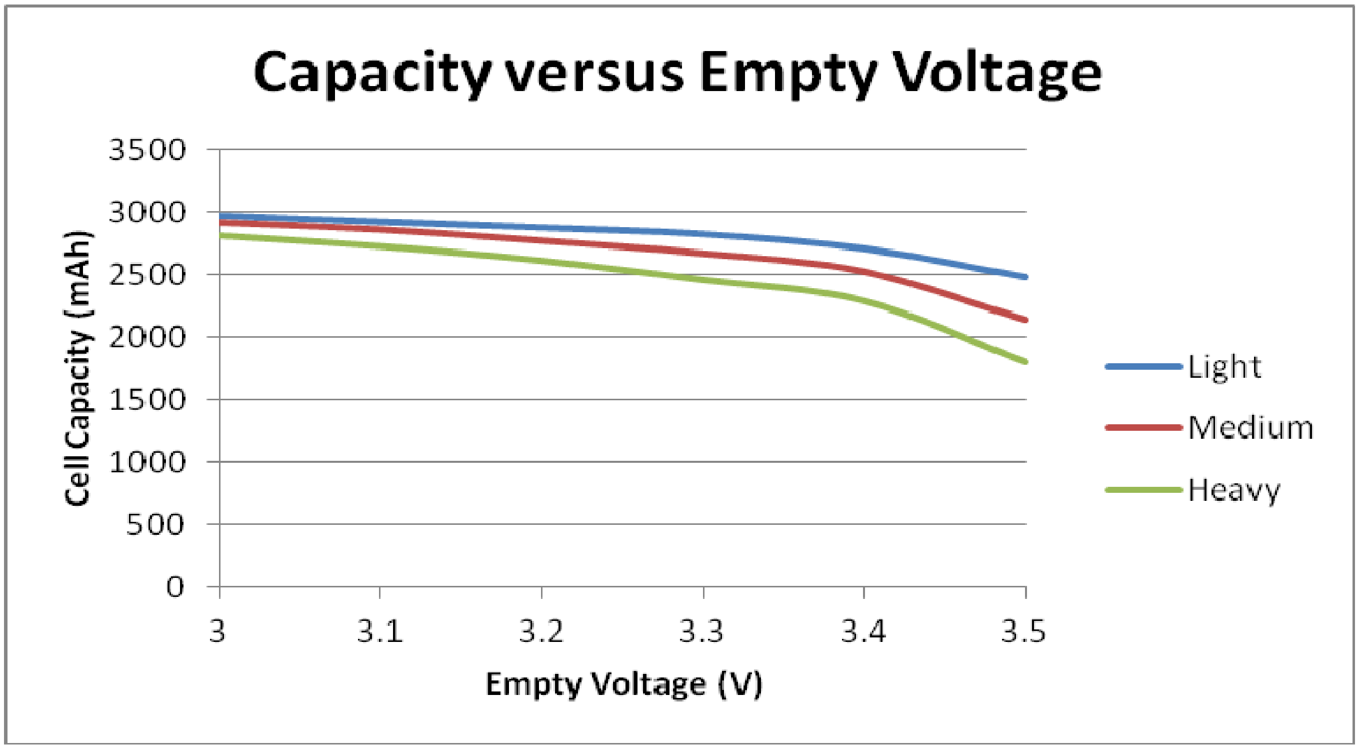


*Figure 6. Cell Capacity versus Empty Voltage*

Lithium Ion cells have a relatively flat discharge voltage curve. Typically the nominal cell voltage is rated near 3.7V (this can vary by 100mV depending on cell manufacturer). The majority of the capacity of the cell is discharged near this nominal voltage. This means that the SOC % per mV is greater around the nominal voltage (the change in SOC will be relatively large compared to the change in voltage). Once a lithium ion cell nears its empty point the voltage will decrease dramatically. In the previous example this happens around 3.4V. ModelGauge uses OCV to estimate capacity. OCV will be higher than the voltage of the cell under load. Notice that after the discharge stops the cell voltage relaxes to a higher voltage. The closer this relaxed OCV is to the nominal voltage the greater the challenge to ModelGauge in estimating SOC. Near the nominal voltage the SOC change with respect to voltage can be as high as 20% for every 100mV. This means that as little as 10mV can equal 2% SOC. Closer to empty the SOC change can be as high as 3% for every 100mV; 10mV becomes insignificant in terms of SOC. Select a lower empty voltage whenever possible for best ModelGauge performance as well as best utilization of the battery.

In *Figure 7* a cell is discharged at a C/2.5 rate to 3.4V. When the load stops the battery voltage recovers to OCV=3.69V. This voltage is right at the nominal cell voltage where %SOC/mV is highest. This discharge profile presents a greater challenge to ModelGauge. If the application allows for it, a lower empty voltage and load current would likely yield much greater ModelGauge accuracy.

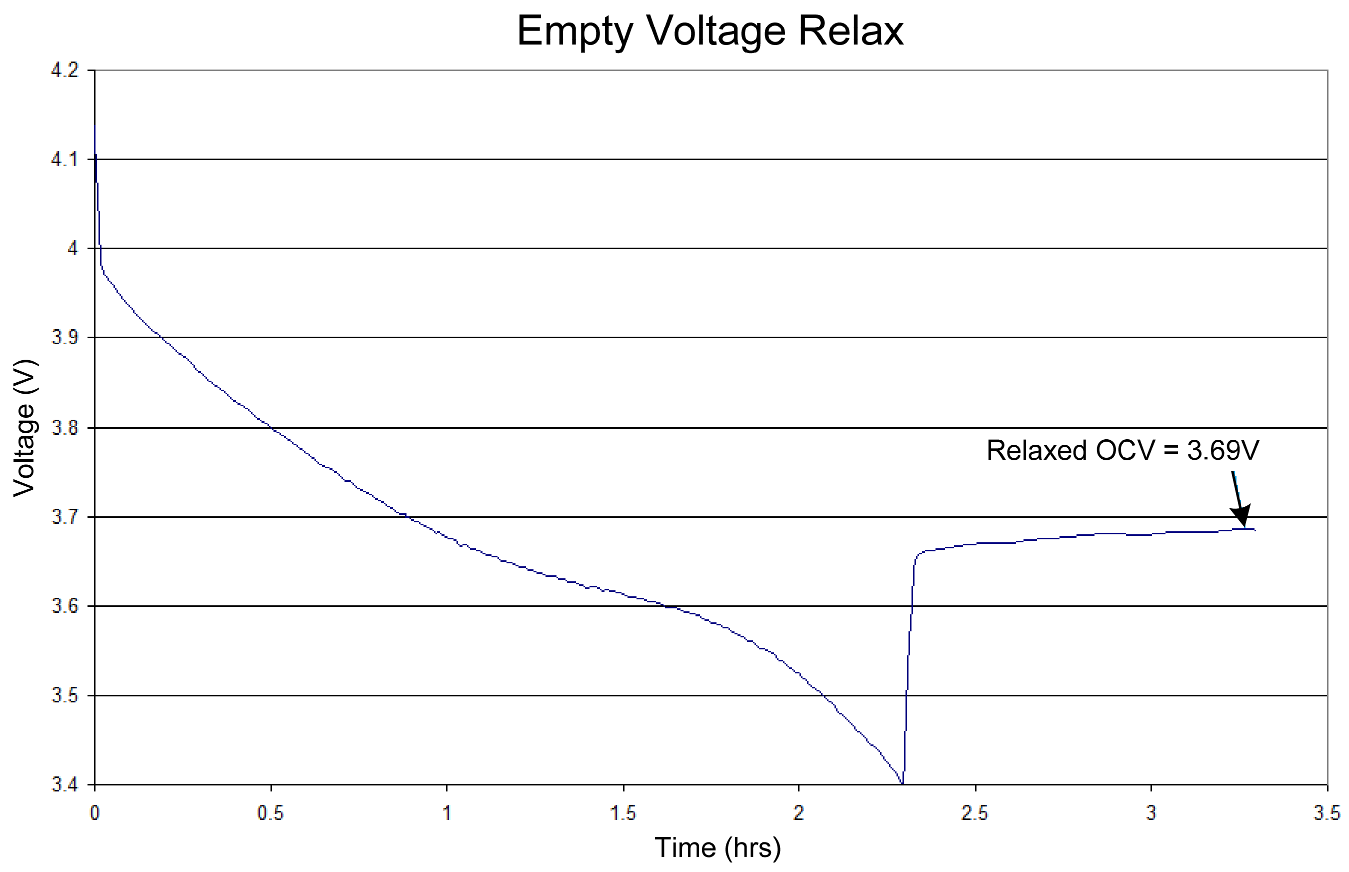


*Figure 7. Cell Relaxation after Discharge Ends*

### 3.2.5 Defining Empty

There are two ways to define when a battery is empty:

1. **From the perspective of the battery**: when discharging further causes damage to the battery. A Lithium cell can be damaged if its voltage drops too low. This number varies between manufacturers and products, but a typical minimum cell voltage is 3V. If the load can operate below 3V, such as with 2.5V logic families, then the load should be controlled to protect the battery. In order to maximize utility of the battery, the load can be active until the load voltage drops to the battery minimum voltage (e.g. 3V to protect the battery).

2. **From the perspective of the load**: when discharging further would cause the load to shut down. If the load cannot operate below the minimum battery voltage, then the battery can be discharged until the load no longer works. This is usually the dominating

factor in consumer applications. In many cell-phone or tablet applications this is limited by the operating range of the radio.

#### 3.2.5.1 Absolute Minimum Voltage and Design Margin

For typical single-cell applications which have critical circuits that shut down above 2.5V, the most important measurement of a fuelgauge is the voltage at which it reports <3% SOC. If this value is 50mV below the minimum at which that circuit can operate, then the application could shut down unexpectedly. The inclination to select an Empty Voltage higher than this minimum is understandable. How much higher should this value be?

#### 3.2.5.2 Increasing Empty Voltage Reduces Battery Capacity Nonlinearly

Consider a battery which is discharged from full at C/1, then charged to full again and discharged at C/10. The current multiplied by the discharge time will result in different values of discharge capacity. A simple first-order approximation that a battery has a constant series resistance will yield this same result - the I*R drop is larger with larger currents, so any given voltage will be reached sooner with a higher current. However, the actual relationship is nonlinear as shown below.

*Figure 8* and Figure 9 show sample data from an ATL 506971 cell. This 2900mAh battery has two very different capacities when discharged at different loads. Consider a fixed Empty Voltage of 3.4V. At a C/2 (1500mA) load, the battery is at 3.4V after 1500mAh of discharge capacity. While at a C/10 (300mA) load, the battery is at 3.4V after 2500mAh of discharge capacity. The fuel gauge has to reconcile the two different capacities and make 0% SOC occur close in time (discharge capacity) to when the empty voltage is reached at the two different loads. The difference in capacity over the two loads is 1000mAh, or nearly 30% of the battery.

The capacity available in a battery is dependent on the load condition (current and temperature) and also on the selection of empty voltage. In some cases the sensitivity to the empty voltage can be very significant. In these situations ModelGauge will tend to be less accurate than if a lower voltage is selected. The resulting error for this project was very poor for 3.4V empty voltage selection, but very reasonable for 3.2V.

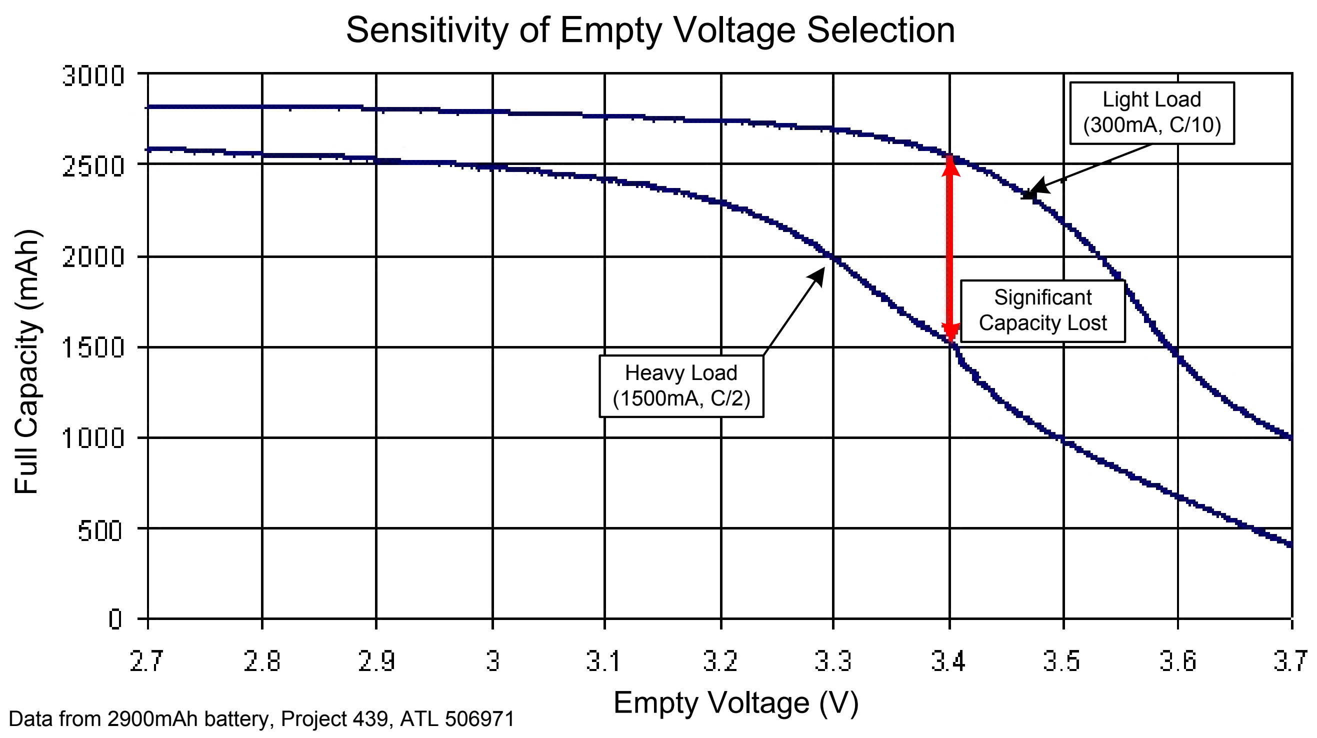
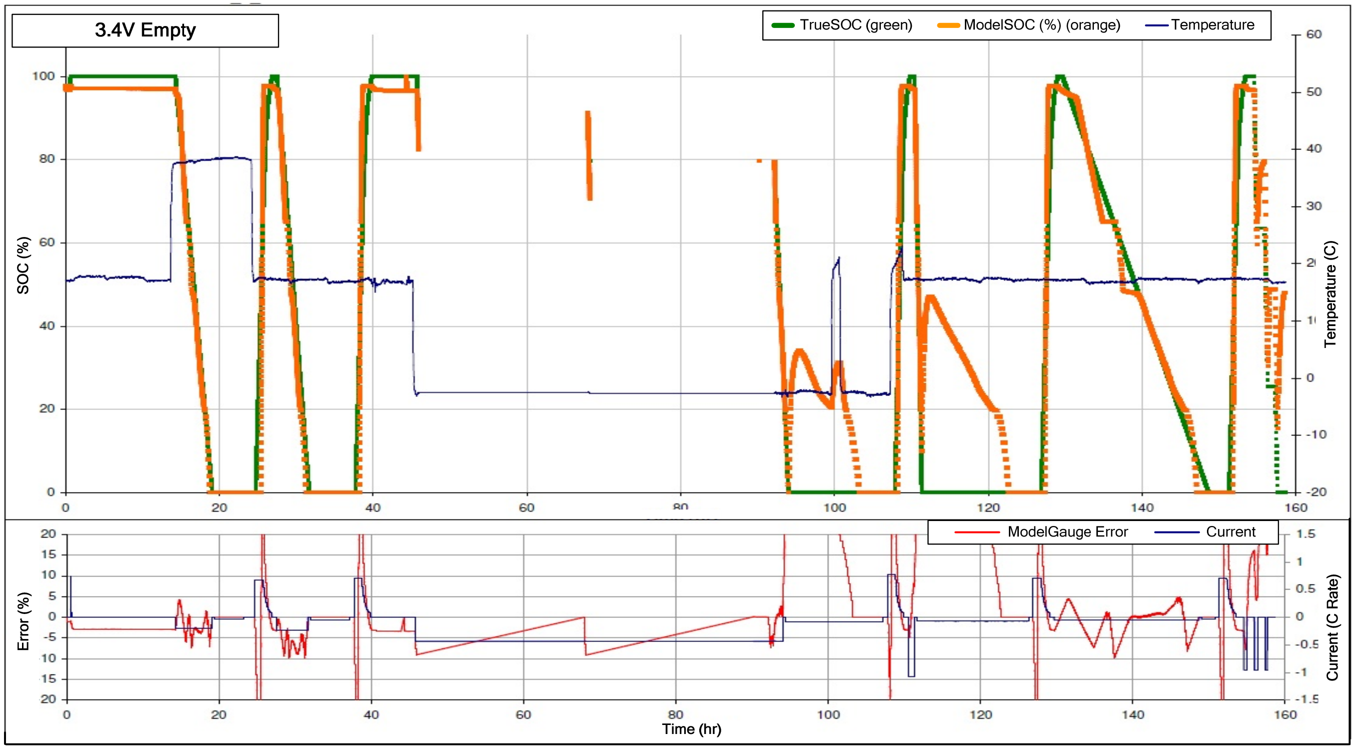


*Figure 8. Cell Capacity Sensitivity to Empty Voltage*



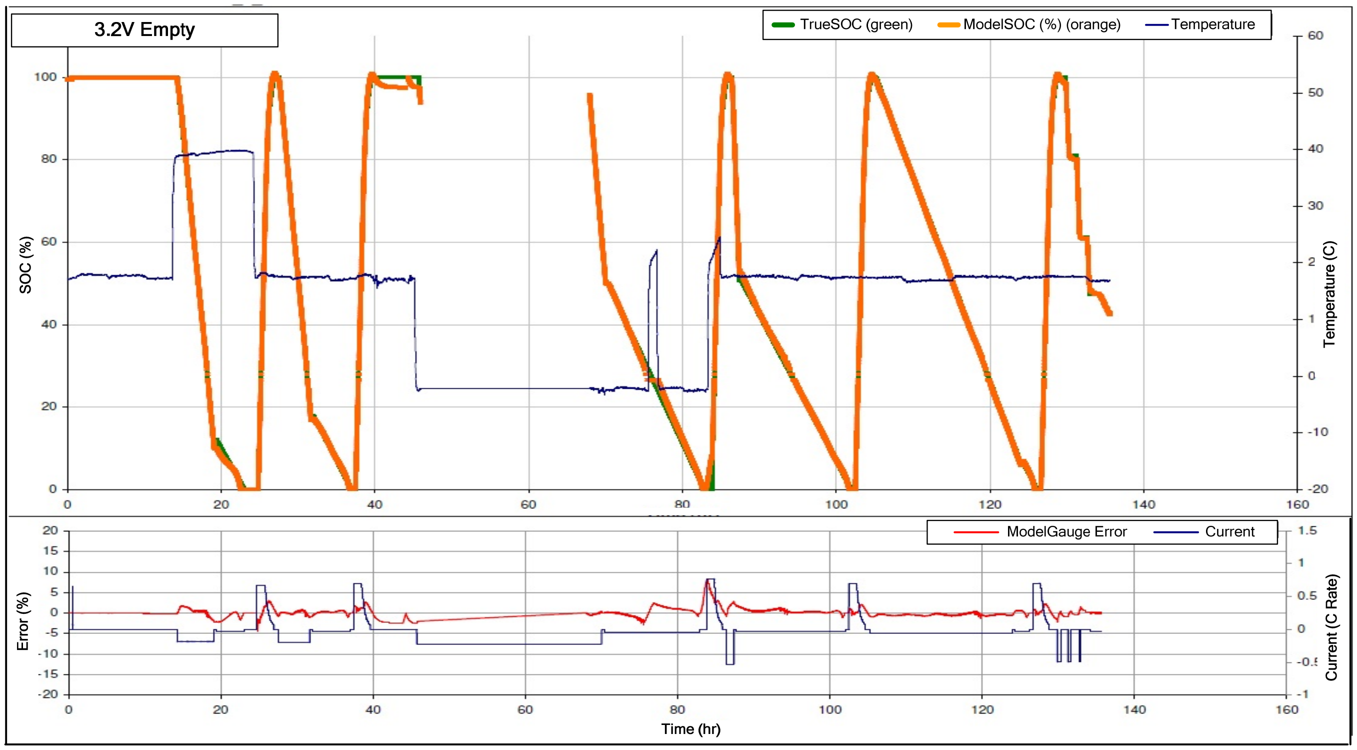


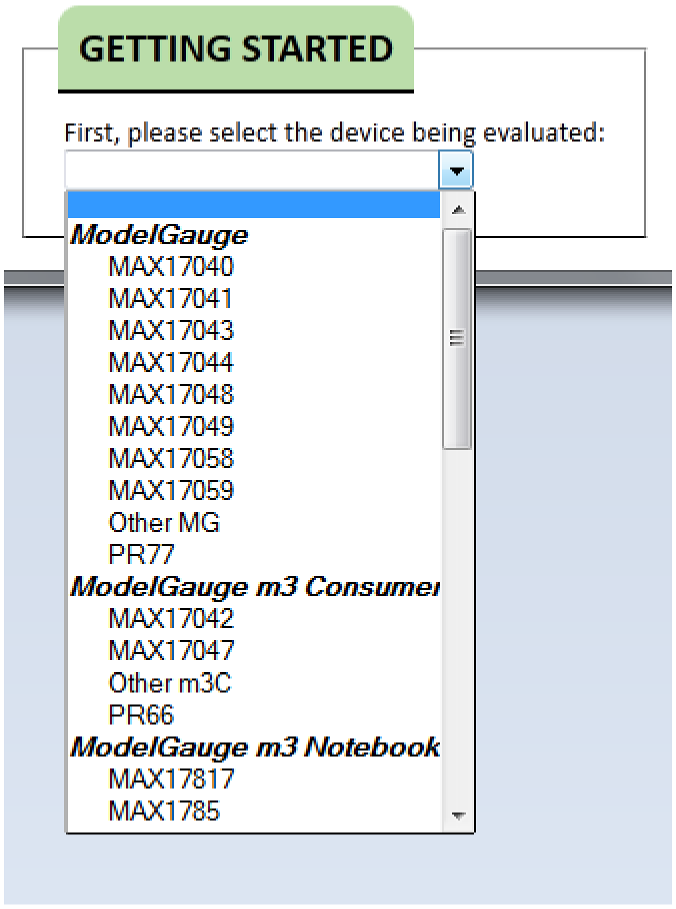
*Figure 9. Example ModelGauge Error versus Empty Voltage*

## 3.3 Battery Characterization Request Form

Correctly pairing an application with a ModelGauge device is critical to accurately estimate the SOC of the battery. Filling out the Characterization Request Form is the first step in this process. Make sure that all parameters are filled out accurately and match the application as closely as possible. Ensure that all of the parameters in the Characterization Form are understood and know how they relate to the application. The online request form shown in *Figure 10* should only be filled out after reading and understanding this entire document.

**Note: At this time the online request form is only accessible to Maxim employees. Contact your local Field Application Engineer to submit a characterization request.**

The online request for can be found here: http://battcave.maxim-ic.com


GETTING STARTED

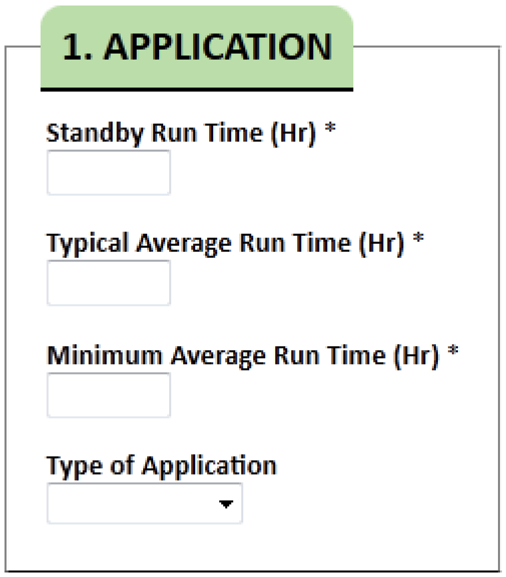
First, please select the device being evaluated:

***ModelGauge***
MAX17040
MAX17041
MAX17043
MAX17044
MAX17048
MAX17049
MAX17058
MAX17059
Other MG
PR77
***ModelGauge m3 Consumer***
MAX17042
MAX17047
Other m3C
PR66
***ModelGauge m3 Notebook***
MAX17817
MAX1785

*Figure 10. Online Cell Characterization Request Page*

### 3.3.1 Request Form Application Section

The first section of the request form as shown in Figure 11 asks for basic information regarding the application. This information will help us sanity check the characterization parameters. Example: A 1000mAh cell with a Typical Active current of 100mA should have a Typical Average Run Time of 10 hours (1000mAh/100mA = 10 hours)



1. APPLICATION

Standby Run Time (Hr) *

Typical Average Run Time (Hr) *

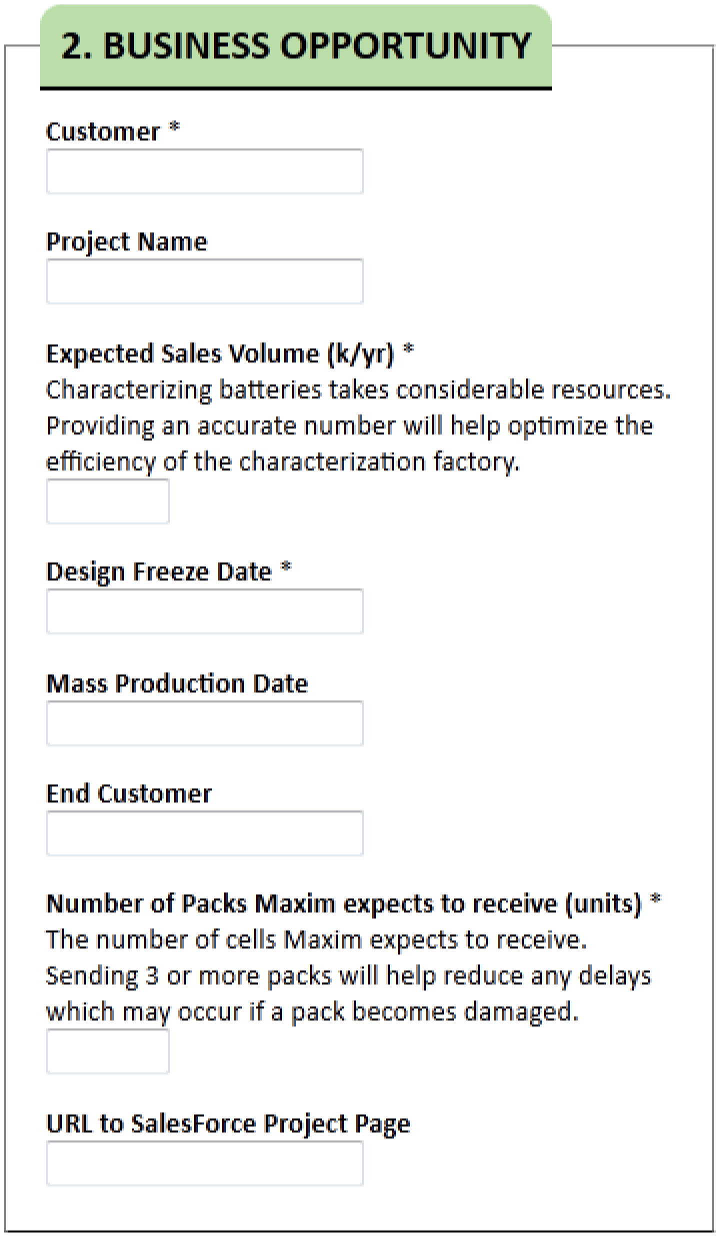
Minimum Average Run Time (Hr) *

Type of Application

*Figure 11. Online Cell Characterization Request Application Section*

### 3.3.2 Request Form Business Opportunity Section

Business Opportunity information as shown in *Figure 12* will help us identify the project and define the projected sales.



**2. BUSINESS OPPORTUNITY**

**Customer** *

**Project Name**

**Expected Sales Volume (k/yr)** *
Characterizing batteries takes considerable resources. Providing an accurate number will help optimize the efficiency of the characterization factory.

**Design Freeze Date** *

**Mass Production Date**

**End Customer**

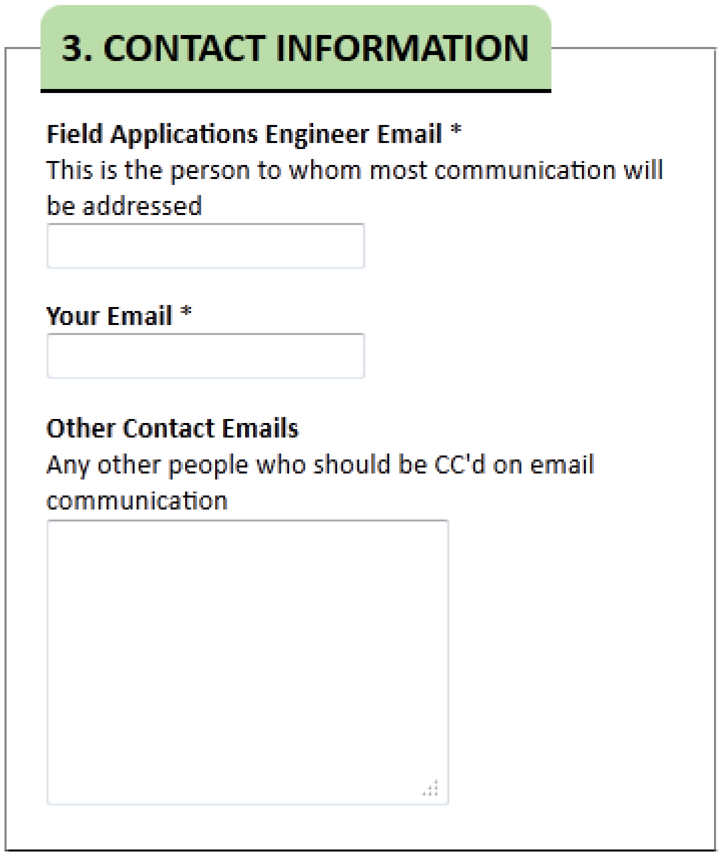
**Number of Packs Maxim expects to receive (units)** *
The number of cells Maxim expects to receive. Sending 3 or more packs will help reduce any delays which may occur if a pack becomes damaged.

**URL to SalesForce Project Page**

*Figure 12. Online Cell Characterization Request Business Opportunity Section*

### 3.3.3 Request Form Contact Information Section

Contact Information section as shown in *Figure 13* will let Maxim know who to contact if there are any questions regarding the characterization parameters and who to send the report to when the project is complete.



3. CONTACT INFORMATION

**Field Applications Engineer Email ***
This is the person to whom most communication will be addressed

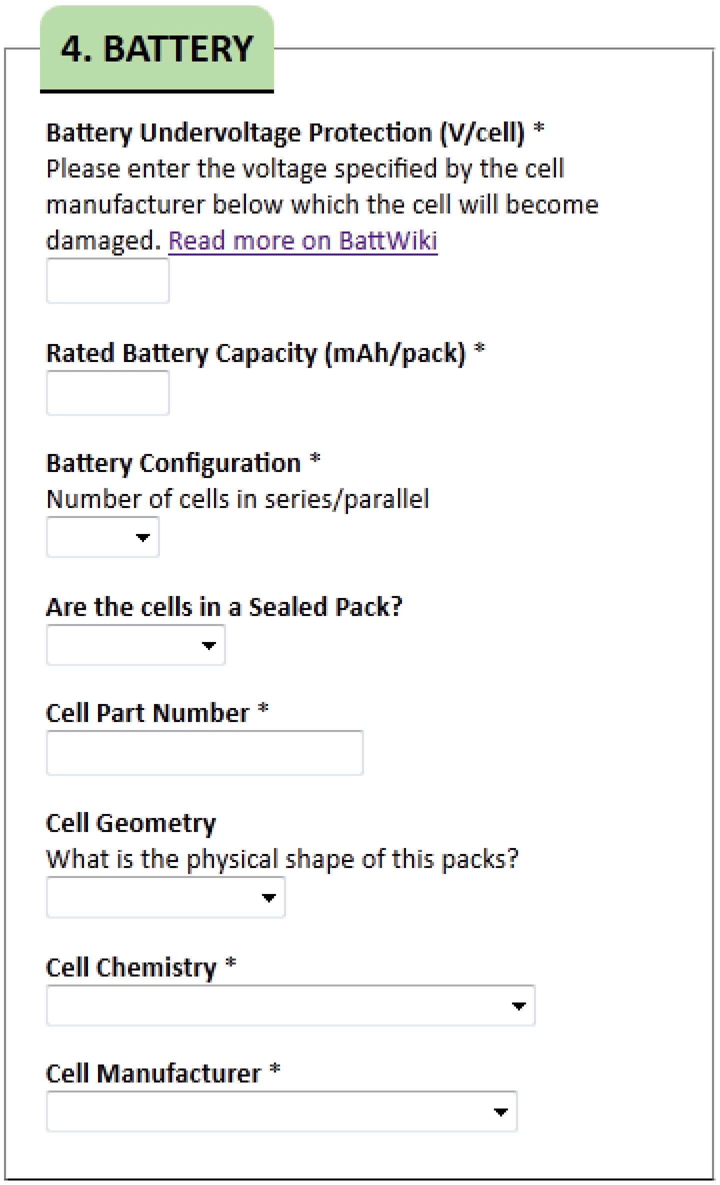
**Your Email ***

**Other Contact Emails**
Any other people who should be CC'd on email communication

*Figure 13. Online Cell Characterization Request Contact Information Section*

### *3.3.4* Request Form Battery Information Section

Battery Information is in regards to the cell itself - cell manufacturer part number and specs. Most of this information can be found in the cell manufacturer data sheet. Figure 14 shows what this field looks like.


**4. BATTERY**

**Battery Undervoltage Protection (V/cell) ***
Please enter the voltage specified by the cell manufacturer below which the cell will become damaged. Read more on BattWiki

**Rated Battery Capacity (mAh/pack) ***

**Battery Configuration ***
Number of cells in series/parallel

**Are the cells in a Sealed Pack?**

**Cell Part Number ***

**Cell Geometry**
What is the physical shape of this packs?

**Cell Chemistry ***

**Cell Manufacturer ***

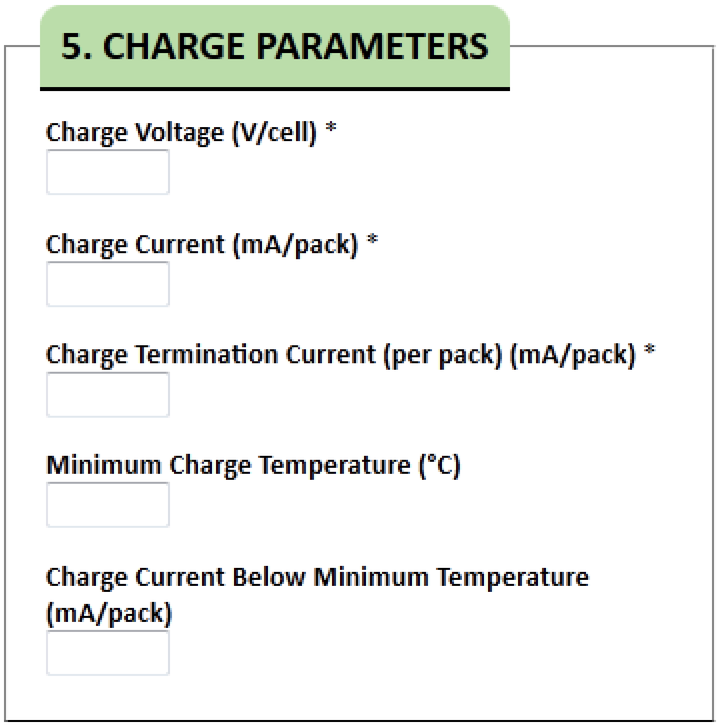
*Figure 14. Online Cell Characterization Request Battery Information Section*

### 3.3.5 Request Form Charge Parameters Section

*Figure 15* shows the Charge Parameters section. These values will set the charge specifications for characterization. Some of the data will be found in the cell manufacturer data sheet and some of it will be defined by the application itself. Examples:

- Charge Current is typically defined by the charger in the application. (This current should be within the specified cell manufacturer range.)
- Minimum Charge Temp and Charge Current Below Minimum Temp are typically found in the cell manufacturer data sheet.

See the Charging the Battery section.



5. CHARGE PARAMETERS

Charge Voltage (V/cell) *

Charge Current (mA/pack) *

Charge Termination Current (per pack) (mA/pack) *

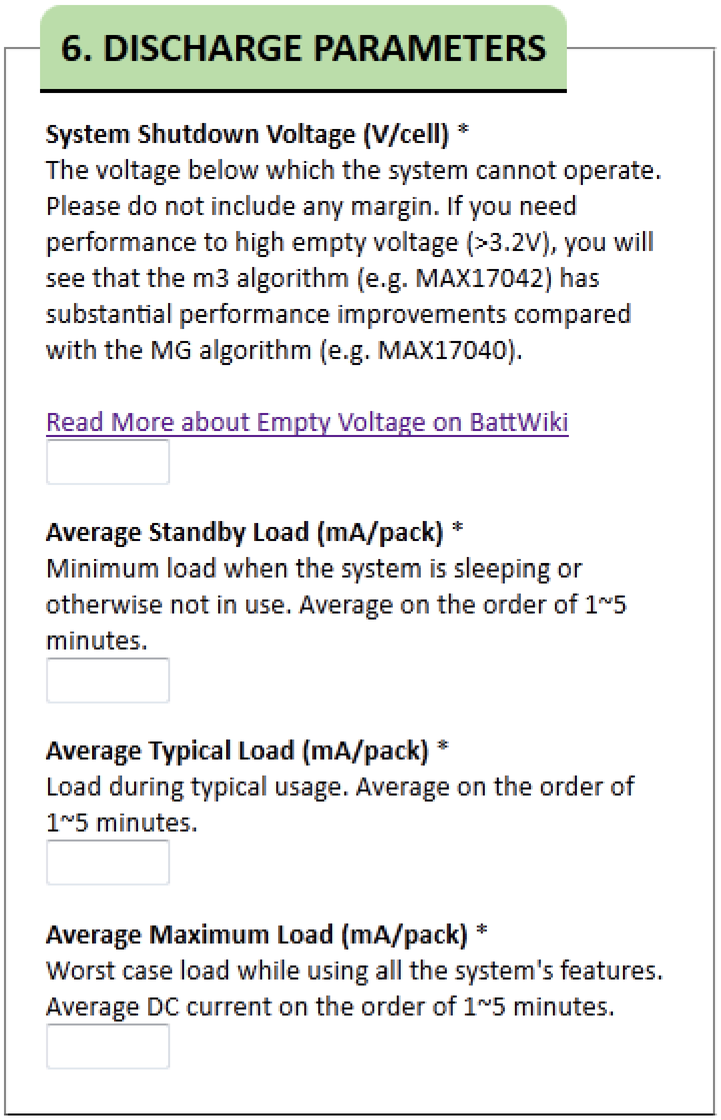
Minimum Charge Temperature (°C)

Charge Current Below Minimum Temperature (mA/pack)

*Figure 15. Online Cell Characterization Request Charge Parameters Section*

### 3.3.6 Request Form Discharge Parameters Section

Discharge Parameters as shown in Figure 16 will set the discharge specifications for characterization. For single cell applications the discharge parameters are typically defined by the minimum operation voltage and loads of the application. See the How Load Affects Empty, Application Loads, and Empty Voltage descriptions.



**6. DISCHARGE PARAMETERS**

**System Shutdown Voltage (V/cell) ***
The voltage below which the system cannot operate. Please do not include any margin. If you need performance to high empty voltage (>3.2V), you will see that the m3 algorithm (e.g. MAX17042) has substantial performance improvements compared with the MG algorithm (e.g. MAX17040).

Read More about Empty Voltage on BattWiki

**Average Standby Load (mA/pack) ***
Minimum load when the system is sleeping or otherwise not in use. Average on the order of 1~5 minutes.

**Average Typical Load (mA/pack) ***
Load during typical usage. Average on the order of 1~5 minutes.

**Average Maximum Load (mA/pack) ***
Worst case load while using all the system's features. Average DC current on the order of 1~5 minutes.

*Figure 16. Online Cell Characterization Request Discharge Parameters Section*

### 3.3.7 Request Form Notes Section

The Notes section is optional. Any additional information or special instructions should be included in this section. See *Figure 17*.

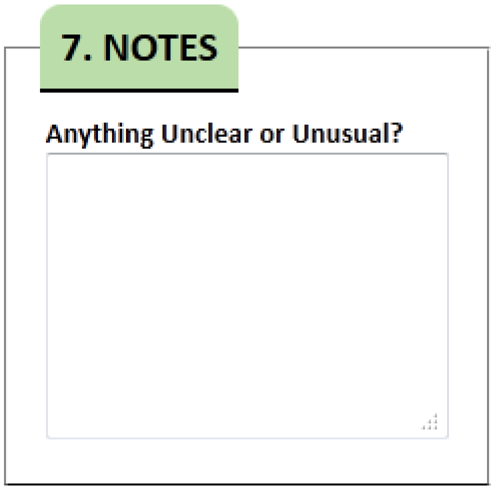


*Figure 17. Online Cell Characterization Request Notes Section*

### 3.3.8 Completing the Characterization Request Form

Staying on schedule is a challenging task with any new design. There are a few things to keep in mind to ensure that the characterization process runs smoothly and the project does not suffer any delays.

- **Missing Parameters** - Fill out the Characterization Request Form fully. Characterization will not be started until all of the required parameters are submitted. Missing parameters will cause the characterization team to have to follow up with the FAE.
- **Unreasonable Parameters** - Make sure that all parameters make sense for the application and the battery. For example, a characterization request for a tablet should not have a 1C discharge rate.
- **Parameter Changes** - Changing the parameters after or during characterization will cause the process to start over.
- **Single Battery Characterization** - Maxim request 3 battery packs for every characterization project submitted. Characterization is performed on 2 batteries and the 3rd is kept as a backup. Sometimes a battery can fail during testing or the data can become corrupted. Characterizing 2 batteries adds redundancy to the process in case of failure. The custom model performance is evaluated on both batteries to ensure that the characteristics are consistent from pack to pack and the model will provide good performance in production.

When you have collected all the information you need, submit the request form and receive your project number. Requests with incomplete information cannot be accepted.

## 3.4 Shipping Packs for Characterization

After the online request form is completed, a project number will be assigned to your project. It is now time to ship cell packs to Maxim for characterization.

### 3.4.1 Project Number

A project number is for reference only and determines no queue order; batteries are not tested sequentially by project number. Please do not provide placeholder information to get a lower project number because the project will neither be queued nor started until we have all the required information.

After your request is accepted, you will get a project number. Please include this number in all correspondence with Maxim, because with the project number the problematic configuration can be compared to that which was intended. It also can confirm that the battery characteristic matches that for which the configuration was prepared.

### 3.4.2 Shipping Instructions

Please ship battery packs including protectors so that their series resistance can be included in the model, which will improve accuracy.

Ship three or four packs so that multiple batteries can be tested. This way a stronger model can be produced and proven on multiple instances of the battery. Having multiple batteries to test along with backups can reduce delays if problems are encountered during testing.

You will receive the shipping address from Maxim after the characterization request has been accepted.

### 3.4.3 Waiting in Queue for Battery Tests

Each project will be placed into the queue, prioritized with respect to business opportunity. Typically a project spends one or two weeks in the queue.

## 3.5 Characterizing the Batteries at Maxim's Lab

Each project will be tested using the parameters from the request form (e.g. charge voltage, charge current, empty voltage, load currents). The batteries are connected to a tester and then placed in a temperature-control chamber (oven). The batteries are tested by charging and discharging at the voltages and loads on the request form. They will be tested at heavy, medium, and light loads for each hot, room and cold temperature. *Figure 18* shows a log of capacity versus voltage for a complete characterization cycle.

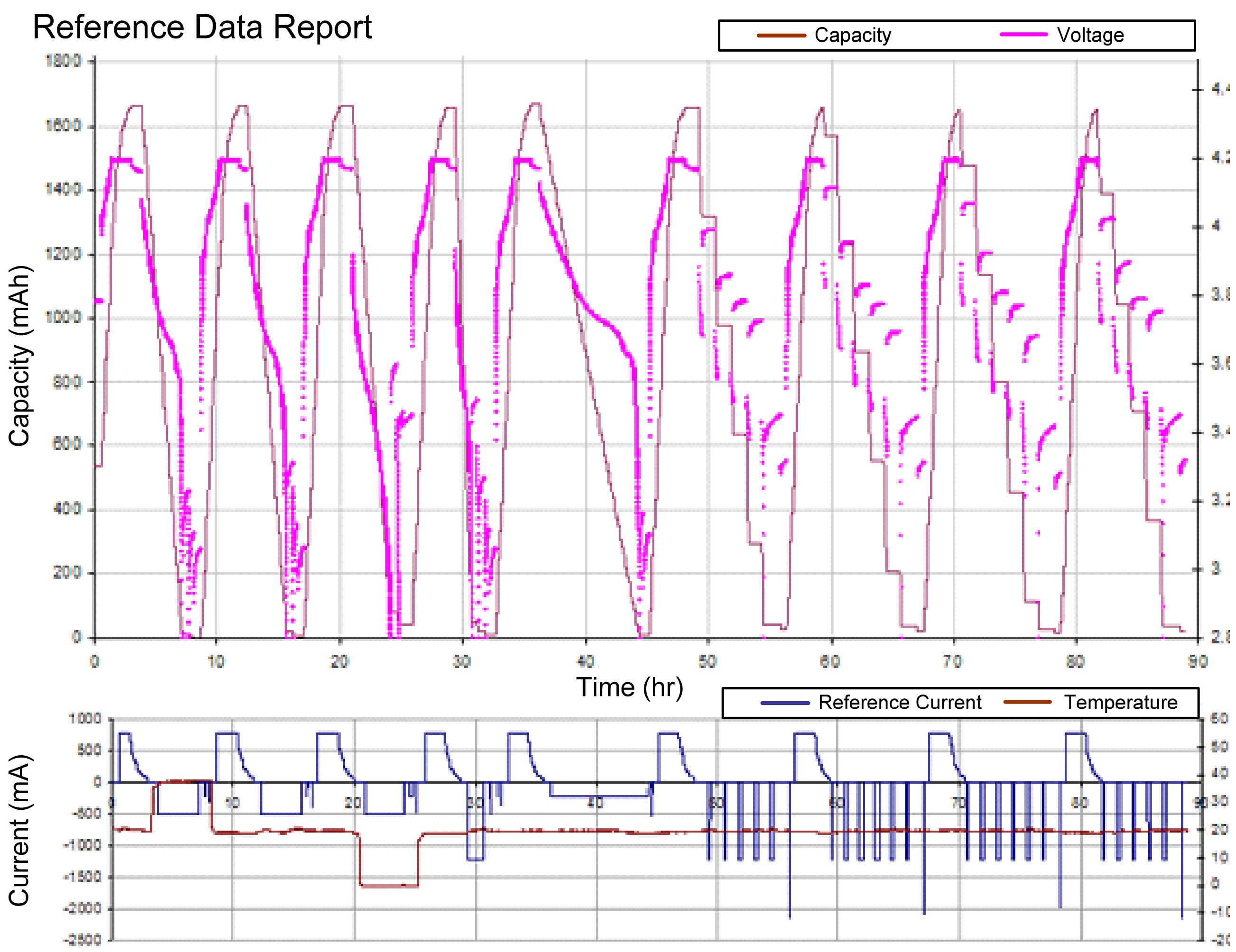


*Figure 18. Example Cell Characterization Cycle*

The ModelGauge characterization procedure is listed below in *Figure 19*. All discharge cycles are followed by a charge to full at 20C and the test pattern will begin with a charge to full at 20C.

**Characterization Procedure**

1. Charge Room (20C)
2. Discharge Medium Load Hot (40C)
3. Charge Room (20C)
4. Discharge Medium Load Room (20C)
5. Charge Room (20C)
6. Discharge Medium Load Cold (0C)

7. Charge Room (20C)
8. Discharge Heavy Load Room (20C)
9. Charge Room (20C)
10. Discharge Light Load Room (20C)
11. Charge Room (20C)
12. Discharge OCV1 Room (20C)

13. Charge Room (20C)
14. Discharge OCV2 Room (20C)
15. Charge Room (20C)
16. Discharge OCV3 Room (20C)
17. Charge Room (20C)
18. Discharge OCV4 Room (20C)

19. Charge Room (20C)

*Figure 19. Characterization Procedure*

## 3.6 Characterization Results

A configuration (custom model) specific to the battery is created from the characterization data using Maxim characterization analysis software. The results are placed in a configuration INI file. The INI file is compatible with the EV Kit software for performance evaluation. A performance report is also provided to show expected performance in the application.

### 3.6.1 Performance Report

The performance report is created and emailed to the FAE along with the configuration INI file. This .pdf file is an easy way to see how the fuelgauge part performed in our tests using the configuration INI file. It will show battery voltage, current, temperature, Model Gauge reported SOC, reference SOC, and error. It is usually 5-6 days of continuous data showing each charge and discharge cycle at different loads and temperatures.

*Figure 20* shows an example of one page of a performance report.

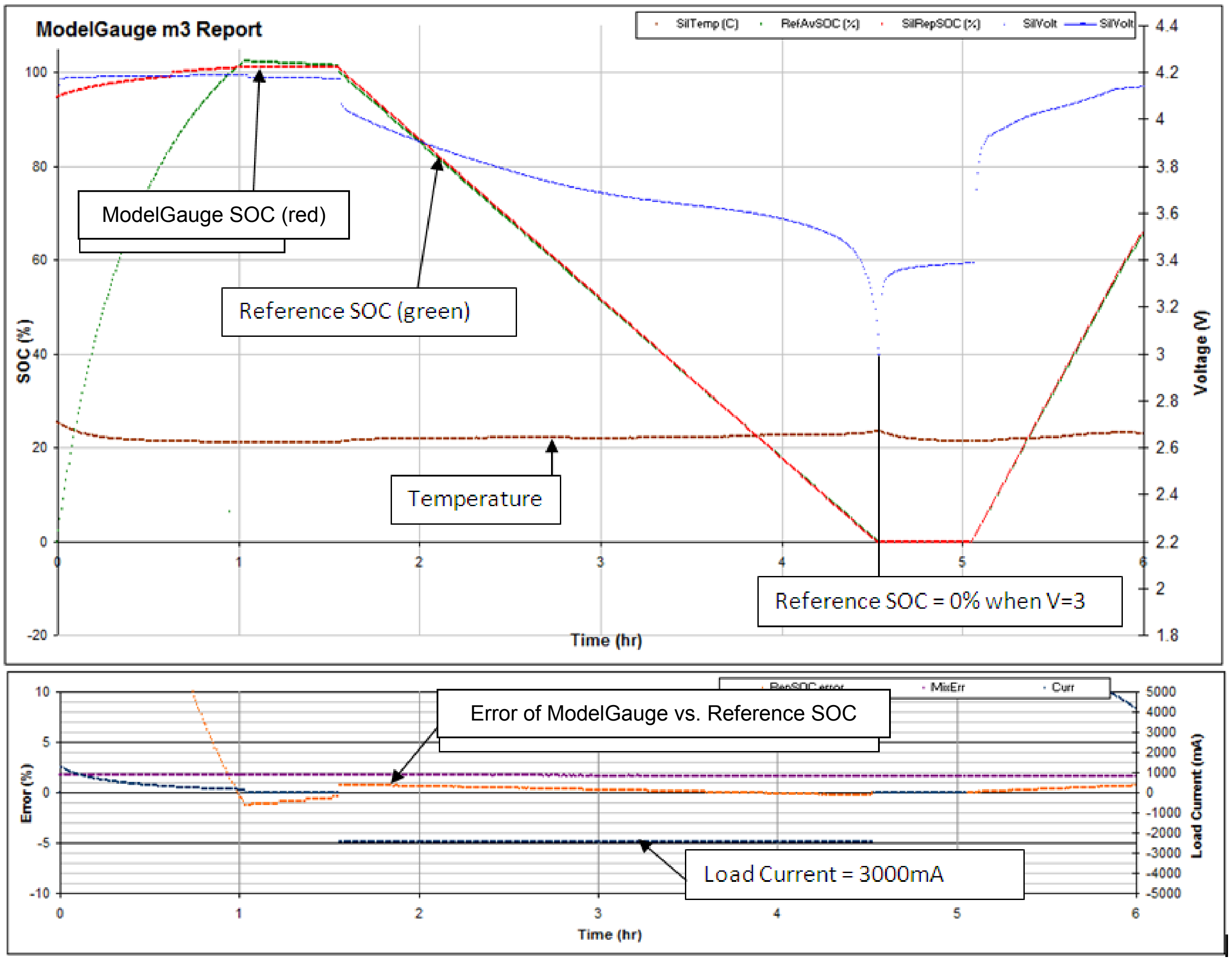


*Figure 20. Example Performance Report*

## 3.6.2 Configuration INI File

This file contains values for many different registers which can vary widely. Only the one particular combination of values in this file will give the expected performance. This file contains the OCV-SOC table, RCOMP, TempCo, and more, all of which are necessary to properly configure the fuelgauge. The INI file is formatted as shown in *Figure 21*.

- The values marked with 0x## will be unique to each INI file. This is a hexadecimal number which may or may not contain the string "0x", e.g. 0x00 or FF. These are all 8 bit values.

- The values marked with ## will be unique to your file. This is a decimal number, e.g. 2.345
- Text marked with %% will be unique to your file. This will be text, and should not be interpreted as any number.

| INI File Contents | Description |
|---|---|
| Device = %% | The name of the device which this configuration is for. Configurations are compatible between all ModelGauge devices |
| Title = %% | Project title used internally at Maxim for identification |
| EmptyAdjustment = ## | |
| FullAdjustment = ## | |
| RCOMP = ## | Starting RCOMP value |
| TempCoUp = ## | Temperature (hot) coeffiecient for RCOMP. Used in UPDATE RCOMP step |
| TempCoDown = ## | Temperature (cold) coeffiecient for RCOMP. Used in UPDATE RCOMP step. |
| OCVTest = ## | OCV Test value in decimal. Used in step 7 |
| SOCCheckA = ## | SOCCheck low value. Used to verify model |
| SOCCheckB = ## | SOCCheck high value. Used to verify model |
| bits = ## | 18 or 19 bit model. See Calculating SOC for details. |
| 0x##<br>...<br>0x## | 32 bytes used for EVKit software only. Discard this data |
| 0x##<br>...<br>0x## | 64 bytes of Model Data begin here. Write these bytes in this order to the first table address at 0x40h. |
| 0x##<br>...<br>0x## | 32 bytes used for EVKit software only. Discard this data |

*Figure 21. Configuration INI File Format*

## 3.7 Hands-on Customer Testing and Customer Feedback

To ensure that problems are debugged before mass-production, the customer should repeat the test and verify performance. See Performance Verification Section for instructions.

# 4 USING MULTIPLE CELL TYPES IN AN APPLICATION

For each battery type used in a given application, a separate characterization should be performed, and the application should load the appropriate model after battery insertion.

There are a few strategies for supporting multiple battery types on the same device in production. The battery should be tested through the normal characterization process for any of these approaches.

## 4.1 Middle Model

After characterization of battery A the performance for battery B can be simulated using model A. The performance will either be good or bad, depending on the similarity of the chemistry. If model A is insufficient to perform on battery B, then it may be possible to develop a middle model to support both batteries. A performance report will show the resulting performance of the middle model on both batteries. If the performance is still insufficient, then consider the "Multiple Models" option listed below.

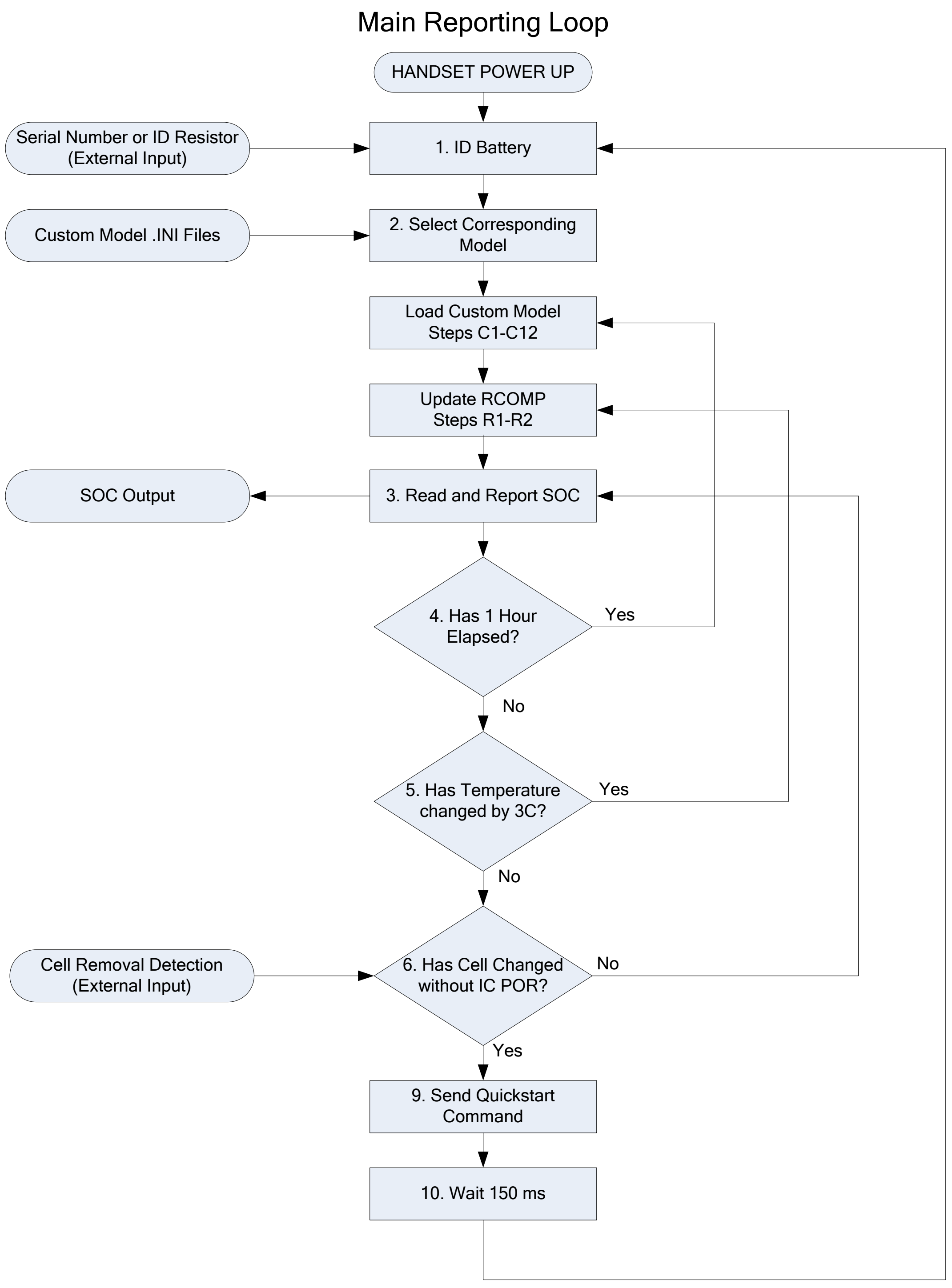
## 4.2 Multiple Models

When a middle model is insufficient, multiple models optimized to each battery type will need to be used. This approach requires the system to identify which battery is installed, and configures the fuelgauge with the appropriate model after battery insertion. To achieve this, the fuelgauge driver should support the following:

1. Nonvolatile memory to store the model parameters associated with each battery.
2. The ability to determine which battery type is installed.

## 5 APPLICATION SOFTWARE IMPLEMENTATION GUIDELINES

This section provides step by step instructions for implementation of all software required to run ModelGauge devices in an application. *Figure 22* provides a flow chart overview of all the steps required to run a ModelGauge device.



Main Reporting Loop
HANDSET POWER UP
Serial Number or ID Resistor (External Input)
1. ID Battery
Custom Model .INI Files
2. Select Corresponding Model
Load Custom Model Steps C1-C12
Update RCOMP Steps R1-R2
SOC Output
3. Read and Report SOC
4. Has 1 Hour Elapsed?
Yes
No
5. Has Temperature changed by 3C?
Yes
No
Cell Removal Detection (External Input)
6. Has Cell Changed without IC POR?
No
Yes
9. Send Quickstart Command
10. Wait 150 ms

*Figure 22. ModelGauge Application Software*

## 5.1 Battery Identification

In systems with only a single cell type, battery identification is not required. In systems with multiple cell types, the host must determine which cell is connected so the proper cell model can be loaded into the IC. There are multiple ways to accomplish this.

### 5.1.1 Captive battery

Battery may be recognized by board identifier or resistor or something installed on the board together with that battery. If the installation is done in a temperature controlled environment, using different thermistors in different packs can be used to identify one pack versus another. Alternatively, the production line could have a way for choosing which flash data to install.

### 5.1.2 Removable battery

Battery could be recognized by any one of the following:

1. Battery security verification. Using a DS2708 for example.
2. Battery ID stored in non-volatile memory such as a DS2502 or DS2708.
3. Thermistor. For example, 10KΩ thermistor installed in battery A and 100KΩ thermistor in battery B. Use significantly different thermistor values to prevent ambiguity between battery ID and battery temperature.
4. Identifying resistor.

## 5.2 Custom Model Selection

In a single cell type application there will only be one model available to load into the IC. If the application uses multiple cells types and a “middle model" does not produce acceptable results, individual custom models for each cell type will be used. Tthe host must identify the cell and load the corresponding model (LoadCustomModel is the same procedure described in detail in the following section).

```
// Procedure to select proper model in multiple cell type applications

CellType = HostIdentifyCellType();

if(CellType == SourceA)
   LoadCustomModel(SourceA);

else if(CellType == SourceB)
   LoadCustomModel(SourceB);

else if(CellType == SourceC)
   LoadCustomModel(SourceC);
```

## 5.3 Data Logging

Consider adding a data logging feature to the main application software to help with debugging. See section 9 for a description on data logging.

## 5.4 Loading the Custom Model

The custom model that is generated by cell characterization is stored in RAM in the ModelGauge IC. The model must be written to the IC any time that power is applied or restored to the device. This generally occurs during battery insertion, and is often required during system boot-up/reboot. After the battery is inserted, the host software must unlock write access to the model, write the model, verify that the model was written properly, and then lock access to the model.

The device does not allow the user to read the model directly. In order to verify the model, the host should write a value beyond the greatest OCV point from the model into the OCV register (memory location 0x0Eh-0x0Fh) and verify that the SOC register (memory location 0x04h-0x05h) matches the desired SOC value from the model. Before updating the OCV register, the host should read the OCV register, so that the original value can be rewritten to the device after the model is verified.

It is also recommended to periodically re-load the model in case an event occurred that might corrupt the custom model. The model can be refreshed once per hour. *Figure 23* details the custom model loading procedure.

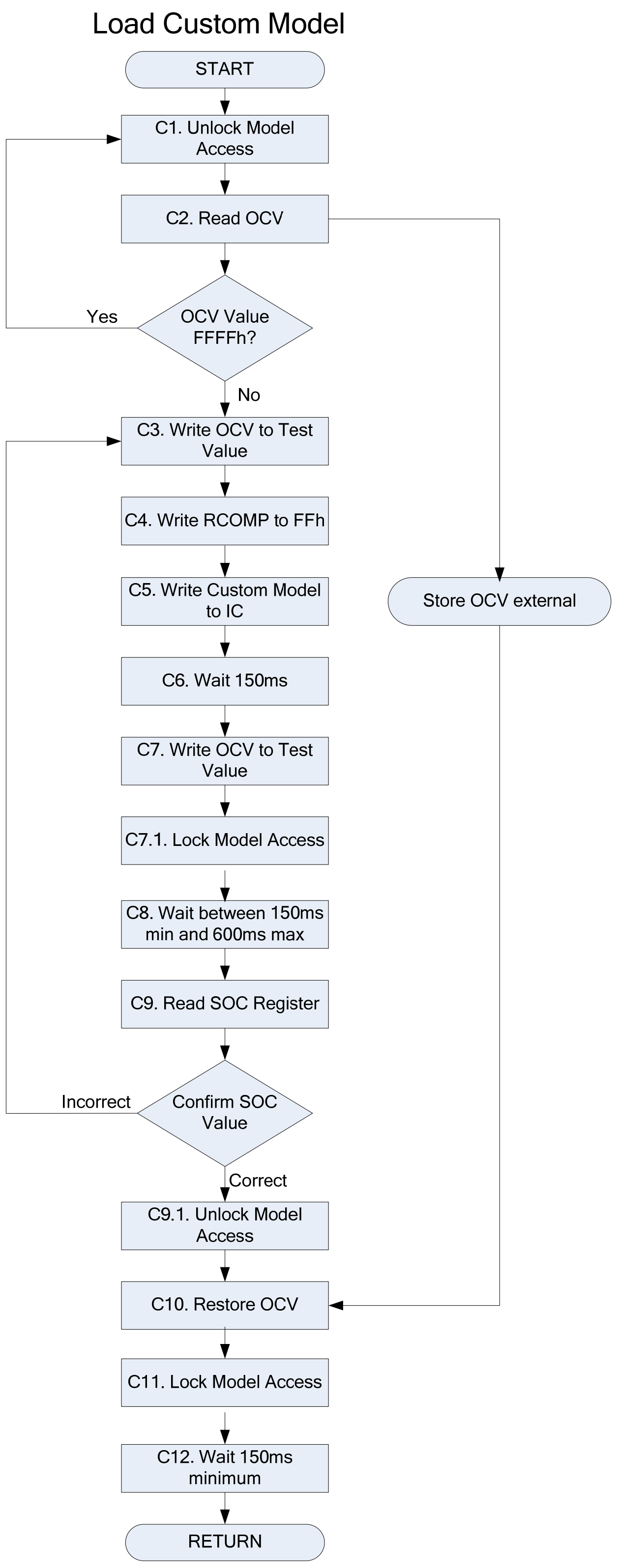


*Figure 23. Flowchart for Loading a Custom Model into a ModelGauge IC*

```
// Step 1.  Unlock Model Access

//   This enables access to the OCV and table registers

WriteWord(0x3E, 0x4A, 0x57);
```

```
// Step 2.  Read OCV

//   The OCV Register will be modified during the process of loading the custom
//   model.  Read and store this value so that it can be written back to the
//   device after the model has been loaded.

Byte original_OCV_1, original_OCV_2;
ReadWord(0x0E, &original_OCV_1, &original_OCV_2);
```

```
// Step 2.5.  Verify Model Access Unlocked

//   If Model Access was correctly unlocked in Step 1, then the OCV bytes read
//   in Step 2 will not be 0xFF. If the values of both bytes are 0xFF,
//   that indicates that Model Access was not correctly unlocked and the
//   sequence should be repeated from Step 1.

if((original_OCV_1 == 0xFF) && (original_OCV_2 == 0xFF))
{
    //Goto Step 1
}
```

```
// Step 3.  Write OCV (MAX17040/1/3/4 only)

//   Find OCVTest_High_Byte and OCVTest_Low_Byte values in INI file

WriteWord(0x0E, INI_OCVTest_High_Byte, INI_OCVTest_Low_Byte);
```

```
// Step 4. Write RCOMP to its Maximum Value (MAX17040/1/3/4 only)

//   Make the fuel-gauge respond as slowly as possible (MSB = 0xFF), and disable
//   alerts during model loading (LSB = 0x00)

WriteWord(0x0C, 0xFF, 0x00);
```

```
// Step 5. Write the Model

//   Once the model is unlocked, the host software must write the 64 byte model
//   to the device.  The model is located between memory 0x40 and 0x7F.
//   The model is available in the INI file provided with your performance
//   report.  See the end of this document for an explanation of the INI file.

//    Note that the table registers are write-only and will always read
//    0xFF. Step 9 will confirm the values were written correctly.

byte model_data[64] =
{
   // Fill in your model data here from the INI file
   // 0x##, 0x##, ..., 0x##
}
for(int i = 0; i < 64; i += 16)
{
    // Perform these I2C tasks:
       START
       Send Slave Address (0x6Ch)
       Send Memory Location (0x40 + i)

       for(int k = 0; k < 16; k++)
       {
           Send Data Byte (model_data[k])
       }
       I2C STOP
}
```

```
// Step 6. Delay at least 150ms (MAX17040/1/3/4 only)

//   This delay must be at least 150mS, but the upper limit is not critical
//   in this step.

sleep_ms(150);
```

```
// Step 7. Write OCV

//   This OCV should produce the SOC_Check values in Step 9

WriteWord(0x0E, INI_OCVTest_High_Byte, INI_OCVTest_Low_Byte);
```

```
// Step 7.1 Disable Hibernate (MAX17048/49/ only)

//    The IC updates SOC less frequently in hibernate mode, so make sure it
//    is not hibernating

WriteWord(0x0A, 0);
```

```
// Step 7.2. Lock Model Access (MAX17048/49/58/49 only)

//   To allow the ModelGauge algorithm to run in MAX17048/9 only, the model must
//   be locked. This is harmless but unnecessary MAX17040/1/3/4

WriteWord(0x3E, 0, 0);
```

```
// Step 8. Delay between 150ms and 600ms

//   This delay must be between 150ms and 600ms.  Delaying beyond 600ms could
//   cause the verification to fail.

sleep_ms(150);
```

```
// Step 9. Read SOC Register and compare to expected result

//   There will be some variation in the SOC register due to the ongoing
//   activity of the ModelGauge algorithm.  Therefore, the upper byte of the SOC
//   register is verified to be within a specified range to verify that the
//   model was loaded correctly. This value is not an indication of the state of
//   the actual battery.

byte SOC_1, SOC_2;
ReadWord(0x04, &SOC_1, &SOC_2);
if(SOC_1 >= INI_SOCCheckA && SOC_1 <= INI_SOCCheckB)
{
    // model was loaded successfully
}
Else
{
    // model was NOT loaded successfully
}
```

```
// Step 9.1. Unlock Model Access (MAX17048/9 only)

//   To write OCV, MAX17048/9 requires model access to be unlocked.

WriteWord(0x3E, 0x4A, 0x57);
```

```
// Step 10.  Restore CONFIG and OCV

//   It is up to the application how to configure the LSB of the CONFIG
//   register; any byte value is valid.

WriteWord(0x0C, StartingRCOMP, Your_Desired_Alert_Configuration);
WriteWord(0x0E, original_OCV_1, original_OCV_2);
```

```
// Step 10.1 Restore Hibernate (MAX17048/49/58/59 only)

//    Remember to restore your desired Hibernate configuration after the
//    model was verified.

Restore your desired value of HIBRT
```

```
// Step 11. Lock Model Access

WriteWord(0x3E, 0x00, 0x00);
```

```
// Step 12. Delay at least 150ms

//   This delay must be at least 150mS before reading the SOC Register to allow
//   the correct value to be calculated by the device.

sleep_ms(150);
```

## 5.5 Update RCOMP

As shown in *Figure 24*, the RCOMP register should be updated when the temperature of the cell changes (by more than 3 degrees in most cases) or any time power is applied to the device, as this register is also a RAM location.

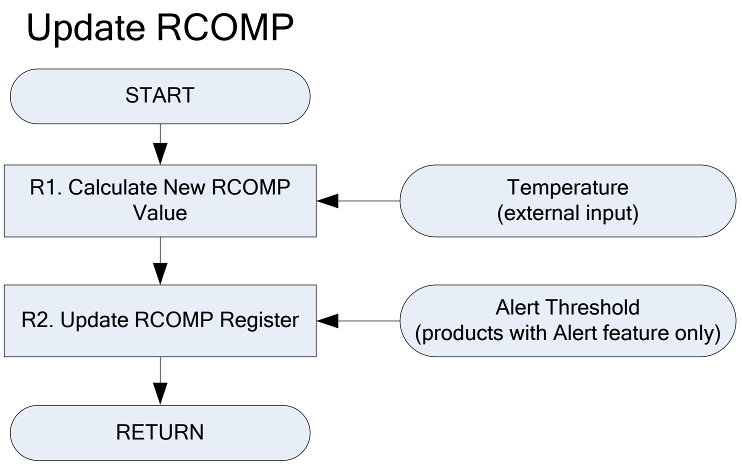

Update RCOMP
START
R1. Calculate New RCOMP Value
Temperature (external input)
R2. Update RCOMP Register
Alert Threshold (products with Alert feature only)
RETURN

*Figure 24. Flowchart for Updating RCOMP*

In this example, the custom RCOMP value at 20°C is stored in the variable StartingRCOMP. Characterization of the cell will indicate that the RCOMP value should be increased by a value of TempCoCold for every degree the temperature is below 20ºC and decreased by TempCoHot for every degree the temperature is above 20°C. The maximum RCOMP value is 255 and the minimum RCOMP value is 0. The new RCOMP is calculated as follows:

```
// RCOMP value at 20 degrees C
int StartingRCOMP;

// RCOMP change per degree for every degree above 20C (TempCoUp_from_INI_file)
int TempCoHot;

// RCOMP change per degree for every degree below 20C (TempCoDown_from_INI_file)
int TempCoCold;

if(Temperature > 20)
{
    NewRCOMP = StartingRCOMP + ((Temperature - 20) * TempCoHot);
}
else if(Temperature < 20)
{
    NewRCOMP = StartingRCOMP + ((Temperature - 20) * TempCoCold);
}
else
{
    NewRCOMP = StartingRCOMP;
}

// Clamp RCOMP to a range from 0x00 to 0xFF
if(NewRCOMP > 0xFF)
{
    NewRCOMP = 0xFF;
}
else if(NewRCOMP < 0)
{
    NewRCOMP = 0;
}
```

Once the new RCOMP value is calculated, write the value to the RCOMP register (memory location 0x0Ch). Memory Location 0x0Dh must always be written or RCOMP will not be updated. It is up to the application how to configure the LSB of the CONFIG register; any byte value is valid.

```
// Update RCOMP/Config Register

WriteWord(0x0C, NewRCOMP, Your_Desired_Alert_Configuration);
```

## 5.6 Reading and Calculating SOC

The custom model for this cell may require a change in the LSB value from the datasheet specification. Nominally, the LSB has a value of $2^{-8}$, but for some custom models, the LSB has to be shifted by one bit so that it now has a value of $2^{-9}$. Note that for an 18-bit model, in the upper byte 1LSb = 1% SOC.

```
// Report State of Charge of the Cell

byte SOC_1, SOC_2;
float SOC_percent;
ReadWord(0x04, &SOC_1, &SOC_2);

if(INI_bits == 18)
{
    SOC_percent = ((SOC_1 << 8) + SOC_2) / 256;
}
else if(INI_bits == 19)
{
    SOC_percent = ((SOC_1 << 8) + SOC_2) / 512;
}
```

## 5.7 Verify Model

ModelGauge devices store the custom model parameters in RAM. The RAM data can be corrupted in the event of a power loss, brown out or ESD event. It is good practice to occasionally verify the model and reload if necessary. Maxim recommends doing this once per hour while the application is active. The following example shows how to verify the model. Alternatively the model can simply be reloaded once per hour without verification.

```
// Write RCOMP and OCV test values to verify model memory

MAX17040WriteWord (0x3E, 0x574A)       //Unlock Model Access
RCOMP0 = MAX17040ReadWord (0x0C)       //Read Original RCOMP
OriginalOCV = MAX17040ReadWord (0x0E)  //Read Original OCV Register
MAX17040WriteWord (0x0E, OCVTest)      //Write OCV Register
MAX17040WriteWord (0x0C, RCOMP0)       //Write RCOMP Register

//This delay must be between 150mS and 600mS. Delaying beyond 600mS could cause the
verification to fail.

Delay (150ms)                        //Delay between 150mS and 600mS

SOC_Check = MAX17040ReadWord(0x04)   //Read SOC Register and Compare to expected
result
if (SOC > SOC_CheckA and SOC <= SOC_CheckB)  //Verify SOC is within expected range
{
   MAX17040WriteWord (0x0C, RCOMP0);
   MAX17040WriteWord (0x0D, OriginalOCV);
}
else
{
   ReloadModel();
}
MAX17040WriteWord (0x3E, 0x0000)     //Lock Model Access
```

## 5.8 MODELGAUGE BATTERY INSERTION HANDLING

Battery insertion is a crucial event for any system-side fuelgauge. Most ModelGauge ICs are located on the system side of the application - meaning when the battery is removed from the application the fuel gauge loses power. The ModelGauge algorithm requires a history of data to obtain the most accurate State-of-Charge (SOC) estimation. Upon battery insertion the device has no data history. ModelGauge relies on the initial voltage reading at power up in order to determine the initial SOC estimation. This makes the accuracy of the initial voltage reading a critical measurement.

### 5.8.1 Initial Battery State Estimation

The initial voltage reading of the battery after a ModelGauge device powers up is used to estimate the State-of-charge (SOC). This voltage reading bypasses the ModelGauge algorithm and is set as the Open Circuit Voltage (OCV is defined as the voltage of the battery in a completely relaxed state; meaning the voltage is not relaxing, settling/changing from a charge or discharge event.) OCV is used to estimate initial SOC based on a look-up table.

Using the initial voltage reading to estimate the battery state is not without risk. However, without any knowledge of the SOC it is the best approach. The following issues could affect the accuracy of the initial SOC estimation.

1. **The Battery was recently charged.** The battery may have been pulled off of a charger before being inserted into the system. In this case the voltage will be higher than the relaxed voltage, and ModelGauge will estimate SOC high.
2. **The Battery was recently discharged.** The battery might have had a load from another device, or potentially the same device - pulled out of an active phone, and re-inserted into the phone. In this case the voltage will be lower than the relaxed voltage, and ModelGauge will estimate SOC low.
3. **Large in-rush current occurs during the initial voltage sampling period.** A large in-rush of current during the initial voltage sampling period can cause a dip in the voltage of the battery. The first voltage reading of the ModelGauge device will be lower than the OCV resulting in a lower SOC estimation.
4. **The ModelGauge device remains powered when the cell is removed.** If the ModelGauge device remains powered by an external supply, such as a charger or decoupling capacitors, when the cell is removed, then the device will continue to measure the voltage although the cell is not present. The device will continue to measure the incorrect voltage resulting in a corrupted SOC estimation. Depending on the circuit, the voltage could be reading 0V, or the charger voltage, or some other voltage. Regardless, these are not valid voltage readings.
5. **Voltage bounces from physical cell insertion.** When a battery is physically inserted into an application the system voltage can bounce due to the contacts connecting/disconnecting. The MAX17048/9 has superior rejection of battery insertion glitches.

In the case where the initial voltage reading is not a true representation of the OCV the initial SOC estimation will have error. However, this error will slowly correct over time. (See the Initial SOC Healing section for an example of this correction.)

### 5.8.2 Initial Voltage Reading Behavior

The MAX17040/1/3/4 and MAX17048/9 manage battery insertion and debouncing differently. The MAX17040/1/3/4 ModelGauge ICs use a sigma-delta ADC, which takes 125ms after power-up to complete the first conversion. The ADC readings are normally 12-bits. However the first sample is acquired with reduced 10-bit resolution in order to speed up the conversion. The first sample is the average voltage over the 125ms sampling period. The initial SOC estimation is available about 300ms after power is applied.

The MAX17048/9 takes 16 voltage samples at power up to determine initial voltage reading. OCV is ready 17ms after battery insertion. The MAX17048/9 provides the following enhancements for handling battery insertion.

1. **Fast first conversions.** The ADC conversions take 1ms each. The conversion packets are single-sample only.

2. **16 burst debouncing.** 16 1ms samples are taken. It takes a total of 17ms to complete the debounce routine. The maximum of the 16 samples is taken as the initial OCV. **Note:** Normal conversions are the average of 4 sequential 1ms samples. However, there is no averaging during debounce; the first OCV is the max of the 16 different 1ms samples.

3. **Battery removal comparator.** The battery is considered removed if the battery voltage drops below an adjustable VRESET threshold. Upon the restoration of the battery voltage, the fuel gauge begins its insertion routine.

4. **12-bit resolution for initial OCV.** The ADC uses 12-bit resolution for initial voltage measurements.

### 5.8.3 Correcting Corrupted Initial Voltage Reading

This method is normally not necessary for the MAX17048/49. In the event that the initial voltage reading is not an accurate representation of the OCV it may be beneficial to try to obtain a more accurate initial SOC estimation. A strong general approach is to utilize the maximum of various voltage measurements:

1. The OCV that is already in the part.

2. The first VCELL seen by the IC.

3. The 2nd VCELL seen by the IC.

Choose the maximum of these measurements, and load that value as OCV.

```
// Variables needed for updating initial voltage reading

byte Volt_MSB, Volt_LSB;
float VCELL1, VCELL2, OCV, Desired_OCV;
```

```
// Step 1.  Read First VCELL Sample

ReadWord(0x02, &Volt_MSB, &Volt_LSB);
VCell1 = ((Volt_MSB << 8) + Volt_LSB);
```

```
// Step 2.  Delay 125ms
// Delay at least 125ms to ensure a new reading in the VCELL register.

delay_ms(125);
```

```
// Step 3.  Read Second VCELL Sample

ReadWord(0x02, &Volt_MSB, &Volt_LSB);
VCell2 = ((Volt_MSB << 8) + Volt_LSB);
```

```
// Step 4.  Unlock Model Access

// To unlock access to the model the host software must write 0x4Ah to memory
// location 0x3E and write 0x57 to memory location 0x3F.
// Model Access must be unlocked to read and write the OCV register.

WriteWord(0x3E, 0x4A, 0x57);
```

```
// Step 5.  Read OCV

ReadWord(0x0E, &Volt_MSB, &Volt_LSB);
OCV = ((Volt_MSB << 8) + Volt_LSB);
```

```
// Step 6.  Determine maximum value of VCell1, VCell2, and OCV

if((VCell1 > VCell2) && (VCell1 > OCV))
{
    Desired_OCV = VCell1;
}
else if((VCell2 > VCell1) && (VCell2 > OCV)
{
    Desired_OCV = Vcell2;
}
Else
{
    Desired_OCV = OCV;
}
```

```
// Step 7.  Write OCV

WriteWord(0x0E, Desired_OCV >> 8, Desired_OCV & 0xFF);
```

```
// Step 8. Lock Model Access

WriteWord(0x3E, 0x00, 0x00);
```

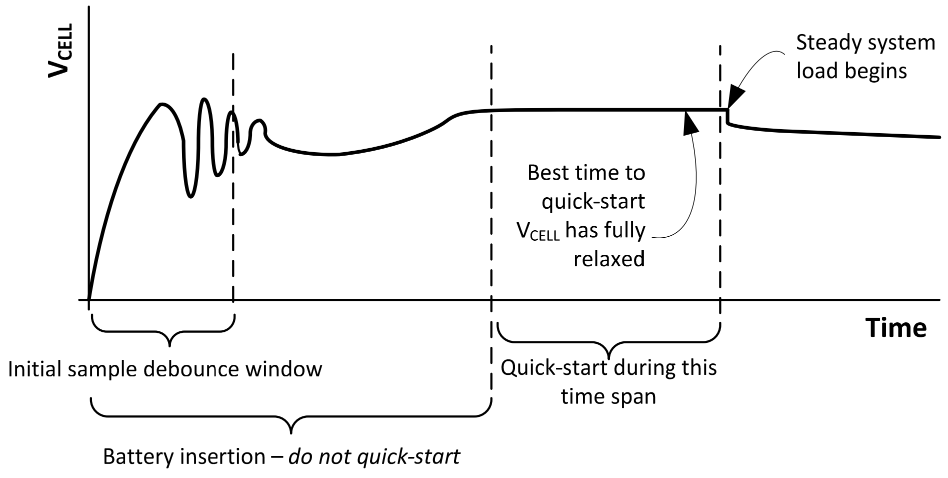
```
// Step 9.  Delay 125ms

// This delay must be at least 150mS before reading the SOC Register to allow
// the correct value to be calculated by the device.

delay_ms(125);
```

### 5.8.4 Quick-Start

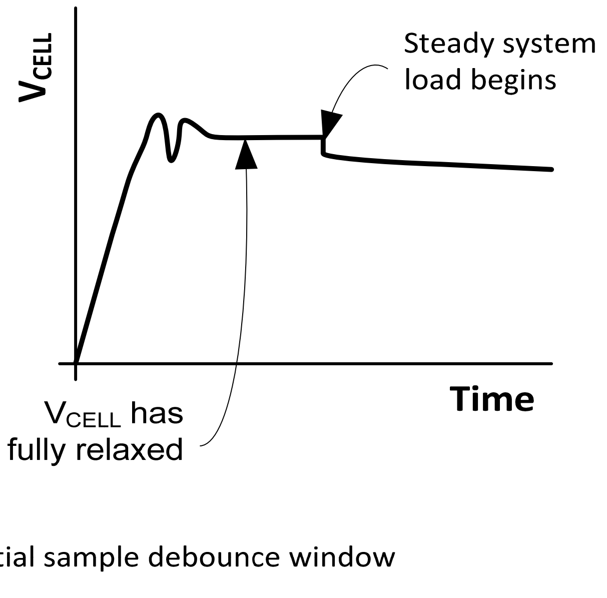
The other method for obtaining a new initial voltage reading is to issue a Quick-Start command. The Quick-Start command will cause the device to re-run the initial voltage sampling routine. The SOC error can be reduced if the cell is at a relaxed voltage at the time of the Quick-Start command. The Quick-Start command will introduce more error into the SOC estimate if the cell voltage is not in a relaxed state at the time the command is issued. The device will automatically recover from this initial error over time. It is often the case that the cell is not in a completely relaxed state when the Quick-Start command is issued. Some initial error is likely, however, in the case of a corrupted initial voltage reading the Quick Start command can help reduce initial SOC estimate.



$V_{CELL}$
Steady system load begins
Best time to quick-start $V_{CELL}$ has fully relaxed
Time
Initial sample debounce window
Quick-start during this time span
Battery insertion – *do not quick-start*

*Figure 25. Quick-start command timing*

*Figure 25* shows an example of what a corrupted initial voltage reading might look like on a scope. As you can see the voltage is noisy during the 125ms initial sampling window. The inrush current period extends well beyond the 125ms window, but eventually settles allowing the battery voltage to recover slightly. This is the time to issue the Quick-Start command. Once the system is fully powered the battery voltage again drops due to the load. A Quick-Start command issued too soon or too late will result in a worse initial SOC estimation. If you believe your system is generating a bad initial SOC then you will need to examine the battery insertion/system power-up waveform to determine if a Quick-Start command is necessary and when to issue the command.



$V_{CELL}$
Steady system load begins
Time
$V_{CELL}$ has fully relaxed
Initial sample debounce window

*Figure 26. Ideal initial reading*

*Figure 26* is an example of an ideal battery insertion/power-up event. A Quick-Start command should not be issued in this case.

The Quick-Start command can be issued in a couple of ways on the MAX17040/1/3/4/8/9 devices.

1. **The Quick-Start command can be issued from software** by writing 4000h to the MODE register. The quick-start command should be used with caution and only sent when the battery is at a relaxed state, as it can often make an error worse if it is sent at the wrong time.

```
// To execute a quick-start write 0x4000 to the MODE register
// Use with caution!
WriteWord(0x06, 0x40, 0x00);
```

2. **The Quick-Start command can be issued from hardware** with a rising edge on an external pin. For the MAX17040/1 the SEO pin must be tied low to configure the device for initiating Quick-Start with a rising edge on the EO pin. The MAX17043/4/8/9 has a dedicated pin, QSTRT that will generate a Quick-Start with a rising edge. This method is useful in situations where the host is unavailable to issue to the command. An external comparator can be used to trigger the Quick-Start at a certain voltage threshold, such as a brown out event where the voltage drops below the device's minimum operating voltage but not low enough to ensure a device reset. (In the event of a brown out the model will need to be reloaded after power is restored. The OCV reading will be preserved during model load.)

### 5.8.5 Initial SOC Healing

Error in initial SOC estimation caused by a non-relaxed battery voltage reading will slowly heal over time regardless of whether there is a load or charge, as shown in *Figure 27*. This is one of the basic advantages of a voltage-fuel-gauge. Errors will converge instead of diverge with time. Voltage fuel-gauging is a long term stability strategy.

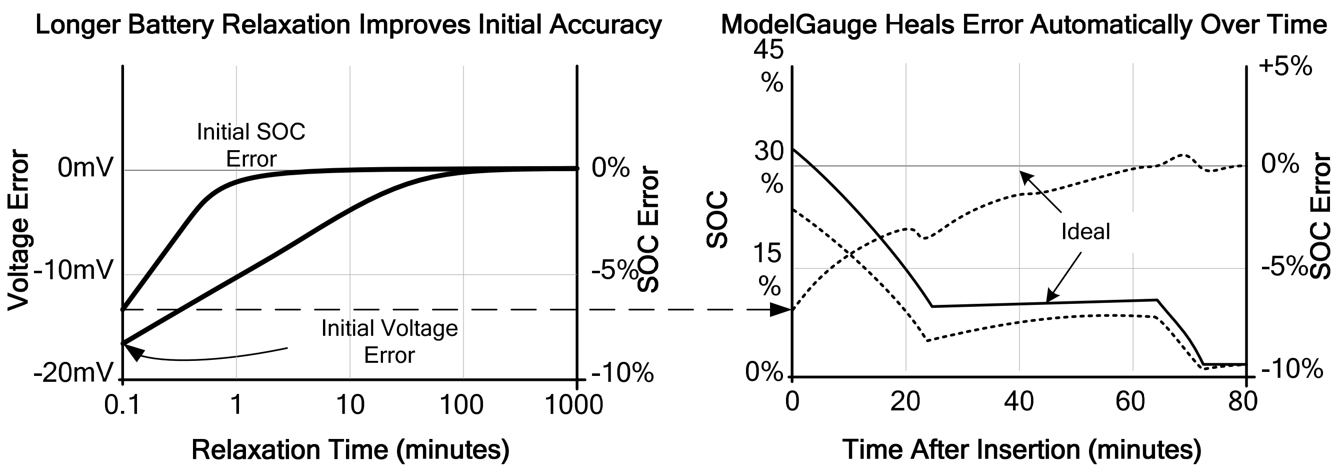


*Figure 27. Initial error correction over time*

## 5.9 Other Software Related Information

### 5.9.1 Considerations Specifically for MAX17048/9

With respect to application software the MAX17048/9 differs from other ModelGauge products as follows:

- **Loading a Custom Model**: When reading the SOC check value, the model must be locked for about 100ms while the first OCV-SOC conversion occurs. This is because the ModelGauge engine is frozen while the model is unlocked.
- **Reading Registers**: In general, block reads are not allowed. Design may be able to advise exactly what barriers exist to block reads.
- **The RCOMP Register**: RCOMP is still located at 0x0D, but it is a part of the CONFIG register at 0x0C. So, to write RCOMP you must read the CONFIG word, two bytes starting at 0x0C, modify the upper byte with the RCOMP value, and write the word back to CONFIG.
- **Hibernate**: The model will not verify if the IC is hibernating, so it must be disabled while loading the model.

### 5.9.2 Reading and Writing IC Register Locations

In all of the software examples in this document reading and writing word locations are defined as follows. All ModelGauge device registers must be written as a complete word. The device will ignore any writes that do not contain all 16 bits of the register.

```
void WriteWord(byte memory_location, byte MSB, byte LSB)
{
    //  Perform these I2C tasks:
        START
        Send Slave Address (0x6Ch)
        Send Memory Location (memory_location)
        Send Data Byte (MSB)
        Send Data Byte (LSB)
        STOP
}
```

All ModelGauge device registers should be read as a complete word.

```
void ReadWord(byte memory_location, byte *output_MSB, byte *output_LSB)
{
    //  Perform these I2C tasks:
        START
        Send Slave Address (0x6Ch)
        Send Memory Location (memory_location)
        RESTART
        Send Slave Address (0x6Dh)
        Read Data Byte (store value in output_MSB)
        Read Data Byte (store value in output_LSB)
        STOP
}
```

### 5.9.3 Configuring Alerts

For all ICs with SOC alert capability, the SOC alert can be configured from 1% to 32% in 1% steps for standard cell models. If the model loaded into the device is 19-bit, then the range changes to 16% and the LSb changes to 0.5%. The alert is always triggered from the same bit location. When upgrading to the MAX1748/49 from an older ModelGauge IC, a 19-bit model can be converted to 18-bit to restore the SOC alert bit weighting to the standard 1%.

For the MAX17048/49, SOC alerts trigger when the 1% weighted SOC bit changes (LSb of the upper byte). For example if the ALSC bit is set and the SOC hex value changes from 0x3200 to 0x31FF or from 0x31FF to 0x3200, the alert will trigger.

# 6 MODELGAUGE LAYOUT CONSIDERATIONS

ModelGauge devices mount on the system side of the application, limiting, but not completely removing any ESD concerns related to layout. The main goal is to minimize voltage measurement error by the IC. To accomplish this there are a few important rules for proper layout. They are as follows:

1. Mount the VDD pin and CELL* pin capacitors as close as possible to the IC. Minimize the loop area of the traces between the IC and the decoupling capacitors. The capacitors are labeled $C_{CELL}$ and $C_{VDD}$ in *Figure 28*.
2. For maximum voltage measurement accuracy, the CELL and GND traces of the MAX1704x circuit should contact the power planes as close as possible to the battery/battery contacts. In *Figure 28*, CELL and GND traces connect directly at the positive and negative battery tabs.
3. The MAX1704x device's exposed pad should be grounded. In *Figure 28*, the exposed pad is used to connect together all other IC pins that are grounded.
4. It is preferred that the MAX1704x is powered directly from the battery pack. Make sure the VDD trace is separate from the CELL trace to prevent errors when measuring voltage.
5. There are no limitations on any other IC connection.

*Figure 28* shows a sample layout using the list rules and *Figure 29* is the reference circuit used for this layout.

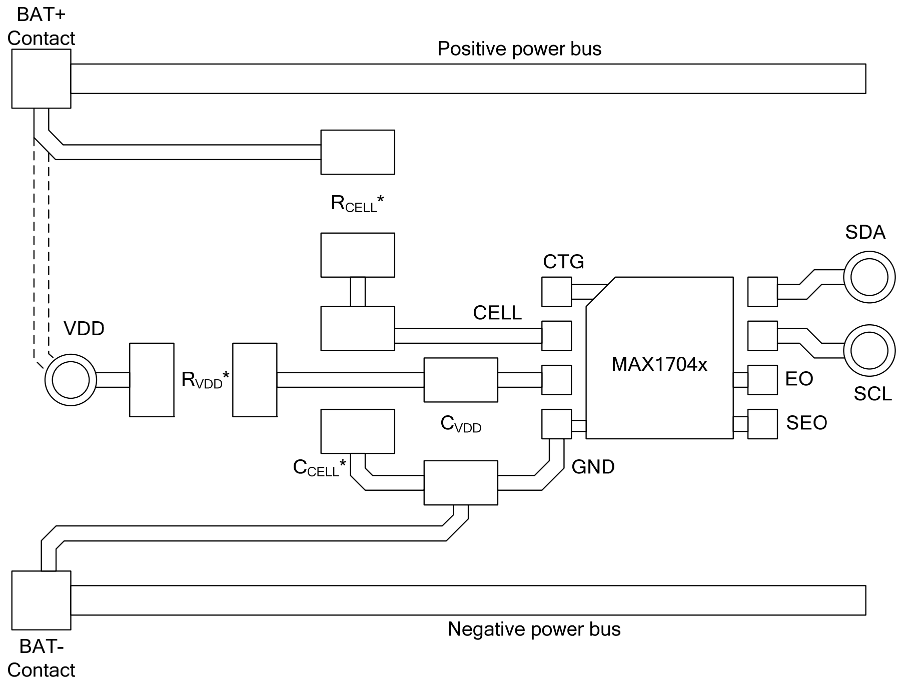


*Figure 28. Example layout of critical paths*

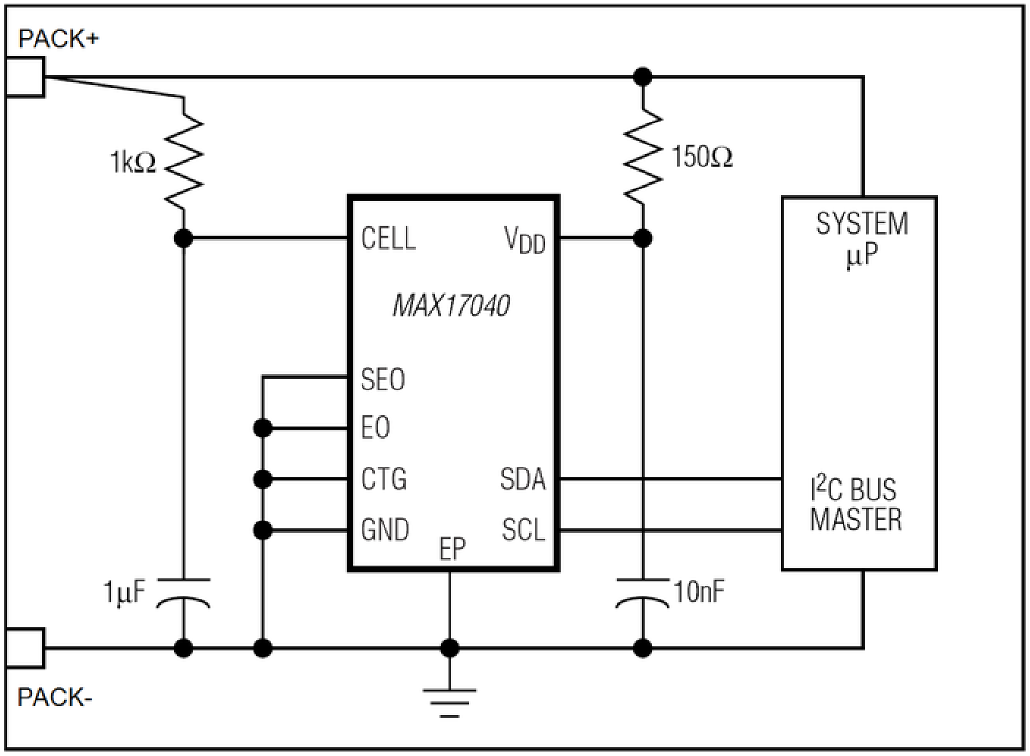


*Figure 29. Circuit used for layout example*

**Series resistors and the CELL pin bypass capacitor are only required on the MAX17040, MAX17041, MAX17043, and MAX17044. All other ModelGauge ICs require only a single bypass capacitor on VDD.*

# 7 PRODUCTION TEST VERIFICATION

After a ModelGauge design is complete, production of the application begins. An ideal test procedure will verify full functionality of the ModelGauge IC in the minimum amount of time. ModelGauge devices shipped from Maxim are already trimmed to provide accurate voltage readings. Production testing should verify connection of the CELL pin and the digital logic/ModelGauge algorithm.

The analog test for a ModelGauge IC is simply a voltage accuracy test. To perform this test, apply a voltage to the CELL pin and read the output of the VCELL register. The measurement should be within the EC table spec. The digital logic test can take the same form as the model verification portion of the code described in the Software Implementation section above.

The code below follows the same format as the Software Implementation section. A more detailed description of code and comments can be found there.

```
//1. Power Up and Voltage Reading Delay
//The ModelGauge IC’s first voltage reading is valid 125ms after power up.
//Add voltage settling time for systems with large supply capacitance.
Power Up System
Delay(125);
```

```
//2. Verify Voltage Reading Accuracy
ReadWord(0x02, VCELL MSB,VCELL LSB);
```

```
//3. Unlock Model Access
WriteWord(0x3E, 0x4A,0x57);
```

```
//4. Write OCV
WriteWord(0x0E,OCVTest High Byte,OCVTest Low Byte);
```

```
//5. Write RCOMP to a Maximum value of 0xFF00h
WriteWord(0x0C, 0xFF,0x00);
```

```
//6. Write the Model
WriteBlock(0x40,ModelTable0,16);  //ModelTable is 16 byte array
WriteBlock(0x50,ModelTable1,16);  //ModelTable is 16 byte array
WriteBlock(0x60,ModelTable2,16);  //ModelTable is 16 byte array
WriteBlock(0x70,ModelTable3,16);  //ModelTable is 16 byte array
```

```
//7. Delay at least 150mS
//This delay must be at least 150mS, the upper limit is not critical in this step.
Delay(150);
```

```
//8. Write OCV
WriteWord(0x0E,OCVTest High Byte,OCVTest Low Byte);
```

```
//9. Delay between 150mS and 600mS
//This delay must be between 150mS and 600mS. Delaying beyond 600mS could cause the
//verification to fail.
Delay(150);
```

```
//10. Read SOC Register and Compare to expected result
//There will be some variation in the SOC register due to the ongoing activity of the
//ModelGauge algorithm.  Therefore, the upper byte of the RCOMP register is verified
//to be within a specified range to verify that the model was loaded correctly.

ReadWord(0x04,&SOC1,&SOC2);

//Compare value
If (SOC1 >= SOCCheckA && SOC1 <= SOCCheckB )
{
    //Model was loaded successful.  Part is working correctly.
}
Else
{
    //Model was not loaded successful.  Retry loading.  If multiple failures, part
    //is not working correctly.
}
```

# 8 PERFORMANCE VERIFICATION

Maxim ModelGauge products offer excellent accuracy, and we recommend that customers verify the performance they achieve in the system matches what is expected from the performance report. Testing the fuel gauge can expose flaws which would otherwise remain undiscovered:

- Incorrect Schematic or datasheet errors
- Less than optimal board layout
- Incorrect firmware implementation or document errors
- Using a different battery or different custom model
- Inaccurate description of system requirements, e.g. load and minimum voltages

Because Maxim tests the batteries under the same conditions as the characterization, the configuration is optimized for the currents and voltages the customer requests. The customer should test performance to verify the custom model, then send that data to Maxim. Closing the feedback-loop in this way, bugs and performance problems can be fixed before mass-production.

## 8.1 At Design Time

If for whatever reason the system will not be able to have a debug firmware, you might need to follow this procedure to understand the fuel gauge behavior:

1. Program the fuel gauge IC with the system firmware
2. Keeping the system fully powered, disconnect the fuel gauge IC from the I2C bus
3. Connect the IC an ModelGauge EVKIT USB to I2C interface
4. Log data using the EVKit PC software

## 8.2 How to Validate Performance

### 8.2.1 Collect Data from the Fuel Gauge

Fuel gauges are complex systems with many parameters and present performance depends on a history of measurements. Collecting sufficient data is a prerequisite to debugging performance problems; see the Data Logging section for detailed instructions.

For ModelGauge voltage fuelgauge, the critical information is Voltage, SOC reported, RCOMP, Temperature, and the time between each log. If a current sense is used to establish reference data, then a coulomb count is also useful to debug and analyze the data.

### 8.2.2 Collect Data from a Reference

Verifying the accuracy of the fuel gauge requires an external measurement to establish a "true" sense of battery state of charge. The best way to do this is through a data log. Critical items to log are time, voltage, current, temperature, and a coulomb counter if available. If a coulomb counter is not available, a complicated pattern can't be setup, but a simple full to empty discharge can be done with a fixed current load. The SOC should follow a straight line to empty. SOC hitting 0% before the battery reaches the empty voltage corresponds to a -% error. The percentage can be calculated by discharging the battery to the empty votlage and looking at how much more runtime was available after the fuelgauge reported 0%. If the battery hits the empty voltage before the SOC reads 0%, the reported SOC is the error.

### 8.2.3 Constructing a Straight Line Reference

The easiest way to construct a reference is with a tool that is measuring the battery in parallel. If a coulomb counter is available, it can be used to build a good representation from full to empty. If a coulomb counter is not available, then an electronic load with a constant current discharge can be used.

A straight line reference is the easiest reference to construct and evaluate performance. Charge the battery to full, relax for 1/2 an hour, then discharge to empty at a constant current. Stop discharging at the empty voltage specified in the characterization request. See *Figure 30*. This test should be done in a temperature controlled environment. When evaluating a ModelGauge fuelgauge that requires external temperature measurement, make sure the RCOMP register is set correctly for the temperature the battery will be tested at. Fuel gauges that compensate for temperature will look like they are reporting more error than what is actually there. An extreme case of this is shown in an example of empty compensation.

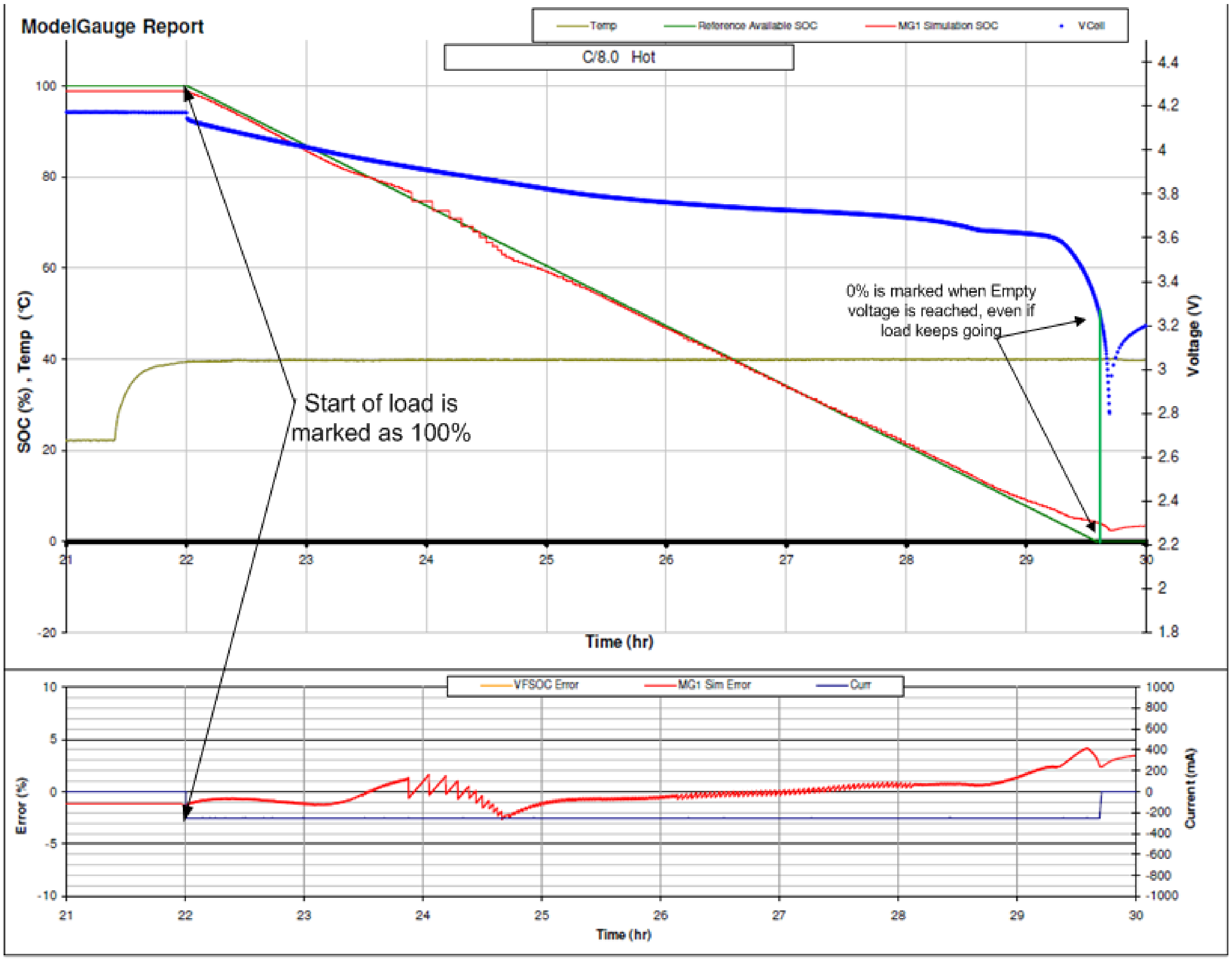


*Figure 30. Validation using a straight line reference*

## 8.2.4 Coulomb Counter Reference

If the test system has a coulomb counter, a wider array of testing can be done than just charging to full and discharging to empty. The trouble with Coulomb Counter references is that there is always drift. The amount of drift is reduced with more accurate current sense instruments, but the problem can never be fully eliminated. The drift can be corrected in the reference by conducting some type of constant test. Maxim's characterization procedures use charging as a control scenario. All batteries are charged at 20 deg C and charged until the same termination current. This allows the full points to be treated as the same capacity, and variations in the capacity at full can be explained by drift and can be canceled out by applying an offset current to the entire dataset.

### 8.2.5 Evaluating Error

All Maxim performance reports measure Fuel Gauge error as Fuelgauge SOC - Reference SOC. Negative error means the fuelgauge reports 0% before the battery hits empty. Positive error means the fuelgauge claims there is capacity left when there really isn't. i.e. If the reference says 15% and the fuel gauge reports 0%, then the error is negative 15% error. The error can be calculated for each data point after the reference is constructed.

## 8.3 Common Mistakes to Avoid

### 8.3.1 Direct SOC-to-Voltage Relationship

There is no direct relationship between instantaneous voltage and state of charge of a battery, which is the reason the ModelGauge algorithm exists. Estimates can be made from observing an open circuit voltage when the cell has to be fully relaxed. This is especially important in the 3.5V - 3.8V range, where millivolts difference can be several percent SOC difference. *Figure 31* shows how the same voltage can represent multiple states of charge.

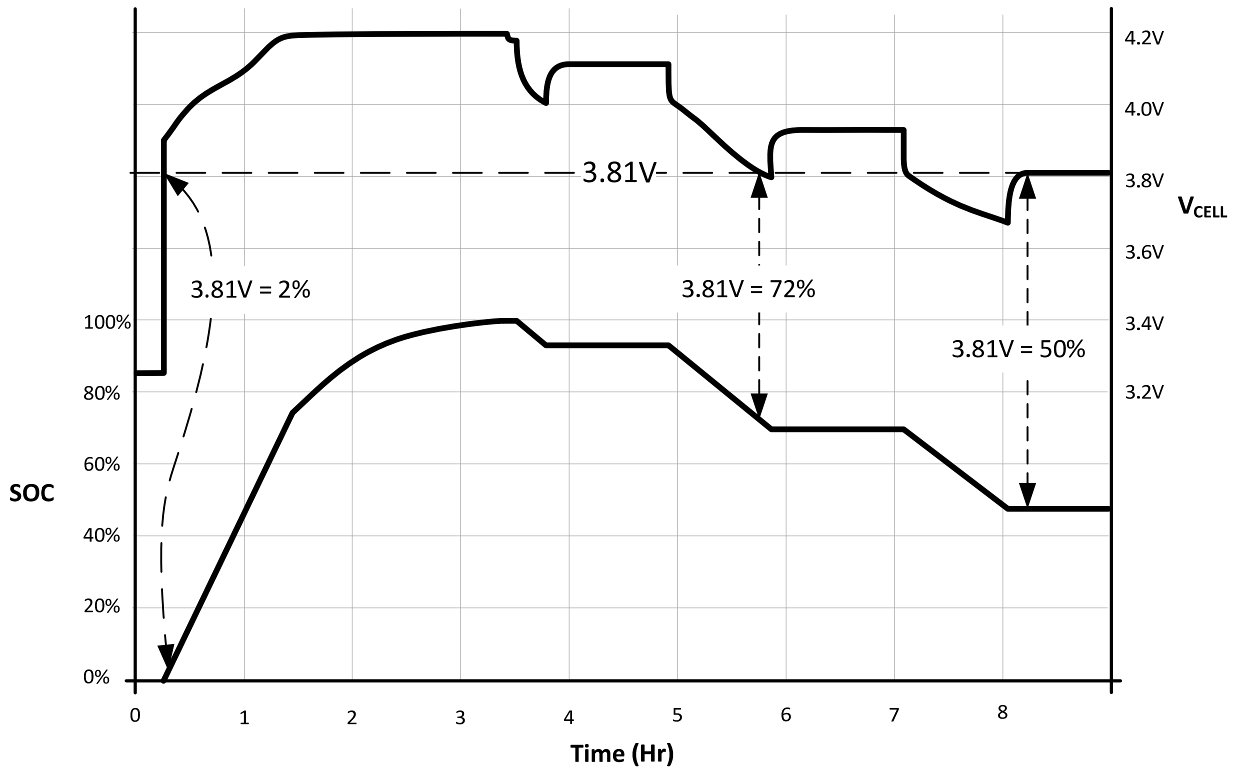


*Figure 31. Cell Voltage is not related to State of Charge*

### 8.3.2 Discharge with Fuelgauge, then discharge with another tool to measure remaining capacity

One conceived test is to discharge the battery with a fuelgauge to a SOC % and discharge the battery on a battery tester to find the remaining capacity. This causes problems in determining the accuracy of the fuelgauge because the capacity available changes with temperature and load conditions. Additionally, due to internal mechanisms, it is possible to hit "empty" on a battery at one current once and hit empty again with a bit more runtime if the battery is allowed to relax longer. This typically happens with high currents for low power batteries. If the Fuelgauge says you have 50% left of a 2000 mAh battery, then 1000 mAh is the capacity available due to the current load and temperature. If the second discharge is done at a different rate or temperature, then the discharge capacity will be different than what the fuelgauge reported. This isn't a fuelgauge error, it’s a testing error.

### 8.3.3 Discharging to a Lower Empty than Where the Model is Focused

Each configuration is based on the empty voltage specified in the characterization request. There is often capacity available below the empty voltage specified. This can be a significant percentage, varying with the magnitude of the load and the empty voltage. To get a sense of the accuracy of the fuelgauge, the battery should be discharged to the empty voltage and no lower. If the battery is discharged below the empty voltage, then the point where it first hits the empty voltage should be the 0% mark.

### 8.3.4 Single Discharge/Charge curve

When a Fuelgauge is first connected to a battery, it has to make an estimate of the State Of Charge. This SOC estimate is based on the initial voltage reading for ModelGauge. The first voltage reading is treated as the Open Circuit Voltage, and is translated into SOC for ModelGauge. If the battery is not relaxed when the first voltage reading is taken, the initial SOC will be incorrect. Some phones and tablets have high board capacitance, causing voltage to drop drastically when the battery is inserted. This causes SOC to be reported low in the beginning. Over time ModelGauge will correct this initial bad estimate and the accuracy will improve. Due to this initial unknown state of the battery, it is recommended that the fuelgauge is connected to the battery and not reset for the duration of a performance test.

# 9 DATA LOGGING AND SUBMISSION

In order to debug and solve problems with fuel-gauge accuracy or other undesirable behavior, Maxim's battery applications team needs a thorough and well-organized log file. This section describes how to collect data and submit it as smoothly as possible.

Remember that any real problems are repeatable and can be logged.

## 9.1 Some Problems Need Special Logging Considerations

The two most common fuel-gauge problems are related to insertion/startup or accuracy. These two cases require different considerations:

### 9.1.1 Insertion and Startup Problems

Make sure to understand how to manage battery insertion. Starting the log immediately is critical because the first moments are when OCV is measured. Repeat the event several times to make sure the problem is repeatable and well understood.

### 9.1.2 Capacity or State Of Charge Accuracy

Make sure to log before and after any of these events:

- Loading a custom model
- Updating RCOMP
- Writing any other register

To understand what the fuel-gauge should do, the battery must be exercised with complete cycles.

### 9.1.3 Always Log Complete Cycles

Being able to see time before and after discharge can help diagnose problems.

A complete cycle is defined as:

1. Constant-current/constant-voltage charging
2. Charge termination
3. Relax
4. Discharge including empty event
5. Relax

If possible, include multiple complete cycles with different current and/or temperature in one continuous log.

## 9.2 How to Collect and Submit Data

### 9.2.1 Submit the Model that was Loaded into the IC During Test

Please submit the model file that was tested so the Applications team can confirm what performance should be expected.

### 9.2.2 Submit the Battery Part Number that was Tested

Seemingly similar batteries can behave very differently, so please help the Applications team confirm that part numbers match exactly.

### 9.2.3 Reproduce the Problem on an EVKit if Possible

The preferred way to diagnose problems is to log data with evaluation kit software, e.g. MAX17040EVKIT. The advantages are:

- Isolates problems with the fuel-gauge from the rest of the system
- Maxim can reproduce the problem with readily-available hardware
- Data log format is already correct
- Requires no software development

### 9.2.4 If the Problem Only Occurs in a System

If using the EVKit is impossible, e.g. the fuel-gauge is mastered by the host processor and the problem does not occur with an EVKit, then software will have to be written to implement error logging.

To avoid unnecessary delays in debugging, reuse the EVKit row and column format. Do this by creating a test log using the EVKit software and any battery. You need only log a moment, because you are most interested in the header. If possible, duplicate its file format:

- Use the same date-stamp format
- Try not to omit any columns, but if one is not logged, please leave the column blank rather than delete it.
- Try to use the original headers, or if you wish to translate the headers into another language, please include the original text.

Include only fuel-gauge IC information. Please do not include any system logs, events, or other information which are not directly applicable to the fuel-gauge IC. The log should include only register readings from the fuel-gauge. If there is a second measurement of battery Voltage, current, or temperature, please include that in a separate file. This type of information can be useful when looking for fuel-gauge errors. Do not log each register with its own row. Collect all registers into a single line, rather than logging each individual read transaction. *Figure 32* shows the MAX17043 EVKit software data log as an example.

| Time | $V_{CELL}$(V) | SOC(%) | Adjusted SOC(%) | Version (Hex) | RCOMP (Hex) | Mode | Alert | Alert Threshold |
|---|---|---|---|---|---|---|---|---|
| **4:39:14 PM** | 3.78750 | 35.05859 | 35.05859 | 2 | 97 | Active | No Alert | 32% |
| **4:39:15 PM** | 3.78750 | 35.05859 | 35.05859 | 2 | 97 | Active | No Alert | 32% |
| **4:39:16 PM** | 3.7869 | 35.0576 | 35.0576 | 2 | 97 | Active | No Alert | 32% |
| **…** | … | … | … | … | … | … | … | … |
| **5:41:38 PM** | 3.5584 | 15.32872 | 15.32872 | 2 | 97 | Active | Alert | 32% |

*Figure 32. Proper Data Logging Format*

## 10 SUMMARY

Any system that accurately measures the state of charge of a cell requires detailed knowledge of the cell and the application. Maxim and the ModelGauge algorithm simplify the entire process to reduce cost and development time while at the same time exceptional results.