

Application Note

MSPM0 Gauge L2 Solution Guide



Eason Zhou

ABSTRACT

This application note describes the level2 gauge solution based on MSPM0. It detects the voltage, current and temperature to calculate the state of charge (SOC) directly. Solution features, hardware introduction, GUI introduction, software introduction, and evaluation will be included in it.

Project collateral discussed in this application note can be downloaded from the following URL: <https://www.ti.com/lit/zip/slaaef5>.

Table of Contents

1 Introduction	3
2 Algorithm Introduction	3
2.1 Battery Basic Knowledge Introduction	3
2.2 Different SOC's and Used Equations	5
2.3 Algorithm Overview	8
3 Gauge GUI Introduction	11
3.1 MCU COM Tool	12
3.2 SM COM Tool	13
3.3 Data Analysis Tool	13
4 MSPM0 Gauge Evaluation Steps	14
4.1 Step1: Hardware Preparation	14
4.2 Step2: Get Battery Model	14
4.3 Step3: Input Customized Configuration	18
4.4 Step4: Evaluation	20
5 MSPM0 Gauge Solutions	23
5.1 MSPM0L1306 + 1 LiCO2 Battery	23
5.2 MSPM0G3507 + BQ76952 + 4 LiFePO4 Batteries	28
6 References	34

List of Figures

Figure 2-1. Battery Discharge Pattern	4
Figure 2-2. Battery Charge Pattern	4
Figure 2-3. SOC-OCV Table	5
Figure 2-4. OCV Calibration and Capacity Accumulation	6
Figure 2-5. Battery Model	6
Figure 2-6. VGauge Software Flow	7
Figure 2-7. Capacity to SOC	7
Figure 2-8. CusRItSoc to SmoothRItSoc	8
Figure 2-9. Algorithm Overview	9
Figure 2-10. Algorithm Performance (By Simulation)	9
Figure 2-11. CT Table Example	11
Figure 3-1. Gauge GUI Functions	12
Figure 3-2. MCU COM Tool Functions	13
Figure 3-3. Data Analysis Tool	14
Figure 4-1. Pulse Discharge Test Case	15
Figure 4-2. Hardware Structure to Get Battery Model	16
Figure 4-3. SmData type	16
Figure 4-4. Generate Parameter File From SMDData	17
Figure 4-5. Battery Circuit Table Input	17

Figure 4-6. Pulse Discharge Example.....	18
Figure 4-7. Gauge_UserConfig.h Setting.....	18
Figure 4-8. Data Structure.....	19
Figure 4-9. tBattParamsConfig Structure.....	19
Figure 4-10. Gauge Mode Setting.....	20
Figure 4-11. Detection Data Input Mode.....	21
Figure 4-12. Communication Data Input Mode Structure.....	21
Figure 4-13. McuData Type.....	22
Figure 4-14. Communication Data Input Steps.....	22
Figure 5-1. Battery SOC.....	23
Figure 5-2. MSPM0 Gauge Hardware Board.....	23
Figure 5-3. MSPM0 Gauge Board Block Diagram.....	24
Figure 5-4. Gauge Board Instructions.....	25
Figure 5-5. MSPM0L1306 Launchpad UART Connection.....	25
Figure 5-6. MSPM0 Gauge Software Project View.....	26
Figure 5-7. Battery Test Case.....	27
Figure 5-8. Battery Test Result.....	27
Figure 5-9. Read Q Values.....	28
Figure 5-10. Current Consumption.....	28
Figure 5-11. MSPM0G3507+BQ75952EVM Block Diagram.....	29
Figure 5-12. MSPM0 Gauge Software Project View.....	30
Figure 5-13. Battery Testcase.....	31
Figure 5-14. Battery Cell Test Result.....	32
Figure 5-15. Battery Pack Test Result.....	32
Figure 5-16. Battery Testcase.....	33
Figure 5-17. Battery Test Result.....	34
Figure 5-18. Battery Test Result.....	34

List of Tables

Table 1-1. MSPM0 Gauge Solution Compare.....	3
Table 2-1. IGauge Key Parameters.....	10
Table 2-2. Capacity Learn Key Parameters.....	10
Table 2-3. Mixing Key Parameters.....	11
Table 4-1. Battery Test Pattern.....	15
Table 4-2. MSPM0 Gauge L2 SOC-OCV Range.....	18
Table 4-3. System Configuration Parameters.....	19
Table 4-4. General Configuration Parameters.....	20
Table 4-5. Mixing Algorithm Related Parameters.....	20
Table 4-6. Capacity Learn Algorithm Related Parameters.....	20
Table 4-7. VGauge Algorithm Related Parameters.....	20
Table 4-8. IGauge Algorithm Related Parameters.....	20

Trademarks

LaunchPad™ is a trademark of Texas Instruments.
 All trademarks are the property of their respective owners.

1 Introduction

There are different Gauge solutions based on MSPM0. [Table 1-1](#) shows the quick compare between them for customers to choose the suitable one. This document focuses on introducing MSPM0 Gauge L2 solution with different setup, including

- MSPM0L1306 Gauge board + 1 LiCO2 battery ([Section 5.1](#))
- MSPM0G3507 Launchpad+ BQ76952 EVM + 4 LiFePO4 batteries ([Section 5.2](#))

Table 1-1. MSPM0 Gauge Solution Compare

	MSPM0 Gauge L1	MSPM0 Gauge L2
Detected parameters	Voltage; Temperature	Voltage; Temperature; Current
Output key parameters	SOC	SOC; SOH; Remain capacity; Cycles
Used methods	Volt Gauge	Coulomb counting + Volt Gauge + Empty/Full compensation + Capacity learn
Suitable application	Output step data with low SOC accuracy	Output percentage data with high SOC accuracy
Suitable battery type	LiCO2/LiMn2O4	LiCO2/LiMn2O4/LiFePO4

Remember MSPM0 Gauge L2 is a pure software code. MCU platform, the AFE or the battery is just used to show the capability of this algorithm. Its features are shown as below:

- Work after MCU power-on without factory calibration or learning cycles.
- Support SOC, SOH, capacities and warning flag output.
- Low requirement for battery chemistry parameters input.
- High precision and reliability

The solution is combined of three parts: hardware, software and GUI. All of them can be found at <https://www.ti.com/lit/zip/slaaef5>. You can also find the MCU code for typical cases under the SDK (mspm0_sdk_xxx\examples\nortos\LP_MSPM0xxxx\battery_gauge).

- The hardware board is used to detect voltage, current and temperature, which will be input into algorithm to calculate SOC. As described for different hardware setup details, see [Section 5](#).
- The software project includes the used gauge algorithm, MCU control and AFE communications. For the description of algorithm, see [Section 2](#). For the typical usage case, see [Section 5](#).
- The GUI is written by python, which can be used to communicate with the gauge board, run test pattern, and do data analysis. For GUI introduction, see [Section 3](#).

2 Algorithm Introduction

In the following section, before giving a description to the software and getting into the detailed algorithm description, a basic idea about the battery knowledge, the used algorithm and how to get the wanted SOC is presented.

2.1 Battery Basic Knowledge Introduction

The gauge algorithm is mostly used to tell users the battery working conditions, and reach a balance between outputting max capacity and extending the battery life. Here, the basic control strategy and the battery performance under these two conditions are shown.

The [Figure 2-1](#) shows a battery discharge pattern for a one-cell LiCO2 battery and the related concept we want to introduce. The red line represents the open circuit voltage (OCV), which means the potential difference between the positive electrode (PE) and the negative electrode (NE) when no current flows and the electrode potentials are at equilibrium. For OCV, it normally can be treated to equal to battery voltage after resting the battery for 1-2 hours. The Blue line means the runtime cell voltage (Vcell). As the battery has internal resistance, you will see a voltage drop between OCV and Vcell with a constant load.

For a LiCO₂ battery, due to the chemical limitation, a full charge voltage threshold (like 4.2V) and an end of discharge voltage threshold (like 3V) will be set to avoid irreversible damage on the battery. That means with different discharge current, you can get different capacities from the battery. Besides, the output capacity is also influenced by the temperature, as the R_{cell} will get reduced while the temperature is increasing. In this gauge solution, the unchangeable capacity is called NomFullCap, which represents the movable lithium ions in the electrode structure and its quantity does not vary with temperature or C-rate of cell usage. The changeable capacity is called CusFullCap, which means the usable capacity by end users and affected by the battery running conditions and threshold setting.

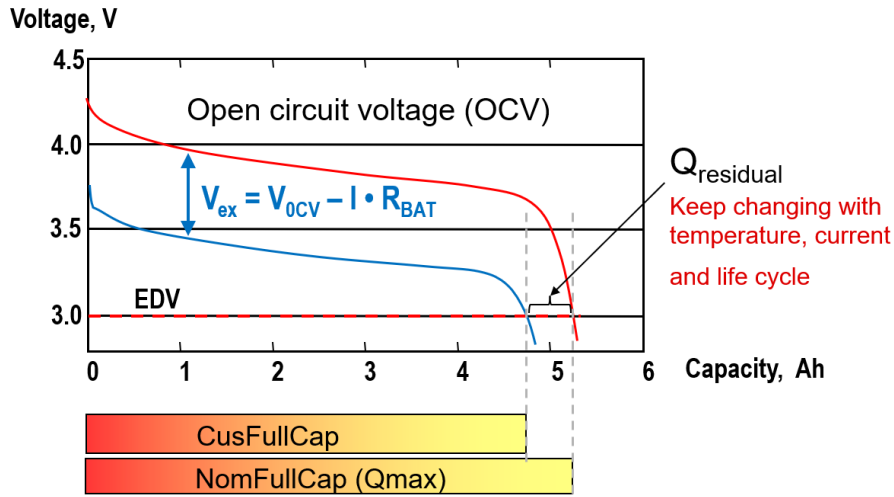


Figure 2-1. Battery Discharge Pattern

The Figure 2-2 shows a battery charge pattern for a normal one cell battery. For a charging condition, it can be simplified into a constant current (CC) window and then a constant voltage (CV) window. At the end of charge, if the charge voltage is constant, the current is reduced in an exponential order. If the current turns to 0, the NomFullCap is obtained. However, to avoid waiting for a long time, actually a terminating charge current is set, like 1/20 capacity (1/20C), which causes a little reduction on CusFullCap, compared with NomFullCap.

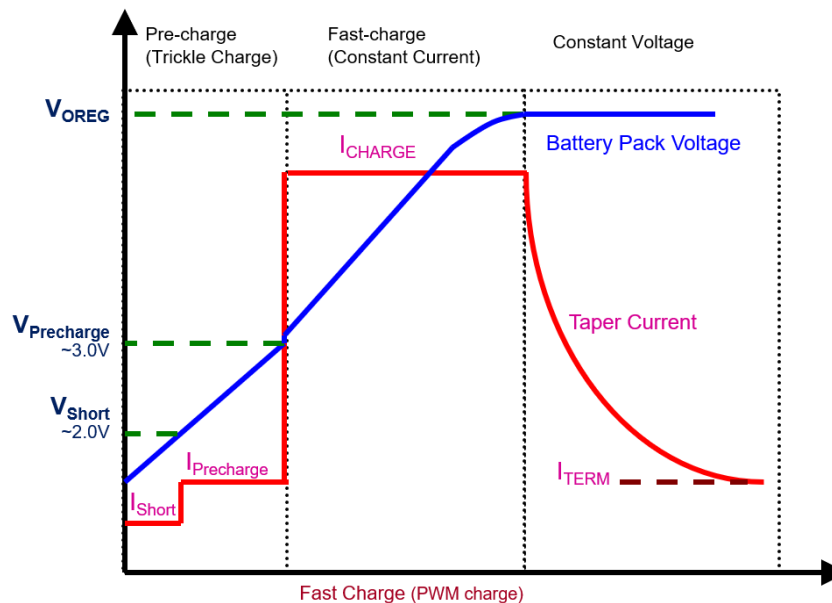


Figure 2-2. Battery Charge Pattern

In a word, NomFullCap is obtained from one OCV to another OCV. CusFullCap is gotten from one V_{cell} to another V_{cell}. So NomFullCap always covers the CusFullCap.

In this gauge algorithm, the NomFullCap range is based on the per saved OCV range in the SOC-OCV table (circuitParamsTable). For the CusFullCap range, it is based on the MaxFullChgVoltThd and EmptyDhgVoltThd setting, and it also changes after self-learning. To leave some margin, it is suggested that you make the OCV range for NomFullCap to be a little larger than the voltage range for CusFullCap.

2.2 Different SOCs and Used Equations

The previous section shows a basic idea about Capacity and OCV. The following sections introduce State-of-charge (SOC), which is finally wanted by end users. Equations are also used to help you under the SOC concept differences.

2.2.1 NomAbsSoc Calculation

If you use the NomFullCap as the normalization factor of the available capacity to the capacity use from a full battery, you will get the normalized absolute State of Charge (NomAbsSoc). This represents the remain percentage of moveable lithium ions in the negative-electrode solid particles. In this algorithm, two strategies are used to calculate NomAbsSoc, which is discussed in the next section. The first is a coulometer with OCV calibration. The second is a battery model filter.

2.2.1.1 Coulometer With OCV Calibration

The common method to update NomAbsSoc is to use coulometer, which is shown in [Equation 1](#) and [Equation 2](#).

$$Q_{use} = I(t) * \Delta t \tag{1}$$

$$NomAbsSoc = \frac{NomFullCap - Q_{use}}{NomFullCap} \tag{2}$$

As coulometer has error accumulation problems. NomAbsSoc is purely calibrated by using the OCV, which is determined after the battery is rested for enough time. An OCV to SOC search table example is shown in [Figure 2-3](#).

$$NomAbsSoc = f(OCV) \tag{3}$$

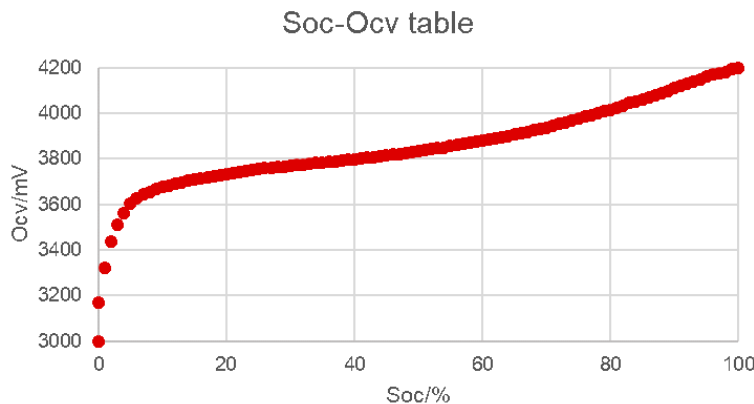


Figure 2-3. SOC-OCV Table

[Equation 2](#) is used to runtime output NomAbsSoc and [Figure 2-1](#) is used to periodically calibrate NomAbsSoc. After two more calibrations, you can get the delta capacity and delta NomAbsSoc. Then, you can calculate the NomFullCap, as shown in [Equation 4](#). Actually, after knowing NomFullCap, [Equation 2](#) can work with acceptable accuracy.

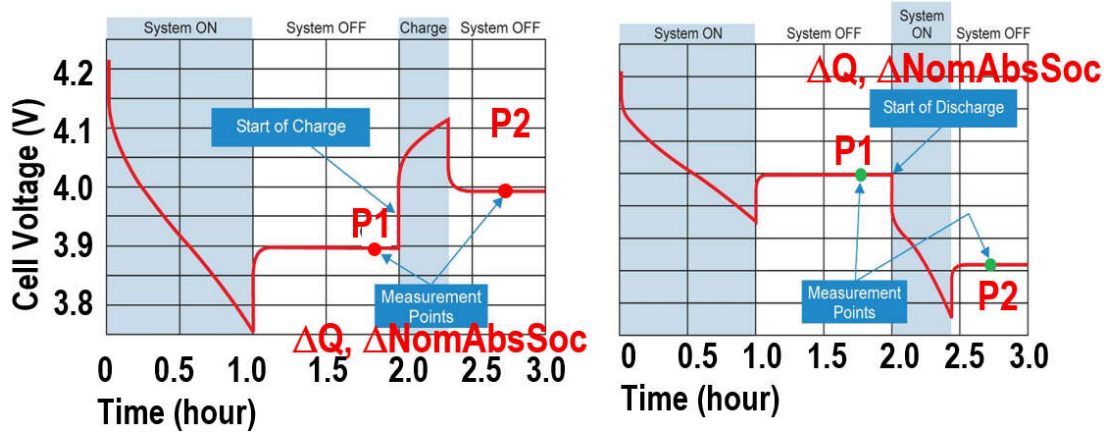


Figure 2-4. OCV Calibration and Capacity Accumulation

$$NomFullCap = \frac{\sum ABS(\Delta Q)}{\sum ABS(\Delta NomAbsSoc)} \quad (4)$$

For a real battery, its NomFullCap will slightly decrease due to the battery getting old. In order to track the capacity decline issue, NomFullCap should be periodically calibrated. Equation 5 is used to represent the capacity decline, named with State-of-Health (SOH). However, in real applications, as the obtained NomFullCap has error, the first obtained NomFullCap is used as the Max NomFullCap.

$$SOH = \frac{NomFullCap[n]}{Max(NomFullCap[n])} \quad (5)$$

2.2.1.2 Battery Model Filter

In Section 2.2.1.1, NomAbsSoc is evaluated based on purely one battery parameter (Voltage or current accumulation). One method is to use the relationship between OCV (positive and negative electrodes balance) and NomAbsSoc (different lithium ion concentrations). Another method is to use the relationship between coulometer (Electron numbers) and capacity (Movable Lithium ion numbers). This kind of strategy requires less computing resources, as only coulometer is needed to work in every cycle. However, it needs to wait until the NomFullCap is generated for 1-2 battery cycles.

Another strategy is used to create a model or a filter to evaluate the NomAbsSoc or even the CusRitSoc based on all the input parameters, like current, voltage, time and temperature. The common methods are equivalent-circuit model, Kalman-filter, neural network and so on. The accuracy of the SOC output mostly lies on the model accuracy. However, more complex model means more MCU computing resources.

An economic method is to use a simplest equivalent-circuit model (a first order RC model), shown in Figure 2-5, to output NomAbsSoc with only voltage input.

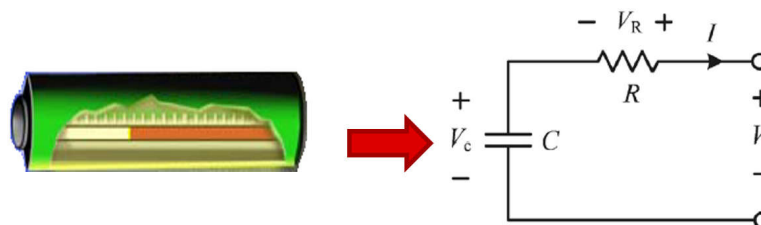


Figure 2-5. Battery Model

Figure 2-6 shows the software flow chart of VGauge. However, in this solution, the SOC lookup table function only needs to be used to search the wanted parameters in this battery model. For the SOC part, it is purely come out from IGauge.

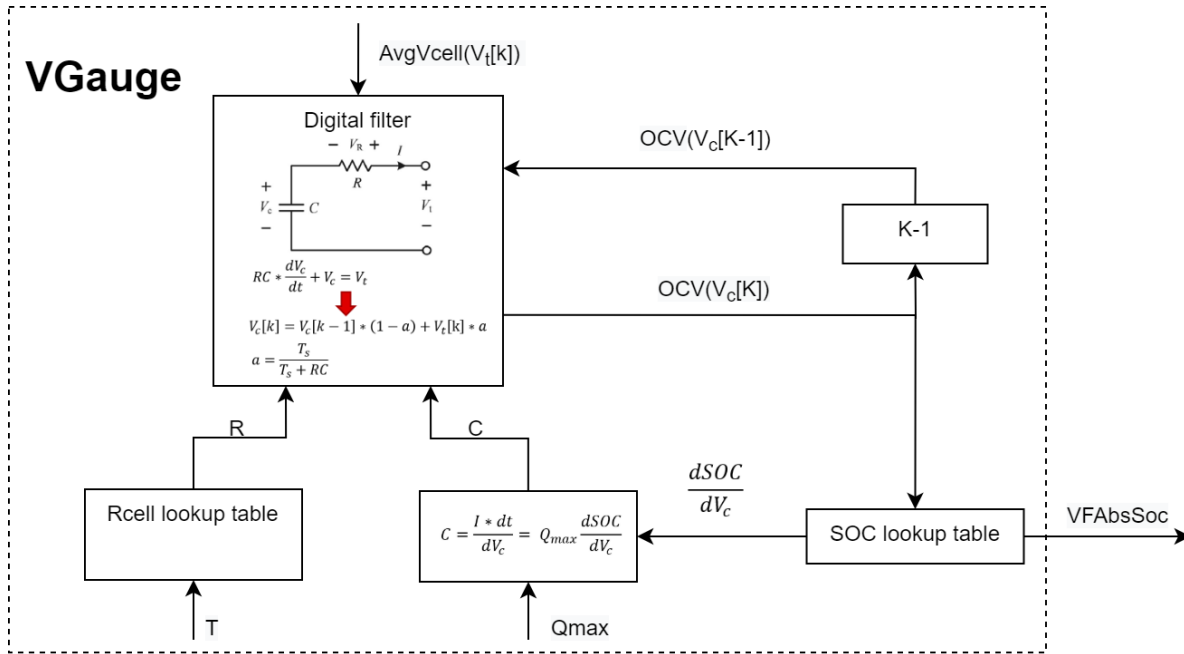


Figure 2-6. VGauge Software Flow

For more about the NomAbsSoc accuracy outputted from VGauge, see [MSPM0 Gauge L1 Solution Guide](#).

2.2.2 CusRltSoc Calculation

For real applications, the wanted SOC by customer is not NomAbsSoc, because the current cannot be controlled at 0 when the battery full and battery empty state is reached, see in [Figure 2-2](#).

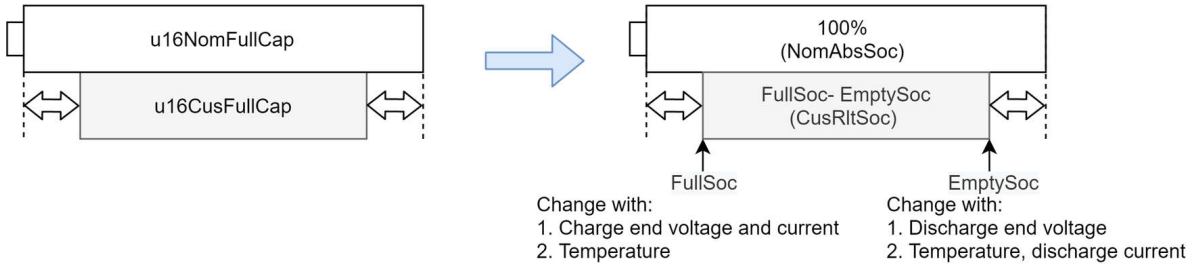


Figure 2-7. Capacity to SOC

Based on the normalization factor: NomFullCap, three new SOC's are gotten to represent the unchangeable capacity, see in [Figure 2-7](#). FullSoc is used to represent the uncharged NomAbsSoc threshold and EmptySoc is used to represent the undischarged NomAbsSoc threshold. CusRltSoc is used to represent the relative SOC, which the normalization factor (FullSoc-EmptySoc) keeps changing with the battery conditions. The equations to calculate the CusRltSoc is shown in [Equation 6](#).

$$CusRltSoc = \frac{NomAbsSoc - EmptySOC}{FullSoc - EmptySoc} \quad (6)$$

The key point to get the CusRltSoc is to get FullSoc and EmptySoc under different conditions. In this gauge algorithm, FullSoc and EmptySoc are recorded and updated into a table under different battery conditions. For this part, refer to [Section 2.3.4](#).

2.2.3 SmoothRltSoc Calculation

The blue line in Figure 2-8 shows a CusRltSoc reaction under a battery charge and discharge test pattern. You can see that there are some data jumps caused from OCV calibration or EmptySoc change.

In order to make the output SOC more acceptable for customers, a SmoothRltSoc is created to make the SOC output to be constant and no sudden jump. For the realizing method, see Section 2.3.4.

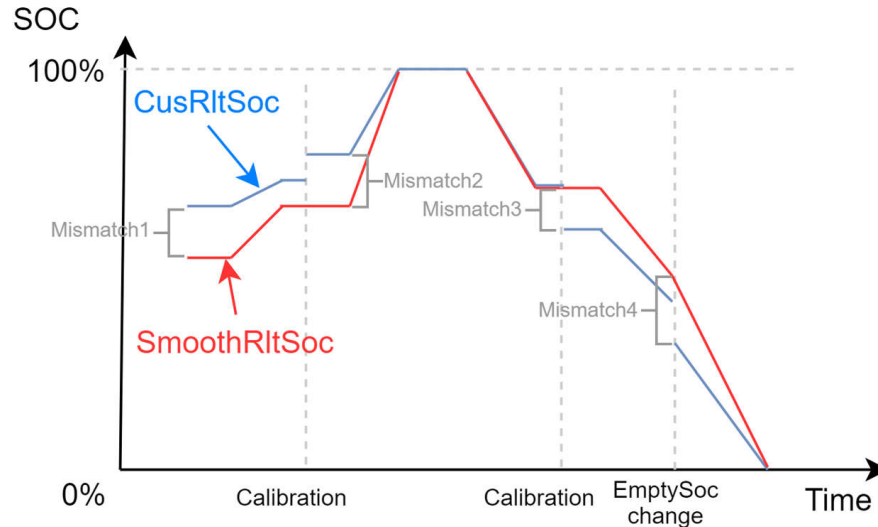


Figure 2-8. CusRltSoc to SmoothRltSoc

2.3 Algorithm Overview

This section provides you with an overview of the introduced gauge algorithm, shown in Figure 2-9. This is only for the battery cell algorithm. For battery pack algorithms, it is just a combination of battery cell algorithms.

This algorithm is based on the coulometer, paired with other methods described before to solve its limitation. It is combined of four parts. The Capacity Learn part is used to detect the battery rest, do OCV calibration and calculate SOH. The VGauge part is used to output the related parameters from the saved class-one battery model. IGauge is a coulometer, used to accumulate the capacity. Mixing part is used to calculate and output NomAbsSoc, CusRltSoc and SmoothRltSoc to customers.

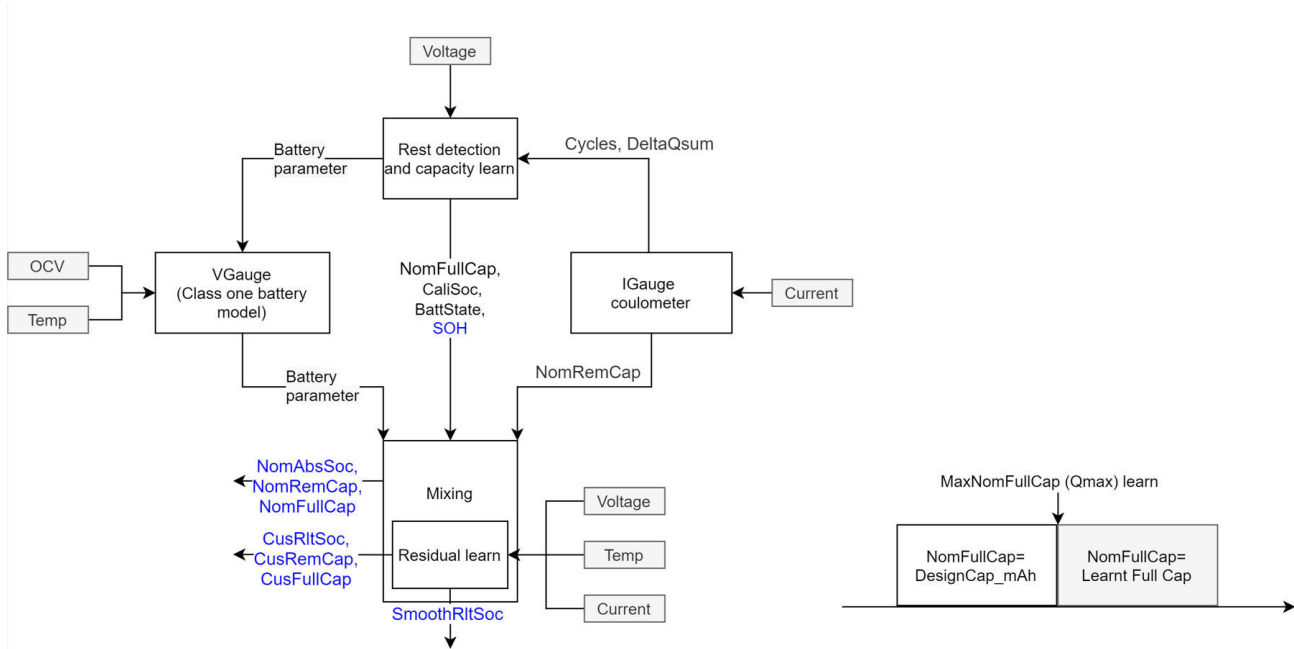


Figure 2-9. Algorithm Overview

Figure 2-10 shows the algorithm performance based on a LiFePO4 battery simulation model, with 3000 random DHG/CHG points, after considering the detection error. The AbsNomSoc error across the battery life is controlled within 3.5%, and the NomFullCap error is controlled within 4%.

Remember that the result is only used to show the algorithm capability to detect NomAbsSoc with a limited condition and it does not ensure the error range of the algorithm in a real application to detect CusRitSoc.

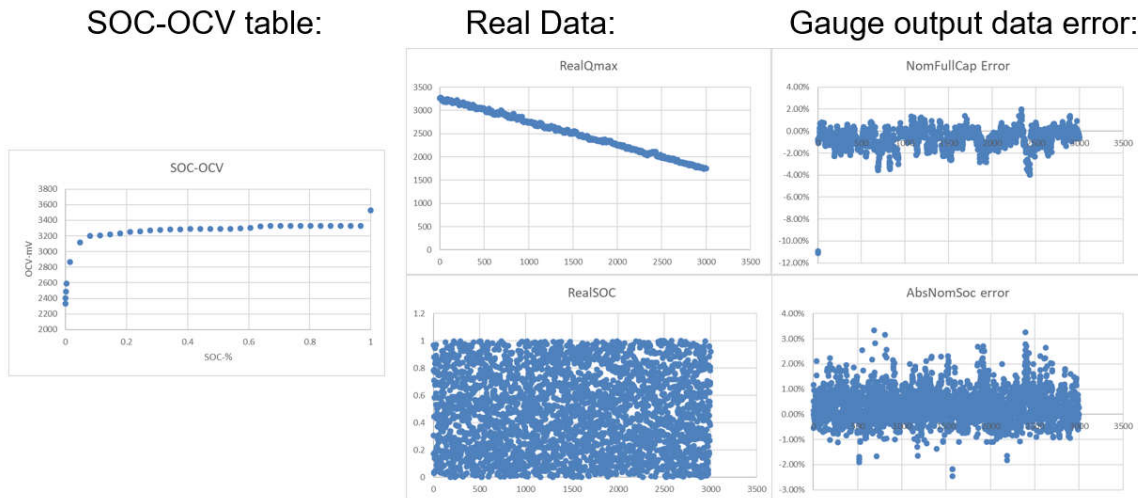


Figure 2-10. Algorithm Performance (By Simulation)

In the following section, a description for every algorithm part and its key parameters outputted to GUI or other functions is provided.

2.3.1 Voltage Gauge Introduction

For the VGauge, it is just used for other function block diagrams to search the battery parameters from a saved battery model: "circuitParamsTable". For more details about the battery model, see [Section 4.2.2](#)

2.3.2 Current Gauge Introduction

The current gauge is just a simple coulometer. It accumulates different types of capacities used for different functions.

The key output parameters of the function are shown in [Table 2-1](#).

Table 2-1. IGauge Key Parameters

Parameter	Comment
u16DhgCycles	Used to tell the user how many cycles the battery has experienced.
iq15DeltaQsum	Used for NomFullCap (Qmax) calculation in CapacityLearn algorithm part.
16IF_NomRemCap_mAh	Runtime calculate and report NomRemCap based on the detected current. It will also be calibrated after getting the OCV.

2.3.3 Capacity Learn Introduction

The capacity learn part has three functions:

- First, is to update battery state, which is used to tell when to do OCV or SOC calibration.
- Second, is to calibrate OCV and SOC. It does the calibration when the battery rest for some time and the voltage drop goes to an acceptable range.
- Third, is to get NomFullCap and SOH, using the method shown in [Section 2.2.1.1](#).

The key output parameters of the function are listed in [Table 2-2](#).

Table 2-2. Capacity Learn Key Parameters

Parameter	Comment
iGaugeDominationFlg	When this flag is set, the u16MaxNomFullCap_mAh is generated and the SOC accuracy goes to an acceptable accuracy
iq15CaliSoc_DEC	This parameter is used by Capacity learn part to record as the calibrated NomAbsSoc for further NomFullCap calculation, after ocvCaliFinishFlg is set. It is also used by Current gauge part to recalculate NomRemCap in the same cycle.
u16CaliOcv_mV	Used to calibrate NomAbsSoc after ocvCaliFinishFlg is set.
u16NomFullCap_mAh	This parameter is used by Capacity learn part to calculate SOH. It is also used by Current gauge part to calculate NomRemCap and discharge or charge cycles.
u16MaxNomFullCap_mAh	This parameter is fixed once the first u16NomFullCap is get and used to calculate SOH.
iq15SOH_DEC	Used to show the capacity decrease.

2.3.4 Mixing Introduction

Mixing algorithm is used to calculate different types of SOC and capacity.

For NomAbsSoc and NomFullCap, it directly comes from IGauge or CapacityLearn.

For EmptySoc, a current-temperature table “AbsEmptySocMatrix” is made to simulate the influence of current and temperature on EmptySoc. A table example is shown in [Figure 2-11](#). One EmptySoc value is used to cover all the real EmptySoc when the battery works in a CT table block range. For example, if $TempThd[1] < Tcell < TempThd[0]$, and $CurtThd[0] < Icell < CurtThd[1]$, EmptySoc[4] is used to represent all the EmptySoc under this condition.

In one block, the real EmptySoc of the left bottom corner is minimum and the real EmptySoc of the right top corner is maximum. The acquired maximum EmptySoc is used to represent the total block for simplifying, the gap between the recorded EmptySoc and real EmptySoc causes the battery dischargeable capacity to shrink, especially in the worst case. In order to reduce this shrink, customers can make the CT table more finely. The max supported temperature threshold number is 6. The max current threshold numbers is 4.

Current-Temperature table example (3x3)

MinTemp	EmptySoc[6]	EmptySoc[7]	EmptySoc[8]
TempThd[1]	EmptySoc[3]	EmptySoc[4]	EmptySoc[5]
TempThd[0]	EmptySoc[0]	EmptySoc[1]	EmptySoc[2]
MaxTemp/0	CurtThd[0]	CurtThd[1]	MinCurt

Figure 2-11. CT Table Example

For FullSoc, after power on, it uses the MaxFullChgVoltThd as OCV to get a default FullSoc first. When the battery is fully charged, the calibrated NomAbsSoc is recognized as the new FullSoc updated into the current-temperature table “AbsFullSocMatrix” under different temperature, same as EmptySoc.

The SmoothRltSoc is obtained by tacking the change of the CusRltSoc with the same target 0% or 100% without sudden change. However, if CusRltSoc suddenly changed to 0% or 100%, SmoothRltSoc changes to 0% and 100% at the same time. Besides, CusRltSoc can be above 0% and 100%. SmoothRltSoc is limited between 0% and 100%.

The key output parameters of the function are listed in [Table 2-3](#).

Table 2-3. Mixing Key Parameters

Parameter	Comment
iq15CusRltSoc	The relative SOC that closely tack the change of EmptySoc and FullSoc, based on NomAbsSoc.
SmoothRltSoc	Used to remove the sudden jump of CusRltSoc.
i16CusRemCap_mAh	The remain capacity can be used by customer with current load and temperature.
u16CusFullCap_mAh	The full capacity can be used by customer with current load and temperature.
i16NomRemCap_mAh	The remain capacity of battery based on circuitParamsTable range.
u16NomFullCap_mAh	The full capacity of battery based on circuitParamsTable range.
iq15AbsEmptySocMatrix	The matrix to record different emptySoc under different temperatures and current. If not initialization, it will be learn automatically.
iq15AbsFullSocMatrix	The matrix to record different fullSoc under different temperatures and it will be learn automatically.

3 Gauge GUI Introduction

Gauge GUI is also a important part of this solution, shown in [Figure 3-1](#). It can be used to record MCU data, run battery test case, and do data conversion. This GUI has three pages.

- First is MCU COM Tool, used to communicate with MSPM0 and record the MCU transmitted battery running data.
- Second is SM COM Tool, used to communicate with the source meter, run battery test case and record the test data sent from the source meter.
- Third is Data Analysis Tool, used to generate battery parameters and evaluate the SOC error.

Pay attention if you are using GUI excuting file to evaluate the solution, the GUI start up time will be a little long, like 2-3 minutes due to the python language limitation. Instead, using GUI source file will have a faster GUI start up speed.

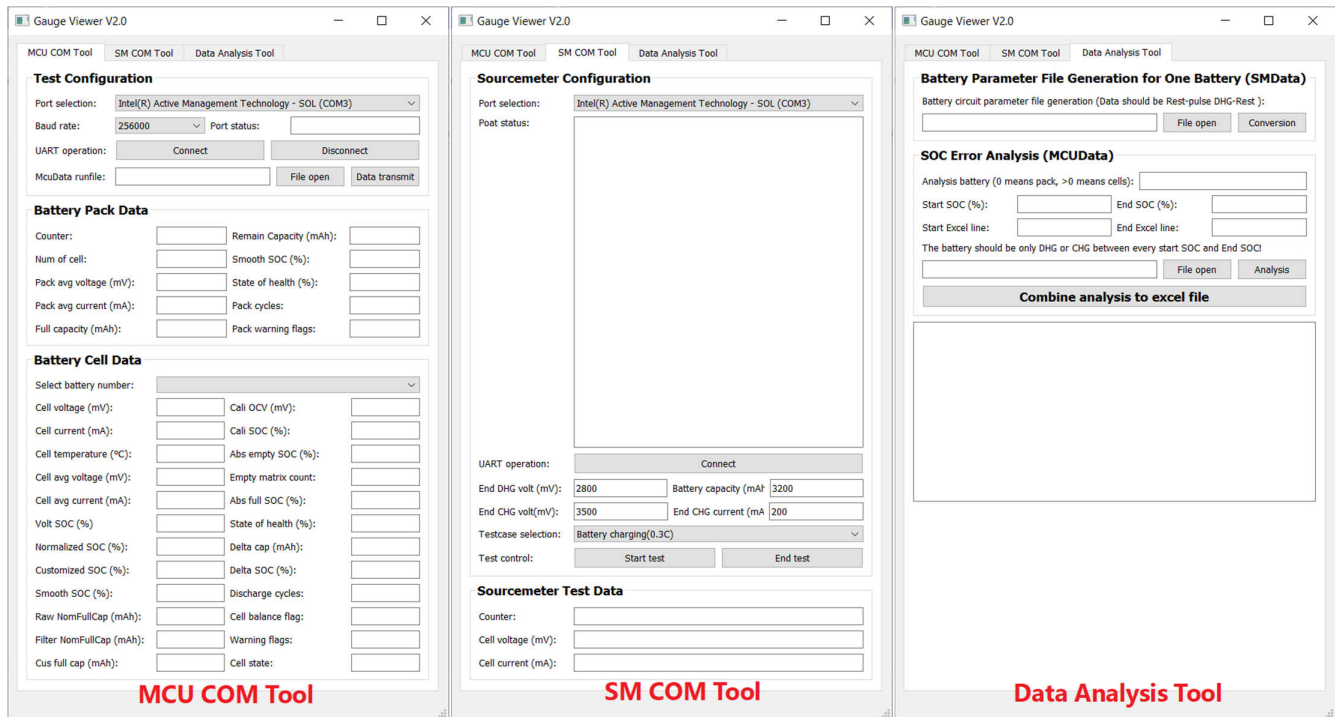


Figure 3-1. Gauge GUI Functions

A Gauge GUI execution file is supplied. The first open will be very long, like 5 to approximately 10 minutes. You can use it for evaluation without installing python. However, if you want to customize the battery test cases under the SM COM Tool, it is recommended that you use the source code. For more details about how to use the GUI, see the next sections.

3.1 MCU COM Tool

The MCU COM Tool, shown in [Figure 3-2](#), will be commonly used by end users. It has two functions:

- The default function is to receive the battery running data from MCU. The default shown data is pack data and cell 1 data. If cell number above 1, you can choose the wanted number following the steps in [Figure 3-2](#). The data is saved automatically in excel with a name “time-McuData.xlsx”, after the test is finished or you stop the test.
- The second function is to load the selected “time-McuData.xlsx” excel file and transmit the cell current, cell voltage and cell temperature data in this file to MCU for algorithm running, paired with the related gauge mode (communication data input mode).

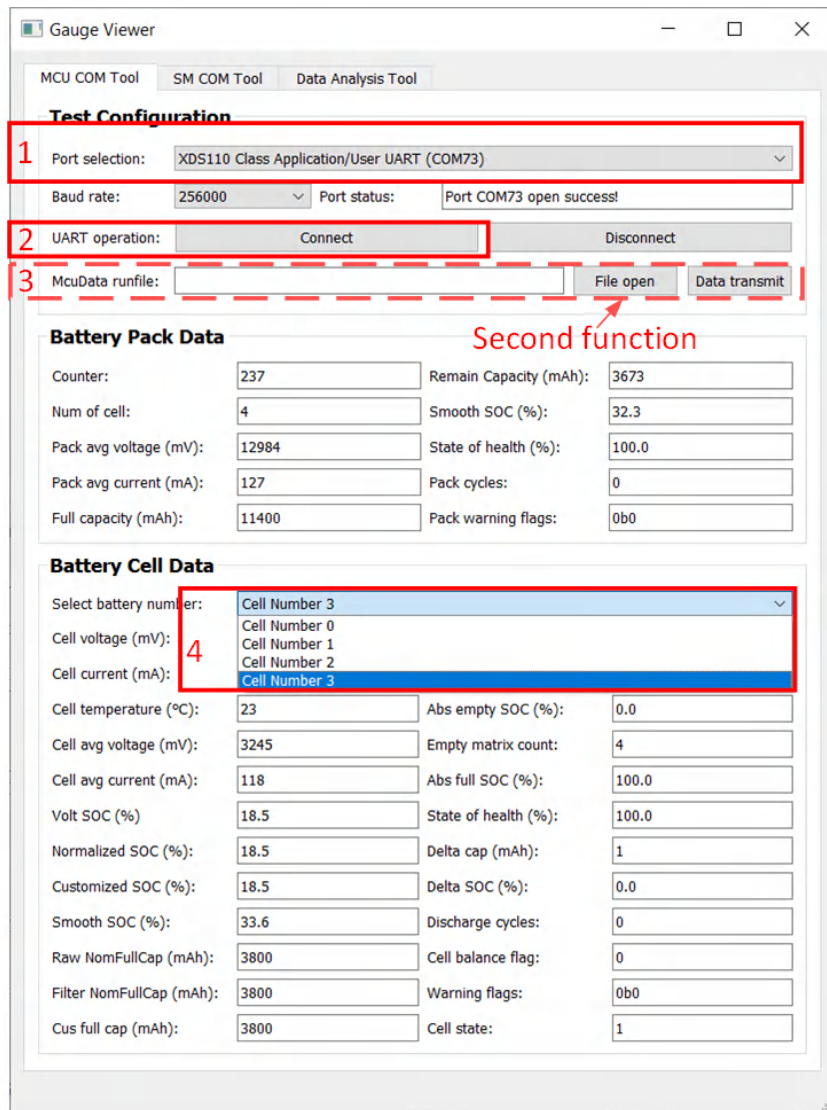


Figure 3-2. MCU COM Tool Functions

3.2 SM COM Tool

For the SM COM Tool, shown in Figure 3-1, it is used to control the source meter to run the battery testcase, and show the data measured by source meter. The record data is saved in excel with a name “time-SmData.xlsx”. If you want to recreate this part for software, you need to at least install NI_VISA. For hardware, you need to buy a USB to rs232 wire and a Keithley 2602A source meter.

As the source meter is not a equipment purely made for BMS, pay attention that it has limitations on the BMS test. For example, it cannot support high current control with high voltage battery. It can trigger the AFE into protection mode with constant current control, because the battery voltage will change above the threshold set in the AFE.

3.3 Data Analysis Tool

Data Analysis Tool is used to do some data conversion and data analysis work.

The first function is the battery parameter file generation, shown in Figure 3-3. It is used to abstract battery circuit information from the pulse discharge datas (SMDData file). Follow the steps, you will get the battery parameter file, for details, see Section 4.2.1.

The second function is to analyze the Gauge accuracy performance. After a real test, MCUData file is generated. The SOC outputted by Gauge at the start of the Excel line and end of the Excel line are priori values. The input of the Start SOC and End SOC by you are posterior values. After you press the analysis button, it first calculates the Qmax by the Start SOC and End SOC input and then generates posterior for every excel line. In a test pastern, there may be more than one DHG and CHG. That means you will also need to do the input in step 2 and step 3 for the same times. At last, you can click step 4, then the GUI will output the final result.

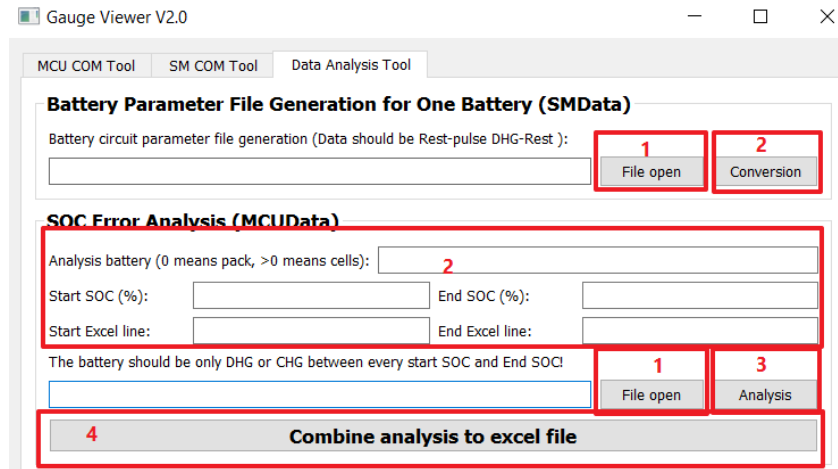


Figure 3-3. Data Analysis Tool

4 MSPM0 Gauge Evaluation Steps

4.1 Step1: Hardware Preparation

Hardware board:

If you want to evaluate this total solution, you need to do the hardware setup first. If you only want to evaluate the gauge software, you just need a LaunchPad™ and input the prepared voltage and temperature data into MSPM0 Gauge algorithm.

Test setup:

In order to do the test and evaluate MSPM0 Gauge performance, you need to prepare a source meter or other battery test machines to control the battery charge and discharge. It will also be helpful if you have a thermo stream to evaluate the gauge performance under different temperature.

4.2 Step2: Get Battery Model

The battery model is obtained by the battery parameter calculation from pulse discharge testcase. It is always good to get the battery model for you project. However, for MSPM0 Gauge L1 with low-discharge current in real application, you do not really need to do the test. You can reuse the default model in the code or get a model related to your battery chemistry from the Web. For higher level MSPM0 Gauge solution, as the SOC calibration accuracy lies on the battery model, it is strongly suggested to get the dedicated battery model.

4.2.1 Battery Test Pattern

For the test machine, you can use any machine that can charge and discharge the battery, and the tested data can be recorded. The paired test machine with the supplied GUI is keithley 2602A source meter, which is controlled through a USB to rs232 wire, paired with NI_VISA.

To get a more accurate model, you need to discharge the battery with low current, like 0.1C for 20 minutes. The rest time after each pulse should be 1-2 hours, then you can take the Vcell as OCV. Finally, with this setting, you will get about 30 points. It is suggested to reduce the discharge current and discharge time at the beginning and at the end in order to catch the voltage rapid change and increase accuracy, especially for LiFePO4 battery.

Note

When doing a battery test, the tested battery should take the PCB and battery socket influence into consideration. Otherwise the tested resistor is smaller than the real circuit resistor.

Table 4-1 shows a suggested test pattern for LiCO₂ / LiMn₂O₄. For LiFePO₄, can refer to it as well.

Table 4-1. Battery Test Pattern

Parameter	Value	Comment
Test temperature	~25°C	
Start voltage (OCV)	4.3~4.4V	Make sure the start voltage is no lower than the application max charge voltage
End voltage (OCV)	2.5~3.0V	Make sure the rest voltage (OCV) is no higher than the application min discharge voltage
Discharge current	0.05C ~ 0.1C (Capacity)	Low current means more point. Suggest to use 0.05C for first and last 5% capacity.
Discharge time	10~20 minutes	Low discharge time means more point. Suggest to use 10 minutes for first and last 5% capacity.
Rest time	1-2 hours	Longer is better. However, 1 hour is enough.

Figure 4-1 shows a battery model example test case. It charges the battery to full (4350mV) and rests it for 1 hour, with the voltage drops to 4322mV. Then, it does a pulse discharge with 20 minutes and rests the battery for 1 hour to get the OCV under different SOC. The test is terminated at 2450mV. After 1 hour rest, the voltage increases to 2864mV. So, the OCV range of the SOC-OCV table is from 4322mV to 2864mV. Start voltage is 4322mV and End voltage is 2864mV, not 2450mV!

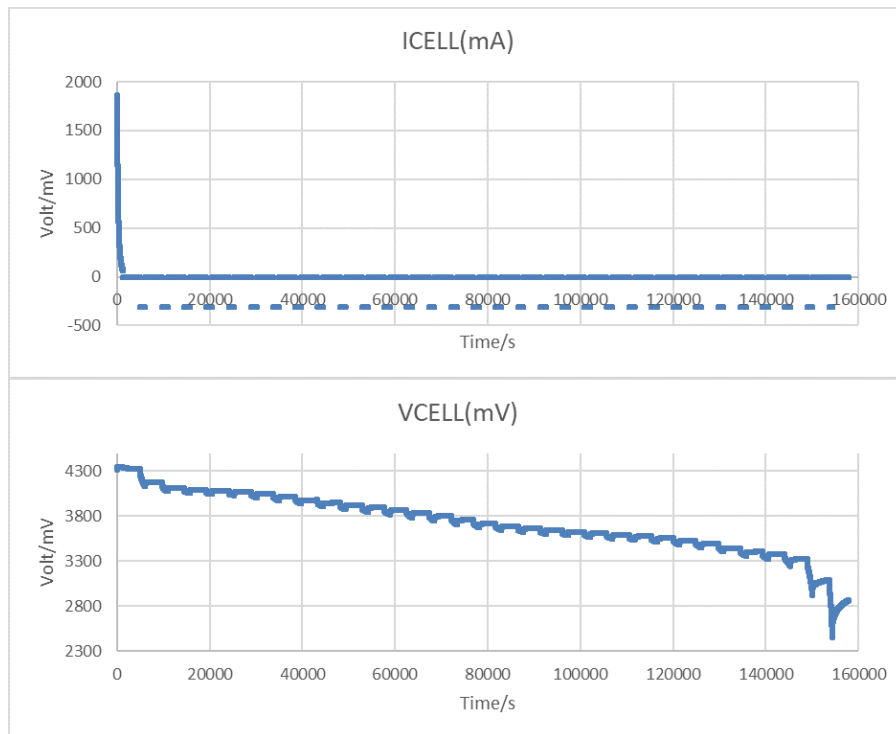


Figure 4-1. Pulse Discharge Test Case

If you use the GUI and the suggested source meter to do the battery test, remember to use the source meter in four wire mode, which can reduce the voltage detection error caused from line resistance. The suggested setup is shown in Figure 4-2. The MCU COM tool is used to get the battery run data. The SM COM tool is used to control the source meter to generate pulse battery charge and collect the voltage and current data to generate the battery parameters later.

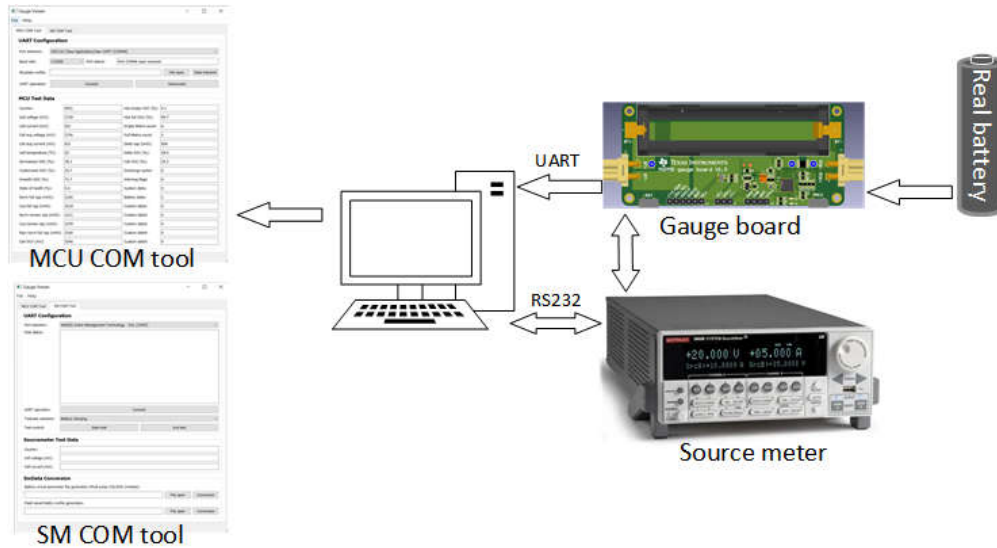


Figure 4-2. Hardware Structure to Get Battery Model

If you use your own test machine to do the test, you can construct the test data according to the SMDData format and using SM COM tool later for battery model generation. Here is the SMDData format. You need to input your tested Vcell and Icell data in Row B and Row C from Line 2. And then name the file with "-SmData.xlsx" at last.

The screenshot shows an Excel spreadsheet titled "Testcase-SmData.xlsx" with the following data:

	A	B	C	D	E	F	G	H
1	Time stamp	VCELL(mV)	ICELL(mA)					
2	0	4200.3	1570.7					
3	1	4200.3	1553.6					
4	2	4200.3	1540.2					
5	3	4200.3	1529.8					
6	4	4200.3	1521.1					
7	5	4200.3	1513.4					
8	6	4200.3	1506.5					
9	7	4200.3	1500.4					
10	8	4200.3	1494.7					
11	9	4200.3	1489.4					
12	10	4200.3	1484.2					

Figure 4-3. SmData type

4.2.2 Battery Model Generation

After you get the battery running data in SMDData format or the MCUDData format (the name should also flow their naming format), you can then use “Battery Parameter File Generation for One Battery” to get the battery model (battery circuit file) in excel and text, by following in the steps in Figure 4-4.

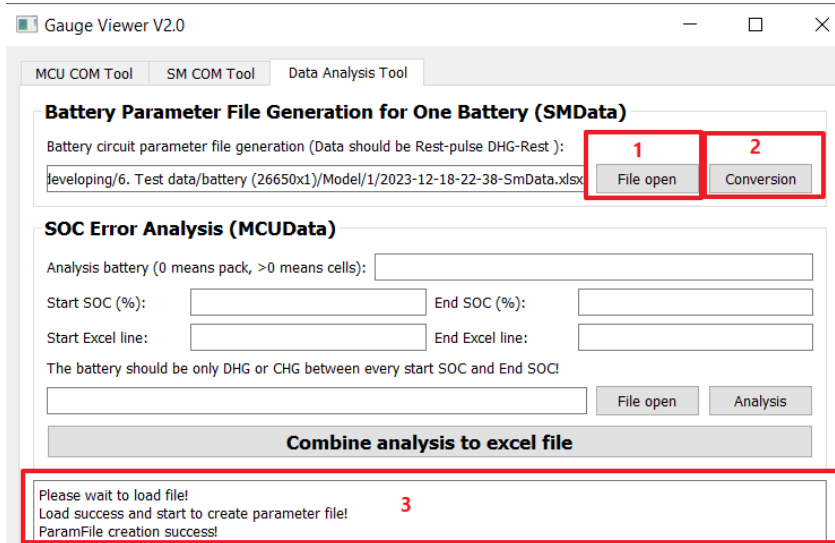


Figure 4-4. Generate Parameter File From SMDData

Copy the generated table in the text into Gauge_UserConfig.c, and the table length into Gauge_UserConfig.h. Then, you can finish the battery circuit table input.

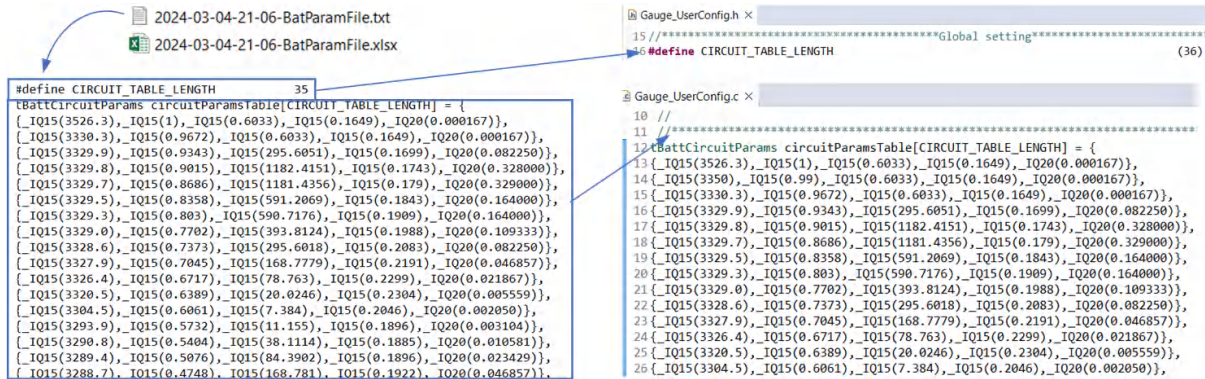


Figure 4-5. Battery Circuit Table Input

The element of the battery circuit table (Battery model) has five combinations:

- The first is OCV (mV).
- The second is SOC.
- The third is Rcell (Ω).
- The fourth is cap factor.
- The fifth is slope rate.

For OCV and SOC, it influences the SOC calibration used in the Capacity learn part. For Rcell and cap factor, it will mostly influence the VFabsSoc accuracy.

A brief introduction is given on how these parameters are generated. As shown in Figure 4-6, OCV equals to the final Vcell before discharging. SOC is obtained after the test with the Qmax at the same time, using Equation 2. Rcell equals to the Ohmic resistance shown in Figure 4-6. The voltage change in one second is treated as the influence of Rcell and its value equals to dOCV(mV)/Current(mA). The cap factor equals to dSOC(%) / dOCV(mV)*Qmax(As) or dSOC(%) / dOCV(V)*3.6*Qmax(mAh). Slope rate equal to dSOC(Dec) / dOCV(mV). For the detailed parameters generation method, see the python source code shared in the development package of this document.

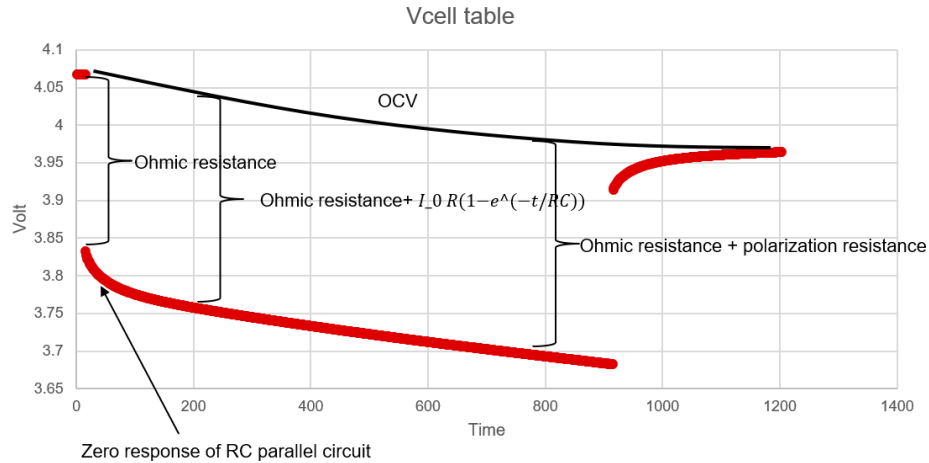


Figure 4-6. Pulse Discharge Example

For high-level MSPM0 Gauge solution, as it takes the residual SOC or battery aging into consideration, it is good to make the circuit table larger than the application voltage range to reserve some buffer. Table 4-2 shows an example for different discharge conditions. Pay attention to the difference between OCV and max charge/discharge voltage.

Table 4-2. MSPM0 Gauge L2 SOC-OCV Range

	Current<0.01C	Current<0.1C	Current<0.5C
Table max OCV	4.3V	4.3V	4.3V
Table min OCV	2.6V	2.6V	2.6V
Application max charge voltage	4.2V	4.2V	4.2V
Application min discharge voltage	2.8V	2.8V	2.8V

4.3 Step3: Input Customized Configuration

First, you need to make some system change based on your applications in "Gauge_UserConfig.h". The common changed parameters are shown in Figure 4-7.

```

//*****Algorithm detection mode selection*****//
#define DETECTION_MODE (COMMUNICATION_DATA_INPUT)
//#define DETECTION_MODE (DETECTION_DATA_INPUT)

//*****Algorithm communication selection*****//
//#define OUTPUT_MODE (NO_OUTPUT)
#define OUTPUT_MODE (UART_OUTPUT)

//*****Global setting*****//
#define CIRCUIT_TABLE_LENGTH (36) //OCV-SOC-Rcell table
#define CELL_NUMBER (4)

```

Figure 4-7. Gauge_UserConfig.h Setting

The explanation of these parameters is shown in [Table 4-3](#).

Table 4-3. System Configuration Parameters

Parameters	Comment
DETECTION_MODE	Affect the algorithm data input source, described in Section 4.4 .
OUTPUT_MODE	Control whether to output tested data to GUI, described in Section 4.4 .
CELL_NUMBER	The cell numbers for the battery pack.
CIRCUIT_TABLE_LENGTH	circuitParamsTable length.

Second, you need to fulfill the data structure configuration in "Gauge_UserConfig.c". A brief introduction to the used data structure in this gauge solution is shown in [Figure 4-8](#).

"tGaugeApplication" represents the battery pack. All the pack related results are saved in "tBattPackParams".

"battGlobalParams_xx" represents every battery cell in the battery pack. All the algorithm data structures are all under it. You need to create the "battGlobalParams_xx" structure same as the battery numbers, and hang it under the "tGaugeApplication", through "battGlobalParamsArray".

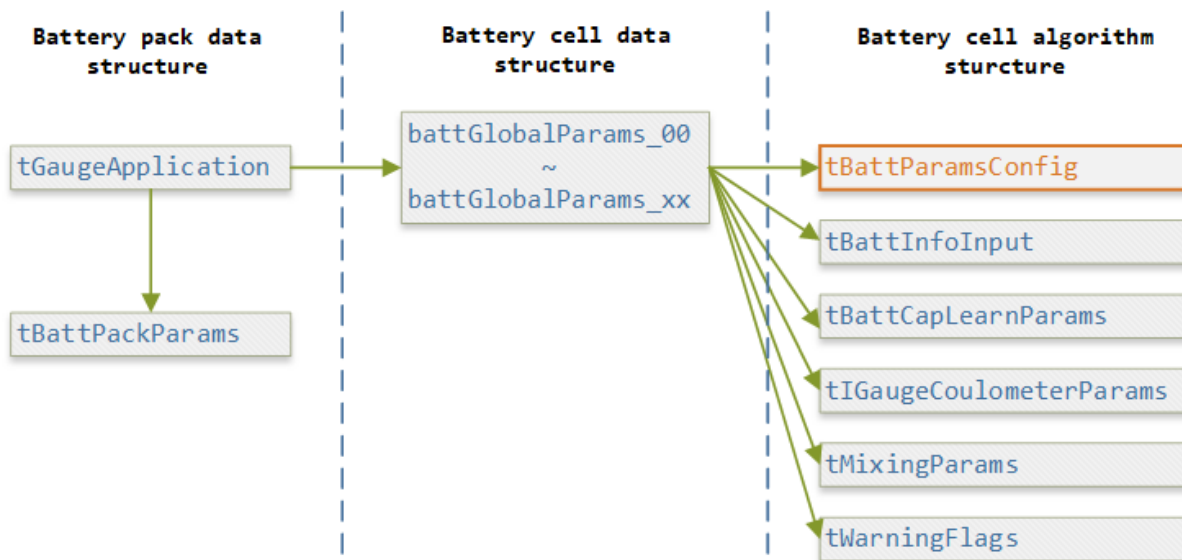


Figure 4-8. Data Structure

The most important data structure is "tBattParamsConfig", shown in [Figure 4-9](#). It contains all the battery parameter settings and algorithm settings.

```

Gauge_UserConfig.c
49 };
50
51 const tBattParamsConfig battParamsCfg = {
52 //*****General configuration parameters**
53 .pBattCircuitParams = circuitParamsTable,
54 .u16DesignCap_mAh = 3200,
55 .u16MinBattVoltThd_mV = 2500, //Need to ensure the battery
56 .u16MaxBattVoltThd_mV = 4300, //Need to ensure the battery
57 .u16MinFullChgVoltThd_mV= 4100, //We advise to set the value
  
```

Figure 4-9. tBattParamsConfig Structure

For easy evaluation, you only need to change the general configuration parameters. These parameters are divided into five parts. A short description is provided for all these related parameters.

Table 4-4. General Configuration Parameters

Parameters	Comment
u16DesignCap_mAh	Design capacity. Just input the standard capacity of battery or the tested one through battery parameter generation test.
u16MinBattVoltThd_mV / u16MaxBattVoltThd_mV i16MaxChgCurtThd_mA / i16MinDhgCurtThd_mA i8MaxChgTempThd_C / i8MinChgTempThd_C i8MaxDhgTempThd_C / i8MinDhgTempThd_C	Battery Vcell, lcell and Tcell threshold. They are reserved to control warning flags when the battery situation is above these parameters.
u16MinFullChgVoltThd_mV u16MaxFullChgVoltThd_mV/i16FullChgCurtThd_mA	Battery full related setting. The battery charge voltage should be in this range. u16MinFullChgVoltThd_mV will help to judge when the battery is full. u16MaxFullChgVoltThd_mV will be used as default Full OCV after MCU power on. When the current is below than i16FullChgCurtThd_mA and the voltage is above u16MinFullChgVoltThd_mV, we will treat the battery is full.
u16EmptyDhgVoltThd_mV	When voltage reaches this value, we treat battery is empty.
u8AvgBattParamsUpdateCount	The average data will be obtained after the settled cycles.

Table 4-5. Mixing Algorithm Related Parameters

Parameters	Comment
i8TempThdx_Ci16CurtThdx_mA	The current and temperature threshold for iq15AbsEmptySocMatrixInput[], iq15AbsEmptySocMatrix[], iq15AbsFullSocMatrix[]. Only for emptySoc, is input provided. You can use divide range you want according to Section 2.3.4 .
iq15AbsEmptySocMatrixInput[]	The pretested EmptySoc matrix input. If all is 0, u16EmptyDhgVoltThd_mV is used as the empty OCV to calculate a related EmptySoc. Then, it will be auto learned after cycling.

Table 4-6. Capacity Learn Algorithm Related Parameters

Parameters	Comment
i16UnloadCurtLowThd_mA i16UnloadCurtHighThd_mA	If the current is between this range, the battery will be treated as rest. Remember to consider the noise of current detection. Otherwise the rest detection will have problem.
u8SOHCalcCycleThd	The battery discharge cycle threshold to do SOH calculation.
iq15DefaultSOH_DEC	Battery default SOH value.

Table 4-7. VGauge Algorithm Related Parameters

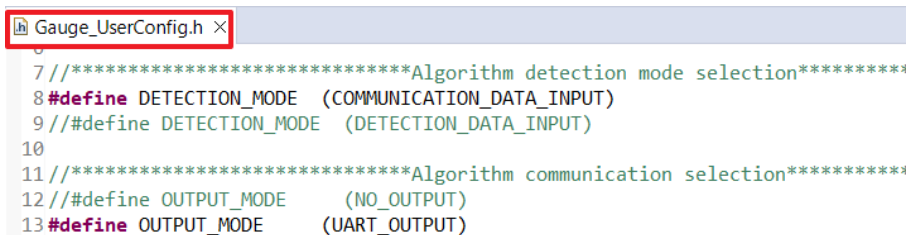
Parameters	Comment
u8CircuitTableLength	Circuit table length.
u8CircuitTableTestTemp_C iq15RcellNegTshift_R iq15RcellPosTshift_R	These parameters are used to evaluate the Rcell under different temperatures. It is by experience and will not affect the performance too much.

Table 4-8. IGauge Algorithm Related Parameters

Parameters	Comment
i16AvgLeakageCurt_mA	Internal leakage current compensation.
i16PassiveDhgCurt_mA	This is reserved for passive discharge current condition.

4.4 Step4: Evaluation

Figure 4-10 shows different evaluation modes used for different conditions, selected in Gauge_UserConfig.h.



```

Gauge_UserConfig.h
7//*****Algorithm detection mode selection*****
8#define DETECTION_MODE (COMMUNICATION_DATA_INPUT)
9//#define DETECTION_MODE (DETECTION_DATA_INPUT)
10
11//*****Algorithm communication selection*****
12//#define OUTPUT_MODE (NO_OUTPUT)
13#define OUTPUT_MODE (UART_OUTPUT)

```

Figure 4-10. Gauge Mode Setting

For different output modes, UART_OUTPUT means enabling data output through universal asynchronous receiver/transmitter (UART). Then you can observe the battery running parameters on the GUI through USB to tool. NO_OUTPUT means terminating the UART data output.

The different detection modes are detailed in the following section. For detection data input mode, it is the common used mode. All the algorithm input is from real data tested by AFE. For communication data input mode, its input data comes from GUI and the one cycle time limitation lies on the UART communication speed.

4.4.1 Detection Data Input Mode

In this mode, you need the MSPM0 Gauge board and a real battery for test. The detection data (Vcell, Icell, Tcell) comes from the real detected signals. The GUI can help to record the battery running data for further analysis.

For software setting, you need to download the gauge code to the launchpad after changing the detection mode to "DETECTION_DATA_INPUT" in "Gauge_UserConfig.h".

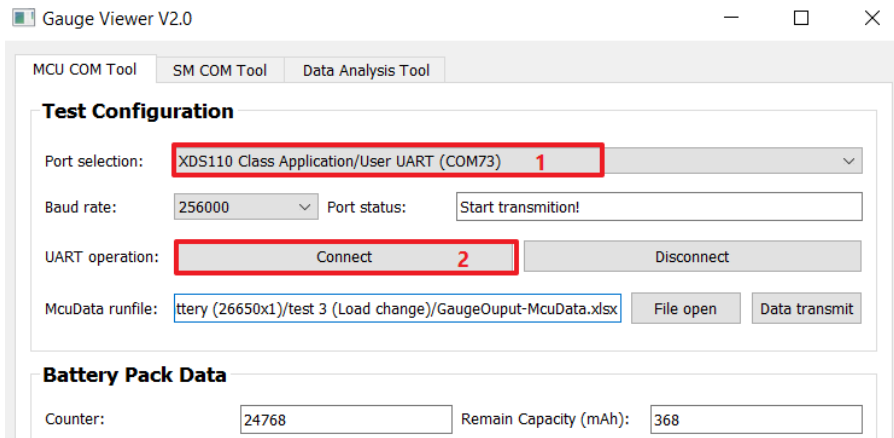


Figure 4-11. Detection Data Input Mode

For GUI control, you only need to select the right port with the name XDS110 Class Application/User UART and then click the connect button. If the MCU already works, you will see the test data at MCU Test Data block. After you click the disconnect button, the data will be saved in excel type under the same address of GUI, with the name "YYYY-MM-DD-HH-MM-McuData.xlsx".

4.4.2 Communication Data Input Mode

For this mode, the battery running data is input from the GUI. It enables you to run the real test case or evaluate the MSPM0 Gauge with only a LaunchPad. This method can remove the need of hardware, increase algorithm running frequency and have no limit to the length of battery running data.

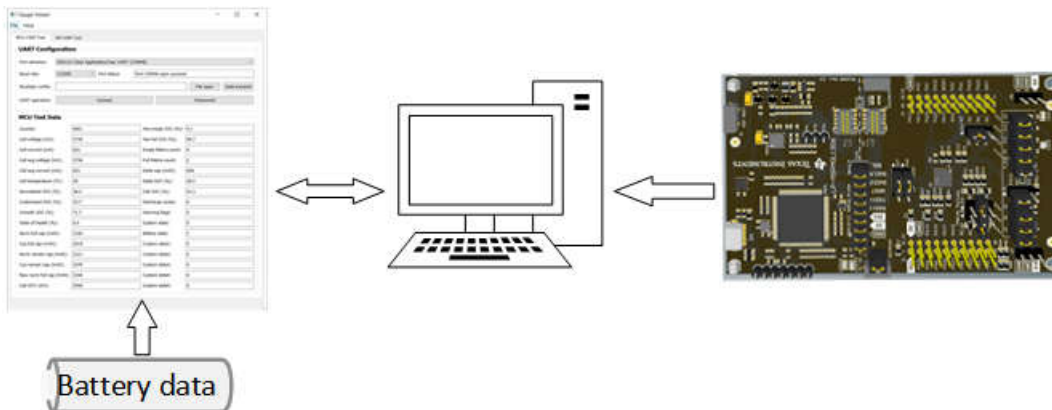


Figure 4-12. Communication Data Input Mode Structure

1. First, to realize this method, you only need a LaunchPad and do the right hardware setting.
2. Second, download the gauge code to the launchpad after changing the detection mode to "COMMUNICATION_DATA_INPUT" in "Gauge_UserConfig.h".
3. Third, you need to have a MCUData file. An introduction is provided on how to transmit a test data into a recognized file by the GUI, especially for those who do not generate the test file from GUI. You need to input the Cell num at column B. And then input every battery's Vcell(mV), Icell(mA) and Tcell(°C) into the same column like the same McuData file. For this, you can generate a McuData file first and refer to that one to do the transmit ion. At last, name the file with "-McuData.xlsx".

Time start	CellNum	Vcell	ICELL(mA)	1_TCELL(C)
0	4	3394.88	998.7	23
1	4	3398.43	998.7	23
2	4	3401.33	998.7	23
3	4	3403.65	998.7	23
4	4	3405.78	998.7	23
5	4	3407.68	998.7	23
6	4	3409.45	998.7	23
7	4	3411.13	998.7	23
8	4	3412.73	998.7	23
9	4	3414.25	998.7	23
10	4	3415.7	998.7	23

Figure 4-13. McuData Type

4. Forth, connect the UART COM port following Figure 4-14 and load the MCUData runfile in MCU COM Tool by clicking the File open button. After clicking the Data transmit button, you need to wait until the port status changes to "Start transmit ion!". The data load time and excel save time will be long if the file is very large. It would be 5~10 minutes.

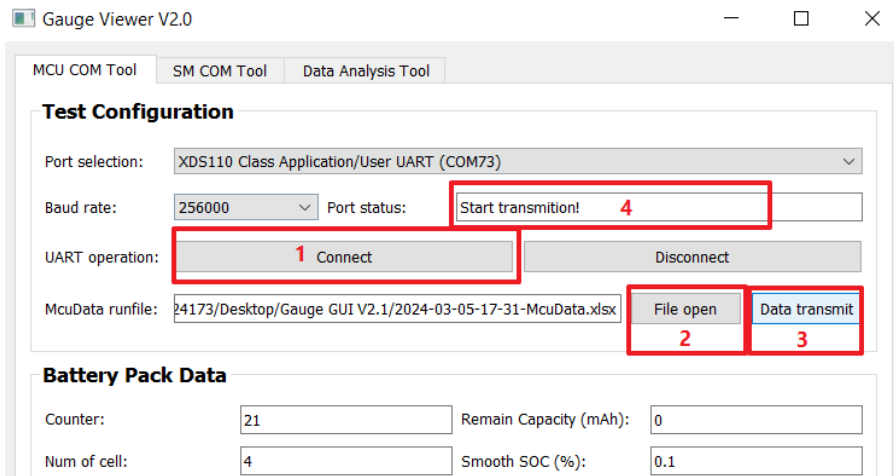


Figure 4-14. Communication Data Input Steps

You will receive the battery running data from MCU shown in MCU Test Data block. After it finishes transmission, the GUI automatically saves the received data under the GUI address.

5 MSPM0 Gauge Solutions

In this section, two different testcases are used to show the capability of MSPM0 Gauge L2:

- Able to switch from one cell detection to multi-cell detection easily
- Able to handle different battery type, especially for LiFePO4, which SOC-OCV table is much flat

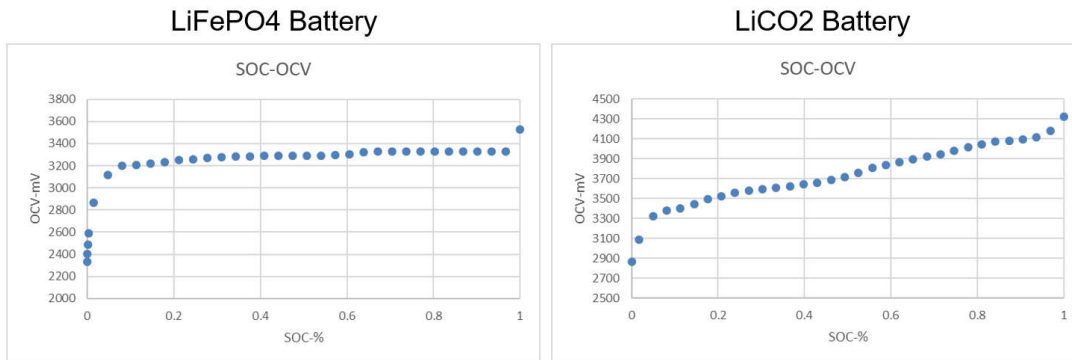


Figure 5-1. Battery SOC

MSPM0 hardware features are also shown, which is why it can be suitable for BMS solutions, including internal high precision analog peripherals and numerous communication methods (CAN/UART/serial peripheral interface (SPI) /inter-integrated circuit (I2C)).

5.1 MSPM0L1306 + 1 LiCO2 Battery

Solution features:

- Total solution takes about 15K flash and 1.3K SRAM.
- Current consumption without universal asynchronous receiver/transmitter (UART) communication (NO_OUTPUT mode) is about 9uA.

Solution advantage:

- A pure one-chip solution with internal analog peripherals
- Self-calibratable current detection with ~1% error
- High performance Gauge algorithm

5.1.1 Hardware Setup Introduction

The hardware board is typically made to evaluate the one-cell battery gauge solution.

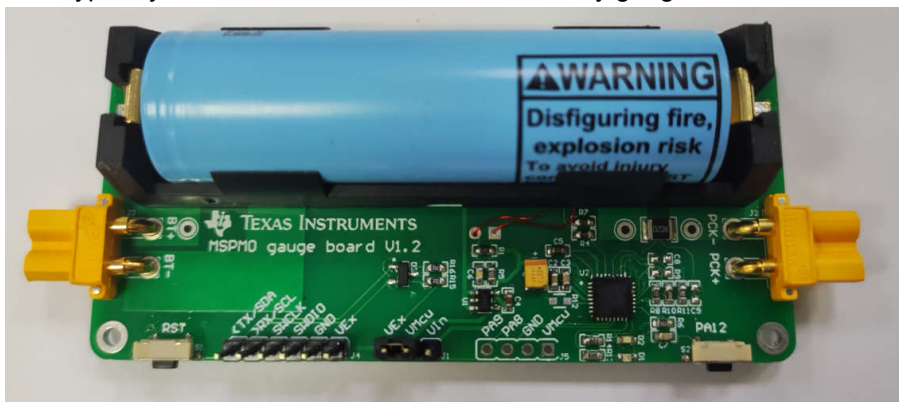


Figure 5-2. MSPM0 Gauge Hardware Board

Figure 5-3 shows the hardware high-level block diagram, showing all the used pins by this demo. The solution tests the current at the analog-to-digital controller (ADC) channel 13, the temperature at ADC channel 5 and voltage at ADC channel 1.

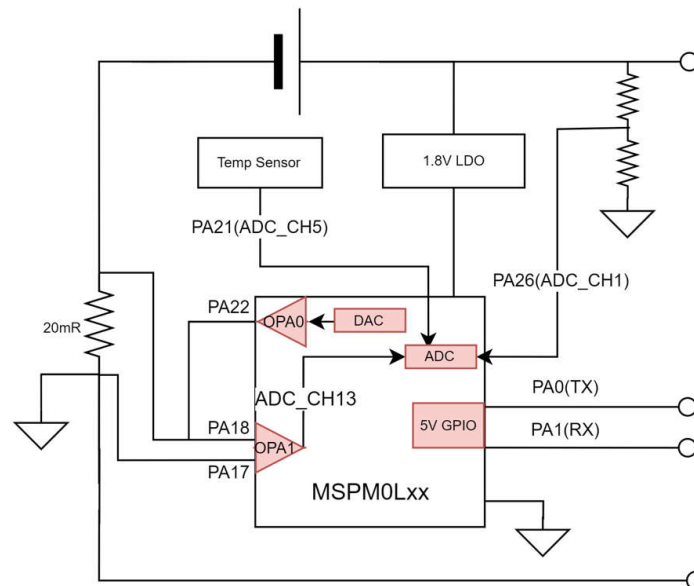


Figure 5-3. MSPM0 Gauge Board Block Diagram

With the internal OPA for current detection, its detection error at room temperature can reach $\pm 0.25\%$ at $\pm 2A$ load. For more hardware introduction and its performance, see [A Self-Calibratable Current Detection Solution Based on MSPM0](#).

The Gauge board instructions is shown in [Figure 5-4](#). Pay attention to the MCU power switch supply jumper. For downloading, connect VMCU to VEx, then the MCU will be supplied with 3.3V, which can ensure the voltage matching with the debugger. For evaluation, connect VMCU to VIn, then the MCU will be supplied with 1.8V-LDO. It can ensure the best analog performance. Besides, when the MCU is powered in around 500ms, the MCU will calibrate the ADC+OPA for current detection. At this time, the current should be 0. Otherwise, there will be a constant current offset.

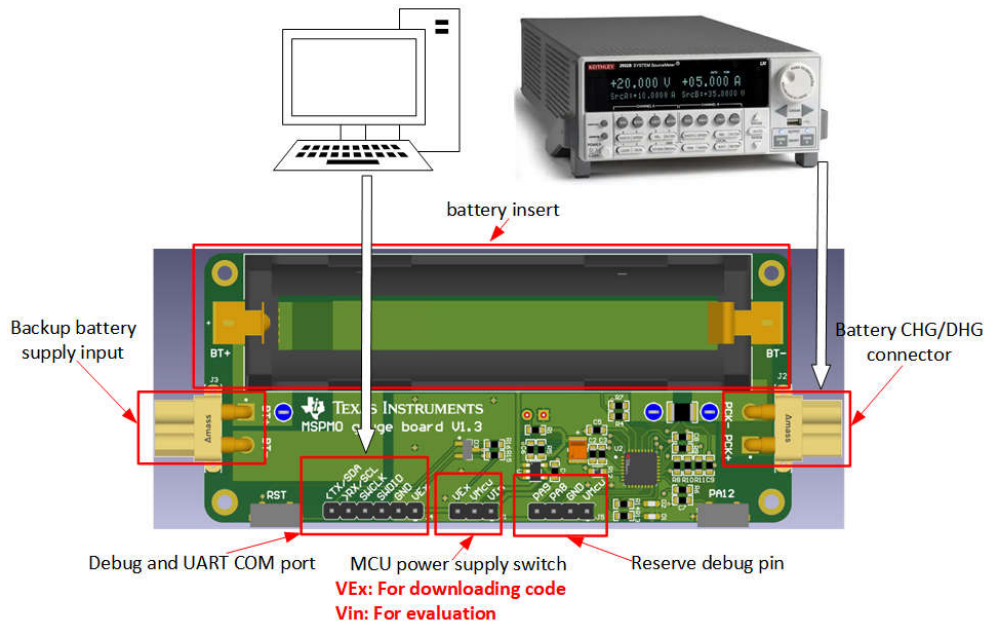


Figure 5-4. Gauge Board Instructions

If you use MSPM0L1306 launchpad in communication data input mode, you need to connect the UART pin as follows, besides of doing the software change.

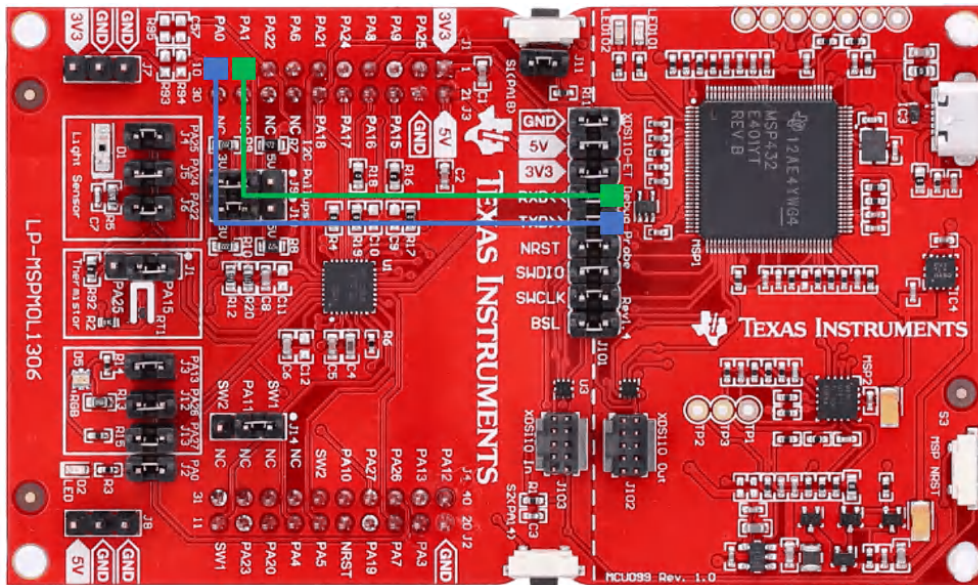


Figure 5-5. MSPM0L1306 Launchpad UART Connection

5.1.2 Software and Evaluation Introduction

Before starting software development and evaluation, it is suggested that you first refer to [MSPM0 Design Flow Guide](#) to have a basic understanding about MSPM0 ecosystem. It will help you quickly catch up the MSPM0 development.

Figure 5-6 illustrates the software project. The project and files related to the gauge algorithm has five parts. For other files, they are same for all the MSPM0 projects.

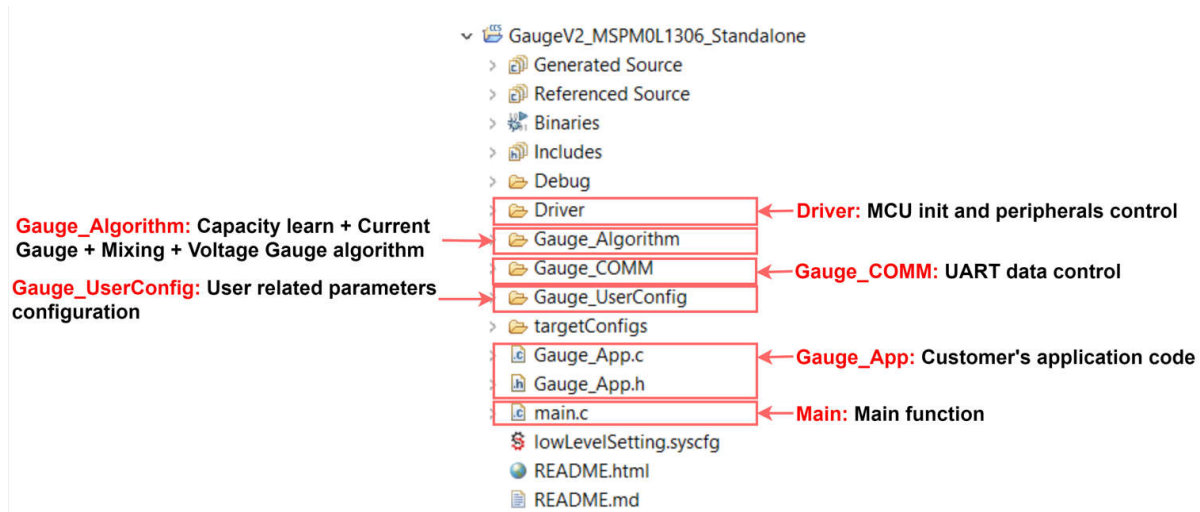


Figure 5-6. MSPM0 Gauge Software Project View

For Gauge_UserConfig part, you will find the description in [Section 4.3](#). The Gauge_Algorithm part introduction is in [Section 2](#). For the Driver part, it includes all the MCU related control. It prepares Icell, Vcell and Tcell data into Gauge_Algorithm. For the Gauge_COMM part, it handles all the UART protocol. For the Gauge_App part, it includes the high-level gauge algorithm calling. This is the place for customers to customize their own functions. For the Main part, it includes the highest system function code.

Remember to follow [Section 4](#) to update the configurations in Gauge_UserConfig folder:

- Generate battery models or use the default one
- Update the configuration in tBattParamsConfig structure
- Update battGlobalParams_xx, and battGlobalParamsArray according to your own battery cells.
- Update the detection mode, communication mode, circuit table length and cell numbers

After you program the MSPM0 through XDS110, you can use the GUI to check and record the results.

5.1.3 Battery Testcases

5.1.3.1 Performance Test

Here is the test based on a 3200 mAh LiCO2 battery, under 25°C. u16MaxFullChgVoltThd setting is 4200mV. EmptyDhgVoltThd setting is 3000 mV.

Note

Make sure the battery is settled before the MCU is powered and the battery is in rest state before testing. Otherwise, the first SOC output is met with an error.

Here is the test pattern:

- Do pulse discharge and pulse charge with different load.
- Constant charge and discharge without rest for 4 cycles with different load .

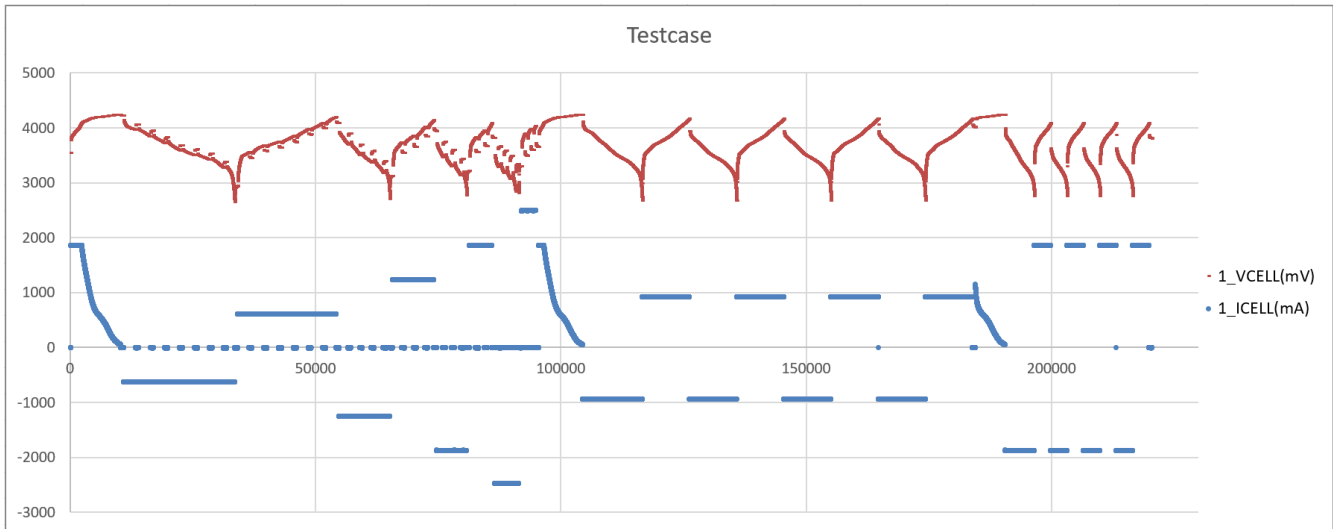


Figure 5-7. Battery Test Case

At the beginning there is an obvious gap for NomSOC, CusSOC and SmoothSOC. See the test results in Figure 5-8. It is caused from first OCV calibration error.

You can find the CusSOC has some gaps at the end of discharge, it is because the influence of EmptySOC. For SmoothSOC, it is flat and no jump at battery rest. All the data is controlled in the 0% to 100% range.

For different NomFullCap, the FitNomFullCap is updated almost after every rest. With the digital filter help, the NomFullCap will get more and more accuracy. After the MaxNomFullCap changes from 0 to a value, it means the output NomFullCap has an acceptable accuracy.

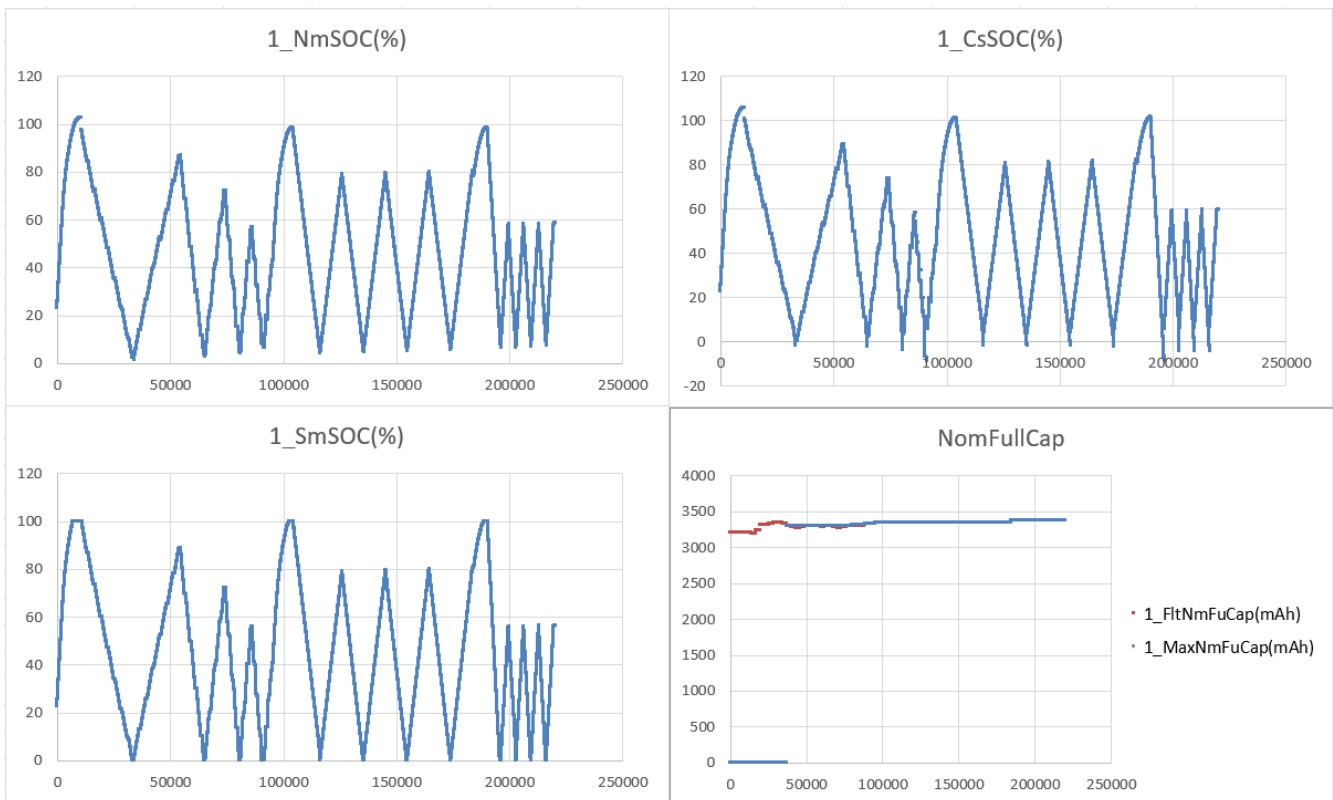


Figure 5-8. Battery Test Result

If you want to check more parameters under debug mode with Q format, you need to right click the value and select the related Q-Value format.

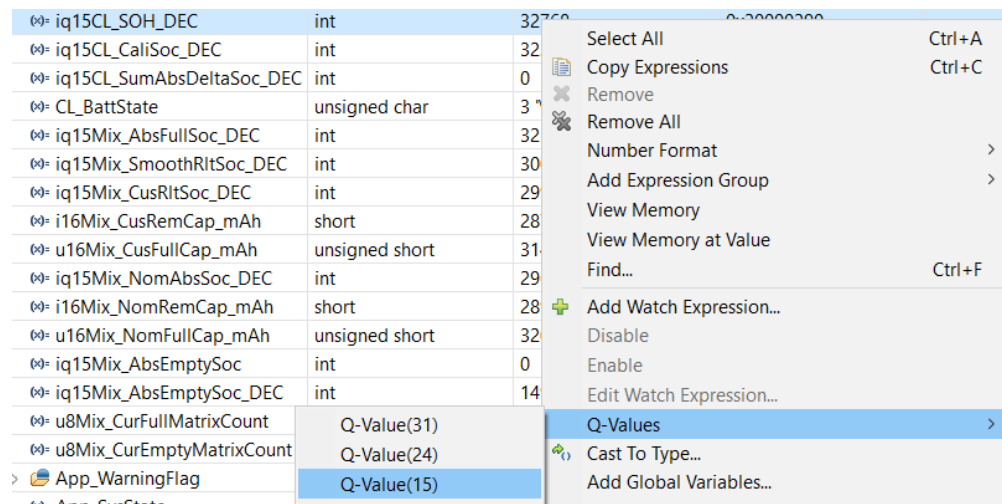


Figure 5-9. Read Q Values

5.1.3.2 Current Consumption Test

As the MSPM0 Gauge board mostly focuses on evaluating the function and doesn't consider how to optimize the power consumption. The current tested based on the Gauge board will be a little high. To optimize it, you need to remove the tantalum capacitor, connect the temperature sensor to GPIO as the GND, and increase the voltage divider resistors.

Here is the current test result, under NO_OUTPUT mode, and removes the tantalum capacitor, the temperature sensor, and voltage divider resistors.

EnergyTrace™ Profile	
Name	Live
System	
Time	56 sec
Energy	1.368 mJ
Power	
Mean	0.0275 mW
Min	0.0073 mW
Max	6.4136 mW
Voltage	
Mean	3.3000 V
Current	
Mean	0.0083 mA
Min	0.0022 mA
Max	1.9435 mA
Battery Life	CR2032: 3 year 30 day (est.)

Figure 5-10. Current Consumption

5.2 MSPM0G3507 + BQ76952 + 4 LiFePO4 Batteries

Solution features:

- Total solution takes about 15K flash and 2.9K SRAM

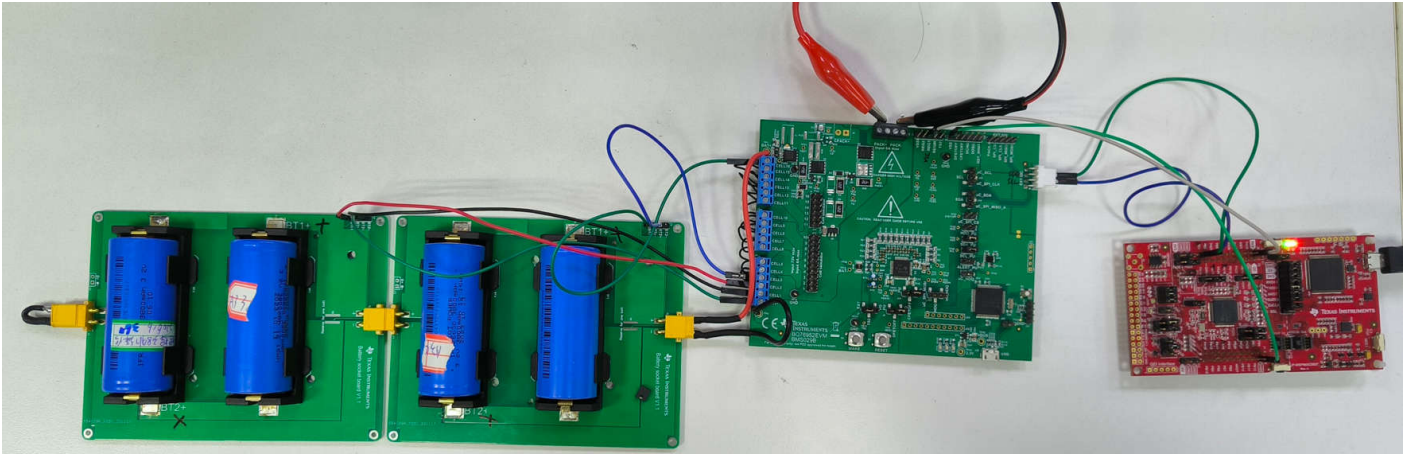
Solution advantage:

- High-performance MCU with CAN integrated
- A combo solution for BMS from TI
- High-performance Gauge algorithm

5.2.1 Hardware Setup Introduction

The hardware board is build up based on MSPM0G3507 Launchpad and BQ76952EVM.

MSPM0G3507 LP + BQ76952EVM



The hardware connection can refer to [Figure 5-11](#). You only need to connect power and I2C between MSPM0 launchpad (SDA: PB3, SCL: PB2) and BQ76952EVM (SDA: J17 PIN3, SCL: J17:PIN2). Remember to remove the cell simulation jumpers and add jumpers for I2C pullup.

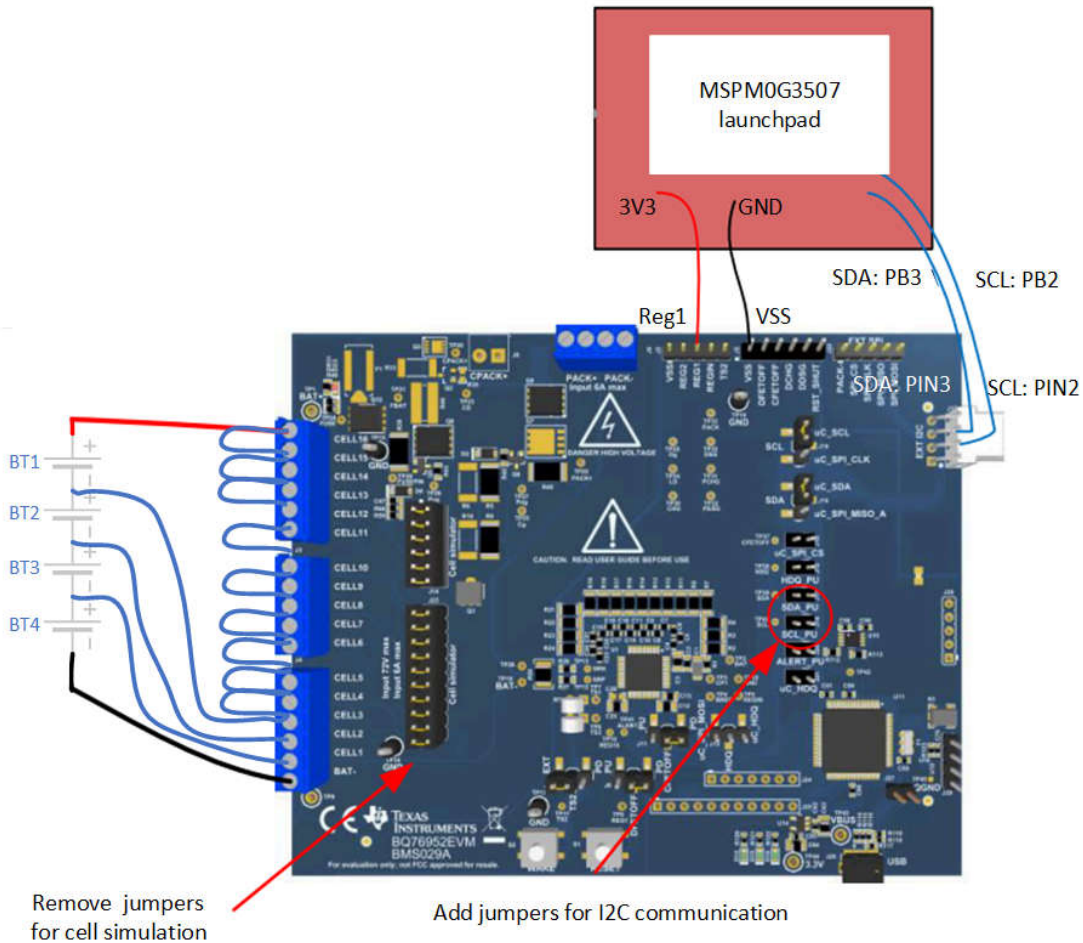


Figure 5-11. MSPM0G3507+BQ76952EVM Block Diagram

If you want to use MSPM0G3507 launchpad in communication data input mode, no other hardware change is needed compared with default MSPM0G3507 launchpad hardware set.

5.2.2 Software and Evaluation Introduction

Before starting software development and evaluation, it is suggested that you first refer to [MSPM0 Design Flow Guide](#) to have a basic understanding about MSPM0 ecosystem. It will help you quickly catch up the MSPM0 development.

Figure 5-12 shows the software project. The project and files related to the gauge algorithm has five parts. For other files, they are same for all the MSPM0 projects.

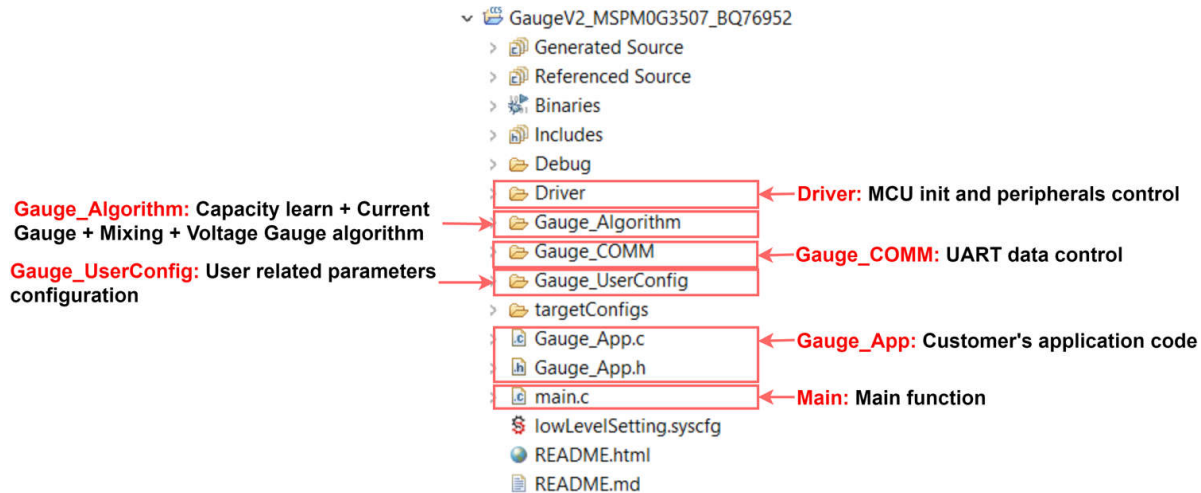


Figure 5-12. MSPM0 Gauge Software Project View

For Gauge_UserConfig part, you will find the description in [Section 4.3](#). The Gauge_Algorithm part introduction is in [Section 2](#). For the Driver part, it includes all the MCU related control. It prepares Icell, Vcell and Tcell data into Gauge_Algorithm. For the Gauge_COMM part, it handles all the UART protocol. For the Gauge_App part, it includes the high-level gauge algorithm calling. This is the place for customers to customize their own functions. For the Main part, it includes the highest system function code.

Remember to follow [Section 4](#), to update the configurations in Gauge_UserConfig folder:

- Generate battery models or use the default one
- Update the configuration in tBattParamsConfig structure
- Update battGlobalParams_xx, and battGlobalParamsArray according to your own battery cells. In this solution, it is four cells.
- Update the detection mode, communication mode, circuit table length and cell numbers

After you program the MSPM0 through XDS110, you can use the GUI to check and record the results.

5.2.3 Battery Testcases

5.2.3.1 Performance Test1 (Pulse Discharge)

Here is the test based on a 3800mAh LiFePO4 battery, under 25°C. u16MaxFullChgVoltThd setting is 3800mV. EmptyDhgVoltThd setting is 2300mV.

Note

Make sure the battery is settled before the MCU is powered and the battery is in rest state before testing, otherwise, the first SOC output is met with an error.

Here is the test pattern: Do pulse discharge for 2 times. The [Figure 5-13](#) shows the condition of a battery Cell in the battery pack. Due to sourcemeter power limitations, only simple tests are run.

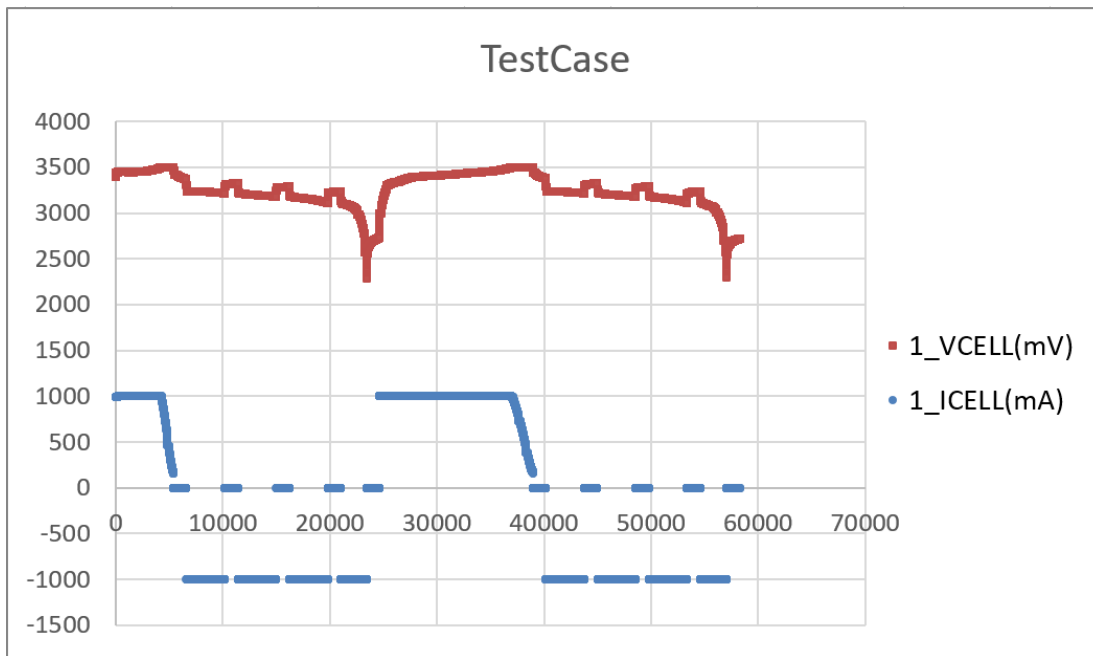


Figure 5-13. Battery Testcase

You can find at the beginning that there is obvious gaps for NomSOC, CusSOC and SmoothSOC. See the test results in [Figure 5-13](#). It is caused from the first OCV calibration error. The reason why it is so high is that the calibration point is under LiFePO4 SOC-OCV flat area.

Here are the results for one cell. With the integrated digital filter, you can find the NomSOC changes follows the real condition very well even in first discharge cycle. However, as the current load is only 1A, you cannot see much influence of EmptySOC in CusSOC.

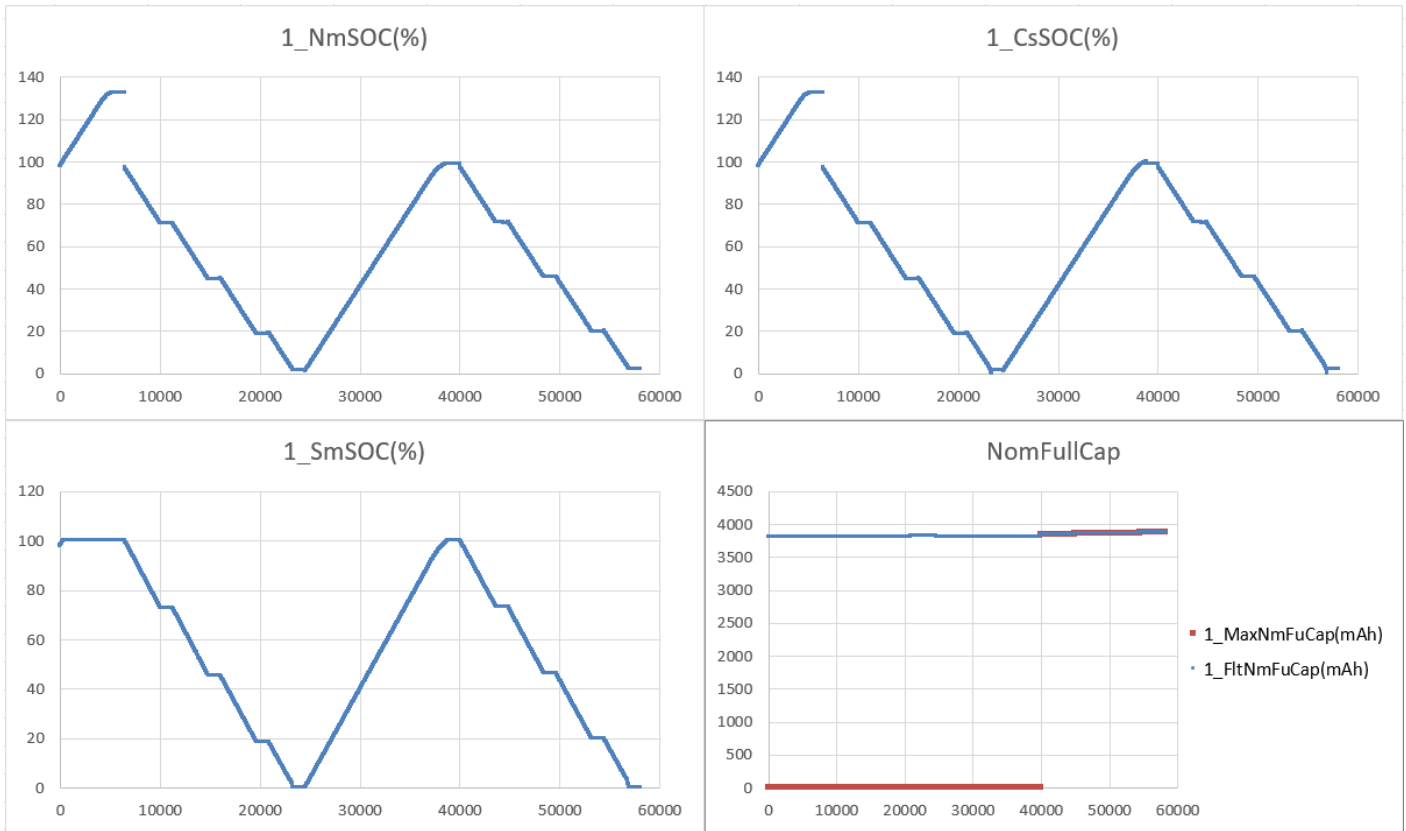


Figure 5-14. Battery Cell Test Result

Here is the result for the battery pack, PackSOC, which follows the minimum SmoothSOC of all the Cells in the battery pack. For PackFullCap, it is the combination of all the CusFullCap, which is influenced by EmptySOC and FullSOC. That is why you will see data jump in the PackFullCap chart. Besides, it is the same for PackRemCap, which is the combination of all the CusRemCap.

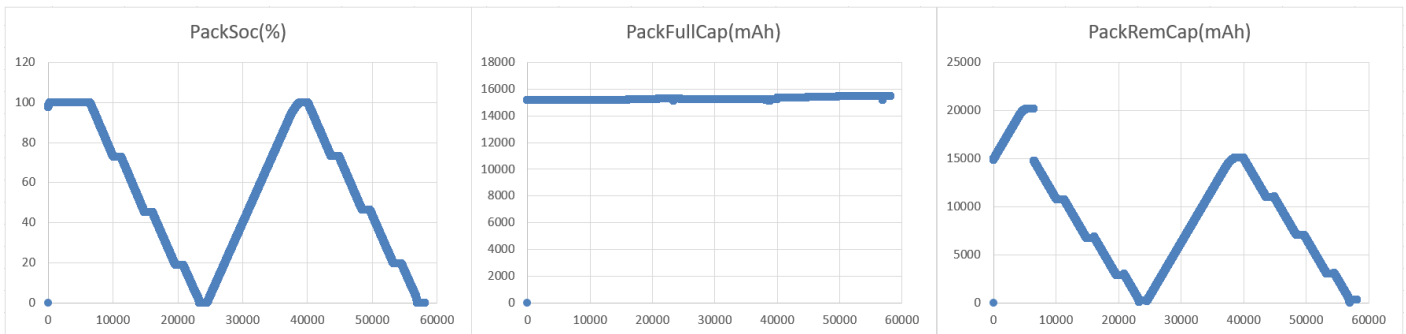


Figure 5-15. Battery Pack Test Result

5.2.3.2 Performance Test2 (Load Change)

Here is the test based on a 3800mAh LiFePO4 battery, under 25°C. u16MaxFullChgVoltThd setting is 3800mV. EmptyDhgVoltThd setting is 2300 mV.

Note

Make sure the battery is settled before the MCU is powered and the battery is in rest state before testing, otherwise, the first SOC output is met with an error.

Here is the test pattern: Do constant discharge 2 times and then change the load. The Figure 5-16 shows the condition of a battery Cell in the battery pack. Due to sourcemeter power limitation, only simple test are run.

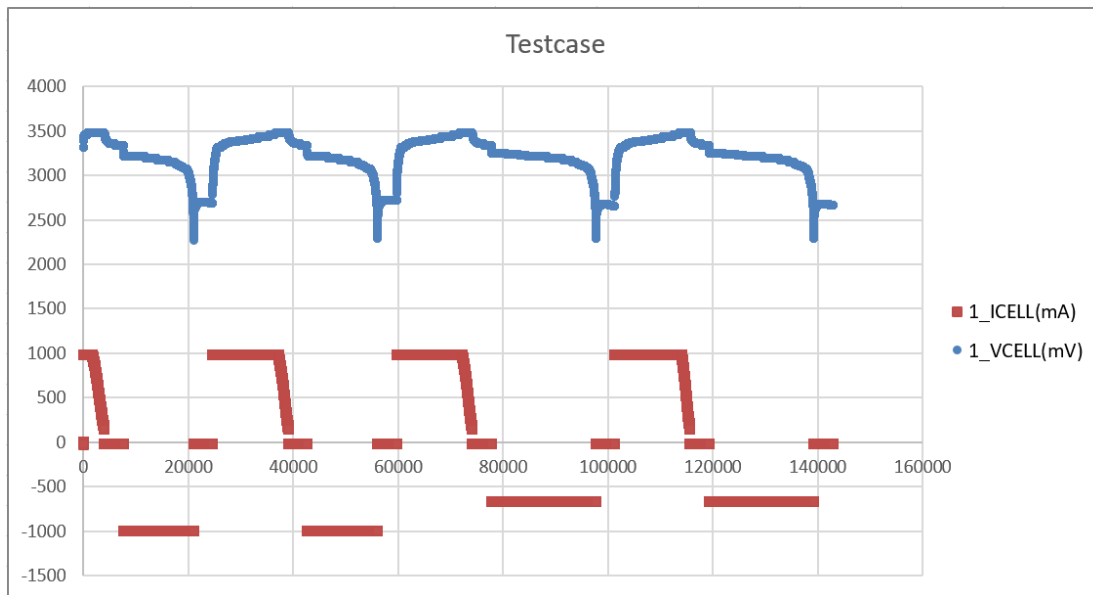


Figure 5-16. Battery Testcase

See the test result in [Figure 5-16](#), you can find at beginning there is a obvious gap for NomSOC, CusSOC and SmoothSOC. It is caused from first OCV calibration error.

Due to residual learn algorithm, you can see that the SmoothSOC can perform perfectly between 0% and 100% when the voltage is reaching the end of discharge voltage (2300mV). Remember at the same time, the EmptySOC needs learning cycles, which means if you do not input iq15AbsEmptySocMatrixInput, the SmoothSOC error will be large when the battery reaches the end of its first discharge voltage.

For different NomFullCap, the FitNomFullCap is updated almost after every rest. With the digital filter help, the NomFullCap gets more and more accurate. After the MaxNomFullCap changes from 0 to a value, it means the output NomFullCap is with an acceptable accuracy.

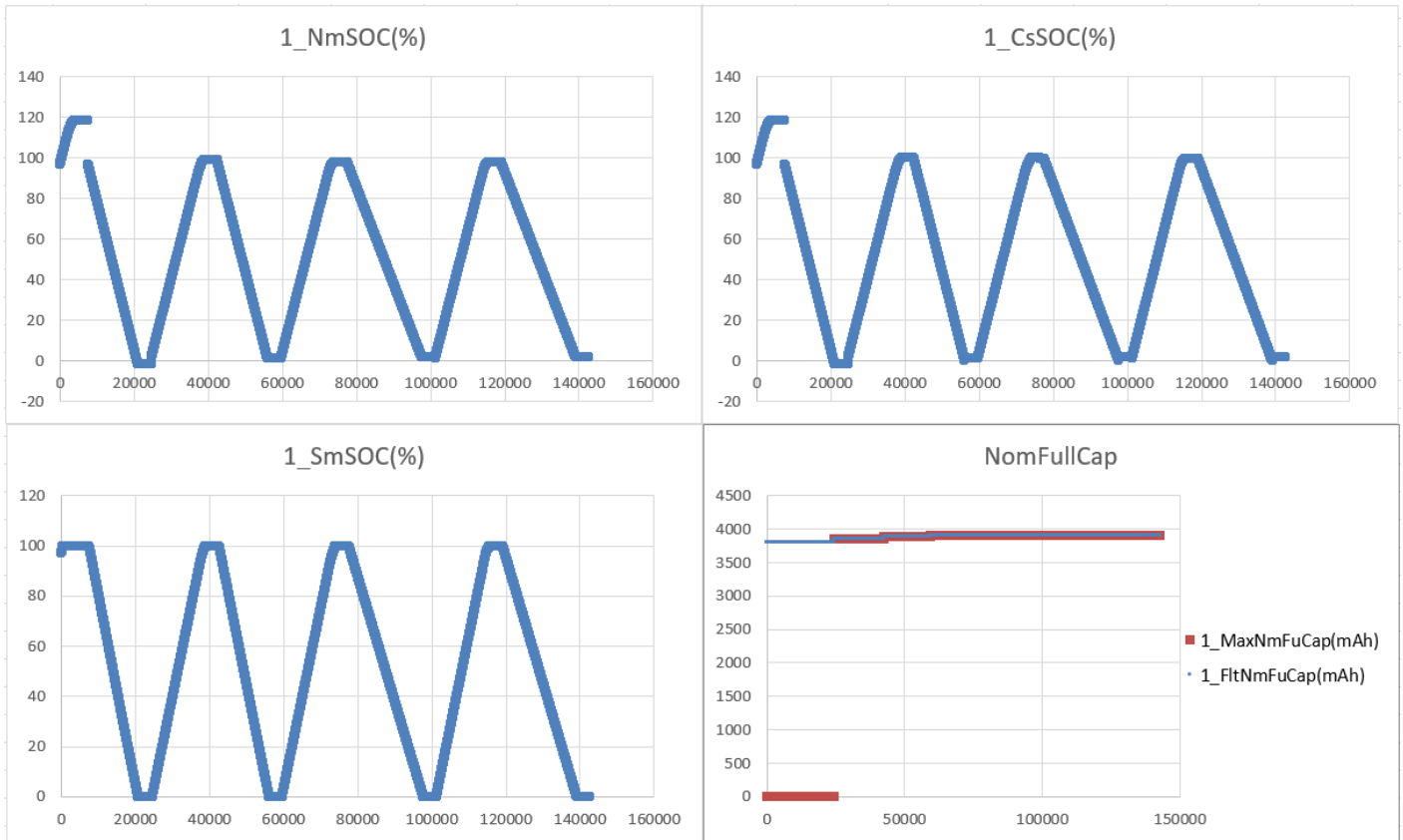


Figure 5-17. Battery Test Result

Figure 5-18 the result for battery pack.

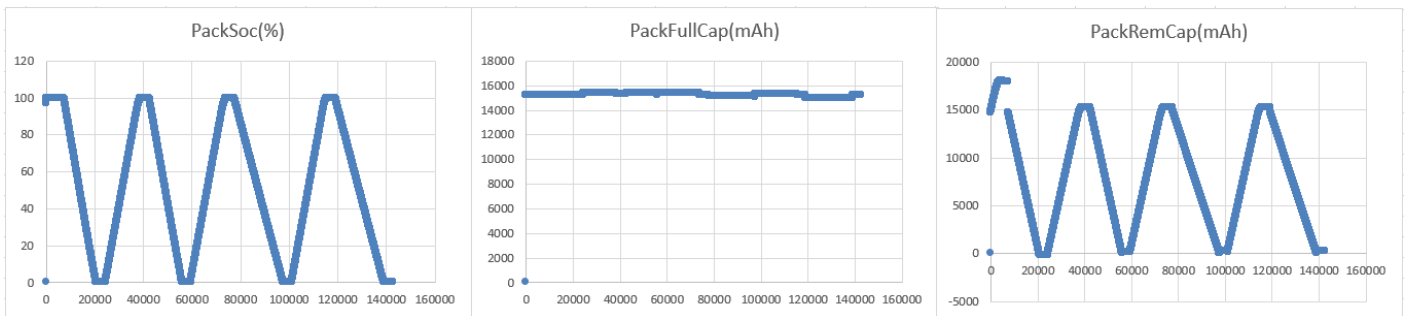


Figure 5-18. Battery Test Result

6 References

- Texas Instruments: [MSPM0 Gauge L1 Solution Guide](#)
- Texas Instruments: [A Self-Calibratable Current Detection Solution Based on MSPM0](#)
- Texas Instruments: [MSPM0 Design Flow Guide](#)

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](https://www.ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2024, Texas Instruments Incorporated