Application Note

MSPM0 Gauge L1 Solution Guide

Eason Zhou

ABSTRACT

This application note is created to build a description about the level1 gauge solution based on MSPM0L130x series. It only detects the voltage to calculate the system-on-chip (SoC) directly. Solution features, hardware introduction, GUI introduction, software introduction, and evaluation follow are included in it.

Project collateral discussed in this application note can be downloaded from the following URL: https://www.ti.com/lit/zip/slaaee3.

Table of Contents

1 Introduction 2
2 Gauge Hardware Introduction 4
3 Gauge Software Introduction 5
3.1 Gauge Algorithm Introduction 5
3.2 Gauge GUI Introduction 6
4 MSPM0 Gauge Evaluation Steps 8
4.1 Step1: Hardware Preparation 8
4.2 Step2: Get the Battery Model 8
4.3 Step3: Input the Customized Configuration 11
4.4 Step4: Evaluation 12
5 MSPM0 Gauge Solution Test Results 15
5.1 Performance Test 15
5.2 Current Consumption Test 16
6 Revision History 16

List of Figures

Figure 1-1. MSPM0 Gauge Hardware Board 2
Figure 1-2. MSPM0 Gauge Software Project 3
Figure 1-3. MSPM0 Gauge GUI Project 3
Figure 2-1. MSPM0 Gauge Board Block Diagram 4
Figure 2-2. Gauge Board Instructions 4
Figure 3-1. MSPM0 Gauge Software Project View 5
Figure 3-2. Battery Model and SoC-OCV Table 5
Figure 3-3. VGauge Software Flow 6
Figure 3-4. MCU COM Tool functions 7
Figure 3-5. SM COM Tool function 7
Figure 4-1. Pulse Discharge Test Case 9
Figure 4-2. Hardware Structure to Get Battery Model 9
Figure 4-3. Battery Circuit Table Generation 10
Figure 4-4. Battery Circuit Table Input 10
Figure 4-5. tBattParamsConfig Structure 11
Figure 4-6. Gauge Mode Setting 12
Figure 4-7. Detection Data Input Mode Structure 12
Figure 4-8. Flash Data Input Mode Structure 12
Figure 4-9. Battery Runfile Generation 13
Figure 4-10. Battery Runfile Copy 13
Figure 4-11. Code Change for Changing Time Step 13
Figure 4-12. Communication Data Input Mode Structure 14
Figure 4-13. Communication Data Input 14

Figure 5-1. Battery Test Case....15
Figure 5-2. Battery Test Result....15
Figure 5-3. Current Consumption Test....16

List of Tables

Table 1-1. MSPM0 Gauge Solution Compare....2
Table 4-1. Battery Test Pattern....8
Table 4-2. MSPM0 Gauge L1 SOC-OCV Range....11
Table 4-3. MSPM0 Gauge L2 SOC-OCV Range....11
Table 4-4. General Configuration Parameters....11
Table 4-5. VGauge Algorithm Related Parameters....11

Trademarks

Python® is a registered trademark of Python Software Foundation.
Microsoft® and Excel® are registered trademarks of Microsoft Corporation.
Keithley® is a registered trademark of Tektronix Inc.
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 L1 solution.

Table 1-1. MSPM0 Gauge Solution Compare

	MSPM0 Gauge L1	MSPM0 Gauge L2
Detected parameters	Voltage; Temperature	Voltage; Temperature; Current
Output key parameters	SOC	S0C; 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

The features of the level1 gauge solution based on MSPM0 are as shown:

- Work after MCU power-on without factory calibration or learning cycles.
- Support SOC (State of charge) and warning flag output.
- Low requirement for battery chemistry parameters input.
- Total solution takes about 6K flash and 1.6K SRAM.
- Current consumption without UART communication (NO_OUTPUT mode) is about 3 µA.

The solution is combined of three parts. All of them can be found at MSPM0 Gauge L1 Development package.

1. The hardware board is used to evaluate the total solution with MSPM0L130x integrated.

Figure 1-1. MSPM0 Gauge Hardware Board

2. The software project based on MSPM0L130x, including the used gauge algorithm.

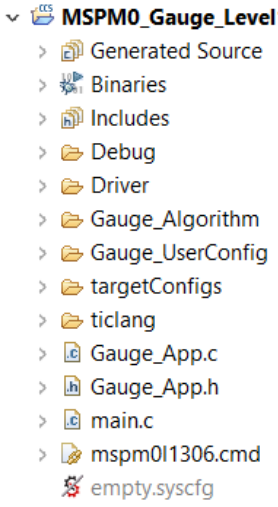

Figure 1-2. MSPM0 Gauge Software Project

3. The GUI is written with Python®, which can be used to communicate with the gauge board, run test pattern by controlling a source meter and generate battery parameters.

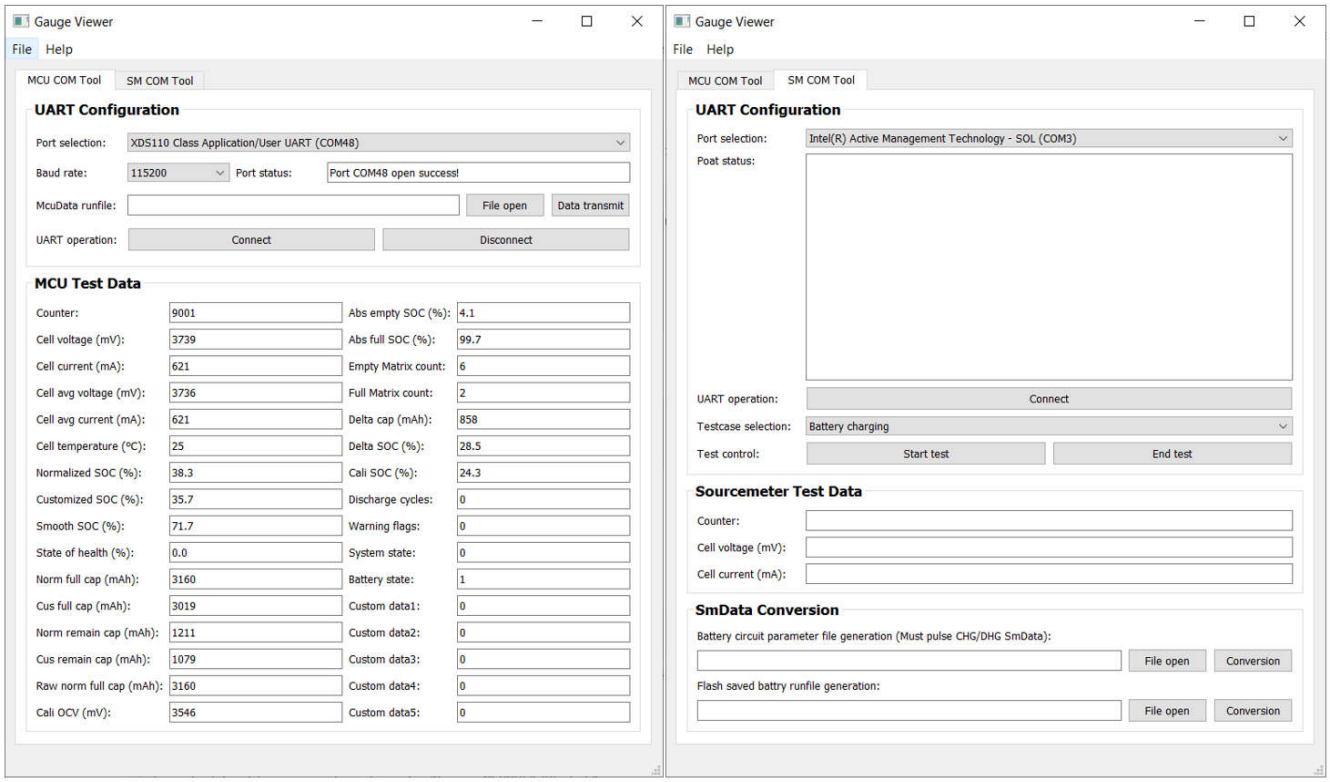

Figure 1-3. MSPM0 Gauge GUI Project

2 Gauge Hardware Introduction

Figure 2-1 shows the hardware high level block diagram. The input parameters are only voltage and temperature tested from ADC channel 1 and ADC channel 5.

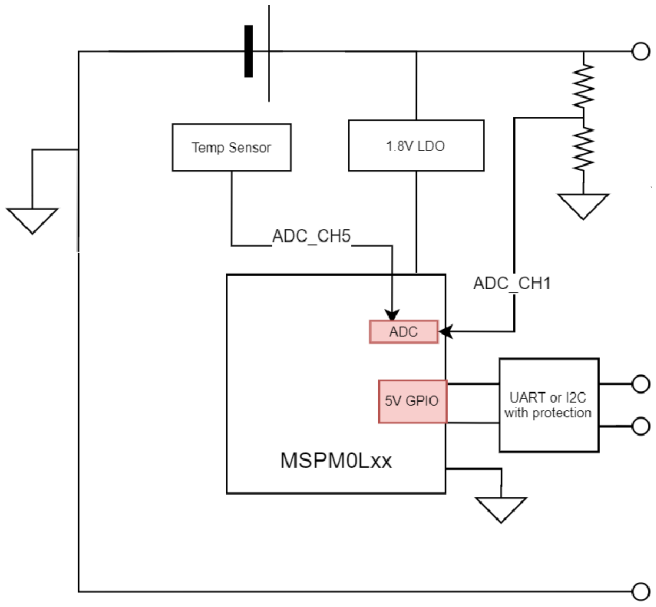

Figure 2-1. MSPM0 Gauge Board Block Diagram

This design can use these MSPM0 features:

- High precision 12-bit ADC for temperature and voltage detection
- 5 V tolerant open-drain I/O with universal asynchronous receiver, transmitter (UART) or I2C function to communicate with masters under different power rails
- Lower to 1.62 V working voltage to support single battery full voltage range
- Lower to 1.1 µA STANDBY current with SRAM retention for battery application

Here we give a quick introduction for the hardware board and how to use it. Insert the battery into the default socket or connect it to the backup battery supply input. Debug and UART COM port is used to connect to PC, which can download the code or communicate with the GUI.

Pay attention for the MCU power switch supply jumper. Connect VMCU to VEx for downloading, then the MCU is supplied with 3.3 V, which can provide the voltage matching with the debugger. Connect VMCU to Vin for evaluation, then the MCU is supplied with 1.8 V LDO and can provide the best analog performance.

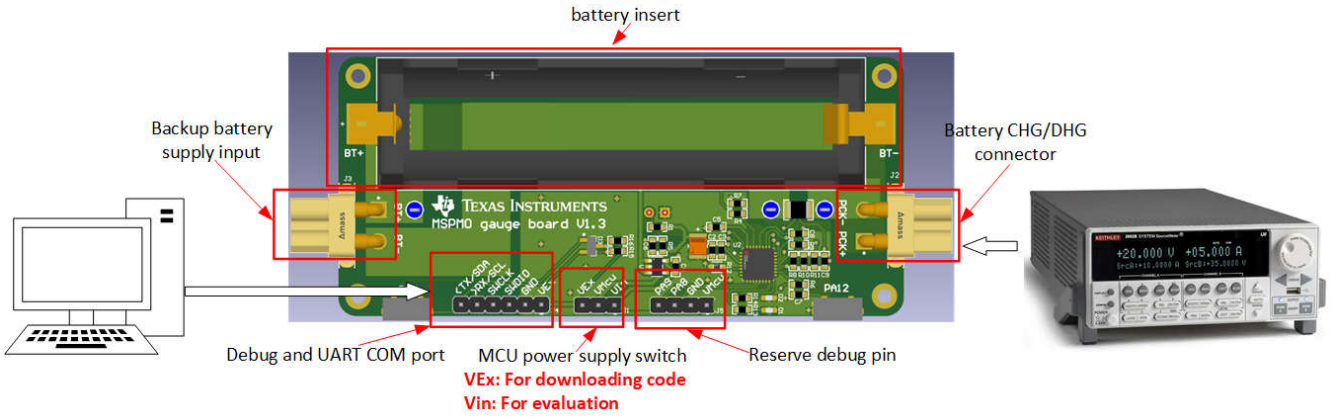

Figure 2-2. Gauge Board Instructions

3 Gauge Software Introduction

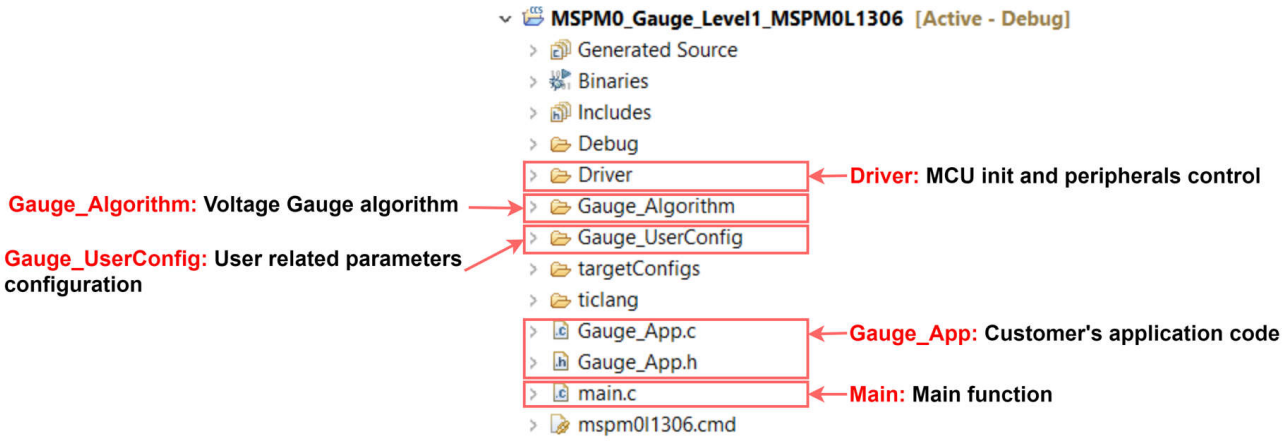

Figure 3-1. MSPM0 Gauge Software Project View

Figure 3-1 shows the software project. The project and files related to the gauge algorithm has six parts. Other files are same for all the MSPM0 projects.

Section 4 is included for the Gauge_UserConfig part.

The the Gauge_Algorithm part is introduced in the last part of this section.

The Driver part includes all the MCU-related peripherals control. The Driver part prepares Vcell, Tcell data into Gauge_Algorithm.

The Guage_App part includes the high-level gauge algorithm calling. This is the place for customers to customize the functions.

The Gauge_Type part includes all the structures used in this project. You can also find some detailed comments.

The Main part includes the highest system function code.

3.1 Gauge Algorithm Introduction

The VGauge treats the battery as a first-order RC model then uses the RC model and the SoC-open circuit voltage (OCV) table to generate VF_SoC. As it uses a low-order circuit model to simulate the battery, the accuracy of VF_SoC is not so high. However, it can help if you know the battery SoC when you do not detect the battery current or do not know the full capacity (AbsFullCap) at the beginning. In the software code, the RC model and SoC-OCV table are saved in “circuitParamsTable”.

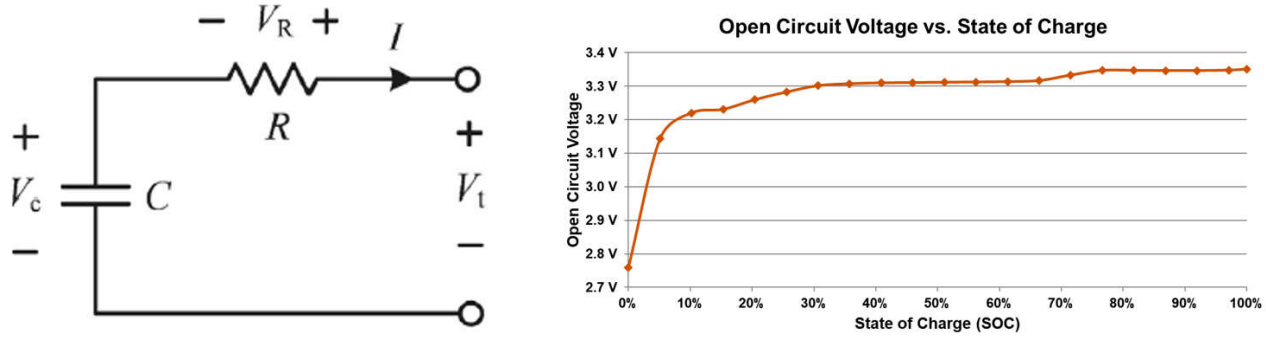

Figure 3-2. Battery Model and SoC-OCV Table

Figure 3-3 shows the software flow chart of VGauge function. The circuitParamsTable (Rcell lookup table and SOC-OCV lookup table) and Qmax are per saved. When the MCU starts working the MCU treats the first AvgVcell as the OCV[K-1], then reads the SOC-OCV table to find the SoC. Rcell and Ccell are calculated and inputted in the model. With the AvgVcell input, a new OCV[K] is calculated, which is treated as a new OCV[K-1] inputted in the model in the next calculation cycle.

In a word, this model is used to evaluate the OCV based on the battery parameters and the AvgVcell input. The SoC is get by searching the SoC-OCV table.

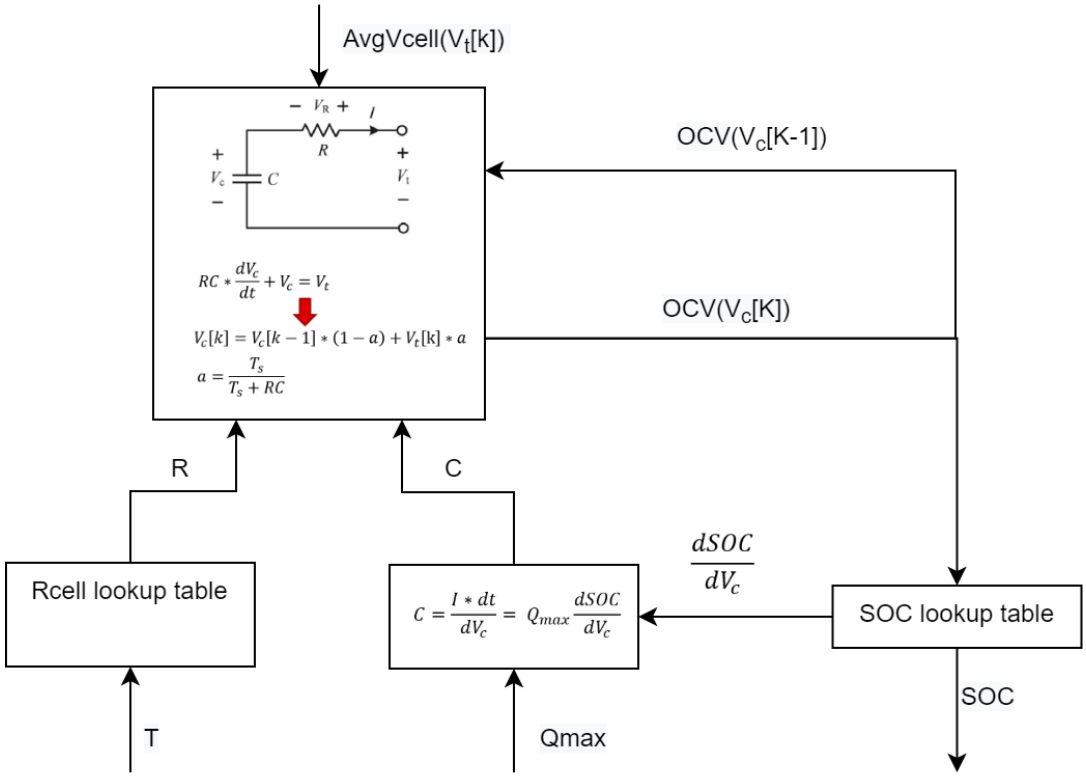

Figure 3-3. VGauge Software Flow

See Current sensor-less state-of-charge estimation algorithm for lithium-ion batteries utilizing filtered terminal voltage for more details about VGauge.

3.2 Gauge GUI Introduction

Gauge GUI is also a important part of this design and can be used for recording MCU data, running battery test case, and do data conversion. This GUI has two 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. Data conversion is also done in this page to pair with the different gauge working mode.

First, observe the MCU COM Tool with two functions shown in Figure 3-4. The first function is to receive the battery running data from MCU. The data is saved automatically in Microsoft® 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).

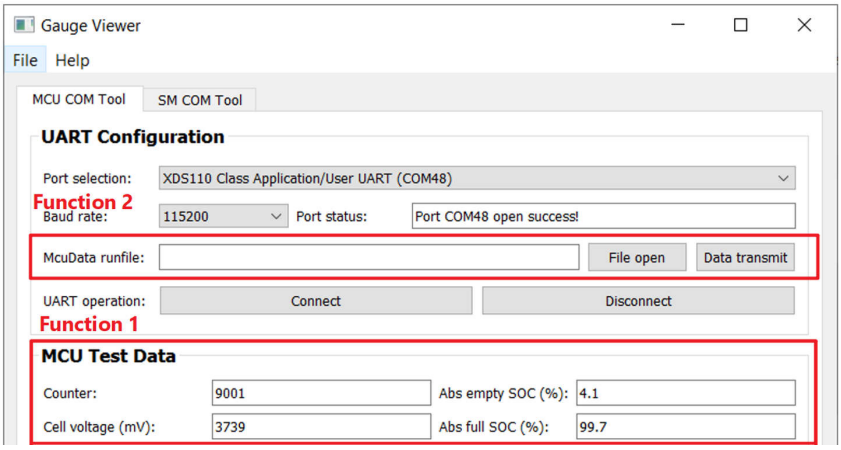

Figure 3-4. MCU COM Tool functions

The SM COM Tool has two functions, shown in Figure 3-5. Function 1 is used to control the source meter to run the battery test case. Then show and record 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, buy a USB-to-rs232 wire and a Keithley® 2602A source meter.

Function2 helps to converge the record data into C files. “Battery circuit parameter file generation” is used to extract the battery parameters, including SOC, OCV and Rcell from a pulse CHG/DHG file, to generate the “circuitParamsTable”. “Flash saved battery runfile generation”, is used to converge the record file into a C file. Then you can save the file into MCU for running, paired with the related gauge mode (Flash data input mode).

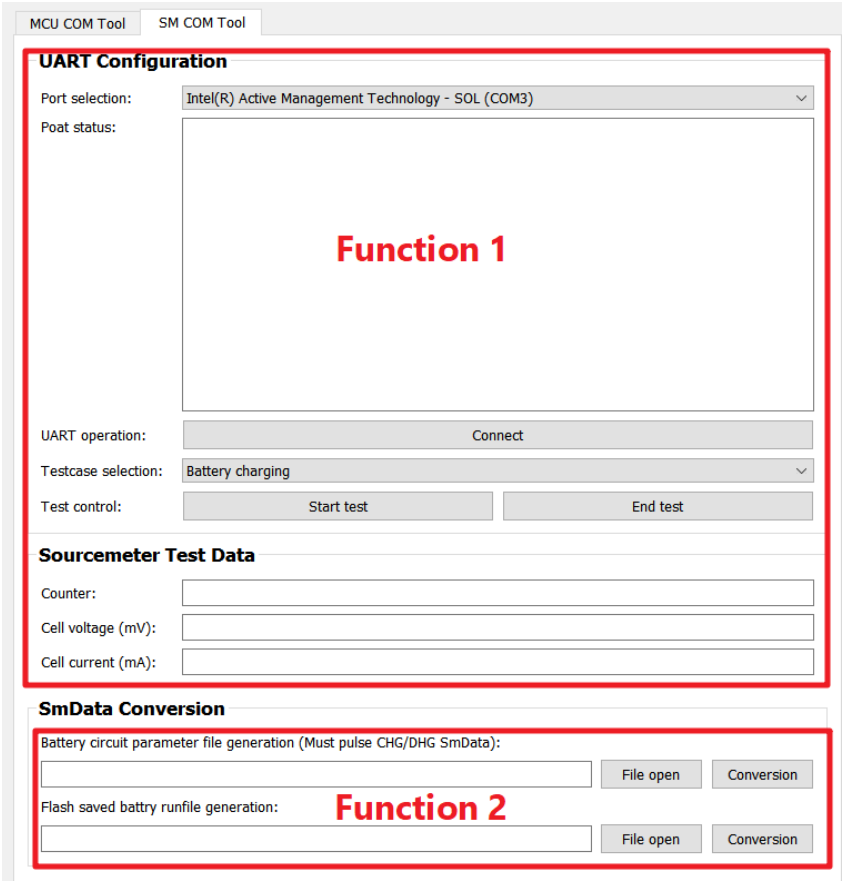

Figure 3-5. SM COM Tool function

A Gauge GUI execution file is supplied; you can use the file for evaluation without installing python. However, if you want to customize the battery test cases under the SM COM Tool, TI recommends that you to use the source code. See the next section for more details about how to use the GUI.

4 MSPM0 Gauge Evaluation Steps

4.1 Step1: Hardware Preparation

Hardware board:

If you want to evaluate this total solution, you need to make the gauge board first. If you only want to evaluate the gauge software, you just need a MSPM0L1306 launchpad and input the prepared voltage and temperature data into MSPM0 Gauge algorithm.

Test setup:

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 is also helpful to use a thermo stream to evaluate the gauge performance under different temperature.

4.2 Step2: Get the Battery Model

Get the battery model from the pulse discharge test case. It is always good to get the battery model for your project. However, for MSPM0 Gauge L1 with low discharge current in real application, you do not really need to do the test. 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 accuracy lies on the battery model, obtain the dedicated battery model.

Use any machine that can charge and discharge the battery for the test machine, and record the tested data. 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, discharge the battery with low current, like 0.1C for 20 minutes. The rest time after each pulse is about 1–2 hours, then you can take the Vcell as OCV. Finally, with this setting about 30 points are obtained.

Table 4-1 shows a suggested test pattern.

Table 4-1. Battery Test Pattern

Parameter	Value	Comment
Start voltage (OCV)	4.3~4.4 V	Make sure the start voltage is no lower than the application max charge voltage
End voltage (OCV)	2.5~3.0 V	Make sure the rest voltage (OCV) is no higher than the application min discharge voltage
Discharge current	0.05C ~ 0.1C	Low current means more point
Discharge time	20 minutes	Low discharge time means more point
Rest time	1-2 hours	Longer is better

Figure 4-1 shows a battery model example test case. In this example, the battery is fully charged (4350 mV) and rests for 1 hour, with the voltage drops to 4322 mV. Then, it does a pulse discharge and rests to get the OCV under different SoC. The test is terminated at 2450 mV. After 1-hour rest, the voltage increases to 2864 mV. So, the OCV range of the SoC-OCV table is from 2864mV to 4322 mV. The start voltage is 4322 mV and the end voltage is 2864 mV under the difference between the OCV and battery voltage.

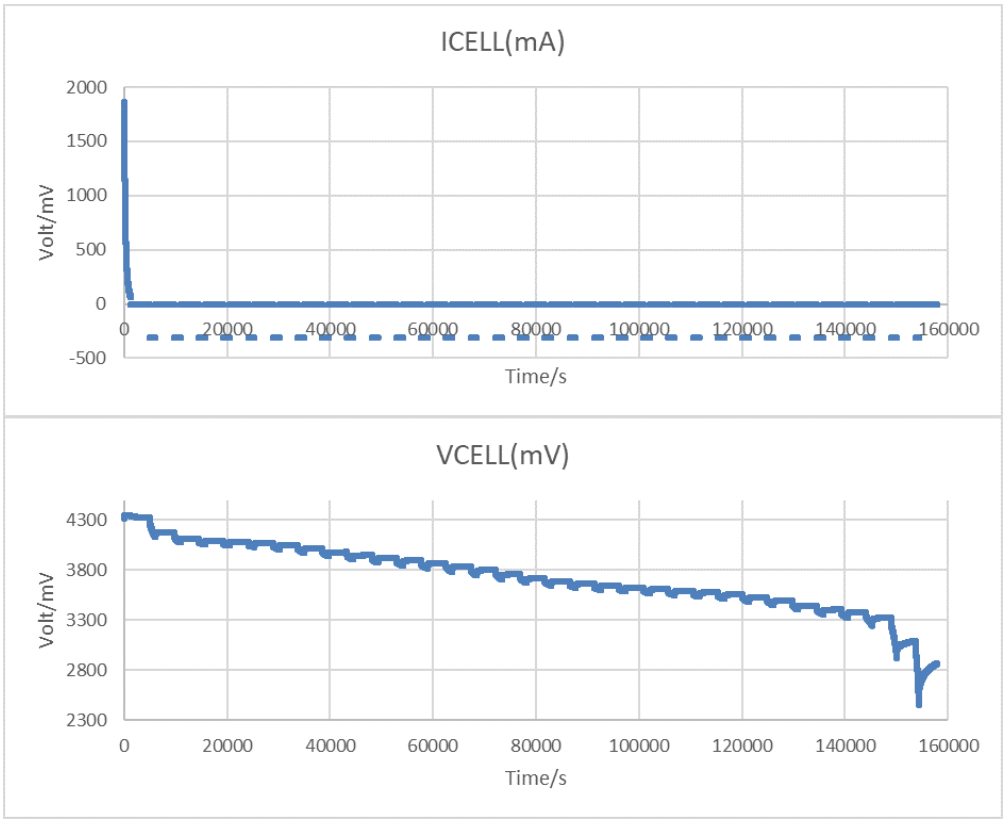

Figure 4-1. Pulse Discharge Test Case

The suggested setup is shown as below. MCU COM tool is used to get the battery run data. 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.

Pay attention to connect source meter in four wire mode, which can reduce the voltage detection error caused from line resistance. Test the battery under real application board, because the application board influences the battery parameters as well, especially the battery resistor.

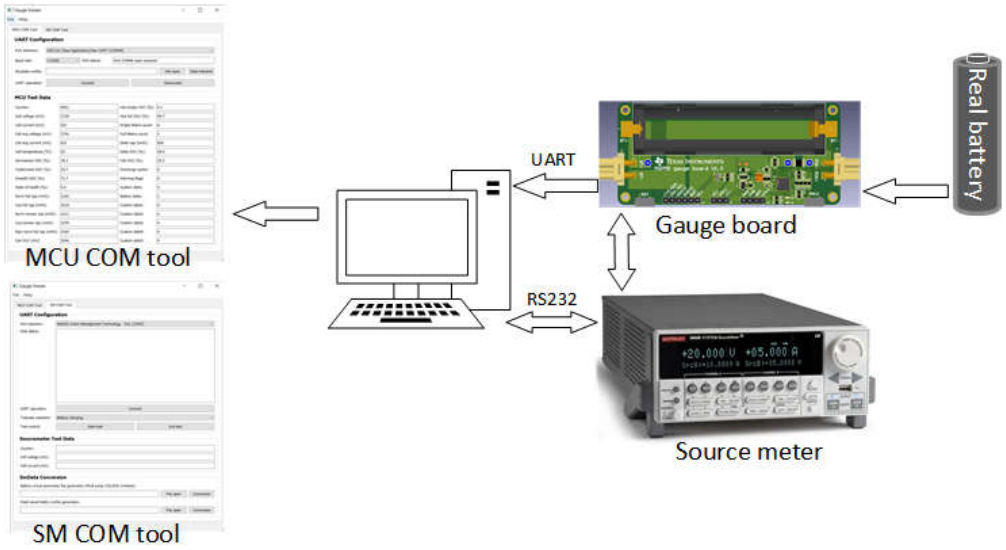

Figure 4-2. Hardware Structure to Get Battery Model

Remember to change the parameters in the Python source code, like the discharge current, end voltage and so on, according to your application. After you get battery running data, you can then use “Battery circuit parameter file generation” to get the battery circuit file in Excel and text, shown in Figure 4-3. The input file can be SMData and MCUData.

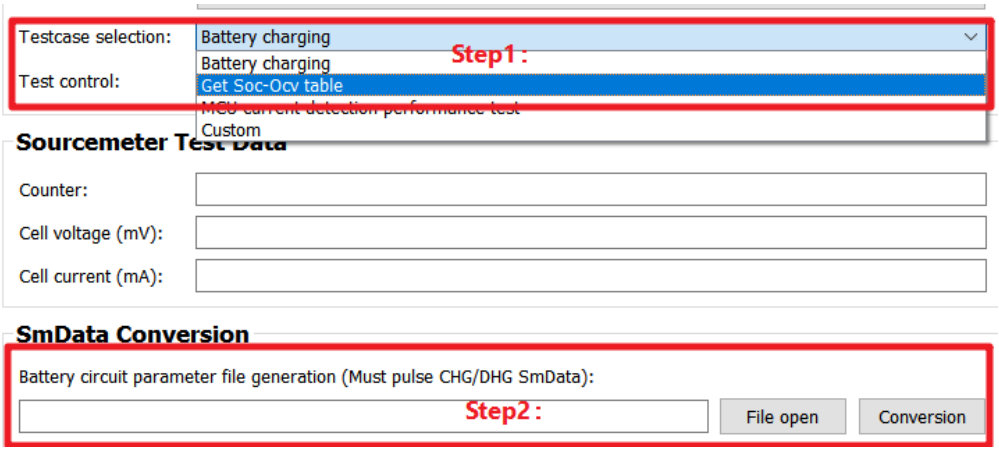

Figure 4-3. Battery Circuit Table Generation

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. The cap factor equals to dSoc(%)/dOcv(mV)*Qmax(As) or dSoc(%)/dOcv(V)*3.6*Qmax(mAh). For other parameters generation method, see the Python source code.

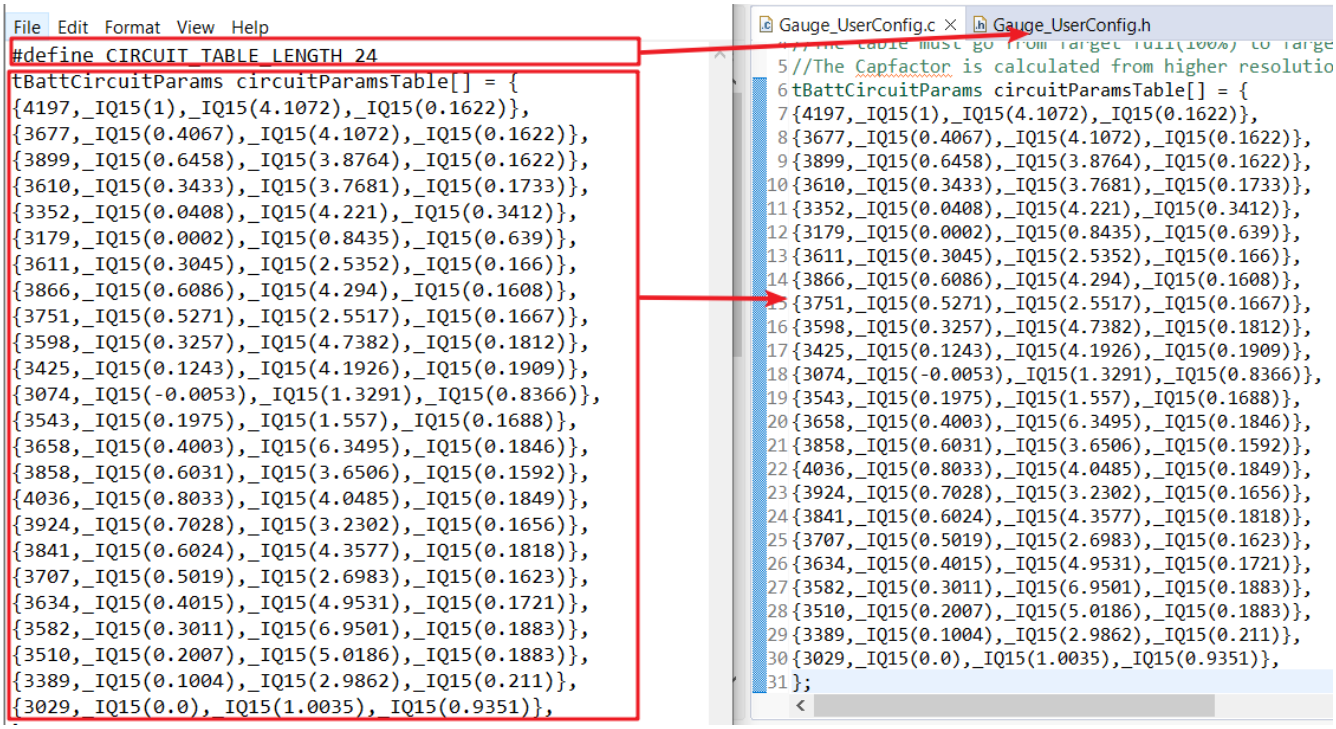

Figure 4-4. Battery Circuit Table Input

Attention: For MSPM0 Gauge L1, it calculates a static SoC, which does not take the residual SoC or battery aging into consideration. That means you need to add some buffer by setting the minimum OCV of SoC-OCV table higher than application min discharge voltage. Table 4-2 shows an example for the suggested minimum OCV for different discharge conditions.

Table 4-2. MSPM0 Gauge L1 SOC-OCV Range

	Current < 0.01C	Current < 0.1C	Current < 0.5C
Table max OCV	4.2 V	4.2 V	4.2 V
Table min OCV	3 V	3.2 V	3.4 V
Application max charge voltage	4.2 V	4.2 V	4.2 V
Application min discharge voltage	2.8 V	2.8 V	2.8 V

For high-level MSPM0 Gauge solution, as it takes the residual SoC or battery aging into consideration, make the circuit table larger than the application voltage range to reserve some buffer. Table 4-3 shows an example for different discharge conditions.

Table 4-3. MSPM0 Gauge L2 SOC-OCV Range

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

4.3 Step3: Input the Customized Configuration

You need to fulfill the configurations of the “tBattParamsConfig” structure in “Gauge_UserConfig.c”. For easy evaluation, change the general configuration parameters.

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 4-5. tBattParamsConfig Structure

Divide these parameters into two parts. A short description for all these related parameters is in Table 4-4.

Table 4-4. General Configuration Parameters

Parameters	Comment
u16DesignCap_mAh	Just input the standard capacity of battery or the tested battery capacity through battery parameter generation test.
u16MinBattVoltThd_mV u16MaxBattVoltThd_mV i8MaxTempThd_C i8MinTempThd_C	Battery Vcell, Tcell threshold. These are reserved to control warning flags when the battery situation is above these parameters.
u8AvgBattParamsUpdateCount	This parameter tells the average data is received after the settled cycles.
u8SysTikShift sysTikFreq	Choose the algorithm running frequency.

Table 4-5. 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 a different temperature. It is by experience and does not affect the performance too much.

4.4 Step4: Evaluation

Before you start, Figure 4-6 shows some settings related to the evaluation in Gauge_UserConfig.h.

Gauge_UserConfig.h

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

//*****************************Algorithm data output mode selection**
//#define OUTPUT_MODE     (NO_OUTPUT)
#define OUTPUT_MODE     (UART_OUTPUT)
```

Figure 4-6. Gauge Mode Setting

For different output modes, UART_OUTPUT means enable data output through UART1. Then you can observe the battery running parameters on the GUI. NO_OUTPUT means terminate the UART data output. This is a good way to debug the algorithm by running it for many cycles in a short time.

The different detection mode is detailed in the next section.

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 MSPM0 analog peripherals. The GUI can help to record the battery running data for further analysis.

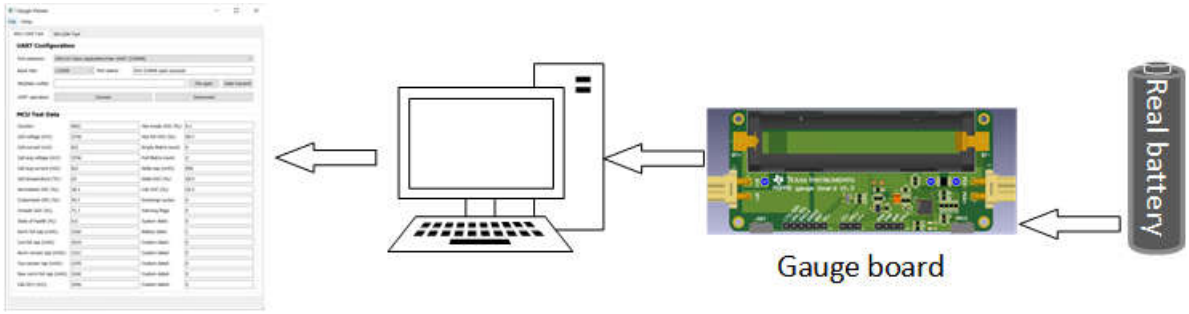

Figure 4-7. Detection Data Input Mode Structure

4.4.2 Flash Data Input Mode

This mode means the battery running data (Vcell, Icell, Tcell) is saved into MCU. This method can remove the need of hardware and increase algorithm running frequency. As you do not need the UART communication, the running frequency is the fastest.

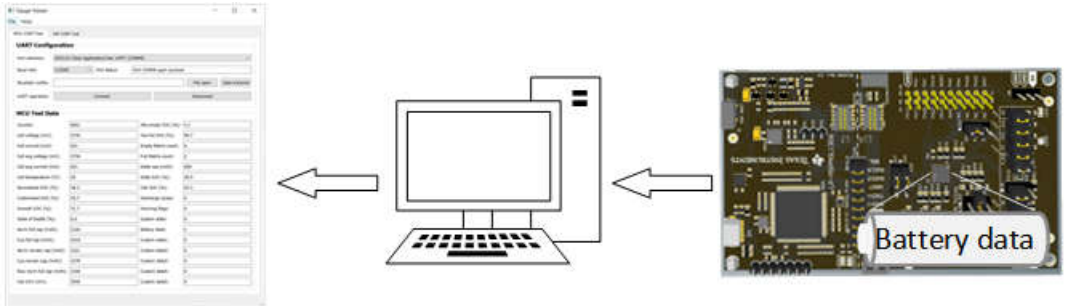

Figure 4-8. Flash Data Input Mode Structure

To realize this method, you need to use the “Flash saved battery runfile generation” function to convert the SMData file or McuData file into C code.

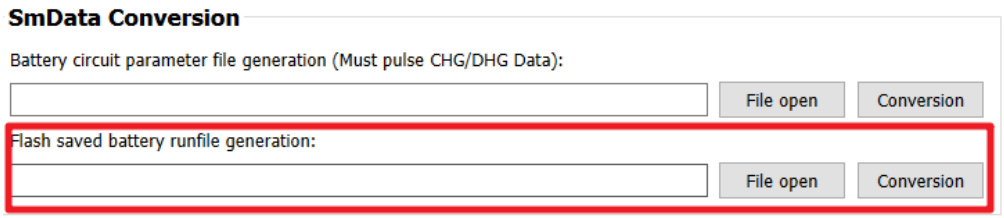

Figure 4-9. Battery Runfile Generation

Copy the code from txt to the C file. After changing the definition of detection mode, run the algorithm with a single launchpad.

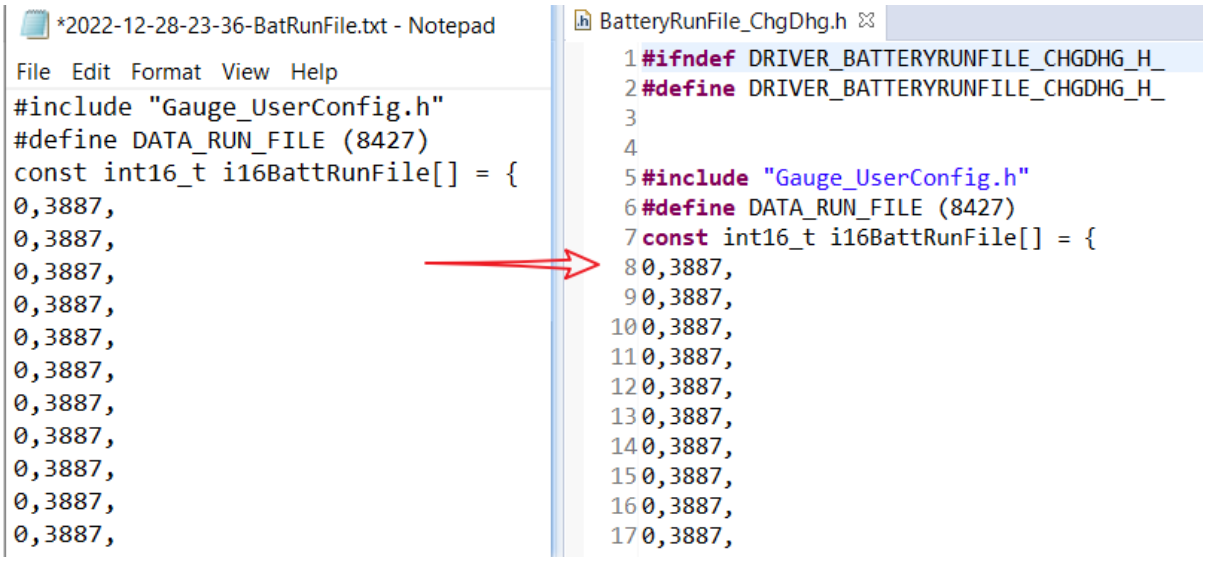

Figure 4-10. Battery Runfile Copy

Remember, as the MCU memory size has limitations, the battery run file input into the MCU cannot be unbounded. If you want to run a long battery cycle test case, you need to change the time step in python code and C code. Reducing the u8AvgVcellDetectPeriod at the same time, if needed.

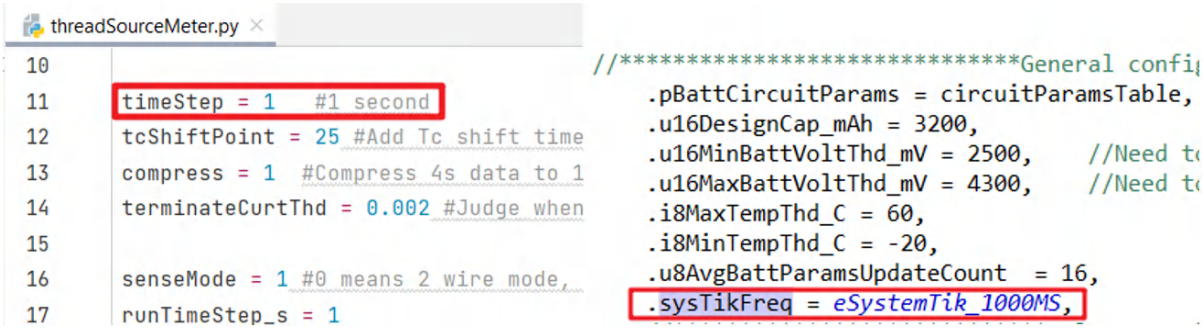

Figure 4-11. Code Change for Changing Time Step

4.4.3 Communication Data Input Mode

The battery running data is input from the GUI for this mode and 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.

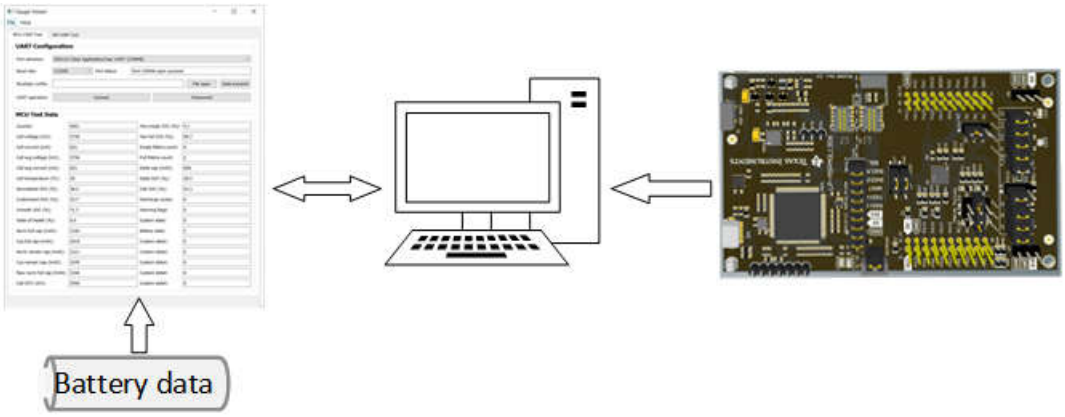

Figure 4-12. Communication Data Input Mode Structure

To realize this method, first connect the UART COM port and load the MCUData run file in MCU COM Tool. After clicking the data transmit button, wait until the port status changes to “Start transmition!”, shown in Figure 4-13. The data load time and Excel save time is long if the file is very large, about 5 to 10 minutes.

The battery running data is returned from MCU. The benefit of the method is that you can load the battery running data over and over again to improve the algorithm and parameter setting.

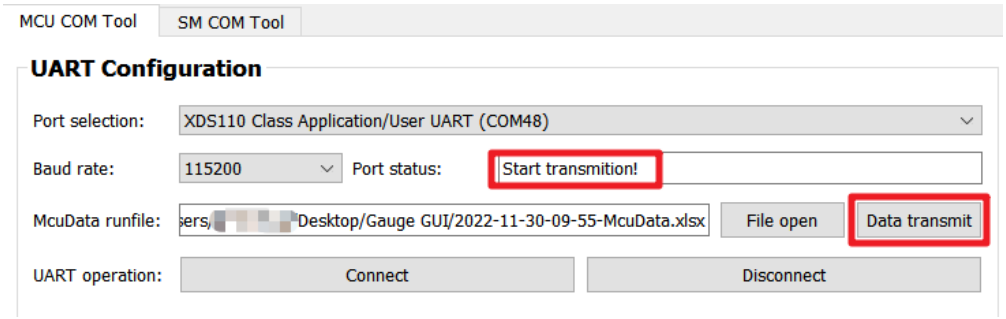

Figure 4-13. Communication Data Input

5 MSPM0 Gauge Solution Test Results

5.1 Performance Test

See the performance of MSPM0 Gauge L1. Here is the test based on a 3100 mAh Lion battery, under 25C.

Here is the test pattern:

1. Charge battery to full (4.25 V) and rest 1 hour.
2. Discharge battery to empty (2.5 V) and rest 1 hour, with 0.5C / 0.3C / 0.1C.
3. Pulse discharge battery with 0.3C.

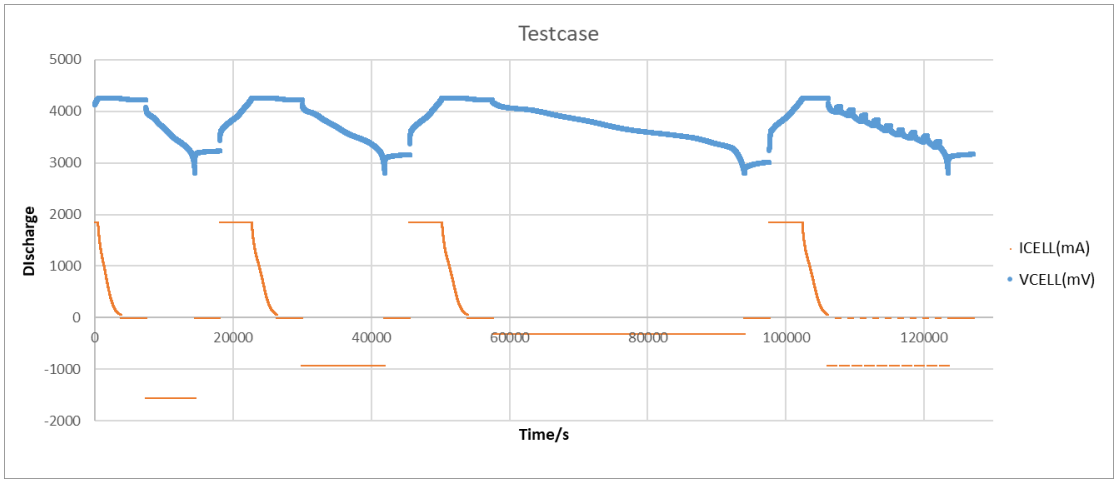

Figure 5-1. Battery Test Case

The battery full capacity is calculated based on the delta SoC get after 1 hour rest and accumulated capacity, then generates the test result. The conclusions are shown in the test results.

- This solution is an excellent choice for constant low-current discharge. You can find that under 0.1C discharge, the SoC error is within 2%. When turns to pulse discharge, the SoC error increases.
- The SoC error increases when the current is high, because of the battery model latency. The maximum error for 0.5C discharge is about 9%. The maximum error for 0.3C discharge is about 4%. The maximum error for 0.1C discharge is about 2%.
- This solution is mostly suitable to output steps instead of percentage to the end user. As it does not test current, the SoC can still change when the battery is resting.

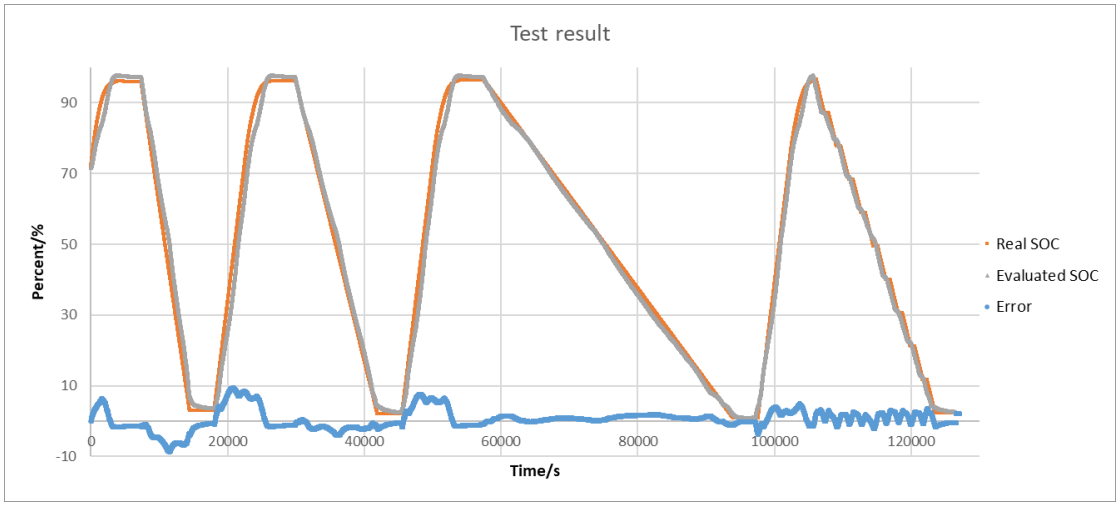

Figure 5-2. Battery Test Result

The advantage of this solution is that:

- Simple hardware setup, only need voltage detection. And that means low average current
- Small software code size
- No accumulating error, compared with coulometer

As this solution mostly lies on the accuracy of battery model, temperature or aging are also two important factors for SoC calculation, which are not considered in this solution. To increase this solution accuracy under more conditions, record the SoC parameter trend affected by temperature or aging, and add factors into the battery model calculation.

5.2 Current Consumption Test

As the MSPM0 Gauge board mostly focuses on evaluating the function, the current test based on the Gauge board is 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. To further improve the current consumption, you can first reduce the ADC sampling and averaging times. Second, reduce the Vcell averaging times (u8AvgBattParamsUpdateCount) and reduce CPU wake-up frequency (sysTikFreq).

Here is the current test result, about 3 µA average current, under NO_OUTPUT mode, and removes the tantalum capacitor, the temperature sensor, and voltage divider resistors. Just used to show the power consumption of MSPM0.

Name	Live
Time	10 sec
Energy	0.094 mJ
⌄ Power	
Mean	0.0093 mW
Min	0.0050 mW
Max	0.0129 mW
⌄ Voltage	
Mean	3.3000 V
⌄ Current	
Mean	0.0028 mA
Min	0.0015 mA
Max	0.0039 mA

Figure 5-3. Current Consumption Test

6 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (July 2023) to Revision A (June 2024)	Page
• Changed content in Table 1-1	2

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 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