Qualcomm Technologies, Inc.

Understanding PMI8998 Fuel Gauge

Application Note

80-VT310-138 Rev. B

March 14, 2017

For additional information or to submit technical questions, go to: https://createpoint.qti.qualcomm.com

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	October 2016	Initial release
B	March 2017	▪ Sections 1.1 Features, 1.2 General description, and 1.4 Fuel gauge pin descriptions: updated to remove mention of the external sense option ▪ Sections 1.5 Generation 3 fuel gauge (PMI8998) vs. previous generation 2 fuel gauge (PMI8996 and PMI8952) and 1.6 Tasks and configurations needed for fuel gauge commercialization: added ▪ Section 2.1.1 Fuel gauge current sensing: updated to remove mention of the external sense option ▪ Sections 2.1.3.2 ESR measurements during discharge and 2.1.3.3 ESR measurements during charge: updated ESR measurement information ▪ Figure 2-1 Round-robin ADC channels and measurement order: updated for v2.1 ▪ Sections 2.1.4.2.2 Overview and 2.1.4.2.4 Capacitance on BATT_THERM: updated for clarity ▪ Table 2-2 Thermistor capacitance values: removed ▪ Table 2-3 AUX_THERM pull-up resistor scaling ratios: updated to new values ▪ Sections 2.1.4.3 AUX_THERM skin temperature beta table and 2.1.4.3.1 AUX_THERM skin temperature thermistor pull-up selection guide: updated for clarity ▪ Sections 2.1.4.4 Battery identification and 2.1.4.5 RID selection: updated for clarity ▪ Section 2.1.5 Battery missing detection: updated for clarity ▪ Sections 2.1.6.1 Charge termination to 2.1.6.3 Auto recharge: corrected incorrect JEITA limit naming convention ▪ Section 2.1.7 Battery current limiting: updated with new information ▪ Section 2.2.1.4 System SoC: added the VBATT_Empty description and clarification ▪ Section 2.2.3 Fuel gauge plots: removed duplicate timing data for simplicity ▪ Section 2.2.3.5 BATT_ID after battery insertion: updated example plots of ESR measurements and added a plot of BATT_ID detection ▪ Section 3.1.1: removed to improve readability ▪ Section 5.1.3 Can the ESR measurements and/or pulses be disabled or their frequency changed?: updated for clarity ▪ Section 5.5.1 Can a third-party fuel gauge be used and the internal fuel gauge be disabled?: updated for clarity

Contents

Figures

Tables

1 Introduction

1.1 Features

- Hardware autonomous with minimal software interaction required
- State of charge (SoC) estimation is done by means of a mixed-mode algorithm using voltage and current information
- Dedicated coulomb counter
- 16-bit dedicated current ADC
  - No software calibration needed
- 15-bit dedicated voltage ADC
  - No software calibration needed
- In- system battery resistance measurements
  - Aids in accurate SoC tracking over temperature and battery aging
- Supports multiple battery profiles in software
- Supports temperature measurements for the battery and Qualcomm® Intelligent Negotiation for Optimum Voltage (INOV) feature
- Hardware-based battery current limiting scheme (BCL) to maximize system performance
- 10-bit round-robin ADC that handles all PMI housekeeping functionalities
- Battery identification
- Battery missing detection
- Supports current sensing via internal battery FET
- JEITA compliant temperature limiting management for charger
- Automatic recharge based on battery voltage or SoC
- Configurable 100% and 0% SoC points, based on current and voltage, respectively

1.2 General description

The fuel gauge offers a hardware-based algorithm that can accurately estimate the SoC by mixing both current and voltage monitoring techniques. This ensures excellent short-term linearity and long-term accuracy. Furthermore, zero current load conditions are not required to maintain accuracy due to the online equivalent series resistance (ESR) measurements.

Using precise measurements of voltage, current, temperature, and resistance, an accurate SoC is delivered over a broad range of operating conditions. High reliability is also achieved via a complex compensation algorithm that takes into account temperature and aging effects, and providing a dependable SoC throughout a battery's entire life.

Configuration registers are provided to fit the requirements of every application.

1.3 Acronyms

Table 1-1 Acronyms

Acronyms	Description
ADC	Analog-to-digital converter
BCL	Battery current limiting
BMD	Battery missing detection
CMA	Conventional memory access
CV	Constant voltage
EOC	End of charge
ESR	Equivalent series resistance
FCC	Fast charge current
FG	Fuel gauge
FV	Float voltage
IMA	Interleaved memory access
OCV	Open circuit voltage
SoC	State of charge

1.4 Fuel gauge pin descriptions

Table 1-2 PMI8998 pin descriptions

Pin	Description
VBATT_SNS_P	Battery plus terminal sense input; connect directly
VBATT_SNS_M	Battery minus terminal sense input; connect directly
IBATT_SNS_P	Defeatured
IBATT_SNS_M	Defeatured
ISNS_SMB_P	Positive current sense from the external parallel charger (VDISCHRG) (optional)
ISNS_SMB_M	Negative current sense from the external parallel charger (VCHRG) (optional)

Pin	Description
VREG_FG	LDO that supplies BIAS for temperature measurements and the ADC; connect a bypass capacitor only—*do not load externally*
VARB	Arbitration circuit that is a part of the charger module, but is the supply to VREG_FG
BATT_ID	Battery ID input to the ADC; also detects the missing battery
BATT_THERM_BIAS	Battery thermistor bias supply
BATT_THERM	Input for battery NTC-type thermistors
AUX_THERM_BIAS	Remote skin temp thermistor bias supply
AUX_THERM	Skin or remote thermistor sensor
GND_FG	Fuel gauge analog ground
REF_GND_FG	Reference ground for fuel gauge controller
GND_PSUB_FG	Substrate ground

1.5 Generation 3 fuel gauge (PMI8998) vs. previous generation 2 fuel gauge (PMI8996 and PMI8952)

Table 1-3 Comparison of fuel gauge, generations 2 and 3

Feature	Generation 2 fuel gauge (PMI8952 and PMI8996)	Generation 3 fuel gauge (PMI8998)
Hardware differences		
Current sensing	▪ Internal current sensing POR ▪ External current sensing option ▪ External current sensing needed for parallel charging ▪ Current sensing up to ~5 A	▪ Internal current sensing POR ▪ External current sensing **not supported** ▪ Internal current sensing POR for parallel charging ▪ Current sensing up to 8.5 A
Fuel gauge memory	▪ Most fuel gauge configurations stored in SRAM ▪ Documentation uses hexadecimal number system for SRAM addressing and configuration ▪ Memory addresses and names mostly constant to all previous PMIs, going back to PMI8994 ▪ Generation 2 uses CMA for profile loading and IMA for fuel gauge memory access	▪ Many fuel gauge configurations moved to SPMI peripherals ▪ Documentation uses decimal number system for SRAM addressing and configuration ▪ Memory addresses and names changed compared to generation 2 ▪ Generation 3 uses IMA for all fuel gauge memory access
ADCs	▪ Dedicated VADC for fuel gauging and battery current limiting ▪ Dedicated IADC for fuel gauging and battery current limiting	▪ Dedicated VADC for fuel gauging and battery current limiting ▪ Dedicated IADC for fuel gauging and battery current limiting ▪ Round-robin ADC for housekeeping on PMI

Feature	Generation 2 fuel gauge (PMI8952 and PMI8996)	Generation 3 fuel gauge (PMI8998)
Housekeeping functions	▪ USB_ID (on PMI8994 and PMI8996) ▪ BATT_ID ▪ BATT_THERM	▪ BATT_ID ▪ BATT_THERM ▪ AUX_THERM ▫ Option for INOV skin temperature measurements **only** ▪ PMI die temperature ▪ Charger die temperature ▪ USB_IN and DC_IN current ▪ USB_IN and DC_IN voltage
Battery current limiting	▪ Voltage and current measurement timing depends on three power levels ▪ Algorithm controlled in software ▪ Mitigation done in software, based on interrupts	▪ Voltage and current measurement timing depends on two power levels ▪ Algorithm primarily controlled by hardware ▪ Mitigation done autonomously in hardware by limits management
Algorithm		
ESR measurements		
Discharge – pulse magnitude	~60 mA	~150 mA
Discharge – pulse timing	Every ~94 s if qualifying transient does not occur	Every ~145.5 s if qualifying transient does not occur
Charging – pulse magnitude	~100 mA	▪ Nonparallel charging; based on Table 2-1 ▪ Parallel charging: ~300 mA static FCC setting on PMI
Charging – pulse timing	Every ~47 s if qualifying transient does not occur	▪ All charging, every ~28 s if qualifying transient does not occur
Rslow mapping	▪ Rslow SoC-based compensation during discharge done in hardware ▪ Rslow SoC-based compensation during charging done in software	All Rslow mapping done in hardware
Slope limiter	▪ 2 programmable registers ▪ Complicated equation to set slope limit	▪ 1 programmable register ▪ Simpler equation to set slope limit
Parallel charging current measurements	▪ Shared external sense resistor between SMB and PMI ▪ Fuel gauge reads 1 total current from SMB and PMI	▪ Separate sensing done between SMB and PMI ▪ Parallel charger communicates current to PMI via the dedicated ISNS_SMB_X ▪ Total current combined in the fuel gauge
New features		
Time to full and time to empty	N/A	Supported in software
Qnovo algorithm	N/A	Fuel gauge provides measurement data needed for Qnovo
Qualcomm® Trepn™ profiler	N/A	Fuel gauge provides measurement data needed for the Trepn application

1.6 Tasks and configurations needed for fuel gauge commercialization

The fuel gauge is a powerful and accurate tool for determining state of charge while still being simple to use, and only requiring a small number of configurations. This section is a guide for customers who are designing for and using the fuel gauge; it supplies all of the necessary configurations to ensure optimal fuel gauge performance.

1.6.1 Pre-board design stage

1.6.1.1 Battery pack design

RID selection

- Supported readable RID range: 1 k–450 k.
  - An RID value must be chosen within this range.
  - It is recommended that separate RID values be chosen for cells from different battery vendors, so individual profiles can be stored in software that are custom specifically to that cell.
    - A 20% tolerance difference is recommended between RID values for multiple cells.
- Default fake battery detection happens at 7.5 kΩ ± 15%, with a typical 5% tolerance RID.
  - This feature prevents charging from taking place in hardware when a ~7.5 kΩ resistor is detected.
  - This range is programmable; for more information about programming this range, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

CAUTION: This feature is enabled by default and is accomplished in hardware. Do not choose an RID near this value unless charging is expected to be prevented.

Thermistor selection and placement

- Supported NTC resistance range: 10 k–100 k.
  - This is the resistance of the thermistor at 25°C.
- Supported NTC beta value range: 3200–4400.
  - When choosing a thermistor, QTI recommends a thermistor with a low beta value for improved temperature measurement accuracy.
  - Qualcomm Technologies, Inc. (QTI) temperature accuracy specifications are based on testing of a 3450 beta value thermistor, and cannot be guaranteed with greater values.
    - Refer to the *PMI8998 Power Management Device Specification* (80-P1087-1), Section 3.5.2, for more information about temperature accuracy and fuel gauge related specifications.
- It is recommended that the NTC be placed on the PCM of the battery pack, away from current carrying traces.
  - Avoid placing the NTC on the board.

- Ensure that the NTC is not placed on a layer above or near a current-carrying trace on the PCM. This can cause inaccurate measurement of the cell temperature, which will affect fuel gauging performance.

Table 1-4 Battery pack recommendations

Min. overcurrent protection threshold for discharge (OCP)	Min. OCP delay time for discharge	Max. undervoltage detection threshold for discharge (UVD)	Min. UVD delay time for discharge	Max. impedance @ 1 kHz	Max. impedance @ 10 kHz
6 A	10 ms	2.4 V	10 ms	60 mΩ	80 mΩ

NOTE: For proper battery current limiting functionality, the minimum OCP and minimum UVD delay timing specifications must be met.

1.6.2 Board design stage

1.6.2.1 Fuel gauge layout

- Fuel gauge layout is critical for both basic functionality and performance.
- Serious issues can arise when the layout guidelines are not followed; pay special attention to layout and follow **all** of the layout guidelines.
- For the fuel gauge layout guidelines, refer to the *PM8998, PMI8998, and PM8005 Layout Design Guidelines* (80-P1086-5A).

1.6.2.2 Battery profile

- A custom battery profile is required when using the fuel gauge.
- Generic profiles are not supported.
- For more information about obtaining a battery profile, see Section 2.2.2.
- It is recommended that each cell vendor have its own RID value and custom profile.
- Allow three to four weeks for the battery characterization process, from the date of delivery of the batteries to the date of receipt of the battery profile.

1.6.2.3 AUX_THERM part selection

- AUX_THERM is an optional input for a skin temperature thermistor for the charger's INOV3 algorithm.
- The measurement technique is similar to BATT_THERM in that it is ratiometric, but the center point is chosen at 45°C.
  - A pull-up value for AUX_THERM measurements must match the thermistor's resistance value at 45°C.
  - Only a center temperature of 45°C is supported.
- INOV3 is a charger function; for more information about INOV3, refer to the charger-related documentation in the *PM8998, PM8005, and PMI8998 Power Management ICs Design Guidelines* (80-P1086-5).

- For more information about AUX_THERM, see Sections 2.1.4.3 and 2.1.4.3.1.

1.6.3 Post-board design – required fuel gauge configurations

1.6.3.1 100% and 0% SoC endpoint matching related configurations

VBATT full

- This is one of the two configurable inputs to the SoC full estimate loop, which affects the 100% SoC prediction algorithm.
- This must be set to 10 mV less than the float voltage setting of the charger.
  - For example, for a 4.4 V battery, this setting should be 4.39 V.
  - If this is not set correctly, 100% SoC will not match well with the IBATT full setting.
- This was previously known as the CC_to_CV threshold.
- For more hardware information about this setting, see Section 2.2.1.4.
- For information about how to configure this in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

IBATT full

- This is one of the two configurable inputs to the SoC full estimate loop, which affects the 100% SoC prediction algorithm.
- IBATT full must be larger in magnitude (more negative) than the charger's termination current so that IBATT full is reached before the charger's termination.
  - For example, if the charger will terminate charge at -100 mA, IBATT full must be set to < -100 mA.
  - In a common example, charger termination current = -100 mA; IBATT full = -150 mA.
  - Keep in mind that when the battery current is negative, it denotes current flowing into the battery.
- This was previously known as the system termination current.
- For hardware information about this setting, see Section 2.2.1.4.
- For information on how to configure this in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

VBATT cutoff

- This is the only configuration for the SoC empty loop, which affects the 0% SoC prediction algorithm.
- This provides a voltage-based threshold to configure when 0% SoC is reported.
- This setting depends on the customer's specific design, requirements, and low battery voltage testing.
- This was previously known as the cutoff voltage threshold.
- For more hardware information about this setting, see Section 2.2.1.4.

- For information about how to configure this in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

VBATT empty

- This is a backup threshold to initiate a software shutdown before a low battery voltage condition can cause a UVLO event.
- It is recommended that this setting be left at 2.8 V by default.
- This was previously known as the volt empty threshold.
- For information about how to configure this in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

1.6.3.2 Charger and battery temperature related configurations

Automatic recharge

- The fuel gauge works with the charger module to support two types of automatic recharge:
  - SoC-based recharge
  - Battery voltage based recharge (default)
- This depends on the customer’s requirements.
- For information about how to configure this in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

Beta coefficient configurations

- The fuel gauge models battery temperature, using an interpolated model of the Steinhart-Hart equation.
  - To align the fuel gauge hardware model with the physical model of the thermistor in the battery pack, coefficients that represent the beta coefficient of the thermistor must be programmed into hardware.
  - Failing to do this may result in a failure to meet temperature accuracy specifications.
- For more information about how to configure this in software, refer to the *MSM8998 BSP Software Design Review Questionnaire* (80-P2484-82).

Thermistor measurement delay configurations

- Some customers require small amounts of capacitance to be placed on the thermistor net, typically for ESD reasons.
- To support this, a delay must be set to ensure that the sampling on BATT_THERM is settled.
  - If this is not configured, and capacitance is placed on the BATT_THERM net, temperature measurements take place immediately after BATT_THERM_BIAS is enabled, causing an unsettled waveform, and resulting in inaccurate temperature measurements.
- For hardware information about this setting, see Section 2.1.4.2.4.
- For information about how to configure this in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

JEITA limit threshold configurations

- Because the fuel gauge is responsible for temperature measurements, JEITA temperature measurement and limiting falls under the fuel gauge settings.
- For hardware information about this setting, see Section 2.1.6.2.
- For information about how to configure this in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

1.6.3.3 Other fuel gauge hardware and software configurations

Monotonic slope limiting configuration

- In hardware, the fuel gauge can limit the magnitude of change of the monotonic SoC between 1.47 s fuel gauge update cycles.
- For hardware information about this setting, see Section 2.2.1.5.1.
- For information about how to configure this in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

Aging and cycle counting

- In software, the fuel gauge can perform capacity learning-based aging and cycle counting.
  - QTI recommends that all customers enable this feature.
- For information about how to configure this in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

2 Detailed module information

2.1 Fuel gauge hardware and additional functions

2.1.1 Fuel gauge current sensing

The fuel gauge measures current via the internal battery FET (BATFET). Current is read as positive when discharging and as negative when charging. The battery current is read every 1.47 s, with a resolution of approximately 150 µA. The battery current and voltage are read synchronously. During this 1.47 s cycle, it takes 160 ms for the ADC to complete a conversion.

2.1.2 Fuel gauge voltage sensing

Battery voltage is measured across the dedicated VBATT_SNS_P and VBATT_SNS_N differential pins. Those pins connect differentially directly to the battery pads and internally feed the dedicated battery voltage ADC. The battery voltage is read every 1.47 s with a resolution of approximately 150 µV. Both the battery voltage and current are read synchronously. During this 1.47 s cycle, it takes 160 ms for the ADC to complete a conversion.

2.1.3 Battery resistance measurements

2.1.3.1 Overview

The battery resistance is a key parameter in determining a battery's health. The fuel gauge hardware models battery resistance is shown in Figure 2-5. The total battery resistance can be broken down into two parts:

- ESR
  - ESR is the resistance seen immediately when a load is applied.
  - Protection circuits are a large contributor to ESR within the battery pack.
  - ESR varies widely over temperature and state of charge.
  - This parameter is measured during device operation.
- Rslow
  - Rslow is the resistance seen after a load is removed and the battery voltage is left to settle.
  - This is represented by R2 and C2 in the battery model in Figure 2-5.
  - This parameter is modeled based on its relationship to ESR over temperature and SoC, where the model was found during battery characterization.

2.1.3.2 ESR measurements during discharge

Active mode

During normal operation, the fuel gauge uses the naturally occurring voltage and current transients on the battery to calculate the real-time battery resistance. The current transient between the current and previous measurement must exceed a 110 mA difference to qualify for an ESR measurement. If naturally occurring transients do not occur within ~145.5 s, the fuel gauge creates its own transient via a pull-down. This is a small pulse (default of 150 mA), done quickly and synchronously with the ADC's V and I measurements (~160 ms), and has minimal impact on the battery life.

Sleep mode

When entering sleep, the software extends the ESR measurements to once every ~382 s.

2.1.3.3 ESR measurements during charge

Single-path charging:

To generate a transient necessary to measure the ESR of the battery, the fuel gauge communicates to the charger module via hardware to make a quick fast-charge current decrement (FCC). This happens at a maximum rate of every ~28 s, for a duration of ~160 ms, and for a current magnitude dependent on values in Table 2-1. If this measurement fails, it then retries an ESR pulse every ~16 s until successful. During the decrement, the fuel gauge takes synchronous V and I measurements to calculate the ESR of the battery. This short decrement allows the fuel gauge to get the information necessary to estimate the ESR with a minimal effect on the charge time.

Parallel charging:

During parallel charging, the ESR pulse magnitude is different than during single path charging. The timing remains the same, but instead of using a value from Table 2-1 for the magnitude, the fuel gauge uses a fixed 300 mA FCC.

NOTE: This is not a decrement of 300 mA from the set FCC, but instead the charger is quickly set to regulate an FCC of 300 mA.

Table 2-1 ESR pulses during single-path charging

FCC code used by the charger	Transition point based on the battery current
No ESR pulses	x > -300 mA
-300 mA	-300 mA > x > -800 mA
-600 mA	-800 mA > x > -1.25 A
-1 A	-1.25 A > x > -2.25 A
-2 A	-2.25 A > x > -3.25 A
-3 A	-3.25 A > x > -4.25 A
-4 A	-4.25 A > x > -5.25 A
-5 A	-5.25 A > x > -6.25 A
-6 A	x < -6.25 A

2.1.3.4 Rslow

Rslow is the second component of the total battery resistance. It is the resistance seen as a battery is left to settle. This resistance is measured during battery characterization and is modeled for both charge and discharge separately in real time by the fuel gauge, as a ratio of ESR that can be scaled by temperature and SoC.

2.1.4 Round-robin ADC measurements

2.1.4.1 Overview

The PMI8998 contains house keeping functionality that is done using a dedicated ADC running in a round-robin manner. Each channel is measured once every fuel gauge cycle (1.47 s). The channel list and order is shown in Figure 2-1. For more information on the round-robin ADC specifications, refer to the *PMI8998 Power Management Device Specification* (80-P1087-1).

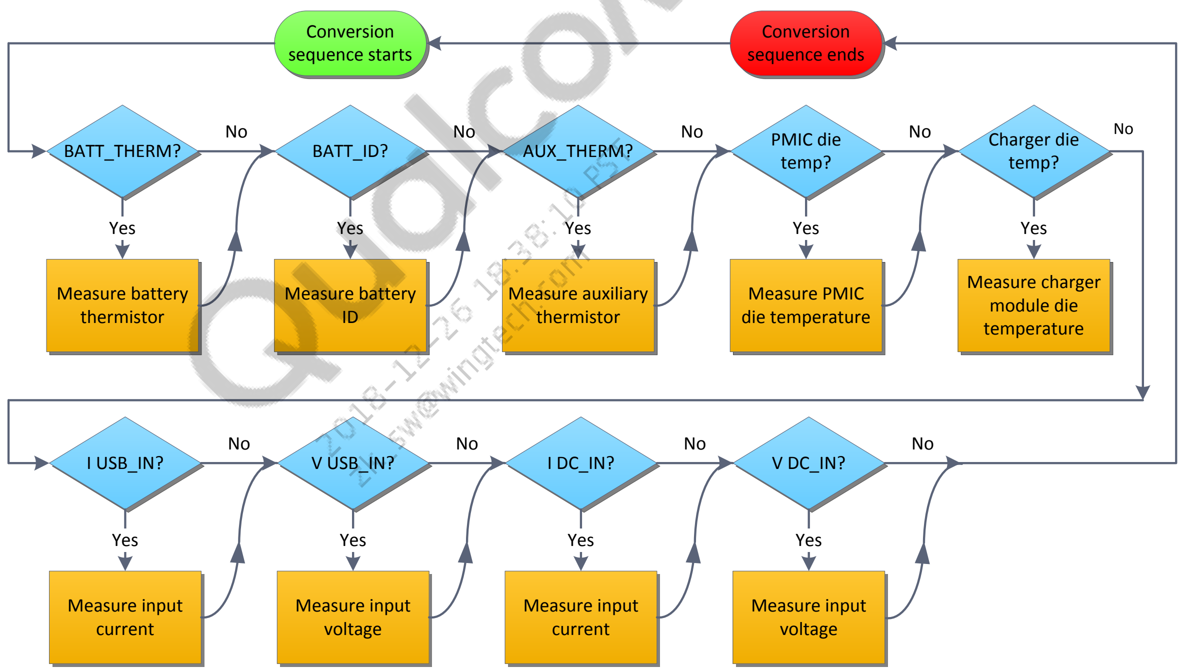

Figure 2-1 Round-robin ADC channels and measurement order

2.1.4.2 Battery temperature sensing and compensation

2.1.4.2.1 Overview

The fuel gauge module monitors the battery's temperature via a dedicated BATT_THERM pin via a round-robin ADC.

Information about the battery's temperature is used by the fuel gauge for two purposes:

- Improve SoC accuracy by adjusting the fuel gauge models based on the temperature.
- Accommodate charger operation with respect to JEITA requirements.

2.1.4.2.2 Functionality

The fuel gauge uses a ratiometric conversion. The advantage of this design is that the pull-up that is used to bias the thermistor is the same regulator internal to the PMIC that supplies the ADC. This allows for a more robust measurement scheme and simplification of the thermistor modeling. The control of the BATT_THERM_BIAS is also automatically handled by hardware. By default, BATT_THERM is measured once every fuel gauge cycle (1.47 s).

NOTE: The regulator (VREG_FG) that supplies BATT_THERM_RBIAS is used for other analog purposes internal to the PMIC and is set based on trim related parameters internal to the PMIC. Therefore, it may not be 2.7 V for all PMICs. This is expected behavior and does not cause any issues.

2.1.4.2.3 BATT_THERM_RBIAS pull-up selection

Due to the ratiometric nature of the temperature measurement scheme, a pull-up resistor must be chosen to match the thermistors resistance value at its center range point (25°C). If the pull-up resistor is not chosen correctly, the temperature accuracy measurements will be out of specification.

2.1.4.2.4 Capacitance on BATT_THERM

During temperature measurement, BATT_THERM_BIAS is enabled before the temperature measurements are sampled. If capacitance is placed on the BATT_THERM node in the battery, or on the board, then the BATT_THERM voltage may not have sufficient time to settle to its final value before the ADC conversion begins. This could corrupt the temperature measurements resulting in a failure to meet the accuracy specification. Capacitance is not officially supported on the BATT_THERM node, but if it is an OEM requirement, there is a programmable delay between when BATT_THERM_BIAS is enabled and when the ADC measurement begins. This is achieved by signaling a timer-only ADC conversion of configurable length (the longer the bit length, the longer the effective delay). This allows the placement of some capacitance on the BATT_THERM net. This functionality has not been validated or simulated for PMI8998; if an issue arises due to capacitance on the BATT_THERM node, it is recommended this delay be maximized and tested again. If this does not resolve the issue, the capacitance must be removed or reduced.

The fuel gauge supports delays of 0 ms (disabled), 1 ms, 4 ms, 12 ms, 20 ms, 40 ms, 60 ms, or 80 ms.

2.1.4.2.5 Beta table and thermistor selection

When selecting a thermistor, it is recommended to choose one with a beta value that is on the lower end of the supported range. This improves accuracy over the entire temperature range as a lower beta value varies less over temperature.

To convert from voltage measurements to temperature readings, the PMIC uses a linear approximation. This approximation uses the coefficients listed in Table 2-2. To function correctly, these coefficients must be programmed into the fuel gauge for the thermistor value selected. Failure to do this can result in inaccurate temperature measurement. Because this is an **approximation**, it is not supported or recommended to use any other means of creating these coefficients, such as using the Steinhart-Hart equation.

Table 2-2 BATT_ THERM beta coefficients

Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
3200	0x8f	0x50	0xff
3205	0x90	0x50	0xff
3210	0x90	0x50	0xff
3215	0x91	0x50	0xff
3220	0x92	0x50	0xff
3225	0x92	0x50	0xff
3230	0x90	0x50	0xff
3235	0x91	0x50	0xff
3240	0x91	0x50	0xff
3245	0x92	0x50	0xff
3250	0x92	0x50	0xff
3255	0x93	0x50	0xff
3260	0x93	0x50	0xff
3265	0x94	0x50	0xff
3270	0x94	0x50	0xff
3275	0x95	0x50	0xff
3280	0x95	0x50	0xff
3285	0x96	0x50	0xff
3290	0x96	0x50	0xff
3295	0x97	0x50	0xff
3300	0x97	0x50	0xff
3305	0x98	0x50	0xff
3310	0x97	0x50	0xff
3315	0x97	0x50	0xff
3320	0x98	0x50	0xff
3325	0x98	0x50	0xff
3330	0x99	0x50	0xff
3335	0x99	0x50	0xff
3340	0x9a	0x50	0xff
3345	0x9a	0x50	0xff
3350	0x9b	0x50	0xff
3355	0x9b	0x50	0xff
3360	0x9c	0x50	0xff
3365	0x9c	0x50	0xff
3370	0x9d	0x50	0xff
3375	0x9d	0x50	0xff
3380	0x9d	0x50	0xff
3385	0x9e	0x50	0xff
3390	0x9c	0x50	0xff
3395	0x9d	0x50	0xff
3400	0x9d	0x50	0xff

Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
3405	0x9e	0x50	0xff
3410	0x9e	0x50	0xff
3415	0x9f	0x50	0xff
3420	0x9f	0x50	0xff
3425	0xa0	0x50	0xff
3430	0xa0	0x50	0xff
3435	0xa1	0x50	0xff
3440	0xa1	0x50	0xff
3445	0xa2	0x50	0xff
3450	0xa2	0x50	0xff
3455	0xa3	0x50	0xff
3460	0xa3	0x50	0xff
3465	0xa4	0x50	0xff
3470	0xa2	0x50	0xff
3475	0xa3	0x50	0xff
3480	0xa3	0x50	0xff
3485	0xa3	0x50	0xff
3490	0xa4	0x50	0xff
3495	0xa4	0x50	0xff
3500	0xa5	0x50	0xff
3505	0xa5	0x50	0xff
3510	0xa6	0x50	0xff
3515	0xa6	0x50	0xff
3520	0xa7	0x50	0xff
3525	0xa7	0x50	0xff
3530	0xa8	0x50	0xff
3535	0xa8	0x50	0xff
3540	0xa9	0x50	0xff
3545	0xa9	0x50	0xff
3550	0xaa	0x50	0xff
3555	0xa8	0x50	0xff
3560	0xa9	0x50	0xff
3565	0xa9	0x50	0xff
3570	0xa9	0x50	0xff
3575	0xaa	0x50	0xff
3580	0xaa	0x50	0xff
3585	0xab	0x50	0xff
3590	0xab	0x50	0xff
3595	0xab	0x50	0xff
3600	0xac	0x50	0xff
3605	0xac	0x50	0xff
3610	0xad	0x50	0xff

Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
3615	0xad	0x50	0xff
3620	0xae	0x50	0xff
3625	0xae	0x50	0xff
3630	0xae	0x50	0xff
3635	0xaf	0x50	0xff
3640	0xaf	0x50	0xff
3645	0xaf	0x50	0xff
3650	0xaf	0x50	0xff
3655	0xaf	0x50	0xff
3660	0xaf	0x50	0xff
3665	0xaf	0x50	0xff
3670	0xb0	0x50	0xff
3675	0xb0	0x50	0xff
3680	0xb1	0x50	0xff
3685	0xb1	0x50	0xff
3690	0xb2	0x50	0xff
3695	0xb2	0x50	0xff
3700	0xb3	0x50	0xff
3705	0xb3	0x50	0xff
3710	0xb4	0x50	0xff
3715	0xb4	0x50	0xff
3720	0xb5	0x50	0xff
3725	0xb4	0x50	0xff
3730	0xb4	0x50	0xff
3735	0xb4	0x50	0xff
3740	0xb5	0x50	0xff
3745	0xb5	0x50	0xff
3750	0xb6	0x50	0xff
3755	0xb6	0x50	0xff
3760	0xb7	0x50	0xff
3765	0xb7	0x50	0xff
3770	0xb8	0x50	0xff
3775	0xb8	0x50	0xff
3780	0xb8	0x50	0xff
3785	0xb9	0x50	0xff
3790	0xb9	0x50	0xff
3795	0xb9	0x50	0xff
3800	0xba	0x50	0xff
3805	0xba	0x50	0xff
3810	0xbb	0x50	0xff
3815	0xbb	0x50	0xff
3820	0xb9	0x50	0xff

Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
3825	0xb9	0x50	0xff
3830	0xba	0x50	0xff
3835	0xba	0x50	0xff
3840	0xbb	0x50	0xff
3845	0xbb	0x50	0xff
3850	0xbc	0x50	0xff
3855	0xbc	0x50	0xff
3860	0xbc	0x50	0xff
3865	0xbd	0x50	0xff
3870	0xbd	0x50	0xff
3875	0xbe	0x50	0xff
3880	0xbe	0x50	0xff
3885	0xbf	0x50	0xff
3890	0xbf	0x50	0xff
3895	0xbf	0x50	0xff
3900	0xc0	0x50	0xff
3905	0xc0	0x50	0xff
3910	0xbe	0x50	0xff
3915	0xbf	0x50	0xff
3920	0xbf	0x50	0xff
3925	0xbf	0x50	0xff
3930	0xc0	0x50	0xff
3935	0xc0	0x50	0xff
3940	0xc1	0x50	0xff
3945	0xc1	0x50	0xff
3950	0xc1	0x50	0xff
3955	0xc2	0x50	0xff
3960	0xc2	0x50	0xff
3965	0xc2	0x50	0xff
3970	0xc3	0x50	0xff
3975	0xc3	0x50	0xff
3980	0xc4	0x50	0xff
3985	0xc4	0x50	0xff
3990	0xc4	0x50	0xff
3995	0xc5	0x50	0xff
4000	0xc5	0x50	0xff
4005	0xc6	0x50	0xff
4010	0xc5	0x50	0xff
4015	0xc5	0x50	0xff
4020	0xc5	0x50	0xff
4025	0xc5	0x50	0xff
4030	0xc6	0x50	0xff

Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
4035	0xc6	0x50	0xff
4040	0xc6	0x50	0xff
4045	0xc7	0x50	0xff
4050	0xc7	0x50	0xff
4055	0xc7	0x50	0xff
4060	0xc8	0x50	0xff
4065	0xc8	0x50	0xff
4070	0xc8	0x50	0xff
4075	0xc9	0x50	0xff
4080	0xc9	0x50	0xff
4085	0xc9	0x50	0xff
4090	0xca	0x50	0xff
4095	0xca	0x50	0xff
4100	0xca	0x50	0xff
4105	0xcb	0x50	0xff
4110	0xcb	0x50	0xff
4115	0xcb	0x50	0xff
4120	0xcb	0x50	0xff
4125	0xcc	0x50	0xff
4130	0xcc	0x50	0xff
4135	0xcc	0x50	0xff
4140	0xcd	0x50	0xff
4145	0xcd	0x50	0xff
4150	0xce	0x50	0xff
4155	0xce	0x50	0xff
4160	0xce	0x50	0xff
4165	0xcf	0x50	0xff
4170	0xcf	0x50	0xff
4175	0xd0	0x50	0xff
4180	0xd0	0x50	0xff
4185	0xd0	0x50	0xff
4190	0xd1	0x50	0xff
4195	0xd1	0x50	0xff
4200	0xd1	0x50	0xff
4205	0xd2	0x50	0xff
4210	0xd2	0x50	0xff
4215	0xd0	0x50	0xff
4220	0xd1	0x50	0xff
4225	0xd1	0x50	0xff
4230	0xd1	0x50	0xff
4235	0xd1	0x50	0xff
4240	0xd2	0x50	0xff

Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
4245	0xd2	0x50	0xff
4250	0xd2	0x50	0xff
4255	0xd3	0x50	0xff
4260	0xd3	0x50	0xff
4265	0xd3	0x50	0xff
4270	0xd4	0x50	0xff
4275	0xd4	0x50	0xff
4280	0xd4	0x50	0xff
4285	0xd5	0x50	0xff
4290	0xd5	0x50	0xff
4295	0xd5	0x50	0xff
4300	0xd6	0x50	0xff
4305	0xd6	0x50	0xff
4310	0xd7	0x50	0xff
4315	0xd7	0x50	0xff
4320	0xd6	0x50	0xff
4325	0xd6	0x50	0xff
4330	0xd6	0x50	0xff
4335	0xd7	0x50	0xff
4340	0xd7	0x50	0xff
4345	0xd7	0x50	0xff
4350	0xd8	0x50	0xff
4355	0xd8	0x50	0xff
4360	0xd8	0x50	0xff
4365	0xd8	0x50	0xff
4370	0xd9	0x50	0xff
4375	0xd9	0x50	0xff
4380	0xd9	0x50	0xff
4385	0xd9	0x50	0xff
4390	0xda	0x50	0xff
4395	0xda	0x50	0xff
4400	0xda	0x50	0xff

NOTE: The bias resistor chosen **must** have the same value as the thermistor at room temperature for accurate BATT_THERM measurements.

2.1.4.2.6 Thermistor placement

QTI recommends the use of a thermistor internal to the battery pack when possible. If this is not an option, it is possible to use an external thermistor placed on the board near the battery. Keep in mind that doing so is an added risk, as this may affect the temperature performance and in turn the fuel gauge and JEITA performance. Take care with the placement and layout around this thermistor as any additional heat sources affect the thermistor's resistance.

2.1.4.3 AUX_THERM skin temperature beta table

Temperature measurements on AUX_THERM are similar in hardware to BATT_THERM in that both use the ratiometric measurement technique and the round-robin ADC. The important difference between the two is that BATT_THERM's half range is centered at 25°C, and AUX_THERM's half range is centered at 45°C. QTI only supports the 45°C half range of AUX_THERM, and only the 45°C related tables will be provided. AUX_THERM's only supported option is to be used for the INOV hardware autonomous thermal management algorithm. Similar to BATT_THERM, AUX_THERM also requires a set of beta coefficients. See Table 2-3 f or the beta coefficient selection.

2.1.4.3.1 AUX_THERM skin temperature thermistor pull-up selection guide

The pull-up resistor for the skin temperature thermistor must be selected to match the NTC's resistance value at 45°C. This is the similar to BATT_THERM, but the center point is now at 45°C, instead of 25°C. This gives more range and better accuracy around the center point of 45°C. Customers can also use the following algorithm to calculate the pull-up resistance value:

1. Check the skin thermistor beta (the lower the beta, the more linear the behavior).
2. Identify the scaling ratio that matches the skin thermistor's beta value from Table 2-3.
3. Pick the pull-up resistor by using the following equation: thermistor room temperature (datasheet nominal value) × scaling ratio.

   For example:

   - A thermistor with 3200 beta and 68 kΩ room temperature value
   - Scaling ratio is 0.509
   - Pull-up value is 68 k × 0.509 = 34.61 k
   - 34.61 k is not a common value, so the closest common resistor value may be used. It is recommended that a maximum of 1% tolerance resistor is used.

Table 2-3 AUX_THERM pull-up resistor scaling ratios

Beta	Scaling ratio	Beta	Scaling ratio	Beta	Scaling ratio	Beta	Scaling ratio
3200	0.509	3505	0.4775	3810	0.448	4115	0.41965
3205	0.5085	3510	0.477	3815	0.4475	4120	0.4192
3210	0.508	3515	0.4765	3820	0.447	4125	0.41875
3215	0.5075	3520	0.476	3825	0.4465	4130	0.4183
3220	0.507	3525	0.4755	3830	0.446	4135	0.41785
3225	0.5065	3530	0.475	3835	0.4455	4140	0.4174
3230	0.506	3535	0.4745	3840	0.445	4145	0.41695
3235	0.5055	3540	0.474	3845	0.4445	4150	0.4165
3240	0.505	3545	0.4735	3850	0.444	4155	0.41605
3245	0.5045	3550	0.473	3855	0.4435	4160	0.4156
3250	0.504	3555	0.4725	3860	0.443	4165	0.41515
3255	0.5035	3560	0.472	3865	0.4425	4170	0.4147
3260	0.503	3565	0.4715	3870	0.442	4175	0.41425
3265	0.5025	3570	0.471	3875	0.4415	4180	0.4138
3270	0.502	3575	0.4705	3880	0.441	4185	0.41335

Beta	Scaling ratio	Beta	Scaling ratio	Beta	Scaling ratio	Beta	Scaling ratio
3275	0.5015	3580	0.47	3885	0.4405	4190	0.4129
3280	0.501	3585	0.4695	3890	0.44	4195	0.4125
3285	0.5005	3590	0.469	3895	0.43955	4200	0.412
3290	0.5	3595	0.4685	3900	0.439	4205	0.4116
3295	0.49945	3600	0.468	3905	0.43855	4210	0.4112
3300	0.499	3605	0.4675	3910	0.4381	4215	0.4108
3305	0.49845	3610	0.467	3915	0.43765	4220	0.4104
3310	0.4979	3615	0.4665	3920	0.4372	4225	0.41
3315	0.49735	3620	0.466	3925	0.43675	4230	0.4096
3320	0.4968	3625	0.4655	3930	0.4363	4235	0.4092
3325	0.49625	3630	0.465	3935	0.43585	4240	0.4088
3330	0.4957	3635	0.4645	3940	0.4354	4245	0.4084
3335	0.49515	3640	0.464	3945	0.43495	4250	0.408
3340	0.4946	3645	0.4635	3950	0.4345	4255	0.4076
3345	0.49405	3650	0.463	3955	0.43405	4260	0.4072
3350	0.4935	3655	0.4625	3960	0.4336	4265	0.4068
3355	0.49295	3660	0.462	3965	0.43315	4270	0.4064
3360	0.4924	3665	0.4615	3970	0.4327	4275	0.406
3365	0.49185	3670	0.461	3975	0.43225	4280	0.4056
3370	0.4913	3675	0.4605	3980	0.4318	4285	0.4052
3375	0.49075	3680	0.46	3985	0.43135	4290	0.4048
3380	0.4902	3685	0.4595	3990	0.4309	4295	0.40435
3385	0.48965	3690	0.459	3995	0.43045	4300	0.404
3390	0.4891	3695	0.45855	4000	0.43	4305	0.40355
3395	0.4886	3700	0.458	4005	0.42955	4310	0.4031
3400	0.488	3705	0.45755	4010	0.4291	4315	0.40265
3405	0.4875	3710	0.4571	4015	0.42865	4320	0.4022
3410	0.487	3715	0.45665	4020	0.4282	4325	0.40175
3415	0.4865	3720	0.4562	4025	0.42775	4330	0.4013
3420	0.486	3725	0.45575	4030	0.4273	4335	0.40085
3425	0.4855	3730	0.4553	4035	0.42685	4340	0.4004
3430	0.485	3735	0.45485	4040	0.4264	4345	0.39995
3435	0.4845	3740	0.4544	4045	0.42595	4350	0.3995
3440	0.484	3745	0.45395	4050	0.4255	4355	0.39905
3445	0.4835	3750	0.4535	4055	0.42505	4360	0.3986
3450	0.483	3755	0.45305	4060	0.4246	4365	0.39815
3455	0.4825	3760	0.4526	4065	0.42415	4370	0.3977
3460	0.482	3765	0.45215	4070	0.4237	4375	0.39725
3465	0.4815	3770	0.4517	4075	0.42325	4380	0.3968
3470	0.481	3775	0.45125	4080	0.4228	4385	0.39635
3475	0.4805	3780	0.4508	4085	0.42235	4390	0.3959
3480	0.48	3785	0.45035	4090	0.4219	4395	0.3955

Beta	Scaling ratio	Beta	Scaling ratio	Beta	Scaling ratio	Beta	Scaling ratio
3485	0.4795	3790	0.4499	4095	0.42145	4400	0.395
3490	0.479	3795	0.4494	4100	0.421	X	X
3495	0.4785	3800	0.449	4105	0.42055	X	X
3500	0.478	3805	0.4485	4110	0.4201	X	X

Table 2-4 AUX_THERM beta coefficients

T (C)	Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
45	3200	0x6A	0x87	0x98
45	3205	0x6A	0x86	0x99
45	3210	0x6B	0x85	0x9A
45	3215	0x6B	0x85	0x9A
45	3220	0x6C	0x84	0x9B
45	3225	0x6C	0x83	0x9C
45	3230	0x6D	0x83	0x9C
45	3235	0x6D	0x82	0x9D
45	3240	0x6D	0x81	0x9E
45	3245	0x6E	0x81	0x9E
45	3250	0x6E	0x80	0x9F
45	3255	0x6F	0x7F	0xA0
45	3260	0x6F	0x7F	0xA0
45	3265	0x70	0x7E	0xA1
45	3270	0x70	0x7D	0xA2
45	3275	0x71	0x7D	0xA2
45	3280	0x71	0x7C	0xA3
45	3285	0x71	0x7C	0xA3
45	3290	0x72	0x7B	0xA4
45	3295	0x72	0x7A	0xA5
45	3300	0x73	0x7A	0xA5
45	3305	0x73	0x79	0xA6
45	3310	0x74	0x79	0xA6
45	3315	0x74	0x78	0xA7
45	3320	0x75	0x77	0xA8
45	3325	0x75	0x77	0xA8
45	3330	0x75	0x76	0xA9
45	3335	0x76	0x76	0xA9
45	3340	0x76	0x75	0xAA
45	3345	0x77	0x74	0xAA
45	3350	0x77	0x74	0xAB
45	3355	0x78	0x73	0xAB
45	3360	0x78	0x73	0xAC
45	3365	0x78	0x72	0xAC

T (C)	Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
45	3370	0x79	0x72	0xAD
45	3375	0x79	0x71	0xAE
45	3380	0x7A	0x71	0xAE
45	3385	0x7A	0x70	0xAF
45	3390	0x7B	0x6F	0xAF
45	3395	0x7B	0x6F	0xB0
45	3400	0x7B	0x6E	0xB0
45	3405	0x7C	0x6E	0xB0
45	3410	0x7C	0x6D	0xB1
45	3415	0x7D	0x6D	0xB1
45	3420	0x7D	0x6C	0xB2
45	3425	0x7E	0x6C	0xB2
45	3430	0x7E	0x6B	0xB3
45	3435	0x7E	0x6B	0xB3
45	3440	0x7F	0x6A	0xB4
45	3445	0x7F	0x6A	0xB4
45	3450	0x80	0x69	0xB5
45	3455	0x80	0x69	0xB5
45	3460	0x80	0x68	0xB6
45	3465	0x81	0x68	0xB6
45	3470	0x81	0x67	0xB6
45	3475	0x82	0x67	0xB7
45	3480	0x82	0x66	0xB7
45	3485	0x83	0x66	0xB8
45	3490	0x83	0x65	0xB8
45	3495	0x83	0x65	0xB9
45	3500	0x84	0x64	0xB9
45	3505	0x84	0x64	0xB9
45	3510	0x85	0x63	0xBA
45	3515	0x85	0x63	0xBA
45	3520	0x85	0x62	0xBB
45	3525	0x86	0x62	0xBB
45	3530	0x86	0x61	0xBB
45	3535	0x87	0x61	0xBC
45	3540	0x87	0x60	0xBC
45	3545	0x87	0x60	0xBD
45	3550	0x88	0x5F	0xBD
45	3555	0x88	0x5F	0xBD
45	3560	0x89	0x5E	0xBE
45	3565	0x89	0x5E	0xBE
45	3570	0x89	0x5E	0xBE
45	3575	0x8A	0x5D	0xBF

T (C)	Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
45	3580	0x8A	0x5D	0xBF
45	3585	0x8B	0x5C	0xC0
45	3590	0x8B	0x5C	0xC0
45	3595	0x8B	0x5B	0xC0
45	3600	0x8C	0x5B	0xC1
45	3605	0x8C	0x5A	0xC1
45	3610	0x8D	0x5A	0xC1
45	3615	0x8D	0x5A	0xC2
45	3620	0x8D	0x59	0xC2
45	3625	0x8E	0x59	0xC3
45	3630	0x8E	0x58	0xC3
45	3635	0x8F	0x58	0xC3
45	3640	0x8F	0x57	0xC4
45	3645	0x8F	0x57	0xC4
45	3650	0x90	0x56	0xC4
45	3655	0x90	0x56	0xC5
45	3660	0x90	0x56	0xC5
45	3665	0x91	0x55	0xC6
45	3670	0x91	0x55	0xC6
45	3675	0x91	0x54	0xC6
45	3680	0x92	0x54	0xC7
45	3685	0x92	0x54	0xC7
45	3690	0x93	0x53	0xC7
45	3695	0x93	0x53	0xC8
45	3700	0x93	0x52	0xC8
45	3705	0x94	0x52	0xC8
45	3710	0x94	0x51	0xC9
45	3715	0x94	0x51	0xC9
45	3720	0x95	0x51	0xCA
45	3725	0x95	0x50	0xCA
45	3730	0x95	0x50	0xCA
45	3735	0x96	0x4F	0xCB
45	3740	0x96	0x4F	0xCB
45	3745	0x96	0x4F	0xCB
45	3750	0x97	0x4E	0xCC
45	3755	0x97	0x4E	0xCC
45	3760	0x98	0x4D	0xCC
45	3765	0x98	0x4D	0xCD
45	3770	0x98	0x4D	0xCD
45	3775	0x99	0x4C	0xCD
45	3780	0x99	0x4C	0xCE
45	3785	0x99	0x4B	0xCE

T (C)	Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
45	3790	0x9A	0x4B	0xCE
45	3795	0x9A	0x4B	0xCF
45	3800	0x9A	0x4A	0xCF
45	3805	0x9B	0x4A	0xD0
45	3810	0x9B	0x49	0xD0
45	3815	0x9B	0x49	0xD0
45	3820	0x9C	0x49	0xD1
45	3825	0x9C	0x48	0xD1
45	3830	0x9C	0x48	0xD1
45	3835	0x9C	0x47	0xD2
45	3840	0x9D	0x47	0xD2
45	3845	0x9D	0x47	0xD2
45	3850	0x9D	0x46	0xD3
45	3855	0x9E	0x46	0xD3
45	3860	0x9E	0x45	0xD3
45	3865	0x9E	0x45	0xD4
45	3870	0x9F	0x45	0xD4
45	3875	0x9F	0x44	0xD4
45	3880	0x9F	0x44	0xD5
45	3885	0xA0	0x44	0xD5
45	3890	0xA0	0x43	0xD5
45	3895	0xA0	0x43	0xD6
45	3900	0xA1	0x42	0xD6
45	3905	0xA1	0x42	0xD6
45	3910	0xA1	0x42	0xD7
45	3915	0xA1	0x41	0xD7
45	3920	0xA2	0x41	0xD7
45	3925	0xA2	0x41	0xD8
45	3930	0xA2	0x40	0xD8
45	3935	0xA3	0x40	0xD8
45	3940	0xA3	0x40	0xD9
45	3945	0xA3	0x3F	0xD9
45	3950	0xA4	0x3F	0xD9
45	3955	0xA4	0x3E	0xD9
45	3960	0xA4	0x3E	0xDA
45	3965	0xA5	0x3E	0xDA
45	3970	0xA5	0x3D	0xDA
45	3975	0xA5	0x3D	0xDB
45	3980	0xA5	0x3D	0xDB
45	3985	0xA6	0x3C	0xDB
45	3990	0xA6	0x3C	0xDB
45	3995	0xA6	0x3C	0xDC

T (C)	Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
45	4000	0xA7	0x3B	0xDC
45	4005	0xA7	0x3B	0xDC
45	4010	0xA7	0x3B	0xDD
45	4015	0xA7	0x3A	0xDD
45	4020	0xA8	0x3A	0xDD
45	4025	0xA8	0x3A	0xDD
45	4030	0xA8	0x39	0xDE
45	4035	0xA9	0x39	0xDE
45	4040	0xA9	0x39	0xDE
45	4045	0xA9	0x38	0xDE
45	4050	0xAA	0x38	0xDF
45	4055	0xAA	0x38	0xDF
45	4060	0xAA	0x37	0xDF
45	4065	0xAA	0x37	0xDF
45	4070	0xAB	0x37	0xDF
45	4075	0xAB	0x36	0xE0
45	4080	0xAB	0x36	0xE0
45	4085	0xAC	0x36	0xE0
45	4090	0xAC	0x35	0xE0
45	4095	0xAC	0x35	0xE1
45	4100	0xAD	0x35	0xE1
45	4105	0xAD	0x35	0xE1
45	4110	0xAD	0x34	0xE1
45	4115	0xAD	0x34	0xE1
45	4120	0xAE	0x34	0xE1
45	4125	0xAE	0x33	0xE2
45	4130	0xAE	0x33	0xE2
45	4135	0xAF	0x33	0xE2
45	4140	0xAF	0x33	0xE2
45	4145	0xAF	0x32	0xE2
45	4150	0xB0	0x32	0xE2
45	4155	0xB0	0x32	0xE3
45	4160	0xB0	0x31	0xE3
45	4165	0xB0	0x31	0xE3
45	4170	0xB1	0x31	0xE3
45	4175	0xB1	0x31	0xE3
45	4180	0xB1	0x30	0xE3
45	4185	0xB2	0x30	0xE3
45	4190	0xB2	0x30	0xE3
45	4195	0xB2	0x2F	0xE4
45	4200	0xB3	0x2F	0xE4
45	4205	0xB3	0x2F	0xE4

T (C)	Beta	C1 hexadecimal	C2 hexadecimal	C3 hexadecimal
45	4210	0xB3	0x2F	0xE4
45	4215	0xB4	0x2E	0xE4
45	4220	0xB4	0x2E	0xE4
45	4225	0xB4	0x2E	0xE4
45	4230	0xB4	0x2E	0xE4
45	4235	0xB5	0x2D	0xE4
45	4240	0xB5	0x2D	0xE4
45	4245	0xB5	0x2D	0xE5
45	4250	0xB6	0x2D	0xE5
45	4255	0xB6	0x2C	0xE5
45	4260	0xB6	0x2C	0xE5
45	4265	0xB7	0x2C	0xE5
45	4270	0xB7	0x2C	0xE5
45	4275	0xB7	0x2B	0xE5
45	4280	0xB8	0x2B	0xE5
45	4285	0xB8	0x2B	0xE5
45	4290	0xB8	0x2B	0xE5
45	4295	0xB9	0x2B	0xE5
45	4300	0xB9	0x2A	0xE5
45	4305	0xB9	0x2A	0xE5
45	4310	0xB9	0x2A	0xE5
45	4315	0xBA	0x2A	0xE6
45	4320	0xBA	0x29	0xE6
45	4325	0xBA	0x29	0xE6
45	4330	0xBB	0x29	0xE6
45	4335	0xBB	0x29	0xE6
45	4340	0xBB	0x28	0xE6
45	4345	0xBC	0x28	0xE6
45	4350	0xBC	0x28	0xE6
45	4355	0xBC	0x28	0xE6
45	4360	0xBC	0x27	0xE6
45	4365	0xBD	0x27	0xE6
45	4370	0xBD	0x27	0xE6
45	4375	0xBD	0x27	0xE7
45	4380	0xBE	0x27	0xE7
45	4385	0xBE	0x26	0xE7
45	4390	0xBE	0x26	0xE7
45	4395	0xBE	0x26	0xE7
45	4400	0xBF	0x26	0xE7

2.1.4.4 Battery identification

The fuel gauge module contains a dedicated BATT_ID pin for battery identification.

Some of the features of battery identification are:

- The battery identification is done automatically when a battery is attached.
- BATT_ID can be used for battery missing detection.

Battery ID detection

Battery ID detection uses a dedicated channel of the round-robin ADC within the fuel gauge. Detection is done by measuring the voltage across the battery ID resistor while enabling one of the three different current sinks. This detection is done on a battery insertion, and then subsequently whenever the software forces a fuel gauge restart. Once the detection is resolved, an interrupt is fired to inform the system about the result. See Table 2-5 for the interrupts available.

The detection is done in the following steps, and is automatically handled by hardware.

1. The detection sequence is started.
2. Pull-up is enabled with the weakest current level (5 µA).
3. First conversion is performed.

   If the converted value exceeds the allowed conversion range, step 3 is repeated with an increased bias current (5 µA→15 µA→150 µA).
4. The value of the battery ID is detected and the corresponding interrupt is triggered for software to calculate the RID value based on the measured voltage.

Table 2-5 Interrupt peripherals

Interrupt	Description
BT_ID	Interrupt to notify the software that the hardware has measured BATT_ID

2.1.4.5 RID selection

- An RID value must be selected within the supported detectable range: 1 kΩ to 450 kΩ.
- A 20% tolerance difference is recommended between RID values for multiple cells.
- 7.5 kΩ ± 15%, with a typical 5% tolerance RID, is designated in the hardware to be used when a debug board or power supply that is unable to be charged is used. This value prevents charging in the hardware and should not be used unless this is the preferred behavior. This value is the hardware default but can be configured in software. For more information about software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software* (80-P2484-74).

2.1.5 Battery missing detection

Battery missing detection (BMD) by default is done via the dedicated BATT_ID pin. If preferred, the BATT_THERM, or both the BATT_ID and BATT_THERM pins, can be used for BMD. When triggered, hardware raises a flag for the software to handle. When BMD is high, the fuel gauge pauses and sends a signal to the charger module to suspend charging. The fuel gauge waits until it detects that the battery is present again before taking a new first SoC estimate and allowing the charger to resume charging.

2.1.6 Charger interaction

2.1.6.1 Charge termination

The fuel gauge is used as the charger termination source by hardware default. This is a simple digital comparison vs. the IADC measurements that take place every 1.47 s. Once the threshold is achieved, the signal is sent to the charger block to terminate charging via hardware.

2.1.6.2 Thermal monitoring (JEITA)

The fuel gauge and charger modules are compliant with the latest JIS8714 and JEITA standard safety requirements. The battery's temperature information measured via BATT_THERM is used to modulate the battery charging voltage and current when temperature is in between programmable ranges (see Figure 2-2).

The fuel gauge contains four settable thresholds: the JEITA_TOO_HOT, JEITA_TOO_COLD, JEITA_HOT, and JEITA _COLD. When the battery's temperature is between JEITA_HOT and JEITA_COLD, the battery must be charged with the default charging current (ICHG) and floating voltage (VFLOAT). When the battery's temperature exceeds either JEITA_HOT or JEITA_COLD (but not JEITA_TOO_HOT and JEITA_TOO_COLD), the respective charging current and floating voltage are modulated to JEITA_CCCOMP and JEITA_FVCOMP. Both JEITA_CCCOMP and JEITA_FVCOMP are user-programmable. If the temperature exceeds the JEITA_TOO_COLD or JEITA_TOO_HOT ranges, charging is terminated until the temperature returns to within a valid range.

2.1.6.3 Auto recharge

The fuel gauge is used as the auto recharge source for both voltage and SoC based recharge schemes. By default, voltage based recharge is configured, but SoC based recharge can be configured in software. For more information about how to configure auto recharge in software, refer to the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

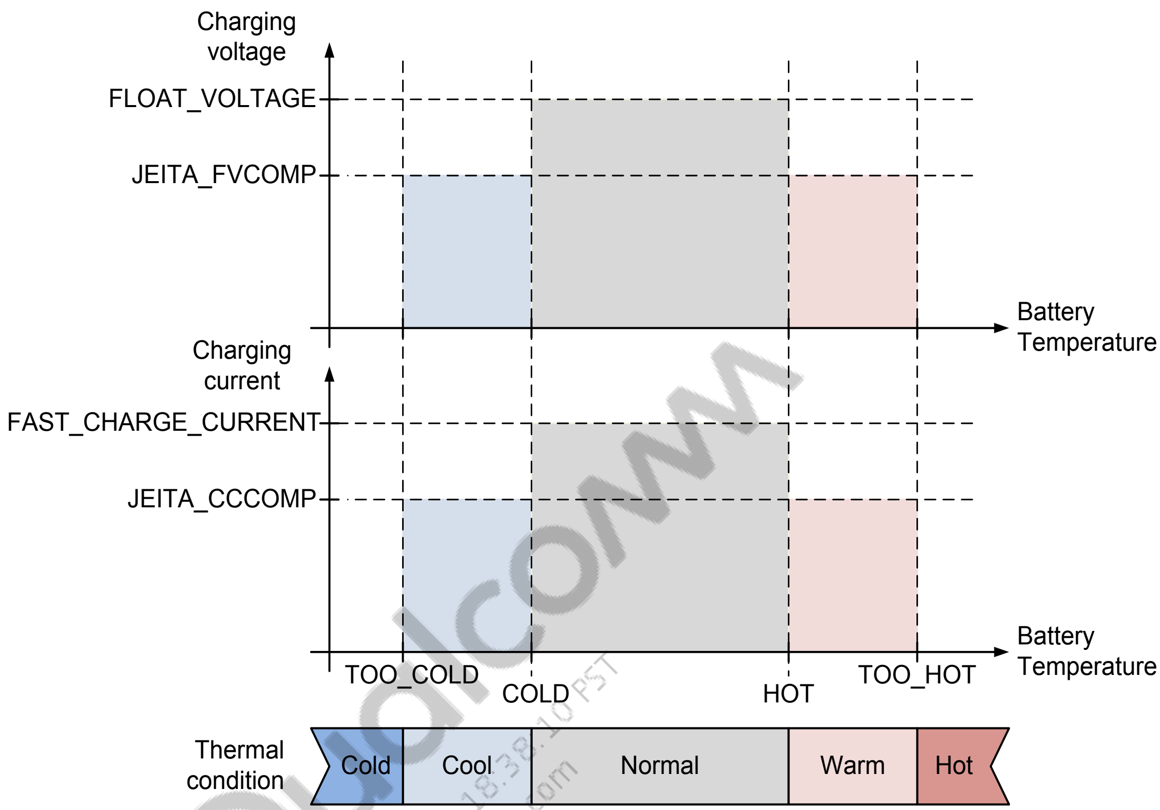

Figure 2-2 JEITA standard regions example

2.1.7 Battery current limiting

Battery current limiting (BCL) is a QTI proprietary function for quickly mitigating high-load conditions that would cause dips in the battery voltage low enough to cause a premature shutdown or crash of the system. BCL is available on the PMI8998 and includes both current and voltage monitoring of the battery. BCL was designed to be easily configured and used, and only needs fine-tuning of a few select parameters. For more information about BCL tuning, refer to *Battery Current Limit (BCL) Overview and Tuning* (80-NM328-709) and the *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

Table 2-6 BCL specifications

BCL mode	Condition	VBATT and IBATT refresh rate
Low power mode (LPM)	Battery current < iBtHpmTh	Every 1.47 s for 160 ms
High power mode (HPM)	Battery current > iBtHpmTh	Every 640 μs

2.2 Fuel gauge algorithm

2.2.1 SoC

2.2.1.1 Overview

The fuel gauge reports SoC in four different layers:

- CC SoC: This is the coulomb-counted SoC represented as a percentage of the typical battery capacity found during characterization.
- Battery SoC: This is calculated using both the CC SoC and the voltage mode correction. This layer adds in the voltage mode correction on top of the coulomb-counted SoC, using the battery model and profile.
- System SoC: This is a filtered version of the battery SoC, and is responsible for the endpoint matching of the cutoff voltage and 0% SoC point, and the termination current and the 100% SoC point.
- Monotonic SoC: This is the final layer of SoC that is displayed to the end user. This layer provides control on the slope and monotonicity of the SoC.

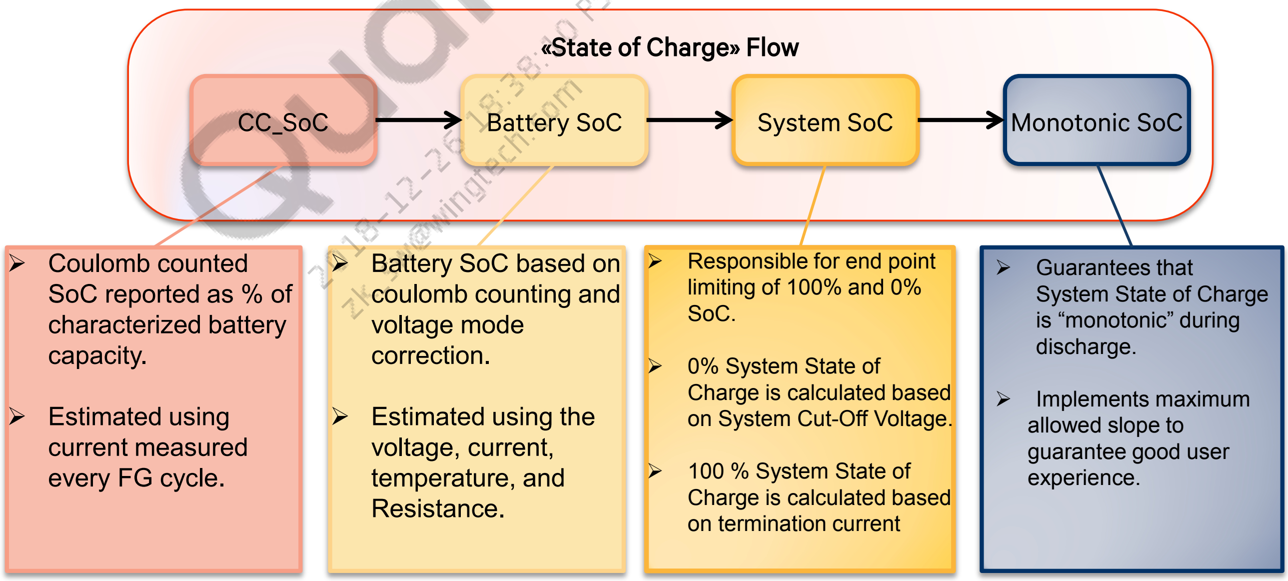

Figure 2-3 SoC flow diagram

2.2.1.2 CC SoC

CC SoC is the first layer of the fuel gauge SoC algorithm, and is responsible for reporting a coulomb-counted value. This SoC is reported as a percentage of the battery capacity, typically found during battery characterization. If capacity learning is enabled, this capacity can change. Note that CC SoC should not be used by customers for coulomb counting as it can be reset to compensate for accumulated error between the coulomb count and the voltage mode estimations. Customers can instead use the newer CC_SoC_SW register that is designed for the purpose of reading a strict coulomb count. This register works similar to CC SoC in that it is a percentage of the capacity, but does not get changed or reset.

2.2.1.3 Battery SoC

Battery SoC is a combination of the coulomb-counted SoC and the voltage mode correction algorithm. The voltage mode algorithm uses measured voltage, current, real-time, measured ESR data, and modeled data of how open circuit voltage (OCV) and Rslow shift over temperature and SoC. This feedback loop uses the battery profile and battery model to calculate the OCV that helps correct any error accumulated during coulomb counting or due to an aged battery cell. This creates a more robust and accurate SoC over many use cases. Figure 2-4 shows how the mixing is performed.

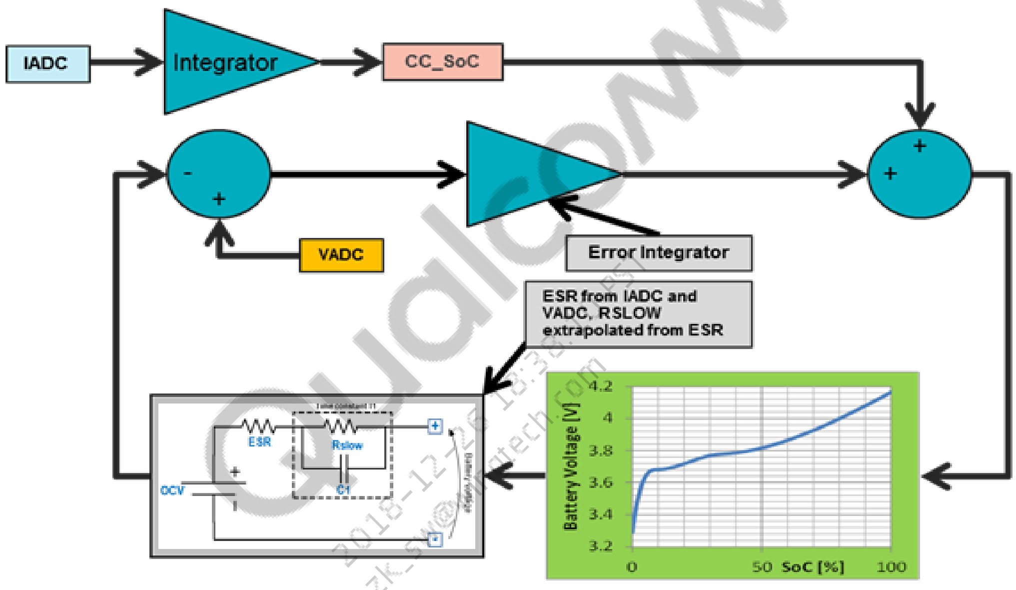

Figure 2-4 Battery SoC algorithm diagram

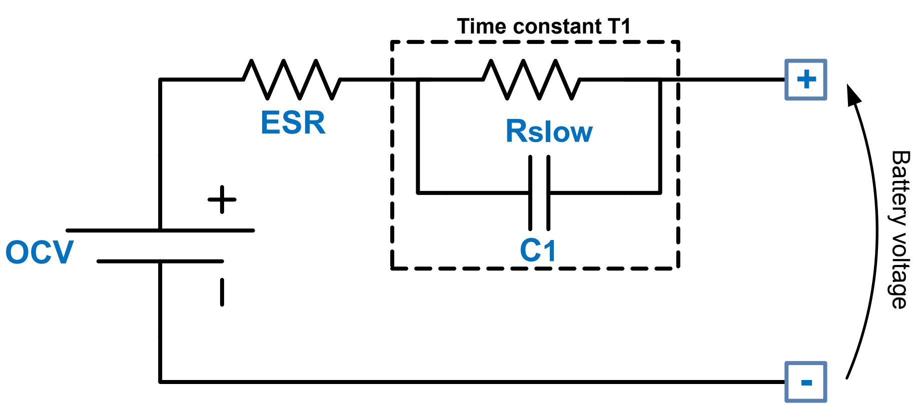

Figure 2-5 Battery model

The fuel gauge voltage mode algorithm uses the battery's electrical model shown in Figure 2-5. This models the electrical behavior of a lithium-ion battery and it includes the following parameters:

- OCV: This is the voltage present at the terminals when no load is applied to the cell, and settled for a long period of time (that is capacitor C1 is completely discharged). OCV varies with battery temperature and the battery's current amount of charge.
- ESR: ESR is the instantaneous resistance measured across the battery terminal during load transitions. It is a combination of the PCM, board impedance, and the limitation of the ability of the battery's chemistry to supply instantaneous current.
- Rslow (C1 time constant): Rslow is the resistance measured across the battery terminal while a battery is left to settle, with no load applied. This pole is a characteristic of all Li-ion batteries. The QTI fuel gauge models Rslow as a ratio of ESR, with additional scaling over temperature, and SoC.

2.2.1.4 System SoC

The system SoC is the intermediate layer between the battery and the monotonic SoC, and allows for full SoC and cutoff SoC configuration.

NOTE: The loop input parameters must be configured in software according to the customer's preferred operation based on the guidance provided below.

Endpoint matching loops

The **full SoC** loop is the feedback loop responsible for matching the 100% SoC point to the IBATT full current setting. The configurable inputs to these loops are IBATT full and VBATT full, both of which need to be set with the guidance provided here to achieve an accurate matching.

The **cutoff SoC** is the feedback loop responsible for matching the 0% SoC point to the VBATT cutoff setting. The configurable input to this loop is VBATT cutoff.

Endpoint matching loop inputs

IBATT full: This is the threshold that is used to define when 100% SoC is reported based on battery current. This should be set such that it is reached before the charge termination current threshold, or 100% SoC may not be reached. Note that the current is reported as negative during charging, so this technically must be more negative than the charge termination current.

VBATT full: This setting is used in the feedback loop responsible for matching the 100% SoC point to the programmed IBATT full termination current. This must be set 10 mV less than the maximum charge voltage of the battery in use to compensate for charger inaccuracy. If this is set incorrectly, poor IBATT full current accuracy is observed.

VBATT cutoff: This setting is used in the feedback loop to report 0% SoC. It is recommended that the value is not dropped too low or it may negatively impact system stability.

VBATT empty: This is a backup battery voltage based threshold used to shut down the device in case 0% is not reached. This should be left as default, unless otherwise required by the customer.

NOTE: The accuracy of these loops is dependent on the correctness of the configuration settings and the battery SoC accuracy. If the battery SoC accuracy is poor, upon reaching the endpoint this inaccuracy directly affects the matching accuracy.

See Figure 2-6 for an example of the full SoC parameter, and Figure 2-7 for an example of both full SoC and cutoff SoC.

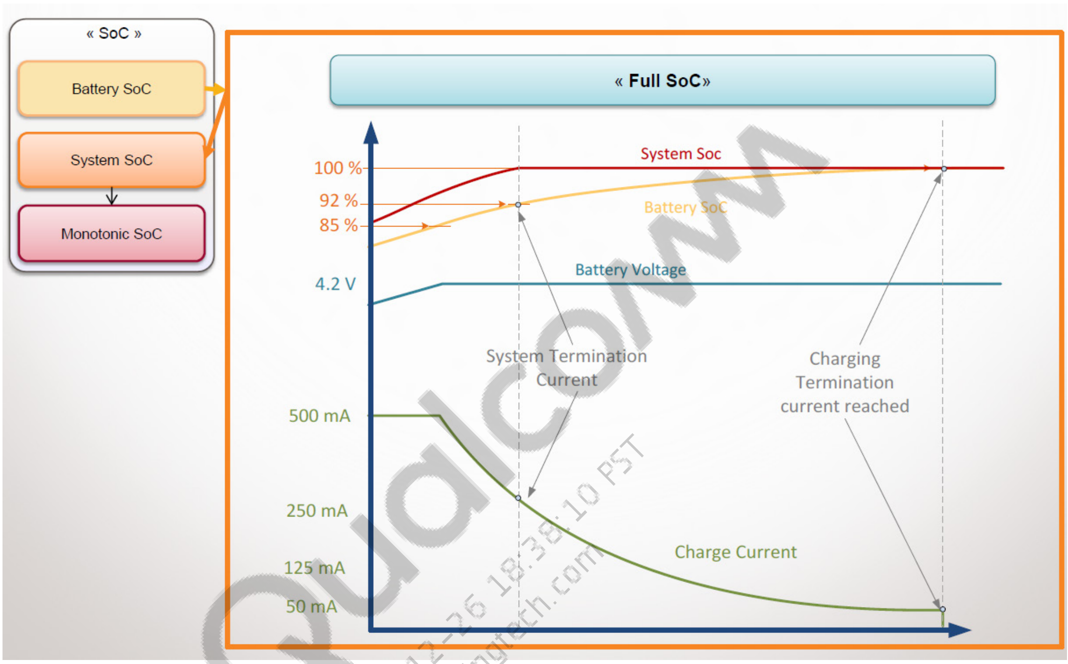

Figure 2-6 System full SoC example

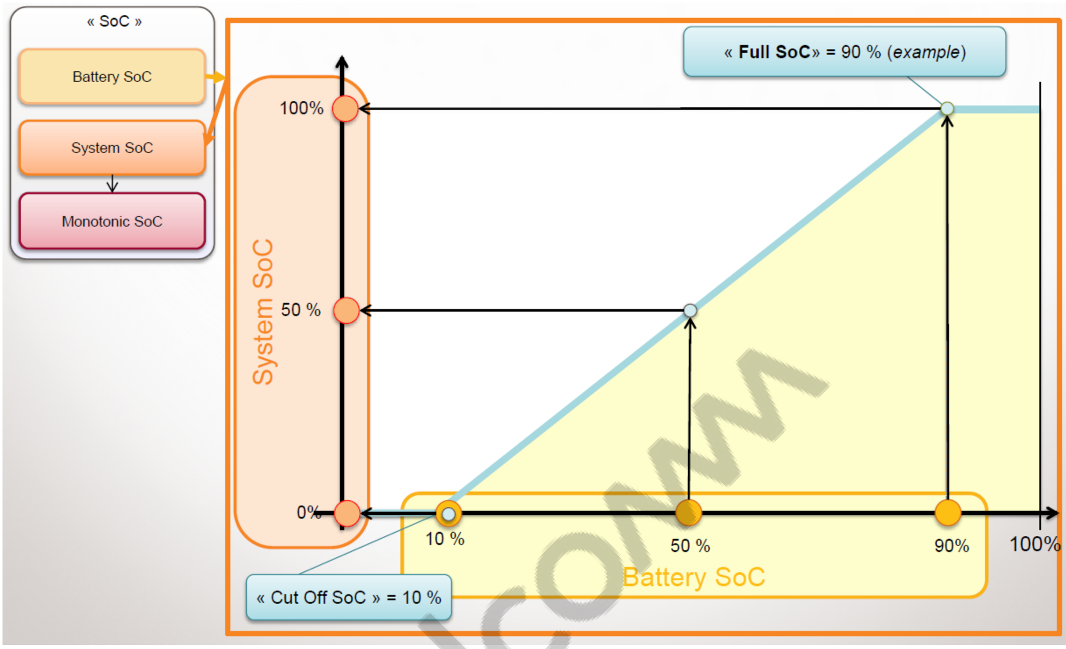

Figure 2-7 System full and cut-off SoC example

2.2.1.5 Monotonic SoC

Because a change in the battery operating conditions could result in an increase or decrease of the available system SoC, filtering is applied to obtain the appropriate monotonic behavior. Monotonic filtering handles situations where the cutoff SoC is increased and then decreased, which can happen due to:

- Increase and decrease of the battery internal resistance value
- Increase and decrease of the battery load current

The slope limiter is required in certain event sequences where the monotonic filter is applied and then a change in the operating conditions occurs that would otherwise cause the clamped system SoC value to propagate directly to the monotonic SoC read by the user, thus creating an unwanted step. The slope limiter prevents steps in the monotonic SoC. See Figure 2-8.

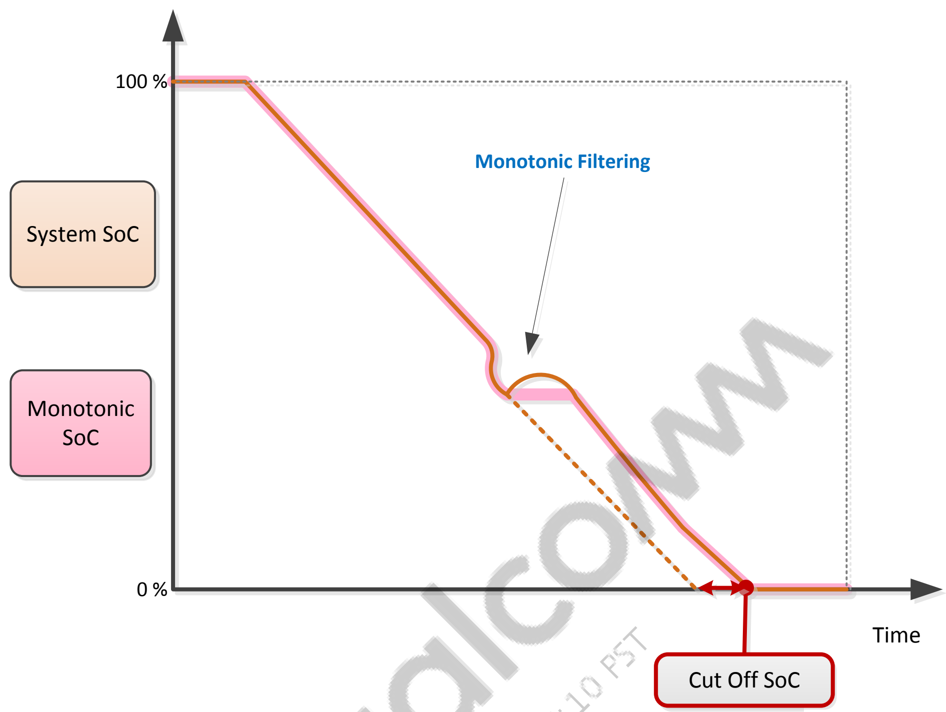

Figure 2-8 Monotonic SoC example

2.2.1.5.1 Slope limiting

Slope limiting depends on the monotonic slope limiter configuration register 0d3[0:7]. It is represented as a percentage of the monotonic SoC. This can be configured in the software. Refer to *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74) for more information.

For example, the monotonic slope limiter configuration is set to decimal 1; it allows a 1% monotonic SoC change every fuel gauge update cycle (1.47 s).

2.2.2 Battery profiling

2.2.2.1 Battery characterization

Customers may submit a battery characterization case type via Salesforce to request battery characterization. Customers may also characterize their batteries themselves using the software tool QBCSW, but cannot to generate the final profile needed. If customers do their own characterization, QTI still must post-process the battery data to generate the final profile. QBCSW is available to customers via Qualcomm® CreatePoint. For more information on QBCSW, refer to the *Battery Characterization Process Application Note* (80-VT310-24).

Profile generation is broken down into two steps:

1. Characterization: Characterization is the process through which the battery impedance, capacity, and OCV relationships are all measured over temperature and state of charge; for both charge and discharge. Only raw data is generated and cannot be used by the fuel gauge until the post processing step.

2. Post-processing: After characterization, high iteration curve fitting is done on the raw data to generate the necessary polynomial coefficients for the fuel gauge models. These polynomials model how Rslow and OCV vary over temperature and SoC, and are necessary for fuel gauge operation. This software is proprietary and not available to customers. The final profile approval process is completed by a stringent review of the battery data of each battery, as well as the averaged profile, to ensure accuracy.

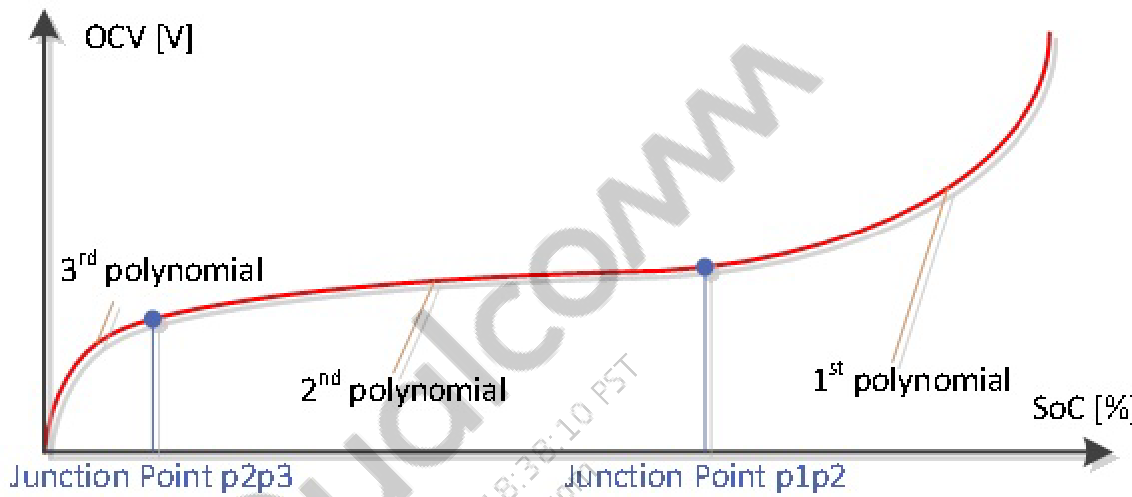

Figure 2-9 OCV vs. SoC curve

2.2.3 Fuel gauge plots

This section gives timing information and example plots of different fuel gauge phenomena and the corresponding high-level explanation.

2.2.3.1 BATT_THERM_BIAS and VREG_FG

Figure 2-10 and Figure 2-11 show the measured examples of when the ADC is taking synchronous voltage and current measurements and battery thermistor measurements. VREG_FG is an internal regulator responsible for supplying the reference to the thermistor BIAS and supplying the ADC when taking measurements. When not regulating, it is approximately ~VBATT.

Figure 2-10 shows BATT_THERM_BIAS enabled for ~30 ms while taking a temperature measurement.

NOTE: BATT_THERM_BIAS is a switch. On PMI8998, BATT_THERM_RBIAS is enabled ~4 ms early, passing ~VBATT before regulating. This is the expected behavior and does not cause any temperature measurement issues.

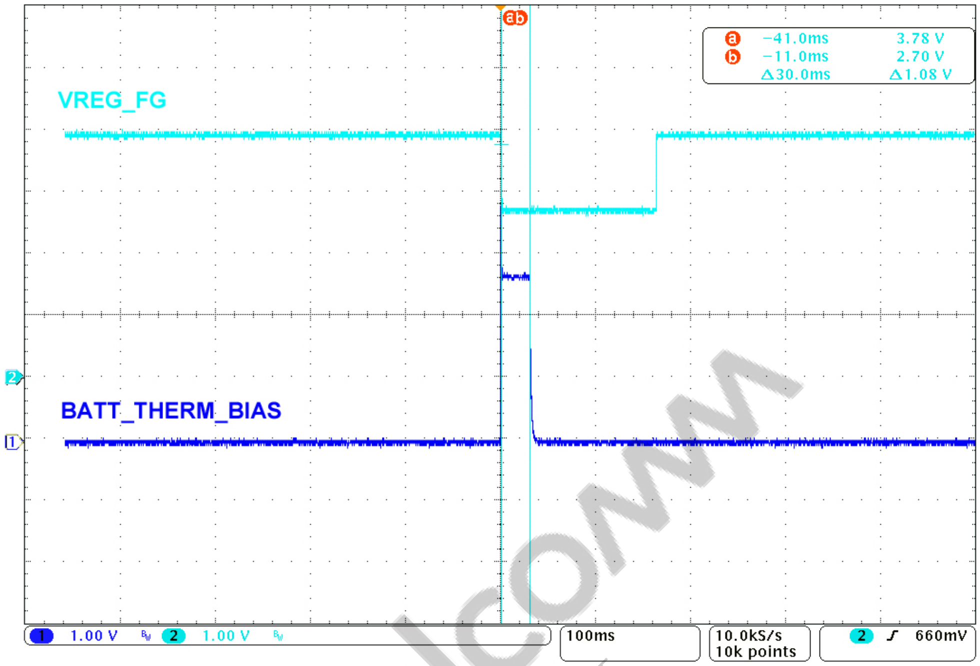

Figure 2-10 BATT_THERM_BIAS measured

Figure 2-11 shows VREG_FG regulating for ~ 163 ms, to supply BATT_THERM_RBAS and supply the IADC and VADC during the synchronous conversions voltage and current conversions.

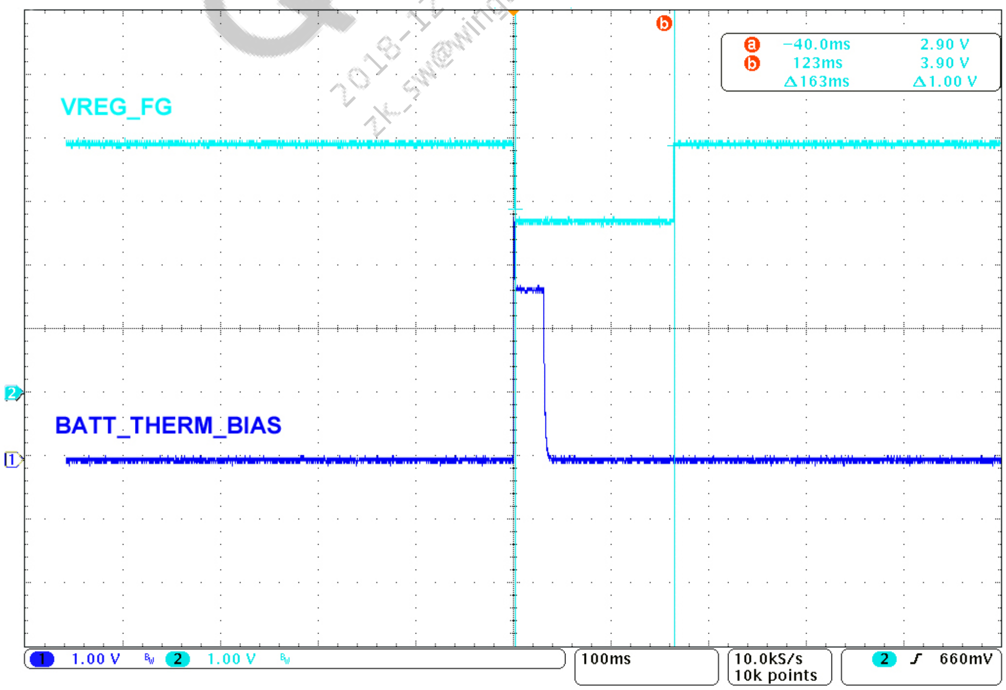

Figure 2-11 VREG_FG measured

2.2.3.2 Fuel gauge cycle

Every 1.47 s, which is the duration of a fuel gauge cycle, VREG_FG regulates for BATT_THERM_BIAS and the fuel gauge V and I ADC, while BATT_THERM_BIAS is enabled to bias BATT_THERM for battery temperature measurements.

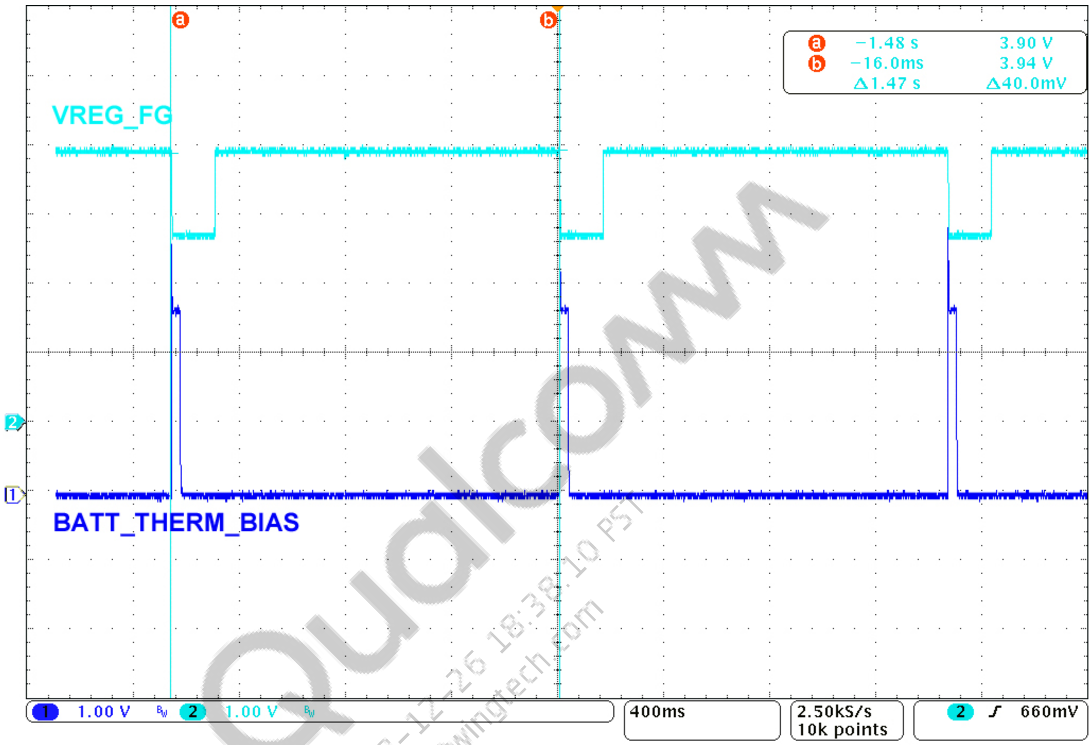

Figure 2-12 Fuel gauge cycle measured

2.2.3.3 Battery resistance measurements

Figure 2-13 and Figure 2-14 show two examples of the fuel gauge creating the necessary load transients to take the ESR measurements.

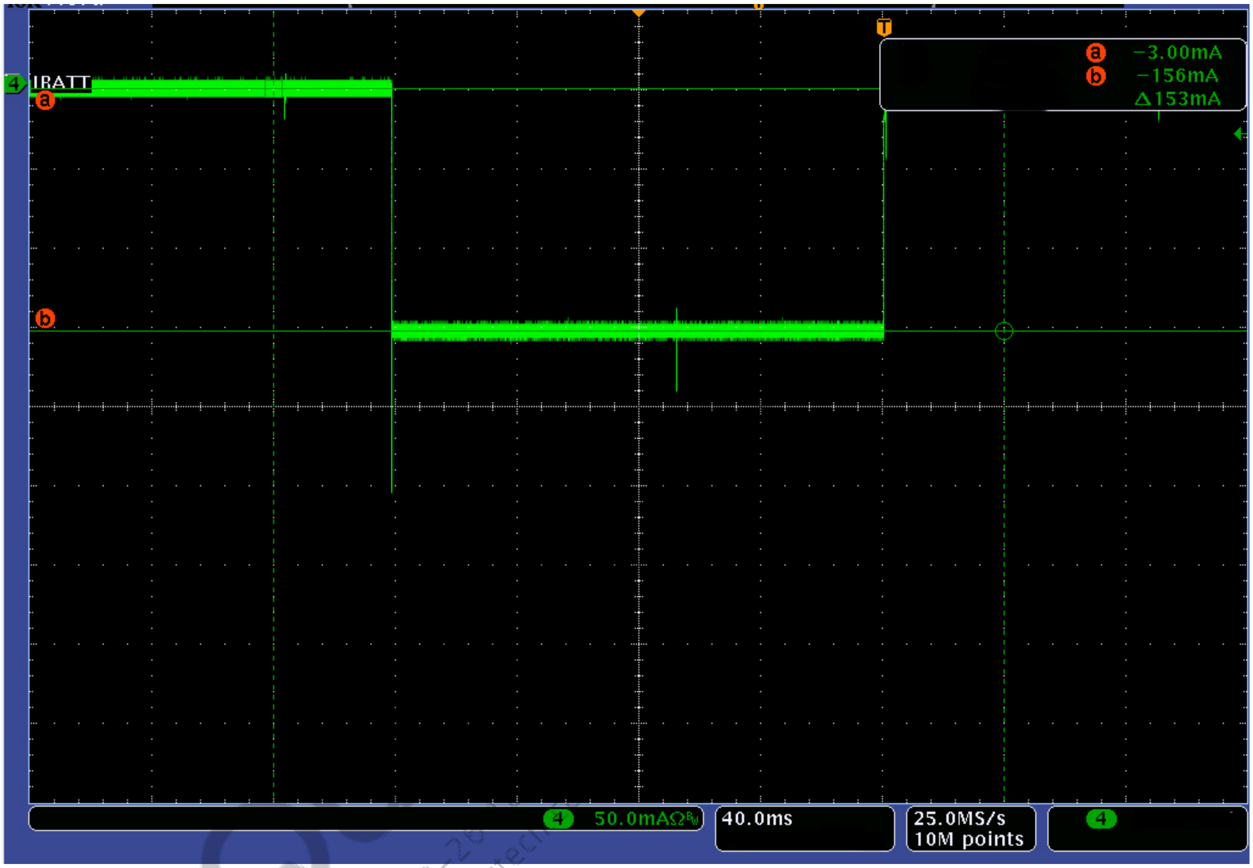

Figure 2-13 ~150 mA ESR pulse example taken on bench with no load

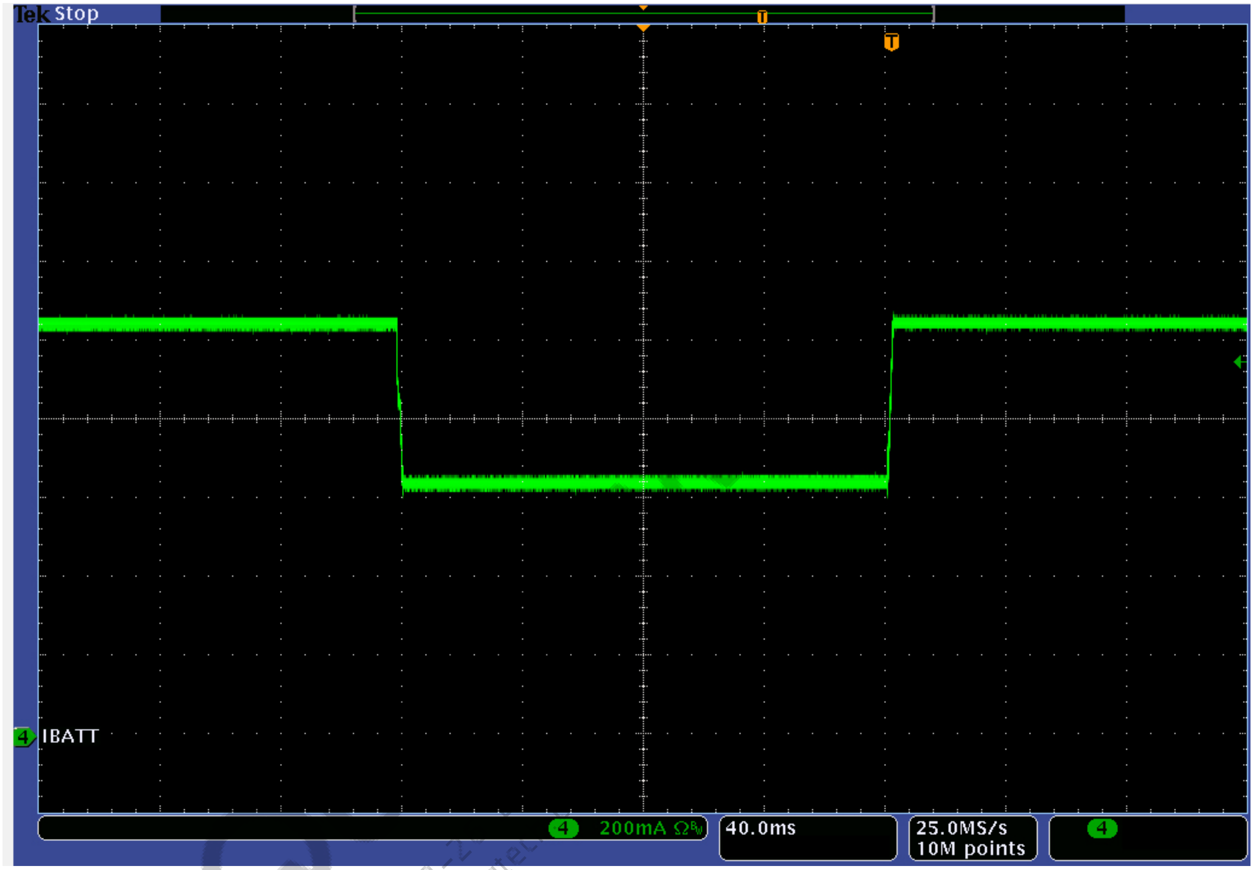

Figure 2-14 Charge current decrement during 1 A charging

2.2.3.4 Fuel gauge current consumption

Every 1.47 s, the fuel gauge consumes ~1 mA of current during synchronous ADC conversions of voltage and current.

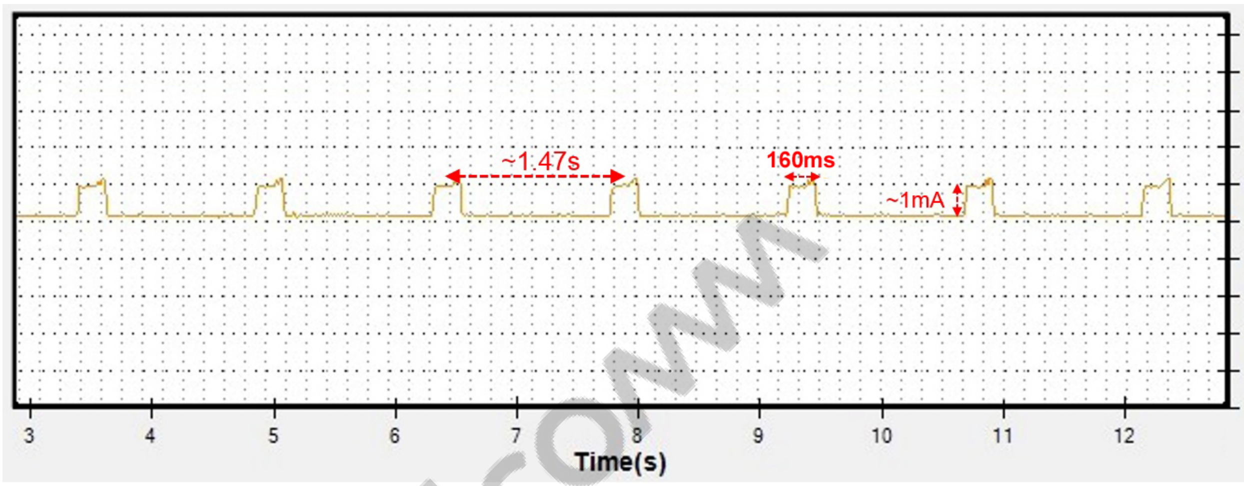

Figure 2-15 Example plot for 1 mA current pulses

2.2.3.5 BATT_ID after battery insertion

Immediately after battery insertion, the hardware automatically does an RID conversion. For more information about how the RID conversion process works, see Section 2.1.4.4. Figure 2-16 shows biasing taking place for BMD; then the current sources toggle for a 100 kΩ resistor. This is expected behavior.

NOTE: The magnitudes of these voltages vary, depending on the RID value used.

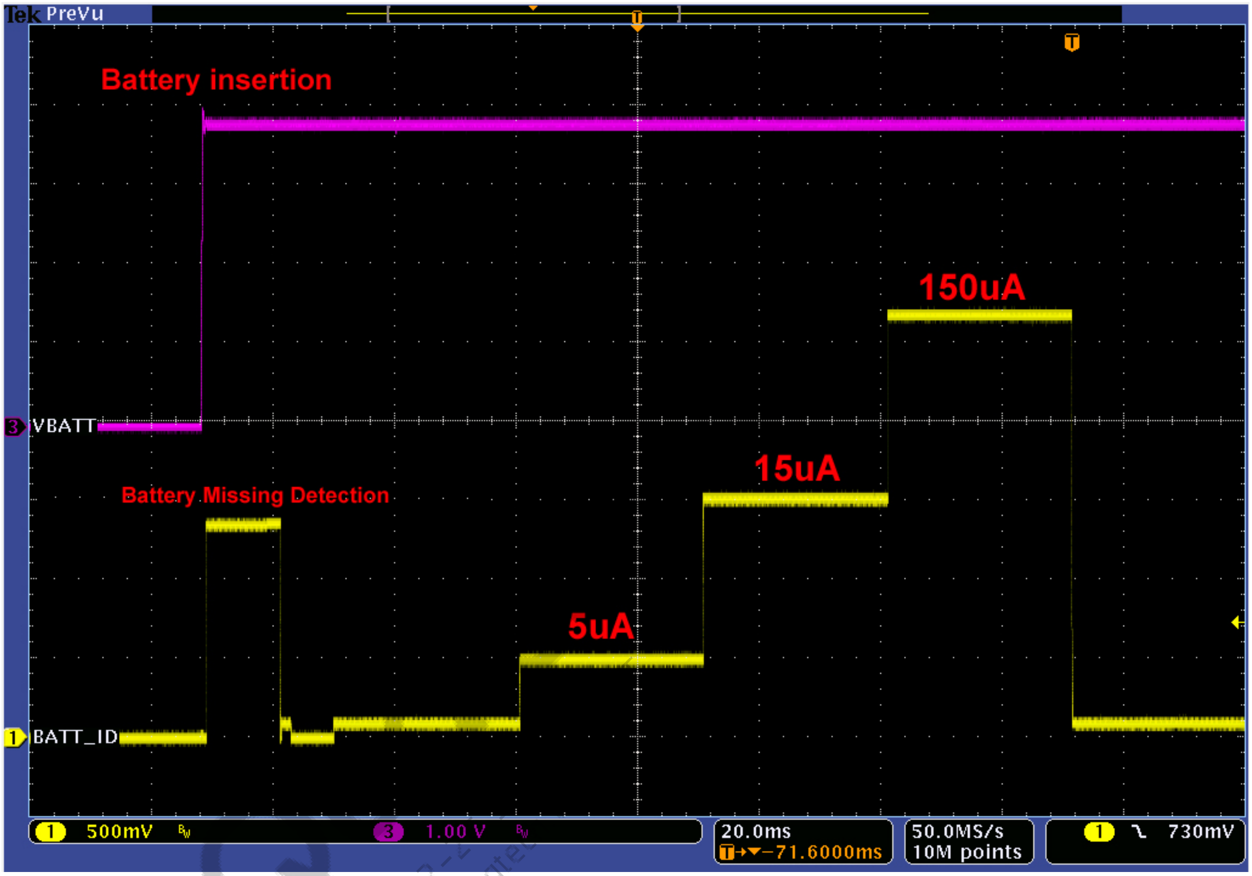

Figure 2-16 100 kΩ RID

3 Register information

3.1 FG_MEMIF_SRAM access

Most of the fuel gauge registers must be accessed using the indirect addressing controls located in the FG_MEMIF peripheral.

Table 3-1 SRAM partitioning

Partition	Start (decimal)	End (decimal)	Size (bytes)
System register range	0	19	19
Battery profile in use	24	61	37
Scratchpad	80	124	44

3.2 Fuel gauge interrupts

Table 3-2 Fuel gauge interrupts

Interrupt	Peripheral	Description	Related SPMI peripheral register
MSOC_FULL	FG_BATT_SOC	Monotonic SoC = 100%	0x4010
MSOC_HIGH	FG_BATT_SOC	Monotonic SoC ≥ high threshold	0x4010
MSOC_EMPTY	FG_BATT_SOC	Monotonic SoC = 0% OR VBATT < Volt_Empty	0x4010
MSOC_LOW	FG_BATT_SOC	Monotonic SoC ≤ low threshold	0x4010
MSOC_DELTA	FG_BATT_SOC	Monotonic SoC change exceeds programmed delta	0x4010
BSOC_DELTA	FG_BATT_SOC	Battery SoC change exceeds programmed delta	0x4010
SOC_READY	FG_BATT_SOC	Fuel gauge completes a first estimate	0x4010
SOC_UPDT	FG_BATT_SOC	Triggers when SoC information is updated (every cycle)	0x4010
BT_TMPR_DELTA	FG_BATT_INFO	Battery temperature change exceeds programmed delta	0x4110
WDOG_EXP	FG_BATT_INFO	Fuel gauge watchdog has expired	0x4110
VBT_LOW	FG_BATT_INFO	Battery voltage < VBATT_LOW threshold	0x4110
BT_MISS	FG_BATT_INFO	Battery missing interrupt	0x4110
DMA_GNT	FG_MEM_IF	Direct memory access granted	0x4410

Interrupt	Peripheral	Description	Related SPMI peripheral register
MEM_XCP	FG_MEM_IF	Interleave memory access exception	0x4410
IMA_RDY	FG_MEM_IF	Ready or end of transaction depending on the configuration	0x4410
RR	FG_ADC_RR	Fresh round-robin data available	0x4510
BT_ID	FG_ADC_RR	Battery ID conversion data available	0x4510

3.3 Fuel gauge RAM register map

Table 3-3 Register map

Register name	Description	Type	Address (dec)	Bit offset	Bit width	LSB	Signed	Value offset	Max range (unsigned)	Pos. range (signed)	Neg. range (signed)	Unit
System_Memory_Partition												
ESR_Current_Threshold	Minimum current difference to allow ESR extraction between two pairs of VBATT/IBATT readings	ac_fixed	0d2	24	8	0.00390625	N	0	0.99609375	N/A	N/A	A
Monotonic_Slope_Limiter_Config	Monotonic SoC change limiter	ac_fixed	0d3	0	8	0.00012207	N	0	0.03112785	N/A	N/A	%
IBATT_Cutoff	Minimum battery current value used for the cutoff SoC estimate	ac_fixed	0d4	0	18	0.00012207	Y	0	N/A	15.9997759	-16.0000201	A
VBATT_Cutoff	Battery voltage set point used to estimate the cutoff SoC	ac_fixed	0d5	0	15	0.00024414	N	0	7.999768147	N/A	N/A	V
IBATT_Full	Battery current set point used to estimate the full SoC	ac_fixed	0d6	0	18	0.00012207	Y	0	N/A	15.9997759	-16.0000201	I
VBATT_Full	Battery voltage set point used to estimate the full SoC: this is the battery float voltage at which the specific battery termination current should be observed	ac_fixed	0d7	0	15	0.00024414	N	0	7.999768147	N/A	N/A	V
ESR_Update_Tight	Maximum increase or decrease (1 ± esrUpdTight %) of ESR with tight filtering	ac_fixed	0d8	0	8	0.00195312	N	0	0.4980456	N/A	N/A	%
ESR_Update_Broad	Maximum increase or decrease (1 ± esrUpdBroad %) of ESR with relaxed filtering	ac_fixed	0d8	8	8	0.00195312	N	0	0.4980456	N/A	N/A	%
ESR_Update_Tight_Low_Temp	Maximum increase or decrease (1 ± esrUpdTightLowTemp %) of ESR with tight filtering in a low battery temperature condition	ac_fixed	0d8	16	8	0.00195312	N	0	0.4980456	N/A	N/A	%
ESR_Update_Broad_Low_Temp	Maximum increase or decrease (1 ± esrUpdBroadLowTemp %) of ESR with relaxed filtering in a low battery temperature condition	ac_fixed	0d8	24	8	0.00195312	N	0	0.4980456	N/A	N/A	%

Register name	Description	Type	Address (dec)	Bit offset	Bit width	LSB	Signed	Value offset	Max range (unsigned)	Pos. range (signed)	Neg. range (signed)	Unit
ESR_Low_Temp_Filter_Threshold	Low temperature threshold in battery ESR filtering to select the right amount of allowed percentage change	ac_fixed	0d9	0	8	0.5	N	0	127.5	N/A	N/A	K
Ki_Discharge_Low_Current	Ki integration coefficient for low discharge current level; used inside the battery SoC voltage mode loop	ac_fixed	0d9	24	8	0.00024414	N	0	0.062255955	N/A	N/A	N/A
Ki_Discharge_Medium_Current	Ki integration coefficient for medium discharge current level; used inside the battery SoC voltage mode loop	ac_fixed	0d10	0	8	0.00024414	N	0	0.062255955	N/A	N/A	N/A
Ki_Discharge_High_Current	Ki integration coefficient for high discharge current level; used inside the battery SoC voltage mode loop	ac_fixed	0d10	8	8	0.00024414	N	0	0.062255955	N/A	N/A	A
Ki_Charge_Low_Current	Ki integration coefficient for low charge current level; used inside the battery SoC voltage mode loop	ac_fixed	0d11	0	8	0.00024414	N	0	0.062255955	N/A	N/A	N/A
Ki_Charge_Medium_Current	Ki integration coefficient for medium charge current level; used the inside battery SoC voltage mode loop	ac_fixed	0d11	8	8	0.00024414	N	0	0.062255955	N/A	N/A	N/A
Ki_Charge_High_Current	Ki integration coefficient for low high current level; used inside the battery SoC voltage mode loop	ac_fixed	0d11	16	8	0.00024414	N	0	0.062255955	N/A	N/A	N/A
Ki_Cutoff	Ki integration coefficient used for the cutoff SoC voltage mode loop	ac_fixed	0d12	8	8	0.00024414	N	0	0.062255955	N/A	N/A	N/A
Ki_Full	Ki integration coefficient used for the full SoC voltage mode loop	ac_fixed	0d12	16	8	0.00024414	N	0	0.062255955	N/A	N/A	A
BSoC_Delta_IRQ_Threshold	Threshold on the battery system SoC change, used to issue the related interrupt. The delta in battery SoC must exceed DeltaSoC by one LSB.	ac_fixed	0d12	24	8	0.00048828	N	0	0.124511655	N/A	N/A	%
MSoC_Delta_IRQ_Threshold	Threshold on the monotonic system SoC change, used to issue the related interrupt. The delta in monotonic SoC must exceed DeltaSoC by one LSB.	ac_fixed	0d13	0	8	0.00048828	N	0	0.124511655	N/A	N/A	%
MSoC_Low_IRQ_Threshold	Low SoC monotonic interrupt threshold	ac_fixed	0d13	8	8	0.00390625	N	0	0.99609375	N/A	N/A	%
MSoC_High_IRQ_Threshold	High SoC monotonic interrupt threshold	ac_fixed	0d13	16	8	0.00390625	N	0	0.99609375	N/A	N/A	%

Register name	Description	Type	Address (dec)	Bit offset	Bit width	LSB	Signed	Value offset	Max range (unsigned)	Pos. range (signed)	Neg. range (signed)	Unit
MSoC_Full_Threshold	Threshold for monotonic SOC reaching 100%	ac_fixed	0d13	24	8	0.00390625	N	0	0.99609375	N/A	N/A	%
MSoC_Minimum_OTG_Threshold	Minimum value of monotonic SoC that allows OTG	ac_fixed	0d14	0	8	0.00390625	N	0	0.99609375	N/A	N/A	%
MSoC_Automatic_Recharge_Threshold	Monotonic SoC threshold for automatic recharge	ac_fixed	0d14	8	8	0.00390625	N	0	0.99609375	N/A	N/A	%
		ac_fixed	0d14	16	8	0.00390625	N	0	0.99609375	N/A	N/A	A
IBATT_Charge_Termination_Threshold	Battery current has reached the charger termination threshold. Used to decide when to terminate charging.	ac_fixed	0d15	8	8	0.00390625	N	0	0.99609375	N/A	N/A	A
VBATT_Empty_IRQ_Threshold	Low battery voltage threshold for the empty interrupt	ac_fixed	0d15	24	8	0.015625	N	2	5.984375	N/A	N/A	V
VBATT_Low_IRQ_Threshold	Low battery voltage interrupt threshold	ac_fixed	0d16	0	8	0.015625	N	2	5.984375	N/A	N/A	V
VBATT_Automatic_Recharge_Threshold	Voltage based automatic recharge threshold	ac_fixed	0d16	8	8	0.015625	N	0	3.984375	N/A	N/A	N/A
ESR_Timer_Discharge_Max	ESR timer config for discharging: maximum value	ac_fixed	0d17	0	16	1	N	0	65535	N/A	N/A	N/A
ESR_Timer_Charge_Max	ESR timer config for charging: maximum value	ac_fixed	0d18	0	16	1	N	0	65535	N/A	N/A	N/A
MSoC_Empty_Use_SoC	Enable SoC 0% as a trigger of an empty interrupt	bool	0d19	4	1	N/A	N	0	Boolean (1 or 0)	N/A	N/A	N/A
MSoC_Empty_Use_Voltage	Enable VBATT less than vBtEmpty as a trigger of an empty interrupt	bool	0d19	5	1	N/A	N	0	Boolean (1 or 0)	N/A	N/A	N/A
Battery_Profile_Memory_Partition												
Discharge_Rseries_to_Rslow_Coefficient	Discharge ESR to the Rslow average scaling factor	ac_fixed	0d34	0	8	0.015625	N	0	3.984375	N/A	N/A	N/A
Charge_Rseries_to_Rslow_Coefficient	Charge ESR to the Rslow average scaling factor	ac_fixed	0d51	0	8	0.015625	N	0	3.984375	N/A	N/A	N/A
Battery_Capacity_Nominal	Nominal battery capacity	ac_fixed	0d58	0	16	1	N	0	65535	N/A	N/A	mAh
Nominal_Float_Voltage	Battery nominal float voltage	ac_fixed	0d58	16	15	0.00024414	N	0	7.999768147	N/A	N/A	V

Register name	Description	Type	Address (dec)	Bit offset	Bit width	LSB	Signed	Value offset	Max range (unsigned)	Pos. range (signed)	Neg. range (signed)	Unit
Scratch_Pad_Memory_Partition												
SW_Learned_Actual_Capacity	Software backup of actual capacity learned (LSB = 1 mAh)	ac_fixed	0d74	0	16	1	0	0	65536	N/A	N/A	mAh
SW_Cycle_Counter_Bin1	Software cycle counter bin 1, number of cycles in the 0–12.5% SoC bin	ac_fixed	0d75	0	16	1	0	0	65536	N/A	N/A	N/A
SW_Cycle_Counter_Bin2	Software cycle counter bin 2, number of cycles in the 12.5–25% SoC bin	ac_fixed	0d75	16	16	1	0	0	65536	N/A	N/A	N/A
SW_Cycle_Counter_Bin3	Software cycle counter bin 3, number of cycles in the 25–37.5% SoC bin	ac_fixed	0d76	0	16	1	0	0	65536	N/A	N/A	N/A
SW_Cycle_Counter_Bin4	Software cycle counter bin, number of cycles in the 37.5–50% SoC bin	ac_fixed	0d76	16	16	1	0	0	65536	N/A	N/A	N/A
SW_Cycle_Counter_Bin5	Software cycle counter bin 5, number of cycles in the 50–62.5% SoC bin	ac_fixed	0d77	0	16	1	0	0	65536	N/A	N/A	N/A
SW_Cycle_Counter_Bin6	Software cycle counter bin 6, number of cycles in the 62.5–75% SoC bin	ac_fixed	0d77	16	16	1	0	0	65536	N/A	N/A	N/A
SW_Cycle_Counter_Bin7	Software cycle counter bin 7, number of cycles in the 75–87.5% SoC bin	ac_fixed	0d78	0	16	1	0	0	65536	N/A	N/A	N/A
SW_Cycle_Counter_Bin8	Software cycle counter bin 8, number of cycles in the 87.5–100% SoC bin	ac_fixed	0d78	16	16	1	0	0	65536	N/A	N/A	N/A
SW_Profile_Integrity_Sts	Software sets this to 1 if a profile is loaded by software. If 0, a default trim profile is loaded.	bool	0d79	24	1	N/A	0	0	N/A	N/A	N/A	N/A
SW_UEFI_Profile_Load_Sts	UEFI sets this to 1 if UEFI loaded a profile	bool	0d79	25	1	N/A	0	0	N/A	N/A	N/A	N/A
SW_UEFI_FG_Restart_Sts	UEFI sets this to 1 if a fuel gauge restart was performed	bool	0d79	26	1	N/A	0	0	N/A	N/A	N/A	N/A
SW_HLOS_Profile_Load_Sts	HLOS sets this to 1 if HLOS loaded a profile	bool	0d79	27	1	N/A	0	0	N/A	N/A	N/A	N/A
Battery_Temperature	Battery temperature reading in Kelvin	ac_fixed	0d81	0	11	0.25	N	0	511.75	N/A	N/A	K
IBATT_Old	Battery current previous cycle	ac_fixed	0d85	0	18	0.00012207	Y	0	N/A	15.999837	-16.0000811	A
VBATT_Old	Battery voltage previous cycle	ac_fixed	0d86	0	15	0.00024414	N	0	7.999768147	N/A	N/A	V
IBATT	Battery current	ac_fixed	0d87	0	18	0.00012207	Y	0	N/A	15.999837	-16.0000811	A
VBATT	Battery voltage	ac_fixed	0d88	0	15	0.00024414	N	0	7.999768147	N/A	N/A	V
IBATT_Low_Pass	Low pass battery current	ac_fixed	0d89	0	24	1.91E-06	Y	0	N/A	16.0000096	-16.0000134	A

Register name	Description	Type	Address (dec)	Bit offset	Bit width	LSB	Signed	Value offset	Max range (unsigned)	Pos. range (signed)	Neg. range (signed)	Unit
VBATT_Low_Pass	Low pass battery voltage	ac_fixed	0d90	0	24	1.91E-06	Y	0	N/A	16.0000096	-16.0000134	V
Battery_SoC	Battery SoC	ac_fixed	0d91	0	32	2.33E-10	N	0	1.00000153	N/A	N/A	%
Voltage_Mode_Correction	Voltage mode correction applied to CC SOC to obtain battery SoC	ac_fixed	0d92	0	32	9.31E-10	Y	0	N/A	2.00000091	-2.00000091	%
Cutoff_SoC	Cutoff SoC	ac_fixed	0d93	0	16	1.53E-05	N	0	0.999985458	N/A	N/A	%
Full_SoC	Full SoC	ac_fixed	0d93	16	16	1.53E-05	N	0	0.999985458	N/A	N/A	%
System_SoC	System SoC	ac_fixed	0d94	0	16	1.53E-05	N	0	0.999985458	N/A	N/A	%
Monotonic_SoC	Monotonic SoC	ac_fixed	0d94	16	16	1.53E-05	N	0	0.999985458	N/A	N/A	%
CC_SoC	Coulomb Counting SoC	ac_fixed	0d95	0	32	9.31E-10	Y	0	N/A	2.00000091	-2.00000091	%
CC_SoC_SW	Software dedicated coulomb counting SoC	ac_fixed	0d96	0	32	9.31E-10	Y	0	N/A	2.00000091	-2.00000091	%
VBATT_Predicted	Predicted battery voltage (related to battery SoC)	ac_fixed	0d97	0	15	0.00024414	N	0	7.999768147	N/A	N/A	V
OCV	Estimated OCV (related to battery SoC)	ac_fixed	0d97	16	15	0.00024414	N	0	7.999768147	N/A	N/A	V
ESR	Battery ESR (related to battery SoC)	ac_fixed	0d99	0	14	0.00024414	N	0	3.999762003	N/A	N/A	Ω
ESR_Voltage_Drop	Voltage drop due to battery resistance (related to battery SoC)	ac_fixed	0d100	0	17	0.00012207	Y	0	N/A	7.99985745	-8.00010159	V
Rslow	Battery Rslow (related to battery SoC)	ac_fixed	0d101	0	14	0.00024414	N	0	3.999762003	N/A	N/A	Ω
Rslow_Voltage_Drop	Voltage drop due to iBtRslw and vBtSlw (related to battery SoC)	ac_fixed	0d102	0	17	0.00012207	Y	0	N/A	7.99985745	-8.00010159	V
IBATT_Low_Pass	Low pass version of battery current for Rslow (related to battery SoC)	ac_fixed	0d103	0	24	1.91E-06	Y	0	N/A	16.0000096	-16.0000134	N/A
Battery_Capacity_Actual	Battery capacity actual	ac_fixed	0d117	0	16	1	N	0	65535	N/A	N/A	mAh
ESR_Timer_Count	ESR timer	ac_fixed	0d118	0	16	1	N	0	65535	N/A	N/A	N/A
ESR_Filter_Log_High_Result	History of the last four ESR extractions hitting the upper increase boundary	ac_int	0d118	17	4	1	N	0	15	N/A	N/A	N/A
ESR_Filter_Log_Low_Result	History of the last four ESR extractions hitting the lower decrease boundary	ac_int	0d118	21	4	1	N	0	15	N/A	N/A	N/A
ESR_Extracted	ESR extracted	bool	0d118	25	1	N/A	N	0	Boolean	N/A	N/A	N/A
Battery_Charging_Status_Old	Battery current is charge current on the old reading	bool	0d118	28	1	N/A	N	0	Boolean	N/A	N/A	N/A
Battery_Charging_Status	Battery current is charge current on current reading	bool	0d118	29	1	N/A	N	0	Boolean	N/A	N/A	N/A

Register name	Description	Type	Address (dec)	Bit offset	Bit width	LSB	Signed	Value offset	Max range (unsigned)	Pos. range (signed)	Neg. range (signed)	Unit
Battery_SoC_Last_IRQ	Battery SoC saved when the last battery SoC interrupt was fired	ac_fixed	0d119	0	16	1.53E-05	N	0	0.999985458	N/A	N/A	%
Monotonic_SoC_Last_IRQ	Monotonic SoC saved when the last battery SoC interrupt was fired	ac_fixed	0d119	16	16	1.53E-05	N	0	0.999985458	N/A	N/A	%
Interrupts_Memory_Partition												
MSoC_Minimum_OTG_Status	Monotonic SoC is too low to allow OTG	bool	0d120	8	1	N/A	N	0	Boolean	N/A	N/A	N/A
MSoC_Automatic_Recharge_Status	Monotonic SoC is below the auto recharge threshold	bool	0d120	9	1	N/A	N	0	Boolean)	N/A	N/A	N/A
IBATT_Charge_Termination_Status	Battery current is less than the target termination current	bool	0d120	10	1	N/A	N	0	Boolean	N/A	N/A	N/A
IBATT_HPM_Status	Battery current is above HPM	bool	0d120	12	1	N/A	N	0	Boolean	N/A	N/A	N/A
VBATT_Automatic_Recharge_Status	Battery voltage is below the auto recharge threshold	bool	0d120	13	1	N/A	N	0	Boolean	N/A	N/A	N/A
VBATT_Float_Voltage_Sts	Battery voltage is above the float voltage threshold	bool	0d120	15	1	N/A	0	0	Boolean	N/A	N/A	N/A

4 Troubleshooting

If an issue is suspected to be fuel gauge related, QTI recommends creating a fuel gauge case via Salesforce. The case will be routed automatically to someone who can assist in debugging this issue.

Figure 4-1 shows example problem areas to choose when submitting a fuel gauge case to ensure that it goes to the appropriate team without delay.

Problem Area 1	Hardware
Problem Area 2	PMIC
Problem Area 3	Battery Interface / BMS / Fuel Gauge

Figure 4-1 Example problem area for a fuel gauge case in Salesforce

Provide the following information when submitting a fuel gauge case:

- Detailed description and steps used to cause or reproduce the issue.
  - For example:
    - Did the issue occur during charging or discharging?
    - What was the battery voltage?
    - Provide the software build information.
    - Failure rate: How many parts have you tested, and how many failed?
- SRAM logs:
  - SRAM logs are the most powerful tool for debugging fuel gauge issues. They will be requested for every issue. Customers must generate them and submit them with the case. These logs should be provided in the proper file format with no SPMI fuel gauge peripheral or charger peripheral logs. Failure to do so might require QTI to request these logs to be recaptured.
  - These are not the usual Linux Android logs. For more information about the process for collecting these logs, refer to *MSM8998 Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

5 Frequently asked questions

5.1 Algorithm and measurement

5.1.1 How do I test the fuel gauge performance?

QTI does not support or provide any tools to customers for SoC accuracy testing. QTI recommends that customers develop their own knowledge and test methods for SoC accuracy testing. QTI tests the fuel gauge extensively and has SoC accuracy data to provide over many use cases upon customer request.

If there are any concerns with fuel gauge performance, QTI recommends that customers create a case, and QTI can review the SRAM logs during normal charge or discharge operation, as well as the layout of the fuel gauge to evaluate its general performance.

5.1.2 Can the fuel gauge cycle of 1.47 s be changed?

The 1.47 s fuel gauge update cycle is fixed in the hardware and cannot be modified. Most concerns with this measurement interval arise because the fuel gauge misses large transients that do not happen during the 160 ms conversion window. QTI's internal testing has shown great performance over many use cases including dynamic loading. This occurs for the following reasons:

- Although the coulomb-count only happens every 1.47 s during an averaging window of 160 ms. The coulomb count averages out over time. The longer the coulomb count runs, the smaller the impact missed transients have on more SoC accuracy. Short bursts of current are averaged out and a good general average is created.
- The fuel gauge does both coulomb counting and voltage mode correction. The voltage mode correction helps to correct any deviations that may be caused due to these quick load transients. The voltage mode also corrects the accumulating error, adjusting for this offset error.

5.1.3 Can the ESR measurements and/or pulses be disabled or their frequency changed?

If the fuel gauge is used to report SoC, the ESR pulses cannot be disabled.

If the fuel gauge is not used in a design to report SoC, the ESR pulses can be disabled, but it is strongly recommended that QTI be contacted before making the decision to not use QTI's fuel gauge to verify that there will not be any unforeseen issues.

NOTE: In many cases when customers decide to use a third-party fuel gauge, the decision is based on incorrect, vague, or misleading information. Contact QTI if there are any concerns with specific features or specifications of the fuel gauge to verify that the concerns are valid.

As for changing the measurement intervals, although it is possible to change the timing on the ESR measurements, QTI strongly recommends against doing so and QTI does not guarantee the fuel gauge SoC accuracy if the intervals have been changed, as this will differ from what has been validated internally.

Consider the following things before changing the ESR timing for discharge and charge:

- Discharge: During discharge, ESR pulses do not always happen. During typical use cases, load transients occur that are large enough for the fuel gauge to update the ESR. If there is a relatively constant load on the battery, the ESR pulses are used to measure the battery resistance. These pulses only last for 160 ms (and only happen every ~146 s), and do not consume much current with 150 mA as the default setting. If customers are concerned that this may impact the days of use (DoU) or other use-time metrics, it is strongly recommended that the calculations be done on the customer's end to verify the very minimal effect on the use time.
- Charge: During charging, the ESR measurements are different from discharge. An external load pulse cannot be used to measure the ESR as the charger is always regulating a constant current or constant voltage. Instead, the fuel gauge communicates to the charger a very quick current decrement based on values in Table 2-1. If there are any concerns that this decrement will affect the charge time, note that these decrements happen at a maximum of every ~28 s and only last for ~160 ms. If there are concerns that this may affect charge time, it is strongly recommended that calculations be done on the customer end to verify the very minimal charge time effect.

5.1.4 What are the 1 mA battery current pulses every 1.47 s?

A fuel gauge cycle is every 1.47 s. During that time, the fuel gauge measures voltage and current. These voltage and current measurements take ~160 ms, and cause the fuel gauge module (ADC) to consume roughly 1 mA of current. See Section 2.2.3.4 for an example plot of these current pulses.

5.1.5 The IBATT full termination current is inaccurate; what could be wrong?

The feedback loop internal to the fuel gauge responsible for matching the 100% point to the system termination current uses two settings: IBATT full and VBATT full. VBATT full must be set to 10 mV less than the programmed float voltage to compensate for the charger's FV inaccuracy or 100% SoC may not be reached. If poor accuracy is noticed, verify that these parameters are set correctly in the software. If these settings are incorrect, but accuracy is still poor, this could be a more serious board level issue, and it is recommended that a case be created via Salesforce to get help in debugging this further.

5.2 Battery thermistor selection and measurement

5.2.1 The battery temperature measurements are out of specification; what could be wrong?

5.2.1.1 Capacitance on BATT_THERM

This is a common issue that is attributed to added capacitance on the BATT_THERM net. Additional capacitance on this net is not recommended and may cause temperature measurement accuracy issues. QTI does not do internal validation with any capacitance on the BATT_THERM net. When checking for added capacitance, the battery pack's internal protection circuit is commonly overlooked, so be sure to check this as well.

If some level of battery capacitance is required on BATT_THERM, verify that the settings explained for the capacitance on BATT_THERM in Section 2.1.4.2.4 are correct.

5.2.1.2 Beta coefficients not programmed correctly

Another common issue is that the beta coefficients provided in Section 2.1.4.2.5 are not programmed into the part. This causes the fuel gauge temperature model to be shifted and read inaccurate temperature measurements. Verify that the correct coefficients are taken from the table and programmed in the software correctly.

5.2.1.3 Pull-up from BATT_THERM to BATT_THERM_BIAS not chosen correctly

The pull-up resistor from BATT_THERM_BIAS to BATT_THERM must match the thermistor's resistance value at 25°C. If these do not match, this affects the measurable temperature range and measurement accuracy.

5.2.2 How should I choose the thermistor value? Does QTI have any tips?

See Section 2.1.4.2.6 on thermistor selection and placement for more information.

5.3 Battery identification resistor selection and algorithm

5.3.1 The battery RID accuracy is out of specification; what could be wrong?

This is a common issue that is attributed to added capacitance on the BATT_ID net. Additional capacitance on this net is **not supported**, and causes RID accuracy measurement issues. When checking for added capacitance, the battery pack's internal protection circuit is commonly overlooked.

5.3.2 What is the supported BATT_ID range and what happens if a value outside of that range is chosen?

Refer to the corresponding PMIC device specification for the supported BATT_ID ranges. If a BATT_ID value is chosen outside of QTI's supported ranges, BMD may mistrigger, and RID measurements cannot be guaranteed.

5.4 Battery characterization

5.4.1 How do I submit batteries for characterization?

Create a battery characterization case type (not a wireless device support). Fill in all necessary information, and send the batteries **to the address in the case**. If you have any questions, use the case to follow-up with QTI.

5.5 Other commonly asked questions

5.5.1 Can a third-party fuel gauge be used and the internal fuel gauge be disabled?

It is technically possible to use a third-party fuel gauge, but QTI strongly recommends against it for the following reasons:

- Loss of necessary features
  - battery current limiting – This is the most important feature on the list, and is a necessary part of QTI's chipsets. This is important for the high and premium tier chipsets. They have the potential to consume large amounts of current in a short time, which, at low SoC or low temperature conditions, can cause system burn-outs. This feature allows the software to mitigate this heavy loading, preventing instability and crashes. This functionality is not easily replicated in the software, and QTI does not provide any support replicating this feature when using a third-party solution.
  - Flash mitigation – Software uses the real-time ESR measurements from the fuel gauge to intelligently prevent flash events at low battery voltages.
  - BATT_THERM and JEITA – The fuel gauge module contains the temperature measurement capability that is used for fuel gauge functions and by the charger for JEITA-compliant compensation by the charger. These functions are in the hardware and would need to be transferred to the software on the customer's end without QTI support.
  - BATT_ID and other RRADC channels – The fuel gauge also contains the modules that control BAT_ID and the rest of the RRADC detection. The fuel gauge module must be enabled if these measurements are required.
- Loss of support on this module
  - If the QTI fuel gauge is not used, QTI will not support replicating any of these functions in software with a third-party solution. This is not an easy task and will take engineering time, resources, and additional cost. It is strongly recommended that customers start a discussion with QTI to ensure that these improvements, whether target specifications or

requirements, are not already met, or if QTI can help to meet certain specifications or requirements before considering a 3rd party solution.

- QTI's fuel gauge is very competitive
  - QTI targets a typical 3% SoC accuracy for all internal testing and, in many instances, this accuracy is better. Contact QTI for SoC accuracy information.
  - QTI's fuel gauge hardware and software contain many useful and competitive features such as combined coulomb count and voltage mode fuel gauging, live ESR estimates, aging, cycle counting and so on. If there are any features in the fuel gauge that are not present, but are preferred, create a Salesforce case and pass this information to the QTI CE team. QTI will do its best to support customer requirements in future versions of the fuel gauge.