Qualcomm Technologies, Inc.

Understanding PMI8994, PMI8996, and PMI8952 Fuel Gauges

Application Note

80-VT310-123 Rev. C

December 11, 2015

For additional information or to submit technical questions go to: https://support.cdmatech.com/

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

Revision history

Revision	Date	Description
A	September 2014	Initial release
B	January 2015	▪ Global changes: ▫ Removed SMB1360 from title ▫ Removed SMB1360 references from document ▪ Section 2.1.3 Battery resistance measurement ▪ Temperature sensing and compensation: Changed section title ▪ Section 2.1.5 Resistance measurements: Updated section to discuss ESR and R_{slow} constants ▪ Section 2.1.6 Battery model: Changed R1 constant to R_{slow} ▪ Section 2.1.7 OCV to SoC mapping in hardware: Added new section to discuss OCV-SOC relationship ▪ Section 2.1.8.2 Battery state of charge: Updated cycle to 1.47 s ▪ Section 2.1.8.3 System state of charge: Updated definition of terms ▪ Section 2.1.8.4.1 Slope limiting: Updated the slope limiting example ▪ Section 2.1.10 Battery profile: Updated the battery profile summary ▪ Section 2.2.1 Battery identification: Updated entire section ▪ Section 2.2.2 Charger interaction: Updated charger variables ▪ Section 3.3.1 Scratch pad: Updated scratch pad registers
C	December 2015	▪ Section 1.2 General description: Made some updates to the general description ▪ Table 1-2 PMI8994 and PMI8996 pin descriptions: Added PMI8996 to the title ▪ Added Table 1-3 PMI8952 pin descriptions ▪ Chapter 2 Detailed module information: Made several updates to the entire chapter; read in its entirety ▪ Section 3.3.1 PMI8994 vs. PMI8952 and PMI8996 register differences: Added this section ▪ Added two chapters: ▫ Chapter 4 Troubleshooting ▫ Chapter 5 Frequently asked questions

Contents

Figures

Tables

1 Introduction

1.1 Features

- Hardware based fuel gauge – minimal software interaction required
- State of charge estimation done entirely by fuel gauge module
- A 16-bit dedicated current ADC
- A 15-bit dedicated voltage ADC
- Supports multiple battery profiles (up to four profiles)
- Battery identification
- Battery missing detection (BMD)
- USB_ID detection.
- Supports both external and internal high side sensing
  - Internal high side sensing is BATFET sensing for minimum low resistance routing
- JEITA detection for charger
- Automatic recharge based on SoC
- Configurable 100% and 0% SoC points
- Configurable monotonic slope limiting

1.2 General description

The fuel gauge offers a new hardware based algorithm that can accurately estimate the SoC by mixing both current and voltage monitoring based techniques. This ensures both excellent short-term linearity and long-term accuracy. Furthermore, full battery cycling or zero current load conditions are not required to maintain accuracy.

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 taking into account temperature and aging effects, and providing a dependable SoC throughout a battery's entire life.

The fuel gauge has the capability to measure the battery pack temperature using an external thermistor. Missing battery detection is also incorporated within the fuel gauge module and is used to accurately monitor battery insertion and removal situations while updating the SoC when a battery is reconnected.

A broad range of configuration registers is provided to fit the requirements of every application.

1.3 Acronyms

Table 1-1 Acronyms

Acronyms	Description
SoC	State of charge
EOC	End of charge
FG	Fuel gauge
BMD	Battery missing detection
ADC	Analog to digital convertor
OCV	Open circuit voltage
ESR	Equivalent series resistance
CV	Constant voltage
FCC	Full charge capacity
BCL	Battery current limit

1.4 Fuel gauge pin descriptions

Table 1-2 PMI8994 and PMI8996 pin descriptions

Pin	Description
BATT_MINUS	Battery minus terminal sense input; connect directly to the battery (-).
BATT_PLUS	Battery plus terminal sense input; connect directly to the battery (+).
CS_MINUS	Current sense resistor minus sense input. It connects to the low side of the current sense element.
CS_PLUS	Current sense resistor plus sense input. It connects to the high side of the current sense element.
VREG_FG1	Bypass capacitor for the internal fuel gauge LDO. It is only used by the fuel gauge and must not be used as a general LDO output.
VREG_FG2	Bypass capacitor for the internal fuel gauge LDO. It is only used by the fuel gauge and must not be used as a general LDO output.
GND_FG	Analog ground for fuel gauge. LDO bypass capacitors connect here.
BATT_ID	Battery ID input to ADC and MIPI BIF interface. It can be used for missing battery detection.
BATT_THERM	Battery temperature input to ADC for measuring pack temperature. It is used for charger safe operation and BMS/fuel gauge.
R_BIAS	Dedicated voltage source for BATT_THERM resistor network biasing.

Table 1-3 PMI8952 pin descriptions

Pin	Description
BATT_PLUS	Battery plus terminal sense input; connect directly
BATT_MINUS	Battery minus terminal sense input; connect directly
CS_MINUS	Current sense resistor—minus side (low side)
CS_PLUS	Current sense resistor—plus side (high side)

Pin	Description
VAA_CAP	LDO output number 1 that supplies fuel gauge circuits; connect bypass capacitor only—*do not load externally*
V_ARB	LDO output number 2 that supplies fuel gauge circuits; connect bypass capacitor only—*do not load externally*
GND_FG	Fuel gauge analog ground; connect to LDO load capacitors
BATT_ID	Battery ID input to ADC and MIPI BIF interface; also detects missing battery
BATT_THERM	Battery temperature input to ADC (measures pack temperature); used by fuel gauge and by charger to ensure safe operation
R_BIAS	Dedicated voltage source for battery-related resistor networks

2 Detailed module information

2.1 Fuel gauge hardware and additional functions

2.1.1 Fuel gauge current sensing

The fuel gauge operates precise coulomb counting by sensing current from and to the battery using current mirroring via the BATFET or by sensing the voltage across the external 10 mΩ sense resistor. Voltage across the sense resistor is read by the dedicated differential pins, CS_PLUS and CS_MINUS. Current is read positive when discharging the battery and is otherwise negative. The CS_PLUS and CS_MINUS pins are internally connected to a dedicated ADC. 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

Information about battery voltage is retrieved across the dedicated BATT_PLUS and BATT_MINUS 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 internal resistance of the battery is a key parameter in determining the unusable charge of the battery. Battery resistance vs. temperature is essential in obtaining an accurate SoC. The fuel gauge takes into account two resistance values as seen in the battery model in Figure 2-3. The total battery resistance can be broken down into two parts:

- ESR
- R1 (R_{slow})

The ESR value is measured while the fuel gauge module is running in real time. Therefore, this value is not mapped in an extensive way. The ESR of a battery is influenced by a battery's temperature and residual capacity. In order to guarantee an accurate SoC estimation, the algorithm relies on a real-time estimate of the actual ESR. The fuel gauge monitors the ESR variation by sampling valid synchronous readings of the battery voltage and current. The data collected is processed by the fuel gauge to achieve the best estimation of the actual ESR.

2.1.3.2 ESR measurements during discharge

In active mode

During normal operation, the fuel gauge will use the naturally occurring voltage and current transients on the battery to calculate the real-time battery resistance. If a naturally occurring transient does not happen within 94 s, the fuel gauge takes an ESR measurement by inducing a small current pulse. This is a small pulse (default is 60 mA) that is done very quickly (~160 ms), to have minimal impact on the battery life.

In sleep mode

While in sleep mode, the fuel gauge will not take ESR measurements, but instead will rely on wake up cycles to take measurements. ESR pulses are disabled in PBS RAM when entering sleep, and enabled when exiting.

2.1.3.3 ESR measurements during charge

During charging, the fuel gauge decrements the charge current momentarily to generate a transient for ESR measurement. The main reason for this is that the charger regulates a constant current. To create the necessary transient instead, the charge current is decremented for a short time, ~160 ms, by a default amount of 100 mA. This short decrement allows the fuel gauge to get the information it needs with a minimal effect on charge time. The charger decrements the charge current every ~47 s, for ~160 ms.

2.1.3.4 R_{slow}

R_{slow} is the second component of the total battery resistance. This is the resistance seen as a battery is left to settle. This resistance is measured during battery characterization and stored in the battery profile. The fuel gauge hardware uses R_{slow} and the temperature information to provide a more accurate total resistance from the real time ESR measurements.

2.1.4 Temperature sensing and compensation

2.1.4.1 Overview

The fuel gauge module is responsible for monitoring the battery's temperature through the dedicated BATT_THERM pin. The temperature measurement circuity is incorporated within the fuel gauge module, and follows the same update cycle of the fuel gauge.

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

- Improves accuracy in the SoC estimation with adjustments of the fuel gauge models based on temperature.

Accommodates the charger (SCHG) operation in order to charge the battery in safe conditions, as per the JEITA requirements. For more information on JEITA/charger interaction refer to

Section 2.2.2.1.

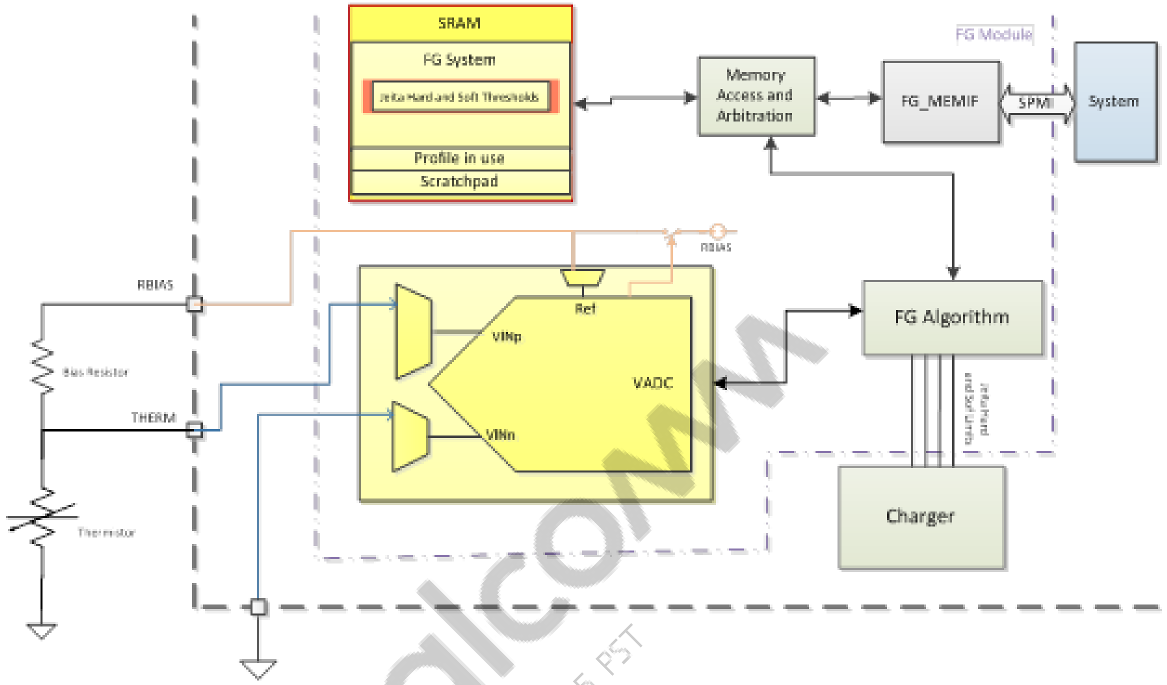


Figure 2-1 Architecture of temperature monitoring

2.1.4.2 Functionality

The fuel gauge uses a ratiometric conversion for temperature measurements. 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 biasing on the R_BIAS is also automatically handled in hardware when there is a conversion request.

2.1.4.3 Thermistor placement

Qualcomm Technologies, Inc. (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 will affect the thermistors resistance.

2.1.4.3.1 Capacitance on BATT_THERM

By default RBIAS is not always enabled, and is controlled by the fuel gauge module. During a period of temperature measurements, RBIAS will be enabled only when needed to conserve power. If capacitance is placed at the BAT_THERM node, then the BAT_THERM voltage may not have sufficient time to settle to its final value before the ADC conversion commences. This causes corrupt temperature measurements that are likely to be out of specification.

CAUTION: QTI **does not** recommend the placement of any capacitance on the BATT_THERM node for any of the fuel gauges mentioned in this document. If any capacitance must be placed, QTI recommends the following calculations and verification be done by the customer to verify proper functionality.

PMI8994

Figure 2-2 can be used to determine the maximum and suggested capacitance values for given thermistor values at room temperature. QTI does not support any capacitance on BATT_THERM for PMI8994 as this was never intended or verified to work in the hardware. Due to a large number of customer requests for some level of support on placing capacitance on the BATT_THERM net, QTI has simulated possible suggested and maximum values that may work. Keep in mind that QTI cannot guarantee the specifications will be met by following these recommendations as these values were never validated in hardware. Following these guidelines must be done at the customer's own risk.

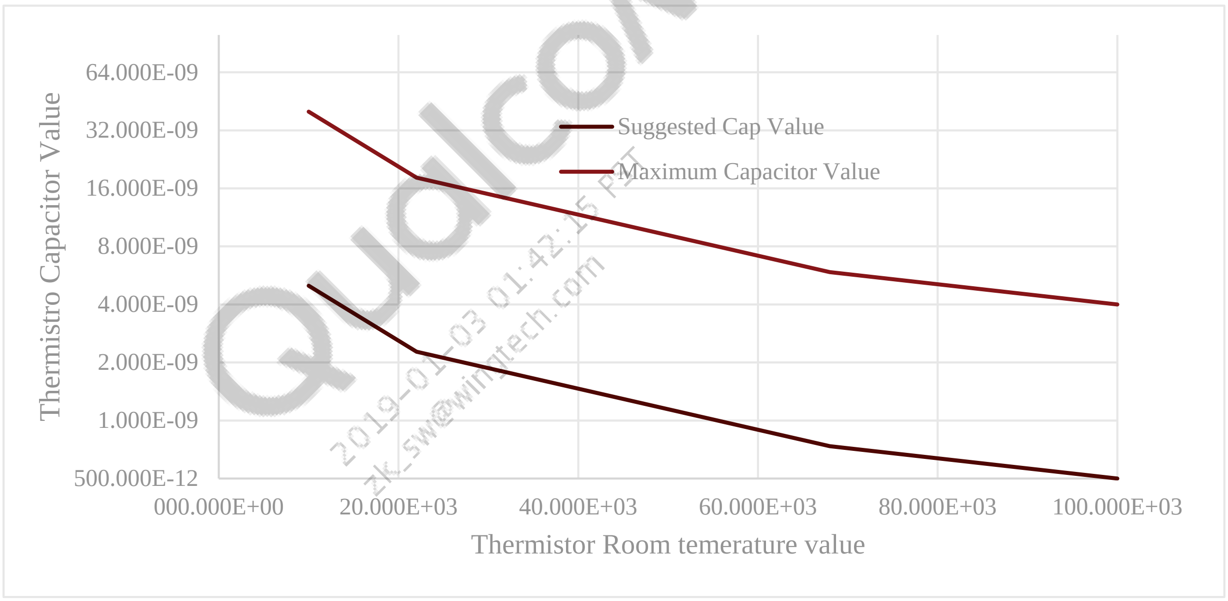


Figure 2-2 Simulated example for PMI8994

PMI8952/PMI8996

A recent addition to PMI8952/PMI8996 is a programmable delay between when R_BIAS is enabled and the ADC measurements begin. 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 that has be verified to work in hardware.

The fuel gauge supports delay times of 0 ms (disabled), 2.56 ms, 5.12 ms, 10.24 ms, 20.48 ms, 40.96 ms, 81.92 ms, or 163.84 ms.

To calculate the recommended setting, and absolute maximum setting, the following equations can be used:

$$C_{therm\ abs\ max} = \frac{163.84}{3} \times \frac{1}{R_{therm,nominal}} \times 10^{-3}$$

$$C_{therm\ rec\ max} = \frac{5.12}{3} \times \frac{1}{R_{therm,nominal}} \times 10^{-3}$$

Table 2-1 lists the results of these calculations for typical nominal thermistor resistances.

Table 2-1 Thermistor capacitance values

Nominal thermistor resistance $R_{therm\ nominal}$	Absolute maximum thermistor capacitance $C_{therm\ abs\ max}$	Recommended maximum thermistor capacitance, $C_{therm\ rec\ max}$ (Applicable only to PMI8950 1.0)
10 kΩ	4.7 µF	0.15 µF
33 kΩ	1.5 µF	0.047 µF
47 kΩ	1.0 µF	0.033 µF
68.1 kΩ	0.68 µF	0.022 µF
100 kΩ	0.47 µF	0.015 µF

Refer to *MSM899X Linux Android PMIC Fuel Gauge Software User Guide* (80-NM328-52) for more information on how to set this in software.

2.1.4.3.2 Beta table and value 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 better matches the modeling in hardware.

To convert from voltage measurements to temperature readings, the PMIC uses a linear approximation of the Steinhart-Hart equation. This approximation uses the coefficients listed in Table 2-2. To function properly, these coefficients must be programmed into the fuel gauge for the thermistor value selected. Failure to do this can result in inaccurate temperature measurement. Since 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 directly.

Table 2-2 Master beta coefficients

NTC	C1	C1-HEX	C2	C2-HEX	C3	C3-HEX
2800	-3.9757	87F3	0.059723	53A5	-0.00646	3E9D
2810	-3.9631	87ED	0.059267	5396	-0.00644	3E99
2820	-3.96	87EB	0.058523	537D	-0.00625	3E+67
2830	-3.9567	8.70E+10	0.057801	5366	-0.00607	3E+36
2840	-3.9429	8.70E+03	0.057426	5359	-0.00604	3E2F
2850	-3.9292	87DB	0.057055	534D	-0.00602	3E+29
2860	-3.9156	87D4	0.05669	5341	-0.00599	3E+22
2870	-3.902	87CD	0.0563	5334	-0.00602	3E2A
2880	-3.8886	87C6	0.055938	5328	-0.006	3E+24
2890	-3.8991	87CC	0.054824	5304	-0.00562	3DC1
2900	-3.8669	87BB	0.055074	530C	-0.00588	30000
2910	-3.8631	87B9	0.054431	52F7	-0.00571	3DD9
2920	-3.8515	87B3	0.054046	52EA	-0.00567	3DCD

NTC	C1	C1-HEX	C2	C2-HEX	C3	C3-HEX
2930	-3.8457	87B1	0.053484	52D8	-0.00554	3DAB
2940	-3.8327	87AA	0.05311	52CC	-0.00559	3DB8
2950	-3.8197	87A3	0.052784	52C1	-0.00556	3DB2
2960	-3.8067	879D	0.052462	52B7	-0.00554	3DAC
2970	-3.7939	8796	0.052145	52AC	-0.00552	3DA7
2980	-3.7811	878F	0.051832	52A2	-0.0055	3DA2
2990	-3.7685	8789	0.051519	5298	-0.00548	3D9C
3000	-3.76	8785	0.051091	528A	-0.0054	3D87
3010	-3.7674	8788	0.050117	526A	-0.00518	3D4D
3020	-3.7548	8782	0.049823	5260	-0.00516	3D48
3030	-3.7422	877C	0.049534	5257	-0.00514	3D44
3040	-3.7298	8775	0.049248	524D	-0.00512	3D3F
3050	-3.7174	876F	0.048967	5244	-0.00511	3D3A
3060	-3.705	8768	0.048689	523B	-0.00509	3D36
3070	-3.6928	8762	0.048416	5232	-0.00507	3D31
3080	-3.706	8769	0.047302	520D	-0.0048	3CEA
3090	-3.6938	8763	0.047037	5205	-0.00479	3CE6
3100	-3.6816	875C	0.046777	51FC	-0.00477	3CE2
3110	-3.6694	8756	0.046521	51F4	-0.00476	3CDE
3120	-3.6574	8750	0.046268	51EC	-0.00474	3CDA
3130	-3.6454	874A	0.046019	5.10E+04	-0.00473	3CD6
3140	-3.6335	8744	0.045774	51DB	-0.00471	3CD2
3150	-3.6217	873E	0.045533	51D4	-0.0047	3CCF
3160	-3.6364	8745	0.044408	51AF	-0.00444	3C8B
3170	-3.6246	873F	0.044174	51A7	-0.00442	3C87
3180	-3.6128	8739	0.043945	519F	-0.00441	3C84
3190	-3.601	8733	0.043719	5198	-0.0044	3C80
3200	-3.5894	872D	0.043497	5191	-0.00438	3C7D
3210	-3.5778	8727	0.043278	518A	-0.00437	3C7A
3220	-3.5663	8721	0.043062	5183	-0.00436	3C77
3230	-3.5552	871C	0.042709	5177	-0.00446	3C90
3240	-3.5492	8719	0.042337	516B	-0.00438	3C7A
3250	-3.5509	871A	0.041752	5158	-0.0042	3C4C
3260	-3.5411	8715	0.041504	5150	-0.00417	3C44
3270	-3.5313	8710	0.04126	5148	-0.00414	3C3C
3280	-3.5215	870B	0.041021	5140	-0.00411	3C34
3290	-3.5119	8706	0.040786	5138	-0.00408	3C2C
3300	-3.5023	8701	0.040554	5130	-0.00405	3C24
3310	-3.492	86FB	0.04018	5124	-0.00415	3C3F
3320	-3.481	86F6	0.039988	511E	-0.00414	3C3C
3330	-3.4701	86F0	0.039799	5118	-0.00413	3C39
3340	-3.4592	86EB	0.039613	5112	-0.00412	3C36

NTC	C1	C1-HEX	C2	C2-HEX	C3	C3-HEX
3350	-3.455	8.60E+09	0.039247	5106	-0.00402	3C1E
3360	-3.4458	8.60E+05	0.039029	50FE	-0.00399	3C17
3370	-3.4366	86DF	0.038812	50F7	-0.00397	3C0F
3380	-3.4274	86DA	0.038601	50F0	-0.00394	3C08
3390	-3.433	86DD	0.037798	50D6	-0.00386	37000000
3400	-3.4224	86D8	0.03763	50D1	-0.00385	370
3410	-3.4117	86D2	0.037465	50CB	-0.00384	37DC
3420	-3.4012	86CD	0.037303	50C6	-0.00383	37D8
3430	-3.3906	86C8	0.037144	50C1	-0.00382	37D3
3440	-3.3802	86C2	0.036987	50BB	-0.00381	37CF
3450	-3.3698	86BD	0.036833	50B6	-0.00381	37CB
3460	-3.3595	86B8	0.036681	50B1	-0.0038	37C7
3470	-3.3773	86C1	0.035566	508D	-0.00359	375B
3480	-3.3673	86BC	0.03541	5088	-0.00358	3754
3490	-3.3569	86B6	0.035266	5083	-0.00357	3750
3500	-3.3467	86B1	0.035125	507E	-0.00357	374D
3510	-3.3365	86AC	0.034987	507A	-0.00356	3749
3520	-3.3263	86A7	0.034851	5076	-0.00355	3745
3530	-3.3162	86A1	0.034718	5071	-0.00354	3742
3540	-3.3062	869C	0.034586	506D	-0.00354	373F
3550	-3.2962	8697	0.034457	5069	-0.00353	373B
3560	-3.2924	8695	0.033919	5057	-0.00359	375B
3570	-3.2841	8691	0.033745	5051	-0.00357	374D
3580	-3.2764	868D	0.033562	504B	-0.00353	373D
3590	-3.2756	868D	0.033206	5040	-0.00343	3704
3600	-3.2741	868C	0.032872	5035	-0.00332	36CF
3610	-3.266	8688	0.032715	5030	-0.0033	36C2
3620	-3.2562	8683	0.032603	502C	-0.0033	36BF
3630	-3.2465	867E	0.032492	5028	-0.00329	36BD
3640	-3.2369	8679	0.032096	501B	-0.00343	3705
3650	-3.2273	8674	0.031981	5017	-0.00342	3702
3660	-3.2177	866F	0.031869	5014	-0.00342	36FE
3670	-3.2082	866A	0.031758	5010	-0.00341	36FB
3680	-3.1988	8665	0.031649	500D	-0.0034	36F8
3690	-3.1894	8660	0.031542	5009	-0.0034	36F5
3700	-3.181	865C	0.031415	5005	-0.00338	36ED
3710	-3.1733	8658	0.031272	5000	-0.00336	360
3720	-3.1681	8656	0.031074	4BF4	-0.00331	36C7
3730	-3.1812	865C	0.030111	4BB5	-0.00321	3690
3740	-3.1719	8658	0.030014	4BAF	-0.0032	368E
3750	-3.1627	8653	0.02992	4BA8	-0.0032	368B
3760	-3.1534	864E	0.029827	4BA2	-0.00319	3689

NTC	C1	C1-HEX	C2	C2-HEX	C3	C3-HEX
3770	-3.1443	8649	0.029737	4B9C	-0.00319	3686
3780	-3.1352	8645	0.029648	4B96	-0.00318	3684
3790	-3.1261	8640	0.02956	4B91	-0.00318	3682
3800	-3.1171	863B	0.029474	4B8B	-0.00317	3680
3810	-3.1082	8637	0.02939	4B86	-0.00317	367D
3820	-3.1291	8642	0.028239	4B3A	-0.003	3624
3830	-3.1201	863D	0.028159	4B35	-0.00299	3622
3840	-3.1111	8638	0.028081	4B30	-0.00299	3620
3850	-3.1022	8634	0.028005	4B2B	-0.00299	361E
3860	-3.0933	862F	0.027931	4B26	-0.00298	361C
3870	-3.0844	862B	0.027858	4B21	-0.00298	361A
3880	-3.0756	8626	0.027787	4B1D	-0.00298	3618
3890	-3.0669	8622	0.027717	4B18	-0.00297	3617
3900	-3.0582	861D	0.027648	4B13	-0.00297	3615
3910	-3.0507	8619	0.027168	4AF4	-0.00311	365D
3920	-3.0438	8616	0.027054	4AED	-0.00308	3651
3930	-3.037	8612	0.026943	4AE5	-0.00306	3645
3940	-3.0334	8611	0.026759	4AD9	-0.00301	3629
3950	-3.0316	8610	0.026538	4ACB	-0.00294	3603
3960	-3.0265	860D	0.026399	4AC2	-0.0029	35EF
3970	-3.0257	860D	0.026165	4AB2	-0.00282	35C5
3980	-3.0197	860A	0.02605	4AAB	-0.00279	35B5
3990	-3.0114	8605	0.025993	4AA7	-0.00279	35B4
4000	-3.003	8601	0.02594	4AA3	-0.00278	35B3
4010	-2.9946	85FD	0.025454	4A84	-0.00294	3603
4020	-2.9863	85F8	0.025397	4A80	-0.00293	3601
4030	-2.978	85F4	0.025341	4A7C	-0.00293	3600
4040	-2.9698	85F0	0.025286	4A79	-0.00293	35FE
4050	-2.9616	85EC	0.025232	4A75	-0.00292	35FC
4060	-2.9535	8.50E+09	0.025179	4A72	-0.00292	35FB
4070	-2.9454	8.50E+05	0.025128	4A6E	-0.00292	35F9
4080	-2.9387	8.50E+01	0.025047	4A69	-0.0029	35F0
4090	-2.9324	85DD	0.02496	4A63	-0.00288	35000000
4100	-2.9261	85DA	0.024876	4A5E	-0.00286	35DB
4110	-2.9433	8.50E+03	0.023812	4A18	-0.00277	35AB
4120	-2.9353	85DE	0.023768	4A15	-0.00276	35A9
4130	-2.9272	85DA	0.023726	4A12	-0.00276	35A8
4140	-2.9192	85D6	0.023684	4A10	-0.00276	35A6
4150	-2.9113	85D2	0.023644	4A0D	-0.00276	35A5
4160	-2.9034	85CE	0.023605	4A0A	-0.00276	35A4
4170	-2.8955	85CA	0.023566	4A08	-0.00275	35A3
4180	-2.8877	85C6	0.023529	4A06	-0.00275	35A2

NTC	C1	C1-HEX	C2	C2-HEX	C3	C3-HEX
4190	-2.88	85C2	0.023493	4A03	-0.00275	35A1
4200	-2.8722	85BE	0.023457	4A01	-0.00275	35A0
4210	-2.8645	85BA	0.023422	49FE	-0.00275	359F
4220	-2.8762	85C0	0.022467	49C0	-0.00272	358F
4230	-2.8703	85BD	0.022395	49BB	-0.0027	3585
4240	-2.8644	85BA	0.022325	49B7	-0.00268	357B
4250	-2.8586	85B7	0.022256	49B2	-0.00266	3571
4260	-2.8567	85B6	0.022107	49A8	-0.0026	3553
4270	-2.8472	85B1	0.02212	49A9	-0.00262	355C
4280	-2.8414	85AE	0.022056	49A5	-0.0026	3552
4290	-2.8339	85AA	0.022031	49A3	-0.0026	3551
4300	-2.8264	85A7	0.022006	49A2	-0.0026	3550
4310	-2.8189	85A3	0.021983	49A0	-0.00259	3550
4320	-2.8184	85A3	0.021263	4971	-0.00269	3582
4330	-2.8128	85A0	0.0212	496D	-0.00267	3578
4340	-2.8072	859D	0.021139	4969	-0.00265	356E
4350	-2.8016	859A	0.021079	4965	-0.00263	3564
4360	-2.7961	8597	0.021021	4961	-0.00261	355A
4370	-2.7906	8594	0.020965	495D	-0.0026	3551
4380	-2.7851	8591	0.02091	495A	-0.00258	3547
4390	-2.7796	858F	0.020856	4956	-0.00256	353E
4400	-2.7741	858C	0.020804	4953	-0.00254	3534

NOTE: The bias resistor *must be* chosen to have the same value as the thermistor at room temperature.

2.1.5 Battery identification

The fuel gauge module contains a dedicated BATT_ID pin for battery identification. This option can be used with any battery pack that contains a battery identification resistor that is within the supported range.

Some features of battery identification are:

- The battery identification is done automatically in hardware by the fuel gauge module.
- BATT_ID can be used for battery missing detection (BMD).

Battery ID detection

Battery ID detection uses a dedicated channel of the VADC within the fuel gauge. Detection is done by measuring the voltage across the battery ID resistor while enabling one of three different current sinks. This detection is done on a battery insertion, and then subsequently whenever software forces a fuel gauge restart. Once the the detection is resolved, an interrupt is fired to let the system know the result. See Table 2-3 for the interrupts available.

The detection is done in the following steps, and is automatically handled inside the fuel gauge.

1. BSI is disabled, and 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. A second conversion is performed to verify stability.
5. Conversion values are checked.
6. The value of the battery ID is detected and and the corresponding interrupt is triggered.

Table 2-3 Interrupt peripherals

Interrupt	Description
BATT_IDENTIFIED_RT_STS FG_BATT	Interrupt to notify that the battery identification is completed.
BATT_ID_REQ_RT_STS FG_BATT	Interrupt to notify that the system has to perform software identification because a smart battery is detected.
BATT_UNKNOWN_RT_STS FG_BATT	Interrupt to notify that the battery is not recognized.

These steps are used to detect a resistor range of 1–450 kΩ.

2.1.6 Battery missing detection

Battery missing detection (formally known as battery removal detection) by default is done via the dedicated BATT_ID pin. If the BATT_ID pin is not used, then the BATT_THERM pin can be used instead. When triggered, the hardware raises a flag for the software to handle. The fuel gauge module does not function while the battery is removed, or is believed to be removed. The fuel gauge waits until it detects that the battery is present again before restarting its estimate and taking a new first SoC estimate.

2.1.7 USB_ID detection

2.1.7.1 PMI8994 and PMI8996

The fuel gauge module is responsible for detecting the resistance on USB ID. When a qualified USB input resistance is applied, the USB connector is between USB_ID and ground. The PMI performs an ADC conversion of the voltage on USB_ID, which is ratio-metric to the voltage on RBIAS. In other words, the resulting ADC output is 2^n × (Vusbid/Vrbias), where n = 12 bits. There is also an analog comparator on the USB_ID input. If no external resistor is present, the pin pulls above the trip point to indicate a missing USB_ID resistor.

The conversion of the USB ID resistor value is done via a dedicated ADC channel. The conversion is scheduled and handled automatically inside the fuel gauge, and is also performed when the fuel gauge algorithm is paused while waiting for the battery.

A double conversion is performed to guarantee the stability and two results should be within a certain range. If this is not the case, the conversion result is marked as unstable and the update is postponed to the next cycle.

The threshold usb_id_recheck_range dictates the allowed change compared to the previous value. See Section 3.3.2 for register information and encoding.

The fuel gauge module detects the value and reports it to the charger module, using internal signals; the detection of any other values is left to the software. The software can read through the converted USB ID value from the charger module registers:

- 0x130F - SMBCHG_USB_CHGPTH_USBID_VALID_ID_7_0
- 0x130E - SMBCHG_USB_CHGPTH_USBID_VALID_ID_11_8

The following equation is used for USB_ID register encoding. The value of the LSB is not fixed in terms of resistance; it depends on the resistance value.

USBID_nominal_value ± USBID_tolerance = 2^12 × (USBID_resistor ± USBID_tolerance)/(24.9 kΩ + (USBID_resistor × USBID_tolerance))

For example, with a 36.5 k USBID_resistor, the nominal code is 0x982. That is calculated using the USB_ID information above.

If USBID_Tolerance = 0x18, then 0x982 ± 0x18 is min = 0x96A and max = 0x99A.

This gives values of 35.59 k minimum and 37.37 k maximum obtained by solving the following equations:

Minimum usbid = USBID_nominal_value - USBID_tolerance =

2^{12} × (minimum usbid res)/(24.9 kΩ + (minimum usbid))

Maximum usbid = USBID_nominal_value + USBID_tolerance =

2^{12} × (maximum usbid res)/(24.9 kΩ + (maximum usbid))

2.1.7.2 PMI8952 and PMI8996

On PMI8952 and PMI8996, reading USB_ID via the fuel gauge is no longer supported. Instead, this has become comparator-based and integrated into the charger module; so this will not be covered in this document.

NOTE: On PMI8996, due to a hardware bug, PMI8996 uses the fuel gauge method of USB_ID detection to support detection of USB_ID = FLOAT, or USB_ID = GND. Refer to *PMI8994/PMI8996 Device Revision Guide* (80-NJ118-4) for more information.

2.1.8 Charger interaction

2.1.8.1 Thermal monitoring (JEITA)

The fuel gauge and charger modules are provided with a soft and hard thermal monitoring designed to be compliant with the latest JIS8714 and JEITA standard safety requirements. The battery’s temperature information is used to modulate the battery charging voltage and current when temperature is in between programmable ranges (see Figure 2-11).

The fuel gauge contains four settable thresholds: the JEITA_hard_hot, JEITA_hard_cold, JEITA_soft_hot, and JEITA_soft_cold thresholds. Any time the battery’s temperature is inside the range between the JEITA_soft_hot, and JEITA_soft_cold, the battery is to be charged with default charging current ICHG and floating voltage VFLOAT. If, at any time, the battery’s temperature exceeds either the JEITA_soft_hot or JEITA_soft_cold (but not the hard limits

JEITA_hard_hot and JEITA_hard_cold), the respective charging current and floating voltage are modulated to ICHG_SOFT and VFLOAT_SOFT. Both ICHG_SOFTand VFLOAT_SOFT are user-programmable.

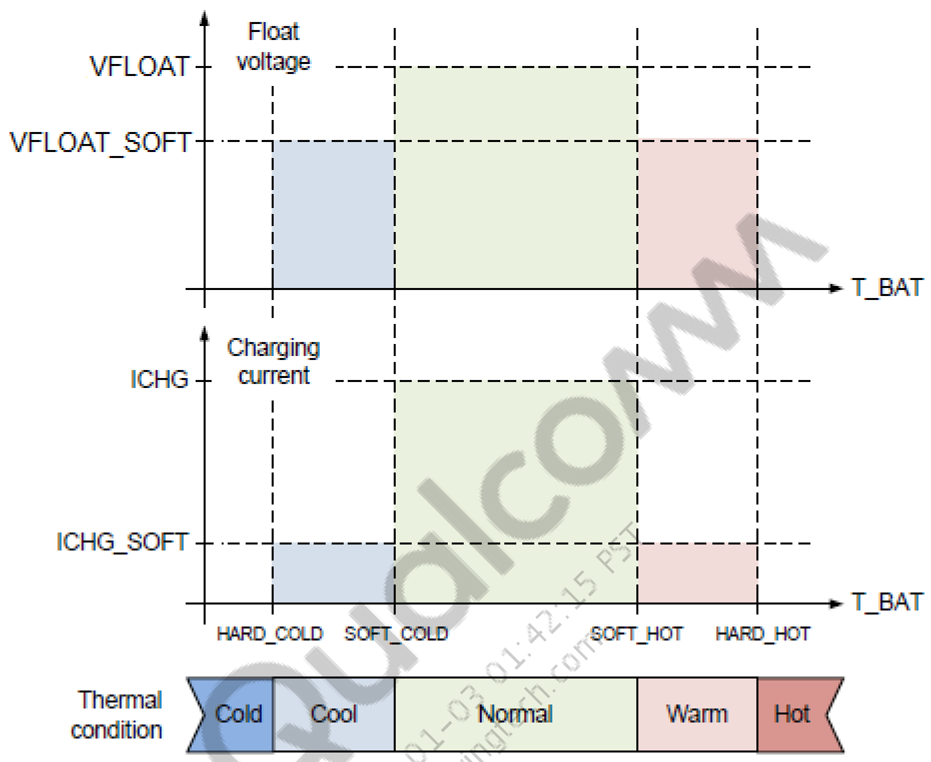


Figure 2-3 JEITA

2.1.9 Battery current limiting (BCL)

The BCL feature is responsible for providing battery current and battery voltage monitoring to the whole system. The update rate of these reading depends on the selected BCL mode.

Three modes are possible:

- LPM: If the battery current is below BCL_LM_Threshold
- MPM: If the battery current is between BCL_LM_Threshold and BCL_MH_Threshold
- HPM: If the battery current is above BCL_MH_Threshold

Table 2-4 lists the timing and behavior of the BCL given the different modes.

Table 2-4 Timing behavior of BCL in different modes

BCL mode	Condition	VBATT and IBATT refresh rate
LPM	Battery current < BCL_LM_Threshold	Every 1.47 s for 160 ms
MPM	BCL_LM_Threshold < battery current < BCL_MH_Threshold	Every 100 ms
HPM	Battery current > BCL_MH_Threshold	Every 640 µs

The fuel gauge inhibits the BCL conversion when the voltage ADCconverts other channels. Therefore, a small period of missing conversion could be expected.

The fuel gauge inhibits the BCL conversion when in reset. Therefore, even if the update of the BCL is enabled, no conversion would actually be performed.

BCL is enabled by default and will occur during boot, except on shutdown/resume during boot. The first BCL cycle is forced to be HPM. This causes BCL to initiate V and I ADC conversions immediately. Status bit 7 of 0x4253 FG_ADC_USR_BCL_VALUES indicates when the BCL values are valid and available.

2.1.9.1 BCL interrupts and status

The BCL registers are available for reading on the SPMI peripherals.

Battery voltage and current interrupts are provided by these registers:

- VBAT_LT_THR_INT: notify battery voltage < VBAT_INT
- IBAT_GT_THR_INT: notify battery current > IBAT_INT

The register bits associated with the BCL values, initial latency reduction and ready flag are located in Table 2-5.

Table 2-5 Register bits associated with BCL values initial latency reduction and ready flag

Peripheral.Register.Bit/Configuration bit	Access type
FG_ADC_USR.BCL_values.Rdy	R
FG_ADC_USR.BCL_mode.Sts	R
FG_ADC_USR.Vbat.Sts	R
FG_ADC_USR.Ibat.Sts	R
FG_ADC_USR.Vbat_cp.Sts	R
FG_ADC_USR.Ibat_cp.Sts	R

When the software provides the BCL conversion values, it sets the following registers:

- FG_ADC_USR.ACCESS_BAT_REQ.CMD = 1 if the software runs on the application processor
- FG_ADC_MDM.ACCESS_BAT_REQ.CMD = 1 if the software runs on the modem

After the read-back of the values, the software clears the bits.

A status bit (FG_ADC.USR_BCL_VALUES.RDY) is provided to notify that valid values are available in the BCL peripheral.

FG_ADC.BCL_VALUES.RDY is cleared on the following conditions:

- Shutdown
- Battery missing

Refresh the registers paused for read-back purposes: in case of a missed conversion value due to a pause in the registers updates (FG_ADC.ACCESS_BAT_REQ.CMD asserted and access granted), the ready bit will also be cleared.

2.2 Fuel gauge algorithm

2.2.1 SoC

2.2.1.1 Overview

The fuel gauge breaks the SoC down into four different layers. The first is the CC_SoC layer. This is the coulomb-counted SoC represented as a percentage of the typical battery capacity found during characterization. The second is battery SoC, which 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. The third is the System_SoC, which is a filtered version of the battery SoC, and is responsible for the end point matching of the cutoff voltage and 0% SoC point and the termination current and the 100% SoC point. The fourth and last is the monotonic SoC, which is the final SoC that is displayed to the end user. The monotonic SoC is the final SoC layer and guarantees monotonicity during discharge.

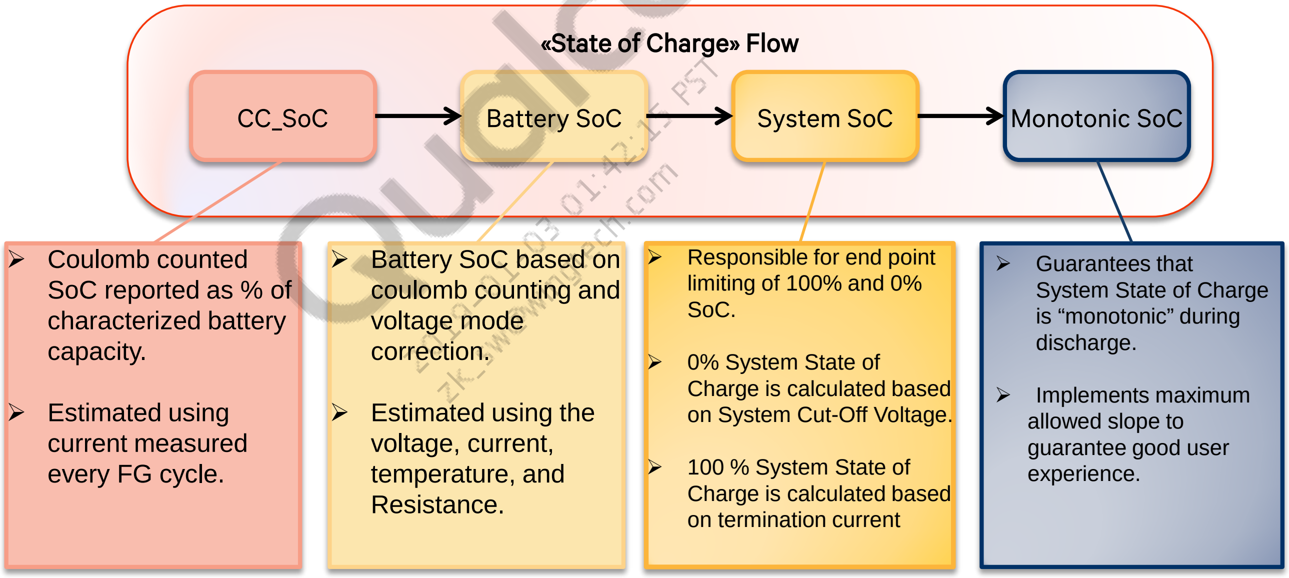

«State of Charge» Flow
CC_SoC
Battery SoC
System SoC
Monotonic SoC
➢ Coulomb counted SoC reported as % of characterized battery capacity.
➢ Estimated using current measured every FG cycle.
➢ Battery SoC based on coulomb counting and voltage mode correction.
➢ Estimated using the voltage, current, temperature, and Resistance.
➢ Responsible for end point limiting of 100% and 0% SoC.
➢ 0% System State of Charge is calculated based on System Cut-Off Voltage.
➢ 100 % System State of Charge is calculated based on termination current
➢ Guarantees that System State of Charge is "monotonic" during discharge.
➢ Implements maximum allowed slope to guarantee good user experience.

Figure 2-4 SoC flow diagram

2.2.1.2 CC_Soc

This is the first layer of the fuel gauge's SoC algorithm, and is responsible for reporting a coulomb-counted value **during constant current charging**. This SoC is reported as a percent of the battery capacity (0x578, 2) found during characterization. It is important to note that CC_SoC (0x570) should not be used by customers for coulomb counting. During CV charging, this value is updated to a new value that no longer is representative of a pure coulomb count. Customers can instead use the newer SW_CC_SOC (only available on PMI8996/PMI8952), which was designed for the purpose of reading a strict coulomb count. This register works similarly to CC_SoC in that it is a percentage of the capacity, but does not get changed or reset during CV charging.

2.2.1.3 Battery state of charge

The battery SoC is generated using the fuel gauge algorithm that measures voltage, current, and real-time ESR measurements. The battery SoC is estimated using the specific battery data collected during the characterization process and the advanced battery model.

In short, the algorithm predicts SoC based on the coulomb-counted value, and uses the voltage mode correction as a feedback loop. This feedback loop uses the battery profile and battery model to back calculate the OCV and help correct any error that may appear on CC_SoC due to aging, etc. This creates more robust and accurate SoC multiple use cases. Figure 2-6 shows how the mixing is performed.

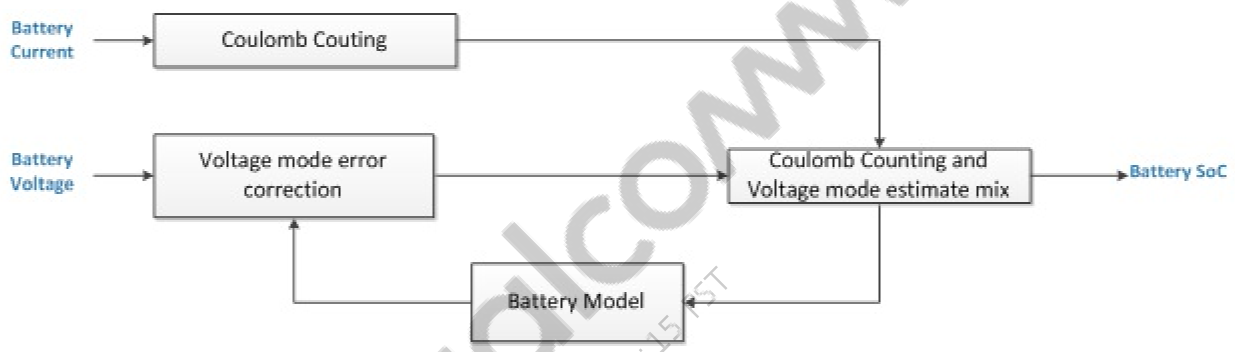

Battery Current
Coulomb Couting
Battery Voltage
Voltage mode error correction
Coulomb Counting and Voltage mode estimate mix
Battery SoC
Battery Model

Figure 2-5 Battery SoC algorithm diagram

Battery model

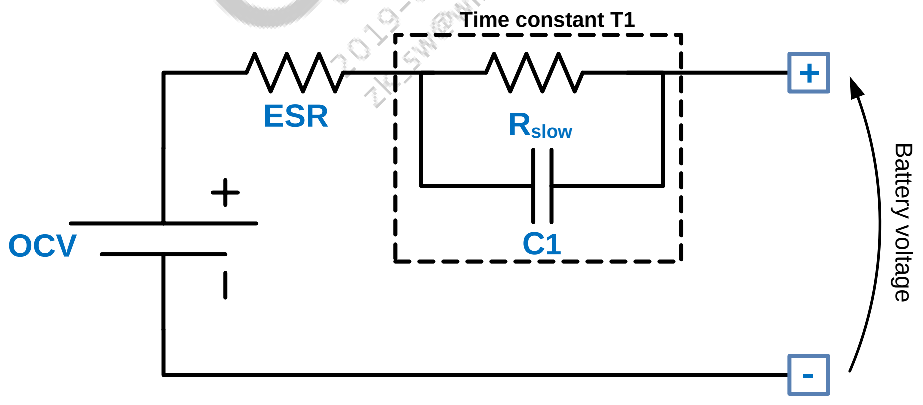


Figure 2-6 Battery model

The fuel gauge algorithm uses the battery's electrical model reported in Figure 2-3. This model replicates the electrical behavior of a real battery and it includes the following parameters:

- OCV: This is the voltage that the battery would express at the output connectors when no current is applied and when the battery is unused or idele for a long time (i.e., model capacitor C1 is totally discharged). OCV varies with battery temperature, SoC, and the battery current state (charged or discharged).

- ESR: This value of resistance affects the instantaneous power the battery can deliver to the load. ESR varies largely with battery temperature and is a function of the battery current state (charged or discharged) and battery age.
- R_{slow}-C1 time constant: For Li-ion battery cells this time constant is usually in the range of minutes and accounts for variation in the battery's SoC upon previous battery use. The same battery, when unconnected, could have the same SoC but show a different voltage at the connectors, based on the use in the previous few minutes. Modeling the R_{slow}-C1 helps account for the hysteretic behavior and improves the estimation of the battery's SoC.

The OCV, ESR, and R_{slow}-C1 values represent the parameters used by the OCV prediction algorithm. The battery model is comprised of OCV, ESR, and R_{slow}-C1. The fuel gauge operates using coulomb counting, and at the same time, continuously monitors the voltage at the battery connectors. This value is then compared to the estimated battery voltage from the electrical model in Figure 2-3 and to the same coulomb count. The results of this comparison are used to accommodate for possible variations of the battery capacity due to aging and temperature.

2.2.1.4 System SoC

The system SoC is the intermediate layer between the battery and the monotonic SoCs, and allows for configuration of full SoC and cutoff SoC. Here are some important terms and their explanations to better understand the system SoC's functionality.

NOTE: Some of these parameters must be configured according to the customer's desired specifications or the fuel gauge will not behave as expected.

Full SoC is a correction based on charge float voltage and termination current, and is applied to the battery SoC in order to estimate the system SoC. This calculation is needed to allow 100% SoC to be displayed when a specific termination current threshold (system termination current), higher than the real termination current (charger termination current) is reached.

Cutoff SoC is based on the system cutoff voltage while in sleep or at low SoC, and on the standby current setting, and is applied to the battery SoC in order to estimate the 0% system SoC. This estimate is needed because during intensive phone usage with a high average load current and high series resistance (low temperature or aged battery) the system cutoff voltage could be reached before the battery is completely empty, thereby reducing the SoC available to the system.

System termination current: This is the current threshold used to report 100% SoC. This should be set higher than the charger termination current so 100% SoC is reported before end of charge, otherwise, 100% SoC may never be reached.

Charge termination current (Termcurrent): This is the charger's termination current setting while using the fuel gauge IADC to report end of charge. This should be set lower than the system termination current.

CC_to_CV: This setting is used by the fuel gauge as the input to the feedback loop responsible for matching the 100% SoC point to the programmed system termination current. This must be set to 10 mV less than the maximum charge voltage of the battery in use. If this is set incorrectly, poor system termination current accuracy will be seen.

System cutoff voltage: This voltage setting is used to report 0% SoC. It is recommended this value is not dropped too low or it may negatively impact the fuel gauge performance and system stability. Make sure, when setting this parameter, that the volt_empty threshold is set 100 mV less than this parameter.

See Figure 2-7 for an example of the full SoC parameter, and Figure 2-8 for an example of both full SoC, and cutoff SoC.

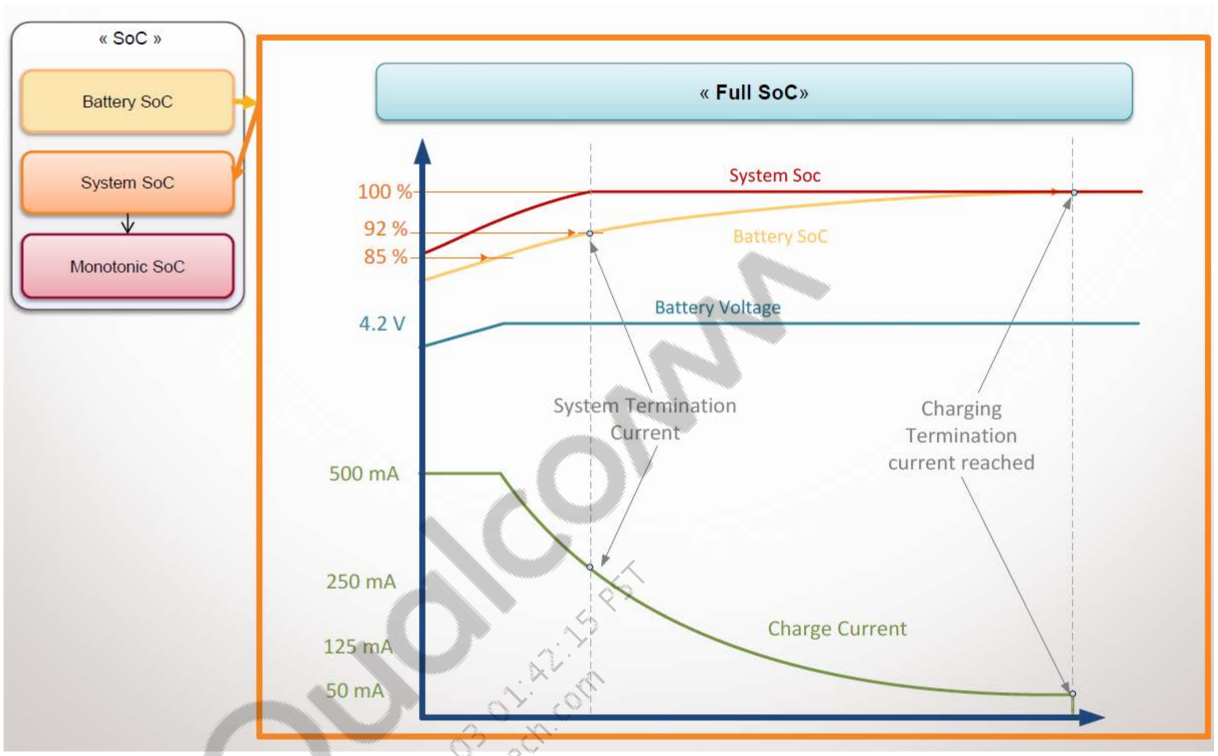


Figure 2-7 System full SoC example

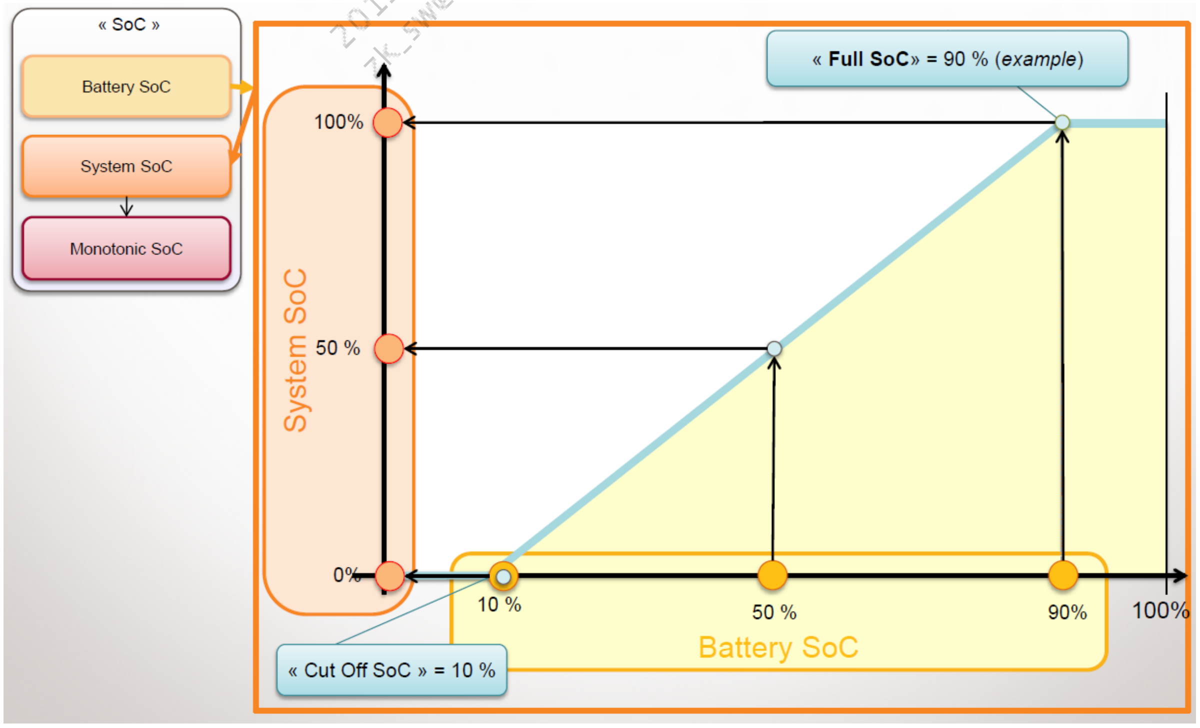


Figure 2-8 System full and cut off SoC example

2.2.1.5 Monotonic state of charge

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

- Increases and decreases of the battery internal resistance value
- Increases and decreases of the battery load current

The default configuration also enforces a slope limiter on the direction allowed, given the current sign.

The slope limiter is needed when:

- 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 undesired step.
- The slope limiter prevents steps in the monotonic SoC.

For an example of this, see Figure 2-10.

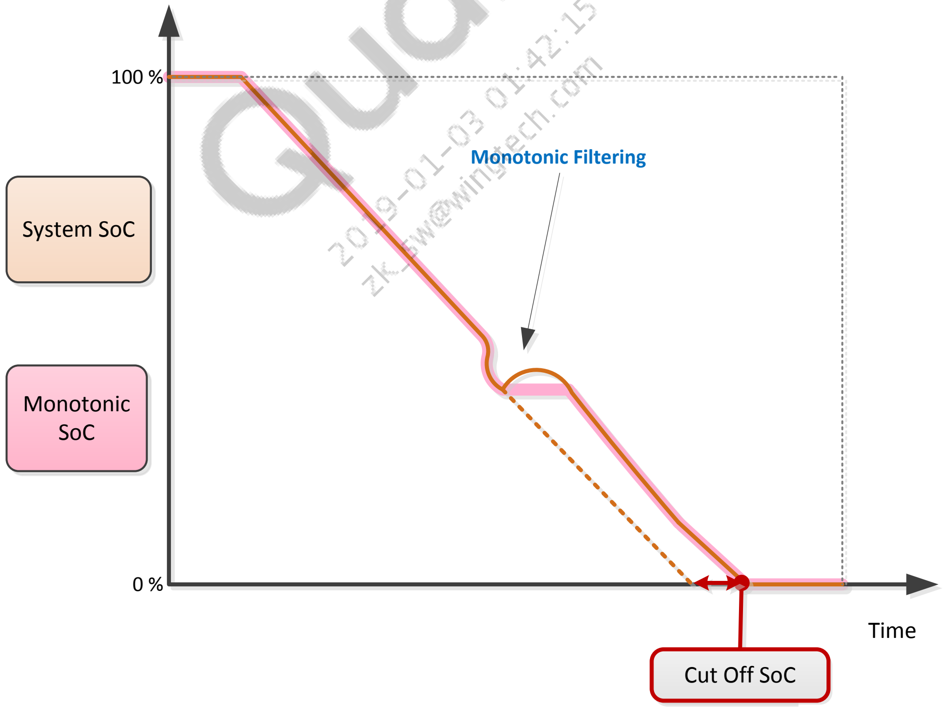


Figure 2-9 Monotonic SoC example

2.2.1.5.1 Slope limiting

Configuration of the slope limiting depends on two parameters: slope_limiter and slope_limiter_coefficient. The maximum change in SoC per cycle (1.47 s) is calculated here:

ΔSoC difference = slope_limiter × slope_limiter_coefficient

Encoding uses an LSB is equal to 1.526e-3%.

- slope_limiter is an unsigned 8-bit number
- slope_limiter_coefficient has a half floating encoding

Here is an example of how to configure the slope limiting:

- slope_limiter = 5
- slope_limiter_coefficient = 45

  = slope_limiter × slope_limiter_coefficient = 225

Then converting using the LSB of 1.52e-3% encoding gives:

225 × .00152% = .342%

The SoC increments/decrements 0.342% each cycle.

2.2.2 Battery profiling

2.2.2.1 Battery characterization

Submission of a battery characterization case type via Salesforce is required to request battery characterization. Customers can also characterize their batteries themselves using the new software tool QBCSW, but will not be able to generate the final profile needed. If customers do their own characterization, QTI will still need to post process the battery data to generate a final profile. The QBCSW is available to customers via CreatePoint. For more information on QBCSW, refer to *Battery Characterization Process Application Note* (80-VT310-24).

The profile generation is broken down into two steps:

1. Characterization: This characterization is very similar to the BMS characterization, but instead QTI now characterizes during charge and discharge, and does not generate resistance tables. The raw data consists of timing data, and OCV vs. SoC data that is very similar to the BMS.

2. Post processing: After the raw data is generated during characterization, high iteration post processing is done on the raw OCV data to generate the most accurate OCV and R_{slow} polynomial coefficients for the fuel gauge polynomial models. Temperature correction information is also collected during this process. This is done by QTI proprietary software that is not available to customers.

2.2.2.2 Battery profile

The fuel gauge requires a battery profile like previous BMS solutions. The battery profile must be merged in the software before the fuel gauge can be used. This has to be done on the customer side.

The new fuel gauge battery profile is very different than the previous BMS profile structures. This is due to the way the fuel gauge hardware works. Instead of the lookup tables used in the previous BMS solutions, the hardware fuel gauge contains models of the OCV vs. SoC relationship, temperature compensation, and R_{slow} resistance. This requires many different coefficients for the different polynomial models within the battery profile. Many of these settings cannot be shown to customers as they are proprietary.

For register settings that are viewable, see Section 3.3.3. For customers, this may be a source of concern as they are not able to read/modify the profiles themselves. To ensure quality, all profiles during and after their generation are checked for accuracy, and approved by QTI's systems team prior to giving to customers.

The OCV-SOC relationship is modeled by three third-order polynomials, connecting at programmable SoC junction points. The OCV-SoC relationship, as well as how this relationship is effected by temperature, is determined during characterization. This data is then processed, modeled, and coefficients are generated for the polynomials that best fit the OCV-to-SoC relationship. A third-order polynomial is also modeled for how the OCV-to-SoC curve is affected by changes in temperature, and these coefficients are also used in the hardware. This creates an accurate OCV-to-SoC model that can be at a maximum ±3% inaccurate, but is typically much lower. See Figure 2-11 for an example on the OCV vs. SoC curve and how it is composed of the polynomials.

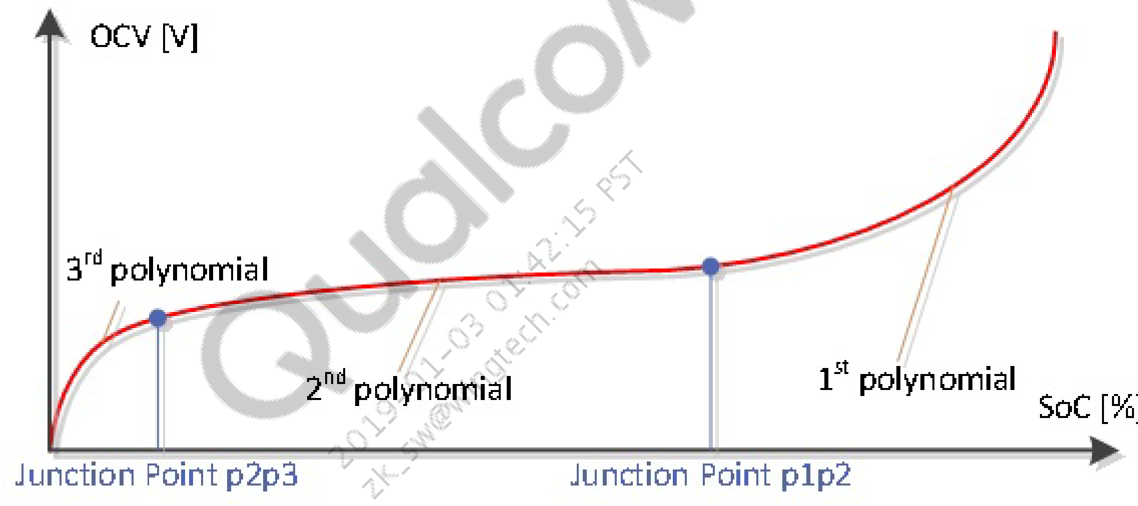


Figure 2-10 OCV vs. SoC curve

2.2.3 Fuel gauge timing plots

This section gives timing information and example plots of different fuel gauge phenomena and the corresponding high-level explanation. Here is a summary of the important timing parameters:

PMI8994, PMI8996, and PMI8952 fuel gauge timing summary

- The fuel gauge update cycle is every 1.47 s. During this time, both the voltage and current measurements are made in the last 160 ms to complete the conversion.
- During charging, ESR measurements happen every ~47 s.
- During discharge, ESR measurements happen at a maximum every ~94 s, if a qualifying transient does not occur before the 94 s timer expires.
- For PMI8994 and PMI8996, the fuel gauge measures USB_ID every 1.47 s, and the conversion time for ADC takes ~42 ms. PMI8952 does not use the fuel gauge for USB_ID detection.

- BATT_THERM is measured every 1.47 s on PMI8994, and every 8 × 1.47 cycles for PMI8952 and PMI8996. The ADC conversion time for this takes ~21 ms.

2.2.3.1 RBIAS/VAA_CAP

Figure 2-12 and Figure 2-13 show an example of when the ADC is taking measurements w.r.t RBIAS being enabled, and VAA_CAP regulating to 2.7 V, which is the source of RBIAS and the ADCs while they are taking measurements.

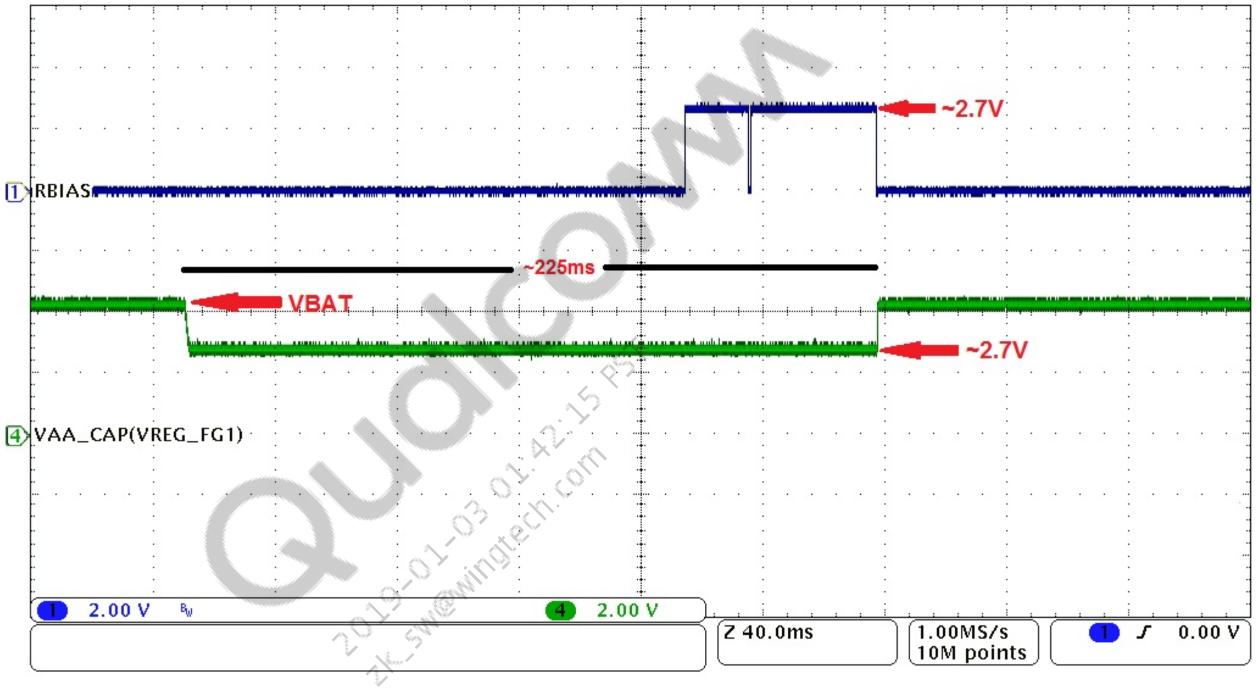


Figure 2-11 Plot for RBIAS and VAA_CAP measuring BATT_THERM and USB_IN

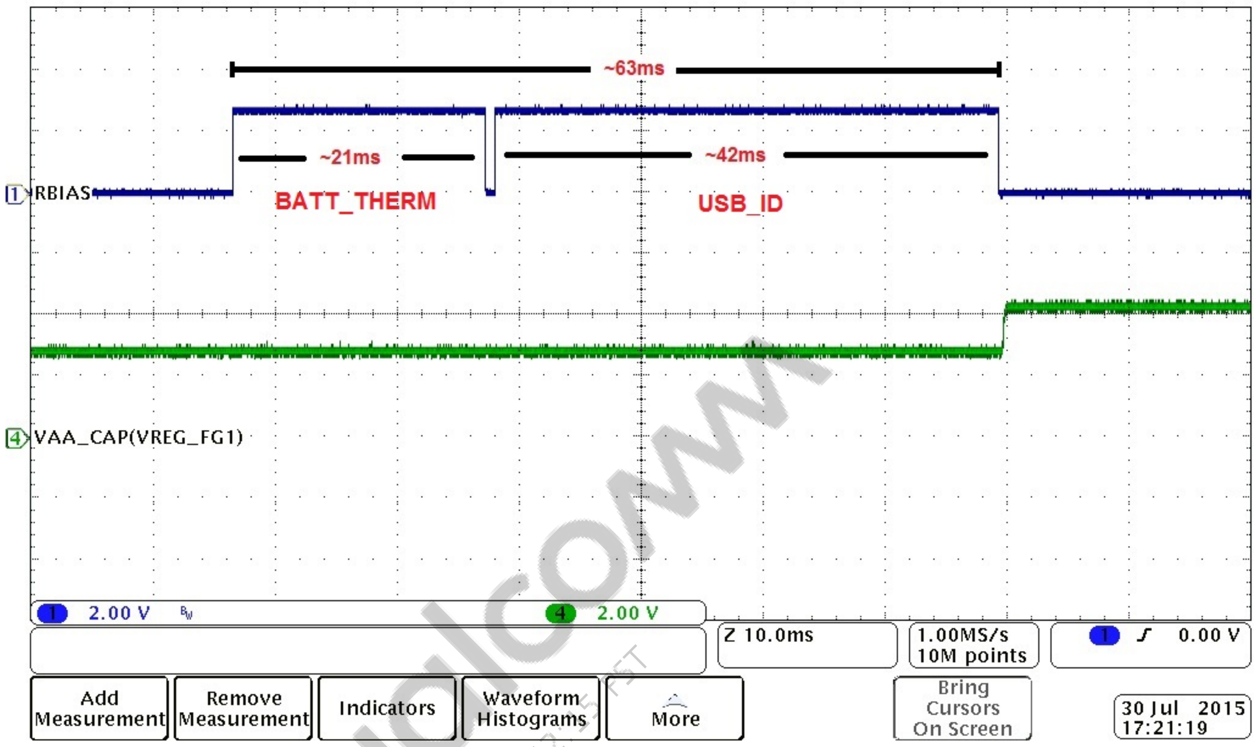


Figure 2-12 Zoomed-in plot for RBIAS and VAA_CAP measuring BATT_THERM and USB_ID

2.2.3.2 Battery resistance measurements

Figure 2-14 and Figure 2-15 are two examples of the fuel gauge creating the necessary load transients to take the ESR measurements.

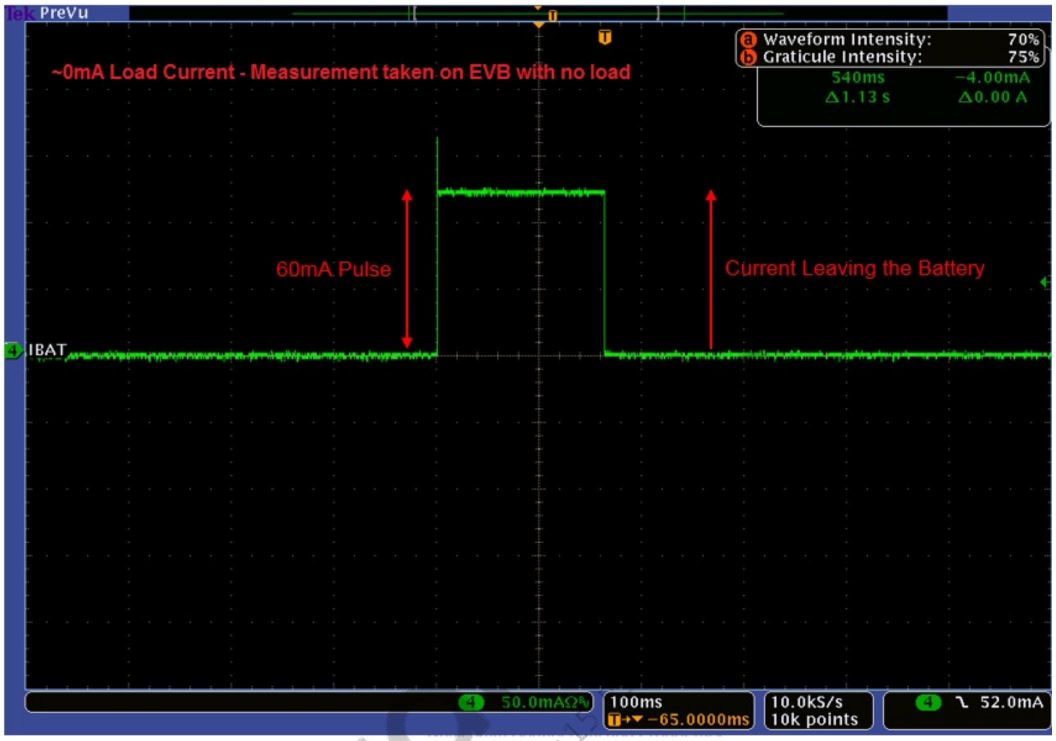


Figure 2-13 Discharge pulse

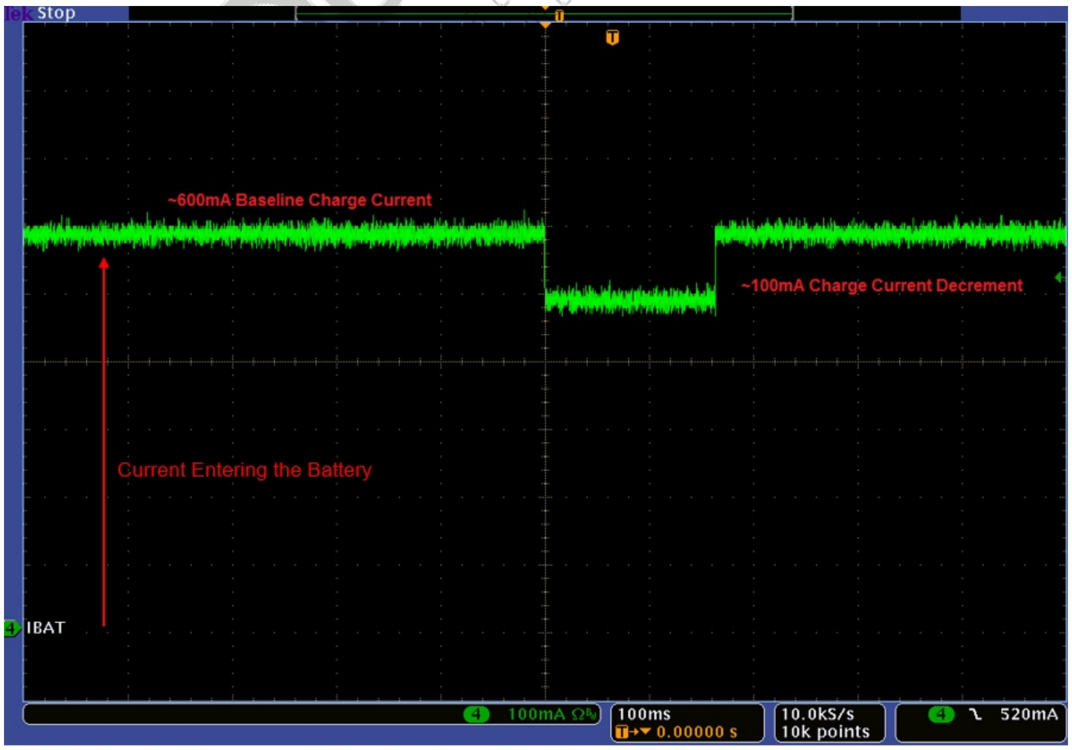


Figure 2-14 Charge current decrement

2.2.3.3 Fuel gauge current consumption

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

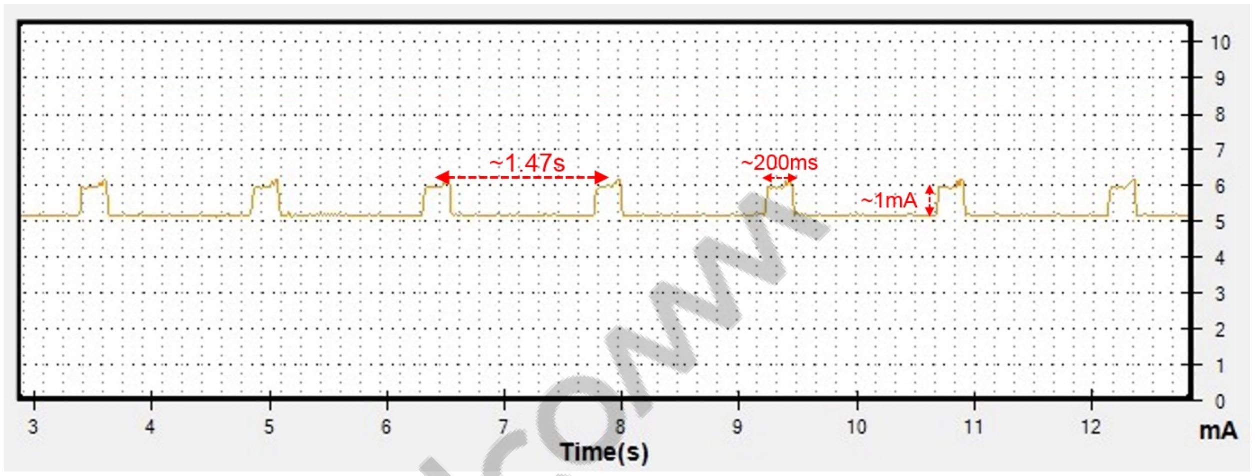


Figure 2-15 Plot for 1 mA current pulses

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.

The block diagram in Figure 3-1 shows the internal memory architecture. The architecture does not envision the use of shadow registers, therefore the access to the fuel gauge register is scheduled in such a way that the read and write access are not interfering with the fuel gauge normal operation.

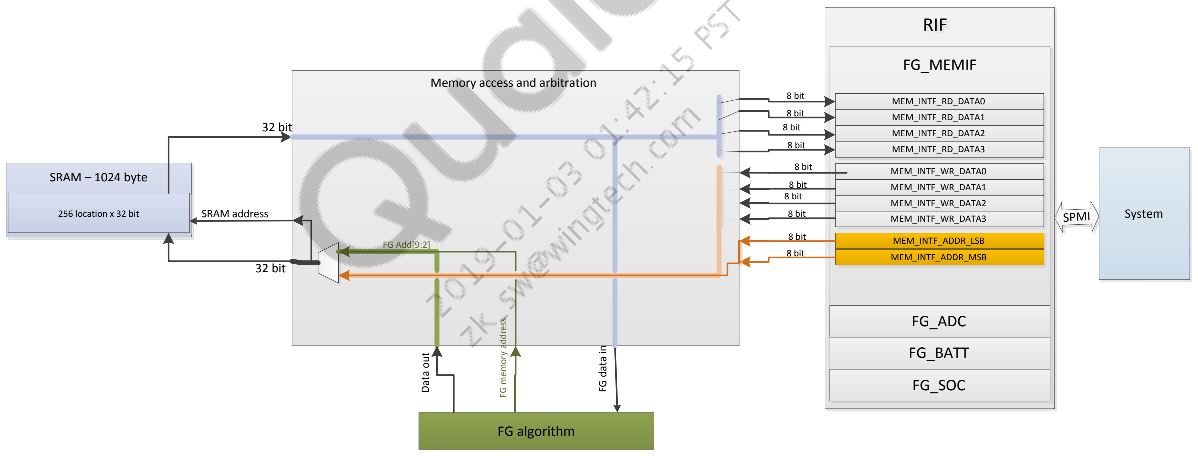


Figure 3-1 Fuel gauge memory architecture

Table 3-1 SRAM partitioning

Section	Start (decimal)	Start MEM_INTF_ADDR	End (decimal)	End MEM_INTF_ADDR	Size (bytes)
System Register range	0	0x400	191	0x4BC	192
Battery Profile in use	192	0x4C0	319	0x53F	128
Scratchpad	320	0x540	511	0x5FF	192

The SRAM size is 512 byte. The SRAM is 32-bit (4 byte) wide even if the fuel gauge is actually accessing the SRAM with a 1 byte granularity.

3.1.1 Half floating encoding (HALF)

The information is stored as half floating point encoding partitioned in:

- Exponent

- Sign
- Mantissa

Here's the procedure to obtain an HALF float encoding from a regular number:

1. Sign = 1 if negative, 0 if positive
2. Get $number_{log} = \log_2(number\_to\_convert)$
3. Round $number_{log}$ to the lower signed integer: int_number_{log}
4. Exponent = $int\_number_{log} + 15$
5. $Mantissa = \left(number\_to\_convert - 2^{\left(int\_number_{log}\right)}\right) * 2^{10}$

Table 3-2 through Table 3-6 report some conversion examples.

Table 3-2 Conversion example 1

Exponent					**Sign**	**Mantissa**									
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Exponent is stored with a +15 bias Exp_value = 2^n where n = exponent – 15 n = 15 – 15 = 0 Exp_value = 2^0 = 1					1 → sign = -1	Mantissa_value									
					0 → sign = 1										
Decimal value = Sign × (Exp_value + Mantissa_value × 2^ (n – 10)) = 1 × (2^0 + 0 × 2^ (0 – 10)) = 1															

Table 3-3 Conversion example 2

Exponent					**Sign**	**Mantissa**									
0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
n = 15 – 15 = 1 Exp_value = 2^0 = 1					1 → sign = -1	Mantissa_value = 0									
Decimal value = -1 × (2^0 + 0 × 2^ (0 – 10)) = -1															

Table 3-4 Conversion example 3

Exponent					**Sign**	**Mantissa**									
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
n = 16 – 15 = 1 Exp_value = 2^1 = 2					1 → sign = -1	Mantissa_value = 0									
Decimal value = = -1 × (2^1 + 0 × 2^ (1 – 10)) = -2															

Table 3-5 Conversion example 4

Exponent					Sign	Mantissa									
0	1	0	0	0	0	0	1	0	0	0	1	1	1	1	0
n = 8 – 15 = -7 Exp_value = 2^-7 = 0.0078125					0 → sign = 1	Mantissa_value = 286									
Decimal value = 1 × (0.0078125 + 286 × 2^ (-7 – 10)) = 0.009994506835938 ≈ 0.01															

Table 3-6 Conversion example 5

Exponent					Sign	Mantissa									
0	0	1	1	0	1	0	1	0	0	0	1	1	1	1	0
n = 6 – 15 = -9 Exp_value = 2^-9 = 0.001953125000000					1 → sign = -1	Mantissa_value = 286									
Decimal value = -1 × (0.001953125000000 + 286 × 2^ (-9 – 10)) = -0.002498626708984≈ -0.0025															

3.2 Fuel gauge interrupts

Table 3-7 Fuel gauge interrupts

Interrupt	Peripheral	Description	Related register
JEITA_SOFT_COLD_RT_STS	FG_BATT	Interrupt to notify that the battery temperature < JEITA_soft_cold threshold.	Indirect addressing: JEITA_soft_cold threshold
JEITA_SOFT_HOT_RT_STS	FG_BATT	Interrupt to notify that the battery temperature > JEITA_soft_hot threshold.	Indirect addressing: JEITA_soft_hot threshold
VBATT_LOW_RT_STS	FG_BATT	Interrupt to notify that the battery voltage < IRQ_volt_min threshold, digital comparison on the ADC value.	Indirect addressing: IRQ_volt_min
BATT_IDENTIFIED_RT_STS	FG_BATT	Interrupt to notify that the battery identification is completed.	
BATT_ID_REQ_RT_STS	FG_BATT	Interrupt to notify that the system has to perform software identification because a smart battery is detected.	
BATT_UNKNOWN_RT_STS	FG_BATT	Interrupt to notify that the battery is not recognized.	
BATT_MISSING_RT_STS	FG_SOC	Interrupt to notify that the battery is missing.	

Interrupt	Peripheral	Description	Related register
BATT_MATCH_RT_STS	FG_SOC	Interrupt to notify that the reconnection of the same battery has been detected	
HIGH_SOC_RT_STS	FG_SOC	IRQ to notify that Monotonics SoC > High SoC Threshold.	Indirect addressing: IRQ_soc_max
LOW_SOC_RT_STS	FG_SOC	IRQ to notify that Monotonics SoC < Low SoC Threshold.	Indirect addressing: IRQ_soc_min
FULL_SOC_RT_STS	FG_SOC	IRQ to notify that Monotonics SoC = 100%.	
EMPTY_SOC_RT_STS	FG_SOC	IRQ to notify that Monotonic SoC = 0%.	Indirect addressing: IRQ_volt_empty
DELTA_SOC_RT_STS	FG_SOC	IRQ to notify that Monotonic SoC change exceeded the specified delta SoC threshold. This interrupt is used to update the system when the SoC changes.	Indirect addressing: IRQ_DeltaSoC_Threshold
FIRST_EST_DONE_RT_STS	FG_SOC	IRQ to notify that the First estimate of SoC is done, cleared on a battery missing event.	
FG_MEM_AVAIL_RT_STS	FG_MEMIF	1= S/W allowed to access FG memory.	
DATA_RCVRY_SUG_RT_STS	FG_MEMIF	1= S/W allowed to update fuel gauge ram contents to assist in boot	Indirect addressing: Restore_xxxx registers
VBAT_LT_THR_INT_RT_STS	FG_ADC_USR	IRQ indicating that VBAT is less than threshold defined in VBAT_INT__THR register.	SPMI FG_ADC_USR peripheral: VBAT_INT
VBAT_LT_THR_INT_RT_STS	FG_ADC_MDM	IRQ indicating that VBAT is less than threshold defined in VBAT_INT__THR register.	SPMI FG_ADC_MDM peripheral: VBAT_INT
IBAT_GT_THR_RT_STS	FG_ADC_USR	IRQ indicating that IBAT is greater than threshold defined in IBAT_INT__THR register.	SPMI FG_ADC_USR peripheral: IBAT_INT
IBAT_GT_THR_RT_STS	FG_ADC_USR	RQ indicating that IBAT is greater than threshold defined in IBAT_INT__THR register.	SPMI FG_ADC_MDM peripheral: IBAT_INT

3.3 Customer-facing register map for PMI8994, PMI8952, and PMI8996

3.3.1 PMI8994 vs. PMI8952 and PMI8996 register differences

This section contains the exposed registers on PMI8952 and PMI8996 that do not exist on PMI8994. Note that the fuel gauge for PMI8952 and PMI8996 are the same, and therefore their register maps match.

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INT F_ADDR start address, byte offset
4	SW_CC_SOC	This is the pure coulomb-count register for PMI8952/PMI8996 that tracks coulombs as a percent of actual capacity (0x578, 2).	SIGN	The SoC is represented by a 32-bit signed number.	0x00000000 = 0% 0x0FFFFFFF = 100% LSB = 1/((2^28)-1)%	0x5BC, 3

3.3.2 Scratch pad

Table 3-8 Scratch pad registers

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ ADDR start address, byte offset
2	V_current_predicted	Given the values of the battery, SoC, battery resistance, and battery current; this is the voltage expected on the battery terminals	UNS	The voltage has the same encoding of the ADC when converting the battery voltage.	0x0000 = 0V 0x7FFF = 5V – 1 LSB LSB = 5/2^15	0x540, 0
2	OCV_Estimate	Is the estimate of the open circuit voltage (OCV) of the battery updated every cycle by the algorithm	UNS	The voltage has the same encoding of the ADC when converting the battery voltage	0x0000 = 0V 0x7FFF = 5V – 1 LSB LSB = 5/2^16	0x544, 2
2	B_Temperature	Reading of the battery temperature	UNS	Battery temperature with 16-bit representation	0x0000 = 0 K 0x1112 = 273.15 K 0x12A2 = 298.15 K LSB = 0.0625 K	0x550, 2
2	ESR_actual	Detected value of battery resistance. This register is mirrored in the ADC peripheral; registers FG_ADC_USR_BAT_RES_X_X, addresses: 0x425C and 0x425D	HALF	Half floating encoding (HALF) is used to represent the battery resistance. Refer to Section 3.1.1 on half floating encoding.	Once the number is converted from his HALF encoding, that number is already the resistance value.	0x554, 2
2	Rslow	Estimate value of the R_{slow} (R1C1) resistance of the battery	HALF	Half floating encoding (HALF) is used to represent the battery resistance. Refer to Section 3.1.1 on half floating encoding.	Once the number is converted from his HALF encoding, that number is already the resistance value.	0x558, 0

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ ADDR start address, byte offset
3	SoC_CutOff	SoC cutoff prediction (or empty SoC)	UNS	The SoC is represented by a 24-bit unsigned number.	0x00000 = 0% 0xFFFFFF = 100% LSB = 1/((2^24)-1)%	0x564, 0
3	SoC_Full	SOC full prediction	UNS	The SoC is represented by a 24-bit unsigned number.	0x00000 = 0% 0xFFFFFF = 100% LSB = 1/((2^24)-1)%	0x564, 3
3	Battery_SoC	Battery state of charge, combination of Coulomb counting and voltage model	UNS	The SoC is represented by a 24-bit unsigned number.	0x00000 = 0% 0xFFFFFF = 100% LSB = 1/((2^24)-1)%	0x56C, 1
4	CC_SoC	State of charge derived from just coulomb counting. Note: This register is reset during CV charging and not reliable for customers to measure for coulomb counting	SIGN	The SoC is represented by a 32-bit signed number.	0x00000000 = 0% 0x0FFFFFFF = 100% LSB = 1/((2^28)-1)%	0x570, 0
2	SoC_System	Battery state of charge corrected with the full and the empty state of charge	UNS	The SoC is represented by a 16-bit unsigned number.	0x00000 = 0% 0xFFFFFF = 100% LSB = 1/((2^24)-1)%	0x574, 0
2	SoC_Monotonic	SoC filtered corrected with slope limiter and filtered for monotonicity. This is the final SoC reported to end user	UNS	The SoC is represented by a 16-bit unsigned number.	0x00000 = 0% 0xFFFFFF = 100% LSB = 1/((2^24)-1)%	0x574, 2
2	actual_capacity_mah	Battery capacity mirrored from battery profile [mAh]	UNS	Battery capacity in mAh.	0x06D6 = 1750 mAh LSB = 1mAh	0x578, 2
2	BID_Detect_value	The latest value of battery ID converted by the ADC in the detection process is stored in this temporary register	UNS	This encoding is the voltage detected due to the bias current used through the BID resistor.	0x00 = 0 0x7F = 1.240 V 0xFF = 2.49 V LSB = 9.8 mV	0x594, 1

1	BID_Detect_additional	Additional information regarding the detected battery ID	UNS	Miscellaneous battery ID related register settings	REG1[1:0] ← 2'b11;// high current
					REG1[1:0] ← 2'b10;// medium current
					REG1[1:0] ← 2'b01;// small current
						0x594, 3

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ ADDR start address, byte offset
					REG1[3:2] ← STATUS_BATTERY_ID_4P;	
					REG1[4] ← STATUS_BID_SMART;	
					REG1[5] ← STATUS_BID_UNKNOWN;	
					REG1[6] ← STATUS_BID_SHORT;	
					REG1[7] ← BATT_PERIPH_battery_detection_error_o | STATUS_BID_POSSIBLE_MISSING;	
2	USB_ID_valid_value	Converted USB ID value	UNS	The latest stable of USB ID converted value	USBID_code = 2^12 × USBID_resistor/ (24.9 kΩ + USBID_resistor) The value of the LSB is not fixed in terms of resistance; it depends on the resistance value. See Section 2.2.3 for more information.	0x598, 0
1	Latest_battery_info	Assorted battery Information	CFG		[1:0] = status_BATTERY_ID	0x5C4, 0
					00 = Profile 1	
					01 = Profile 2	
					10 = Profile 3	
					11 = Profile 4	
					[4:2] = System Status	
					SYSTEM_CHARGING_CC: 3'b010	
					SYSTEM_CHARGING_CV: 3'b001	
					SYSTEM_FULL: 3'b011;	
				-	SYSTEM_EMPTY: 3'b111;	
					SYSTEM_DISCHARGING: 3'b100;	
					SYSTEM_ISMISSING: 3'b000;	
					SYSTEM_AUTOMATIC_RECHARGE: 3'b101;	
					SYSTEM_SUPPLEMENTAL: 3'b110;	
					[5] = READY	

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ ADDR start address, byte offset
					[6] = chk_status_boot_done: asserted when end of the connection sequence is done, de-asserted when in battery missing	
					[7] ← Integrity bit	
2	Voltage_Shadow	Last battery voltage measurement.	UNS	The voltage has the same encoding of the ADC when converting the battery voltage	0x0000 = 0V 0x7FFF = 5V – 1 LSB LSB = 5/2^15	0x5CC, 1
2	Current_Shadow	Last battery current measurement.	SIGN	The current has the same encoding of the ADC when converting the battery current	0x0000 = 0A 0x7FFF = 5A – 1 LSB 0x8000 = - 5A LSB = 5/2^15	0x5CC, 3

3.3.3 System RAM

Table 3-9 System RAM registers

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ADDR start address, byte offset
	cutoff_voltage	System cut off voltage; used for the estimate of the empty SoC.	UNS	The voltage has the same encoding of the ADC when converting the battery voltage	0x0000 = 0V 0x7FFF = 5V – 1 LSB LSB = 5/2^15	0x40C, 0 0x40C, 1
2	System termination current	Termination current for purpose of estimating the full SoC. Note: This value must be negative.	SIGN	The current has the same encoding of the ADC when converting the battery current	0x0000 = 0A 0x7FFF = 5A – 1 LSB 0x8000 = - 5A LSB = 5/2^15	0x40C, 2 0x40C, 3
2	Stby_current	Standby current value for the cutoff SoC estimate. Note: This value should be positive.	SIGN	The current has the same encoding of the ADC when converting the battery current	0x0000 = 0A 0x7FFF = 5A – 1 LSB 0x8000 = - 5A LSB = 5/2^15	0x410, 0 0x410, 1
1	slope_limiter	Multiplied with slope_limiter_coefficient defines the slope limiter.	UNS	Convert hex to decimal	Number	0x430, 2
2	slope_limiter_coefficient	Slope limiter is multiplied by slope	HALF		-	0x430, 3

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ADDR start address, byte offset
		coefficient to define the maximum change in the SoC.		Half floating encoding (HALF)		0x434, 0
2	thermistor_c1_coeff	1st degree coefficient that represent the linear component of the ratio to Temperature relationship.	HALF	Half floating encoding (HALF)	–	0x444, 2 0x444, 3
2	thermistor_c2_coeff	2nd degree coefficient that represent the quadratic component of the ratio to temperature relationship.	HALF	Half floating encoding (HALF)	–	0x448, 0 0x448, 1
2	thermistor_c3_coeff	3rd degree coefficient that represent the cubic component of the ratio to temperature relationship.	HALF	Half floating encoding (HALF)	–	0x448, 2 0x448, 3
1	IRQ_DeltaSoC_Threshold	Threshold on the monotonic system SoC change, used to issue the related interrupt.	UNS	The SoC is represented by a 8-bit unsigned number	0x00 = 0% 0xFF = 100% LSB = (100/255)%	0x450, 3
1	JEITA_soft_cold	Jeita soft cold temperature threshold.	UNS	Temperature offset that is added to 243 Kelvin	0x00 = 243K = 243 K 0x1E = 243K + 30 = 273 K 0x7F=243K + 128 = 371 K LSB = 1 K	0x454, 0
1	JEITA_soft_hot	Jeita soft hot temperature threshold.	UNS			0x454, 1
1	JEITA_hard_cold	Jeita hard cold temperature threshold.	UNS			0x454, 2
1	JEITA_hard_hot	Jeita hard hot temperature threshold	UNS			0x454, 3
1	IRQ_volt_min	Low battery voltage interrupt threshold.	UNS	The voltage has the same encoding of the ADC when converting the battery voltage	0x00 = 2.5 V 0x52 = 3.3 V LSB = 5/2^9	0x458, 0
1	IRQ_soc_max	High SoC interrupt threshold.	UNS	The SoC is represented by an 8-bit unsigned number.	0x00 = 0% 0xFF = 100% LSB = (100/255) %	0x458, 1
1	IRQ_soc_min	Low SoC interrupt threshold.	UNS	The SoC is represented by an 8-bit unsigned number	0x00 = 0% 0xFF = 100% LSB = (100/255) %	0x458, 2

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ADDR start address, byte offset
1	IRQ_volt_empty	Low battery voltage threshold for the empty interrupt.	UNS	This battery voltage is an offset from 2.5 V, with the encoding reported below	0x00 = 2.5V 0x52= 3.3V LSB = 5 / 2^9	0x458, 3
1	ARECH_LowSoC_Threshold	Monotonic SoC threshold for automatic recharge.	UNS	The SoC is represented by an 8-bit unsigned number	0x00 = 0% 0xFF = 100% LSB = (100/255)%	0x45C, 1
1	battery_id_1	Battery ID for profile 1	UNS	The encoding reported is always valid voltage wise, but the corresponding resistor value depends on the bias current.	0x00 = 0 0x7F = 1.240 V 0xFF = 2.49 V LSB = 9.8 mV	0x460, 1
1	battery_id_2	Battery ID for profile 2	UNS			0x460, 2
1	battery_id_3	Battery ID for profile 3	UNS			0x460, 3
1	battery_id_4	Battery ID for profile 4	UNS			0x464, 0
2	battery_id_recheck_range	Battery resistance code tolerance for recheck purpose.	HALF	See Section 2.1.3	See Section 2.1.3	0x464, 1 0x464, 2
2	BID_Tolerance_RANGE1	Battery resistance code tolerance for current range = 150 µA	HALF			0x464, 3 0x468, 0
2	BID_Tolerance_RANGE2	Battery resistance code tolerance for current range = 15 µA	HALF			0x468, 1 0x468, 2
2	usb_id_change_irq_range	Change threshold for the USB ID change interrupt	UNS	This is the range used to identify a specific resistor value	See Section 2.2.3	0x46C, 2 0x46C, 3
2	usb_id_recheck_range	Change threshold for considering the usbid value as stable.	UNS	This is the range used to identify a specific resistor value	See Section 2.2.3	0x470, 0 0x470, 1
2	USBID_ACA_TOL	Tolerance on the ACA_X usb id resistor value.	UNS	This is the range used to identify a specific resistor value	See Section 2.2.3	0x470, 2 0x470, 3
1	FGALG_syscbits_1	Slope limiting configurations	CONFIG	Reserved	-	0x4B0, 1
				Reserved	-	
				Reserved	-	
				Reserved	-	
				Reserved	-	

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ADDR start address, byte offset
				[5:5]: enable_monotonic_filter	1 = enable 0 = disable	
				[6:6]: controls the enable of the slope limiter	1 = enable 0 = disable	
				Reserved	-	
1	alert_cfg	SoC configurations	-	[0:0]: IRQ_use_voltage_hyst	Config	0x4B0, 3
				[1:1]: Empty_from_voltage: config bit to assert empty on empty voltage reached	1 = true 0 =false	
				[2:2]: Empty_from_SoC: config bit to assert empty on 0% monotonic SoC	1 = true 0 =false	
				[3:3]: DBG_delta_soc_every_cycle: control bit to force the assertion of delta soc interrupt every cycle	1 = true 0 =false	
				Reserved	-	
				Reserved	-	
				Reserved	-	
1	bid_config_1	Battery ID configurations		[1:0]: bid_active_profiles: Define the number of active profiles.	00 = Profile 1 active	0x4B8, 0
					01 = Profile 1 and Profile 2 active	
					10 = Profile 1, Profile , and Profile 3 active	
					11 = All profiles active	
				[2:2]: bid_smart_active: allows the detection of a smart battery	1 = true	
					0 =false	

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ADDR start address, byte offset
				[3:3]: bid_identification_enable; enable the battery identification	1 = true, 0 =false	
				[4:4]: P1_range: current range for profile 1	0 = 150 µA range	
					1 = 15 µA range	
				[5:5]: P2_range: current range for profile 2	0 = 150 µA range	
					1 = 15 µA range	
				[6:6]: P3_range: current range for profile 3	0 = 150 µA range	
					1 = 15 µA range	
				[7:7]: P4_range: current range for profile 4	0 = 150 µA range	
					1 = 15 µA range	
1	bid_config_2	Battery ID configurations		Reserved		0x4B8, 1
				[2:1]: smart_default_select; battery profile to use when a smart battery is detected	00 = Profile 1	
					01 = Profile 2	
					10 = Profile 3	
					11 = Profile 4	
				[4:3]: bid_profile_fallback_select; battery profile to use when the identification is not successful	00 = Profile 1	
					01 = Profile 2	
					10 = Profile 3	
					11 = Profile 4	
				Reserved	-	
				Reserved	-	
				Reserved	-	
				Reserved	-	

3.3.4 Battery profile

Table 3-10 Battery profile registers

Byte size	Register name	Register description	Data type	Encoding description	Encoding	MEM_INTF_ADDR start address, ByteOffset
2	Termcurrent	Termination current used for a specific battery	SIGN	The current has the same encoding of the ADC when converting the battery current	0x0000 = 0A 0x7FFF = 5A – 1 LSB 0x8000 = - 5A LSB = 5/2^15	0x4F8, 2

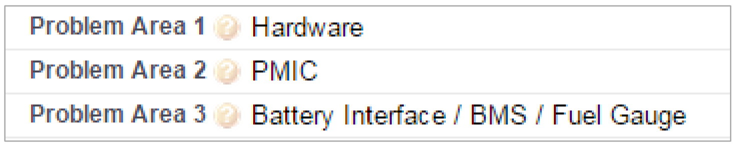
4 Troubleshooting

When experiencing an issue suspected to be a fuel gauge issue, QTI recommends creating a fuel gauge case via Salesforce. The case will be routed to QTI's fuel gauge module leads who can assist in debugging any issues. When creating a case, provide the following information:

- Detailed description and steps to reproduce the issue.
  - A few questions to answer:
    - Charging or discharging?
    - Battery voltage?
    - Timing information, if applicable
    - New SW or HW?
- Failure rate: The number of parts that have been tested and the number of parts with the issue.
- Logs:
  - If applicable, SRAM logs recorded during the issue. The should be provided in the proper file format (no SPMI logs in the .txt file format should be provided).
    - These are not the usual Linux Android logs. For more information on the process for taking these logs, refer to *MSM899x Linux Android PMIC Fuel Gauge Software User Guide* (80-NM328-52), section 3.7.1 and the *Application Note: Fuel Gauge Data Collector* (80-NU716-1).

NOTE: Do not provide QTI with SRAM logs mixed in with any other logs that contain SPMI or driver logs. The SRAM logs should be in a separate text file. If the logs are mixed with other types of logs, QTI may not accept these logs and request that the logs be regenerated in the correct format.

  - If applicable, separate the Linux Android logs with charger and fuel gauge debug options enabled.



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

5 Frequently asked questions

5.1 Algorithm and measurement FAQs

5.1.1 How do I test the fuel gauge performance?

Due to the complex and proprietary nature of the hardware fuel gauge, and the intricacies of testing it accurately and correctly, QTI does not support customer fuel gauge evaluation. QTI tests the fuel gauge extensively and has SoC accuracy data to provide over many use cases. Create a fuel gauge case in Salesforce for the specific PMI chip under evaluation and this information can be provided.

Most fuel gauge issues stem from poor layout or incorrect configurations. If there are any concerns with fuel gauge performance, it is recommended that a case may be created, and QTI can review the SRAM logs during normal charge/discharge operation, as well as the layout of the fuel gauge. If there is anything wrong with the fuel gauge performance, this review will catch it.

5.1.2 Can the fuel gauge measurement cycle be altered?

The 1.47 s fuel gauge update cycle is fixed in hardware and cannot be modified. Most concerns with this measurement interval stem from fears that the fuel gauge may miss large transients that do not happen within the 200 ms conversion. QTI's internal testing has shown good performance with dynamic loading (GSM pulses, for example), and that on average, the coulomb-count will average out in time to be accurate enough to meet the QTI target typical accuracy specification of 3%.

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 do so.

Although it is technically possible to change the timing on ESR measurements, QTI strongly recommends against doing so and QTI cannot guarantee the fuel gauge's performance 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 will occur that will be large enough for the fuel gauge to use to update the ESR. If, for whatever reason, there is relatively constant load on the battery, then ESR pulses will be used to measure battery resistance. Remember that these pulses only last for a few hundred milliseconds (and only happen every ~90 s), and do not consume much current with

60 mA as the default setting. If there is concern with the impact that this may have on Days of Use (DoU) or other use-time metrics, it is strongly recommended the calculations be done on the customer's end to prove that this will have a minimal effect on use-time.

- Charge: An external load current pulse cannot be used to measure ESR as the charger will compensate too quickly for this additional load, causing the measurement to be affected or missed. To fix this, the fuel gauge communicates to the charger a very quick current decrement of approximately 100 mA, and use this load transient to measure ESR during charge. If there are any concerns that this decrement will affect charge time, remember that these decrements happen every ~50 s and only last for a ~200 ms, and the current decrement during charge is only 100 mA. If there are concerns that this will affect charge time, it is strongly recommended that calculations be done to verify the very minimal 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 ~200 ms, and cause the fuel gauge module (ADC) to consume roughly 1 mA of current. See Section 2.2.3.1 for an example plot of these current pulses.

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

The feedback loop internal to the fuel gauge responsible for this function uses two settings as inputs in matching the 100% point to the system termination current: the system termination current setting and the CC_to_CV setting. This setting needs to be set to 10 mV less than the programmed float voltage to make sure this prediction loop works correctly. There is no specification for this accuracy as it depends on the fuel gauge's accuracy, which depends on the test setup, layout, etc. If poor accuracy is noticed, verify that the parameter is set correctly in the SW. If this setting is set correctly, but accuracy is still suffering, this could be a more serious board level issue, and it is recommended a case be created via Salesforce to get help in debugging this further.

5.2 Battery thermistor selection and measurement FAQs

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

This is a common issue that is attributed to added capacitance on the BATT_THERM net. Additional capacitance on this net is not supported on PMI8994, and will cause temperature accuracy measurement issues. When checking for added capacitance the battery pack's internal protection circuit is commonly overlooked. Ensure to check that too.

PMI8996 and PMI8952 support up to a limited amount of capacitance on the thermistor node. See Section 2.1.4.3.1 for more information.

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

See Section 2.1.4.3.2 on thermistor selection and placement for more information on this.

5.3 Battery identification resistor selection and algorithm FAQ

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 will cause RID accuracy measurement issues. When checking for added capacitance, the battery pack's internal protection circuit is commonly overlooked. Ensure to check that too. No added capacitance is supported.

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, RID measurement accuracy cannot be guaranteed.

5.4 Battery characterization FAQs

5.4.1 How do I submit the batteries for characterization?

Create a battery characterization case type, not a wireless device support case type. Fill in all necessary information, and send batteries to the address on the case. If you have any questions, use the case and QTI will be happy to help.

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
  - BCL – This is the most important feature on this list, and is a necessary part of QTI's chipsets, especially the high-tier chipsets, as they have the potential to consume large amounts of current in a short time. This feature allows the software to mitigate heavy loading at low battery voltage, preventing instability and crashes that would be caused by the battery voltage dropping too low. This functionality is not easily replicated in the

    software, and QTI will not provide support to copy this feature when using a third-party solution.
  - Flash mitigation – Software uses the real-time ESR measurements from the fuel gauge to 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 compensation. These functions are in the hardware and would need to be transferred to the software on the customer's end with no support from QTI.
  - BATT_ID and USB_ID(PMI8994 only) – The fuel gauge also contains the modules that control BAT_ID detection and USB_ID detection. The fuel gauge module must be enabled if these functions 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 and resources, not to mention the cost of any additional ICs. These can be avoided by using QTI's fuel gauge solution. The factors need to be weighed against the improvement a third-party IC may bring. It is strongly recommended that customers start a discussion with QTI to make sure that these improvements, whether target specifications or requirements, are not already met, or if QTI can help to meet certain specifications or requirements.
- QTI's fuel gauge is very competitive
  - QTI targets a typical 3% SoC accuracy for all internal testing. Contact QTI for SoC accuracy information to compare QTI's fuel gauge accuracy vs. those of the competitors.

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, etc. If there are any features in the fuel gauge that are not present, but are desired, create a Salesforce case and pass this information to QTI's CE teams. QTI will do its best to support customer requirements in the current or future versions of the fuel gauge.