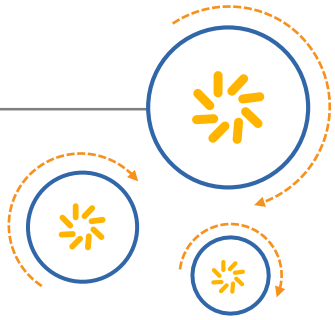
Qualcomm Technologies, Inc.



MSM899x Linux Android PMIC Fuel Gauge Software

User Guide

80-NM328-52 D

January 12, 2016

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

Revision history

Revision	Date	Description
A	August 2014	Initial release
B	January 2015	Updated with API information
C	March 2015	Added Sections 2.2.18, 2.2.8, 2.2.11, 2.2.13, and 2.2.18; updated Sections 2.2.5.1 and 2.2.12
D	January 2016	Updated document to support MSM8996 Added Sections 2.2.5 and 2.2.18; updated Sections 2.1, 2.2.5, 2.2.10, 2.2.12, 2.2.20, 2.2.22, 3.2, 3.4, 3.5, 3.7, and Figure 2-1

Contents

Figures

Tables

1 Introduction

1.1 Purpose

This document describes the programmable features for the fuel gauge (FG) and the software driver configuration for PMI8994 and PMI8996 chipsets. It is recommended to have a complete understanding of the hardware specifications to properly configure the charging parameters in the device tree file. Refer to *PMI8994 Power Management IC Device Specification (Advanced Information)* (80-NJ118-1).

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

Shading indicates content that has been added or changed in this revision of the document.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 Overview

2.1 General description

The Fuel Gauge (FG) module is a hardware implementation of an algorithm that accurately estimates the battery's state of charge by mixing current monitoring with the voltage-based technique. This method ensures short-term linearity and long-term high accuracy. Furthermore, full-battery cycling and zero-current load conditions are not required to maintain the accuracy.

When precise measurements of voltage, current, and temperatures are used, an accurate state of charge estimate is delivered over a broad range of operative conditions. High reliability is achieved through a complex compensation for temperature and aging effects, providing a dependable state-of-charge (SOC) estimate throughout the battery life.

The FG allows for measuring the battery pack temperature via an external thermistor. Missing battery detection monitors for battery insertion or removal, and updates the SOC when a battery is reconnected.

The FG interfaces with the battery charger module to do the following:

- Enable top-off charging when the device is connected to a power supply for an extended period
- Disable USB on-the-go (OTG) functionality when battery SOC falls below a programmable threshold

NOTE: Use this document concurrently with *PMI8994 Fuel Gauge HW/SW Control* (80-VT310-123), as it focuses on software aspects and supports for the PMI899x FG solution.

2.2 Functional description

NOTE: Numerous changes were made in this section.

The FG monitor is based on a QTI-proprietary algorithm. This algorithm guarantees accurate estimation of the battery residual capacity, i.e., SOC, against different user cases, battery conditions, and ages. The algorithm relies on Coulomb counting and voltage-based techniques, which ensures short-term linearity and long-term high accuracy at the same time.

The current state of the battery is continuously updated by monitoring the voltage at the battery connectors, the total charge exchanged from and to the battery, and the battery temperature. These values are used to detect the battery condition and provide an estimate of the residual capacity, according to the following values:

- Calculation of the total charge present in the battery (Coulomb count)
- Variation to the charge value returned by the adopted battery model (voltage-based model)

2.2.1 FG current sensing

The FG operates precise Coulomb counting by sensing current from and to the battery with the voltage across the 10 mΩ sense resistor. Voltage across the sense resistor is read by the dedicated differential pins CS_P and CS_N. Current is read positive when discharging the battery or is negative otherwise. The CS_P and CS_N pins are internally connected to a dedicated analog-to-digital converter (ADC). The battery current is read every ~1500 ms with a resolution of approximately 150 µA. The battery current and voltage are read synchronously.

The current is accessible through the power supply framework (software read) as POWER_SUPPLY_PROP_CURRENT_NOW.

2.2.2 FG battery voltage sensing

Information about battery voltage is retrieved across the dedicated BATT_P and BATT_N differential pins. These pins are connected to the battery pads and internally feed the dedicated battery voltage ADC. The battery voltage is read every ~1500 ms with a resolution of approximately 150 µV. Both the battery voltage and current are read synchronously.

The voltage is accessible through the power supply framework (software read) as POWER_SUPPLY_PROP_VOLTAGE_NOW.

2.2.3 BAT_ID pin

The FG module is provided with a dedicated BAT_ID pin for battery missing detection and battery model identification. This option can be enabled any time the system includes battery packs provided with a specific pin for battery ID resistor connection. The BAT_ID pin is internally pulled up.

The detection is done automatically inside the FG. The detection is repeated with an increased bias current until a find is matched (5 µA→15 µA→150 µA).

In the Smart Battery Range, the value of the resistor is identified with 5 µA current sink, which is used for smart batteries. Hardware identification of the different ID resistors in this range is not supported. The 5 µA has an enable flag because the BSI module must identify the smart battery.

The BAT ID accessible through the power supply framework (software read) as POWER_SUPPLY_PROP_RESISTANCE_ID.

2.2.4 Thermistor pin

The FG module integrates circuitry for a battery temperature reading. Figure 2-1, shows the default RBIAS and THERM pin connection, with a thermistor placed externally and in close proximity to the battery pack.

NOTE: The following graphic has been updated.

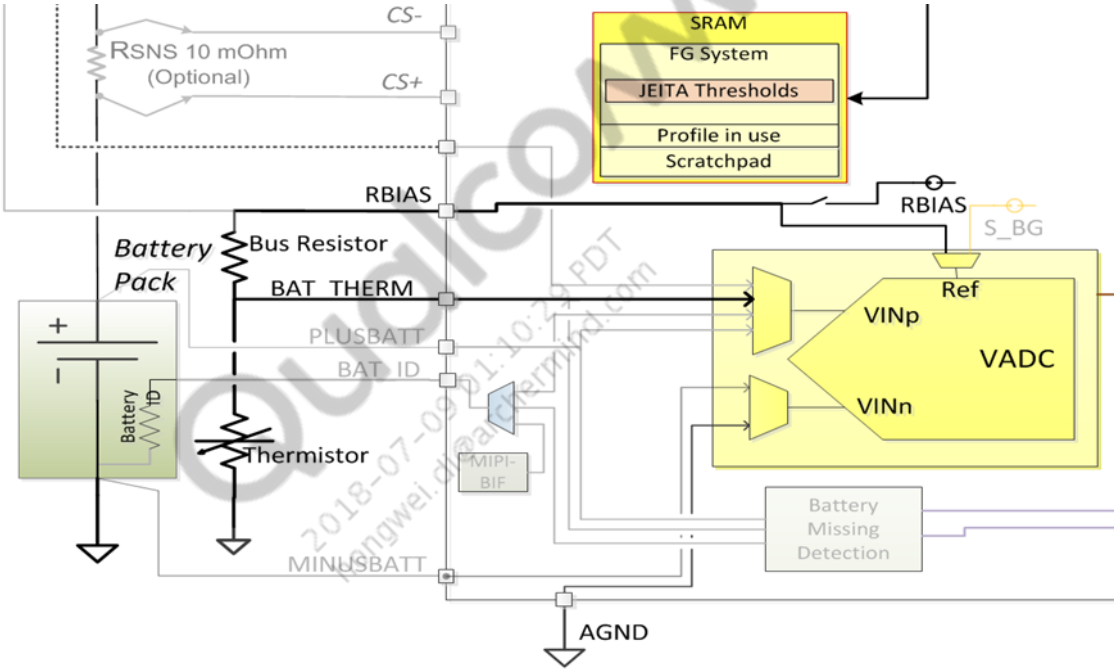


Figure 2-1 Temperature monitoring architecture

This configuration accurately measures the battery temperature. In more advanced battery packs, the thermistor is integrated into the battery case and connected externally through a dedicated pin. In this case, the THERM pin is intended to be connected to this specific battery pin to continue providing battery temperature readings. In addition, the THERM pin can be used the same as the BAT_ID pin for battery missing detection.

An internal comparator is connected to the THERM pin to detect a voltage drop in the event of a disconnected battery. The RBIAS pin provides biasing for the external net of a pull-up resistor and thermistor. The biasing on the RBIAS is handled automatically when there is a conversion request.

2.2.5 JEITA hard cold and hard hot hysteresis

NOTE: This section was added to this document revision.

JEITA hard cold and hard hot hysteresis is done entirely in the software to reduce the JEITA_HARD_HOT (COLD) limit upon a JEITA_HARD_HOT (COLD) event, and then to return the limit to the upper value upon entering JEITA_SOFT_HOT. Hysteresis requires new dtsi file properties. Hot and cold hystereses are disabled by default. Temperatures are in decidegrees C°.

qcom,cold-hot-jeita-hysteresis – A tuple of two as follows:

- Index[0] is cold hysteresis
- Index[1] is hot hysteresis

2.2.6 Temperature monitoring

2.2.6.1 Battery temperature sensing

The PMI899x FG monitors battery temperature through the dedicated THERM pin. If the battery pack is provided with an integrated thermistor, the THERM pin is to be connected to the related terminal on the battery case.

PMI899x uses the battery temperature information for two purposes:

- In accordance with JEITA requirements, it accommodates the switching charger operation to charge the battery in safe conditions.
- It improves accuracy in the SOC estimation with fine adjustment of the FG algorithm.

Any time the battery temperature is sampled, the BIAS pin is enabled and provides biasing for the voltage divider; including the pull-up resistor R2 and the battery's thermistor.

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

The voltage across the thermistor is sensed at the THERM pin and it is internally translated into real temperature information according to the known thermistor NTC beta. The NTC beta value is a sensitive parameter for proper operation of the FG and the CHARGER module. For PMI899x, this beta value is handled by the software and loaded to SRAM.

To configure the FG thermistor beta value, the beta coefficients must be set according to the master beta coefficients in *PMI8994 Fuel Gauge HW/SW Control* (80-VT310-123).

- qcom,thermal-coefficients – Byte array of thermal coefficients for reading battery thermistor. This array must be exactly 6 bytes long.

  Example

  Match the B value with the beta coefficients 80-VT310-123:

```
B    C1 hex    C2 hex    C3 hex
4050 85EC      4A75      35FC
Set in DTSI file as (list them in reverse HEX order):
qcom,thermal-coefficients = [ec 85 75 4a fc 35]
```

- SBL – Can manage this task, but adds 1.5 sec to boot time due to SRAM access. By default, this feature is not enabled.

```
..\boot_images\core\systemdrivers\pmic\config\msm8994\pm_config_target.c
```

  - Edit FgSramAddrDataEx_type fg_sram_data table to enable the writing to SRAM.

```
fg_sram_data[SBL_SRAM_CONFIG_SIZE] =
{
…
//Thermistor Beta Coefficents:
    { 0x0444, 0x86D8, 2, 2, PM_DISABLE_CONFIG }, //thermistor_c1_coeff:
default = 0x86D8;
    { 0x0448, 0x50F1, 0, 2, PM_DISABLE_CONFIG }, //thermistor_c2_coeff:
default = 0x50F1;
    { 0x0448, 0x3C11, 2, 2, PM_DISABLE_CONFIG }  //thermistor_c3_coeff:
default = 0x3C11;
};
```

2.2.7 Soft thermal monitor (JEITA)

The FG and CHARGER modules include a soft thermal monitor for compliance with JIS8714 and JEITA safety requirements. The battery temperature is used to modulate battery charging voltage and current when the temperature is in between programmable ranges (Figure 2-2).

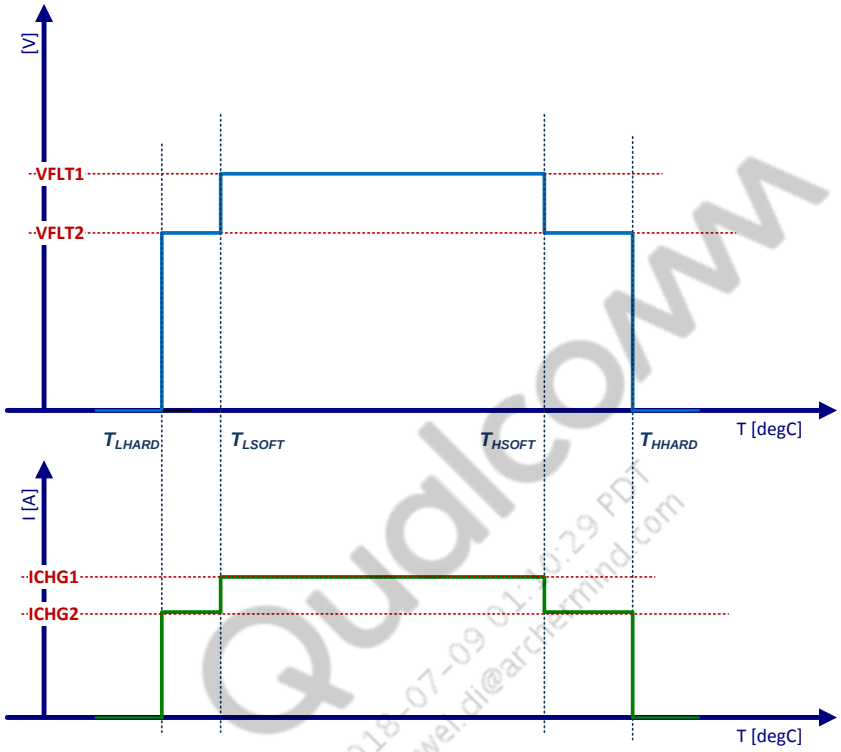


Figure 2-2 JEITA

The FG module has dedicated registers for JEITA_soft_hot and JEITA_soft_cold thresholds. When the battery temperature is inside the range of the JEITA_soft_hot and JEITA_soft_cold thresholds, the battery can be charged using the default charging current ICHG and floating voltage VFLT. Refer to *MSM8994.LA Charger Software* (80-NM328-56).

The respective charging current and floating voltage are modulated to ICHG1 and VFLT1 when the battery temperature meets the following conditions:

- Exceeds either the JEITA_soft_hot and JEITA_soft_cold
- Does not exceed the JEITA_hard_hot and JEITA_hard_cold *hard* limits

Refer to *MSM8994.LA Charger Software* (80-NM328-56) for information on how to program ICHG1 and VFLT1.

- The HLOS software setting for these parameters are available in the dtsi file as follows:

File	Temperature in decidegrees C°
`qcom,warm-bat-decidegc`	Warm battery
`qcom,cool-bat-decidegc`	Cool battery
`qcom,hot-bat-decidegc`	Hot battery
`qcom,cold-bat-decidegc`	Cold battery

- JEITA_soft_cold and JEITA_soft_hot are also accessible (read/write) through the power supply framework as:
  - POWER_SUPPLY_PROP_WARM_TEMP
  - POWER_SUPPLY_PROP_COOL_TEMP

SBL software can also be used to set up this data by set editing the following file:

- ...\boot_images\core\systemdrivers\pmic\config\msm8994\pm_config_target.c

```
FgSramAddrDataEx_type fg_sram_data[SBL_SRAM_CONFIG_SIZE] =
{   //JEITA Thresholds:
    //SramAddr,  SramData, DataOffset, DataSize, EnableConfig
    { 0x0454, 0x23,  0,  1, PM_DISABLE_CONFIG }, //JEITA Soft Cold
Threshold:  default = 0x23
    { 0x0454, 0x46, 1,  1, PM_DISABLE_CONFIG }, //JEITA Soft Hot
Threshold:  default = 0x46
    { 0x0454,   0x1E, 2,  1, PM_DISABLE_CONFIG }, //JEITA Hard Cold
Threshold:  default = 0x1E
    { 0x0454, 0x48, 3, 1,  PM_DISABLE_CONFIG }, //JEITA hard Hot
Threshold:  default = 0x48
….
}
```

NOTE: Hexadecimal data in this case is the temperature offset added to 243 Kelvin as follows:

- 0x00 = 243K = 243 K
- 0x1E = 243K + 30 = 273 K
- 0x7F=243K + 128 = 371 K LSB = 1 K

2.2.8 Battery ESR estimation

The value of the equivalent series resistance (ESR) varies widely by the following factors:

- Temperature
- Battery residual capacity, that is, SOC
- Battery model

To guarantee an accurate SOC estimation, the algorithm must rely on a real estimate of the actual ESR. The FG monitors the ESR variation by sampling valid synchronous readings of the battery voltage and current. The data collected is post-processed to achieve the best estimation of the actual ESR. These pulses are generated every 90 sec and the entire procedure is run exceptionally, not actually affecting the battery lifetime.

The FG algorithm also issues an ESR estimating current pulse each time the battery temperature changes by at least ±6°C and in the absence of valid readings. This procedure is useful because the ESR strongly varies with the cell's temperature.

Software read access to this data is through the power supply framework POWER_SUPPLY_PROP_RESISTANCE.

The FG hardware issues ESR pulses to help measure the real resistance of the battery. However, this activity can raise the rock bottom current draw of the device. Because ESR measurements are unnecessary in low current conditions, the software disables them when going into suspend.

2.2.9 System termination current

The current has the same encoding of the ADC when converting the battery current.

NOTE: This value must be negative (system charging).

The configurable device tree data (dtsi) file is used if the customer desires to change behavior when 100% is reported vs. when charging actually stops (i.e., the EOC set by `qcom,iterm-ma`). This parameter allows 100% of the SOC to be displayed when a specific termination current threshold (system termination current) higher than the real termination current (charger termination current) is reached.

```
qcom,fg-iterm-ma = <150>; /* ex: 150mA */
```

shows an example of a full SOC.

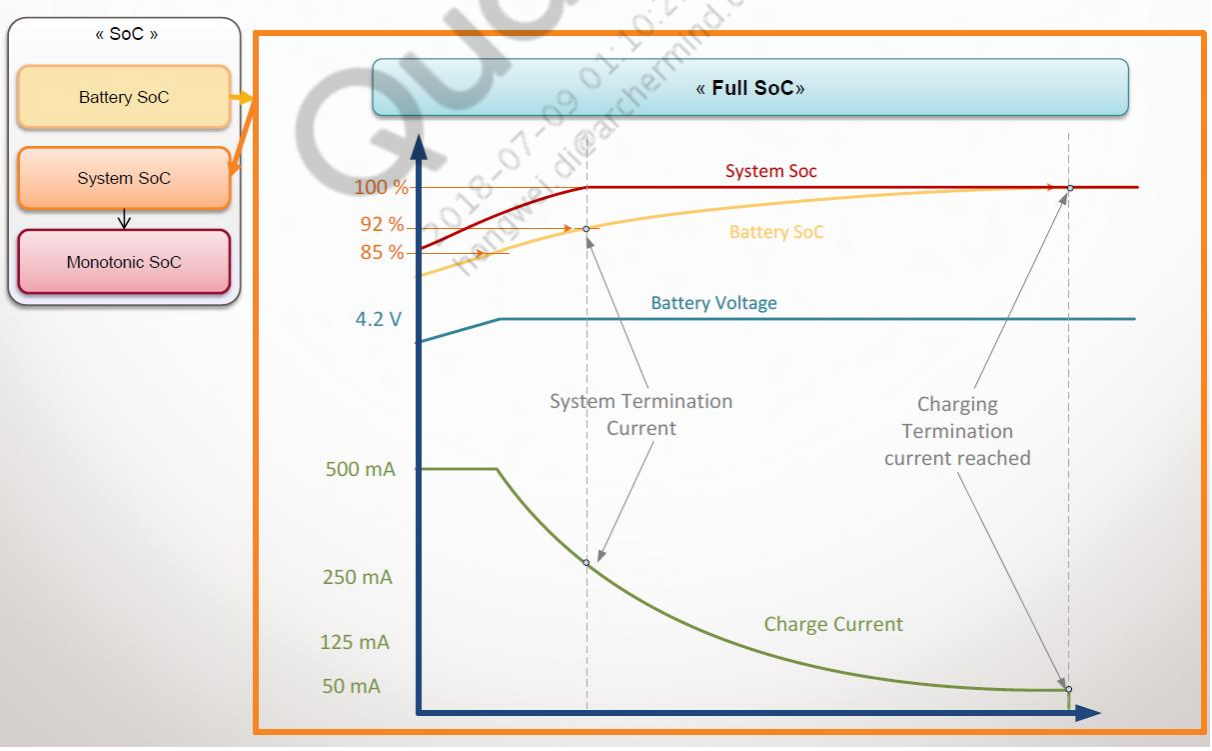


Figure 2-3 System full SOC example

2.2.10 Fuel gauge termination current

Fuel gauge termination current is used to terminate charging when the source of the detection is the FG ADC (not the analog charger terminate current done previously).

If the charger is using the fuel gauge ADC current to determine End of Charge (EOC), the termination current must be configured in the fuel gauge.

Allow this register to be configured via the device tree property:

```
qcom,fg-chg-iterm-ma = <100> ;  // e.g. 100mA to terminate
```

2.2.11 Battery profile termination current

The configurable device tree data in the battery data profile data is currently not being used. The setting the same as `qcom,iterm-ma`. (if the customer chooses EOC by the analog charger sensor termination current).

```
qcom,chg-term-ua = <100000>; /*ex: 100mA */
```

2.2.12 System cutoff voltage

System cutoff voltage is the set point that affects the estimate of the 0% SOC estimate (3.0 V to 3.8 V).

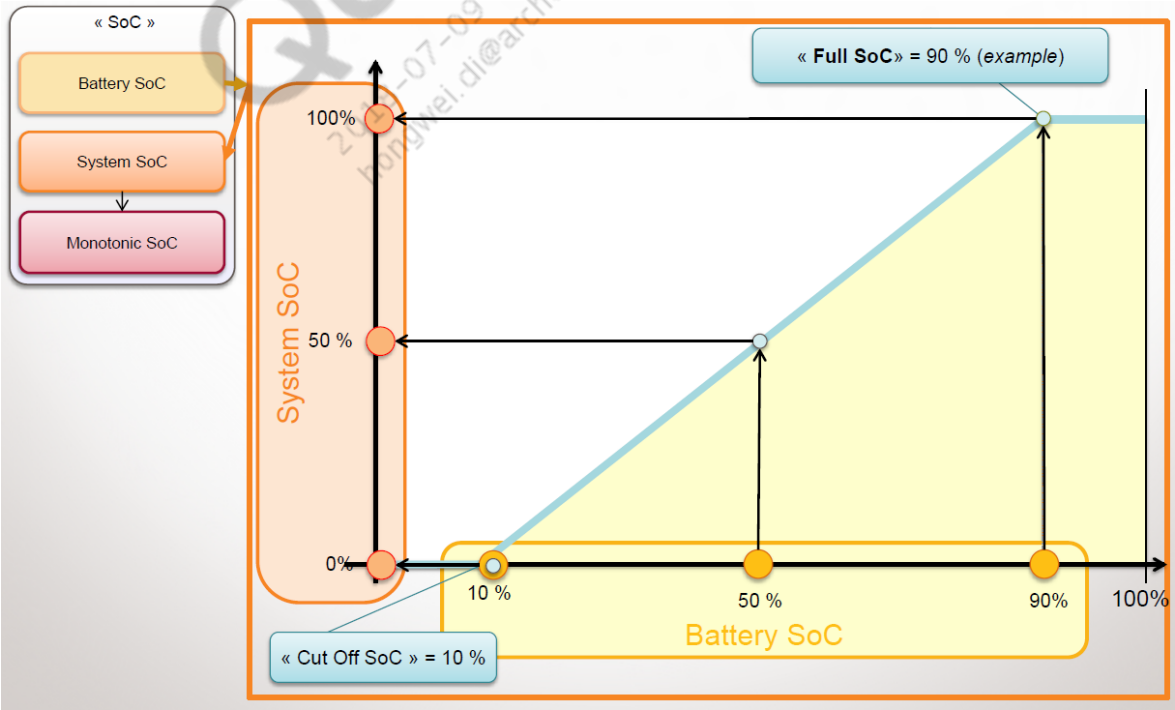


Figure 2-4 System full and cutoff SOC example

The configuration for this data is in the .dtsi file.

```
qcom,fg-cutoff-voltage-mv = <3000>; /* example 3V */
```

This configuration must be set higher (default higher setting is 100 mV) or equal to the empty SOC voltage (IRQ_volt_empty).

2.2.13 Conditional FG restart at boot up by estimated battery voltage

NOTE: Numerous changes were made in this section.

If the predicted voltage based on the SOC and battery current or resistance at boot up differs from the actual voltage by more than the defined threshold amount, the FG re-estimates the first SOC when the FG driver probes.

The configurable threshold is a parameter in the device tree file (dtsi):

```
qcom,vbat-estimate-diff-mv = <30>; /* 30mV default setting */
```

2.2.14 CC_to_CV threshold set point

The constant charge to constant voltage (CC_to_CV) voltage point is used as a possible source other than the charger for when to detect entering into Taper/CV mode. When the float voltage adjustment algorithm is active, the CC_CV set point requires to be set closer to the float voltage. A configurable device tree data (dtsi) file is:

```
qcom,fg-cc-cv-threshold-mv = <4340>;
```

Specify the CC_CV set point for FG to 4340 mV (currently default) which is 10 mV less than the float voltage (currently default to be 4350 mV) configured. This setting is required to properly notify an EOC.

NOTE: Only use this property if the qcom,autoadjust-vfloat property is used in the Charger driver.

2.2.15 Resume charging based on SOC

Feature TBD pending future release.

Resume charging based on the SOC can be configured for automatic recharge in the FG. This configuration, along with the recharge threshold source configuration in the charger, recharges based on the FG. Resume the SOC to resume the charge set in the percentage unit in the dtsi file:

```
qcom,resume-soc = <95>;
```

2.2.16 Standby current

Feature TBD pending future release.

Standby current is an additional parameter used for the cutoff SOC (0%) which is estimated during low current discharge. This parameter provides a current minimum that the FG hardware uses to calculate the SOC. This value derates the SOC and makes sure the device has enough capacity at a low %SOC to run in Active mode for a short period before shutting down.

```
qcom,fg-ibatt-standby-ma = <TBD>;
```

2.2.17 External/internal current sense

The running parameter is required for parallel charging. The switch must occur before parallel charging is turned on. The DTSI setting for this feature is as follows:

```
qcom,ext-sense-type;
```

Using this DTSI setting enables external sense on the current sense channel used by the FG.

2.2.18 IRQ_volt_empty

This IRQ notifies when the Monotonic SOC = 0% (low battery voltage threshold for the empty battery interrupt) and has the following properties:

- Range – 3.0 to system cutoff voltage
- Recommended setting/default = System cutoff voltage – 50 mV (might need more headroom than 50 mV value, depending on system)
- Software forces a 0% upon an empty interrupt; user space is notified via the power supply framework. The user space reads 0% SOC and immediately shutdown to prevent UVLO.
- Configurable as a device tree data

```
qcom,irq-volt-empty-mv
```

2.2.19 Programmable delay between RBIAS and BAT_THERM

NOTE: This section was added to this document revision.

If capacitance is placed at the BAT_THERM node, the BAT_THERM voltage might not have enough time to settle to its final value before the ADC conversion begins. To resolve this issue, a programmable delay is placed chronologically between the application of V_rbias and the onset of the temperature ADC conversion.

The delay is configurable in the device tree file settings. The default value is 0 microseconds.

```
qcom,fg-therm-delay-us
```

2.2.20 USB ID detection

The FG module detects the value and reports it to the charger module through shadow registers using internal signals. The qpnp-smbcharger software reads through the converted USB ID value from the charger module registers for OTG device detection.

- 0x130F – SMBCHG_USB_CHGPTH_USBID_VALID_ID_7_0
- 0x130E – SMBCHG_USB_CHGPTH_USBID_VALID_ID_11_8

2.2.21 Battery age detection

NOTE: Numerous changes were made in this section.

The Battery Learning Capacity algorithm discovers battery capacity via Coulomb counting.

NOTE: The battery age algorithms are QTI intellectual property; details are not fully disclosed. The open source code is in the kernel driver and is subject to change at any time.The Battery Learning Capacity algorithm uses the following information:

- Temperature
- Qualified starting point of the SOC of the battery
- Allowable increment and decrement of each charge cycle to qualify for each learning cycle

2.2.22 Device tree parameter settings for battery age detection

Table 2-1 Battery age detection settings

Parameter	Description
qcom,capacity-learning-on	Boolean property to have the fuel gauge driver attempt to learn the battery capacity when charging using coulomb counting; takes precedence over qcom,capacity-estimation-on
qcom,cl-max-increment-deciperc	Maximum percentage that the capacity can rise as the result of a single charge cycle; this property corresponds to 0.1% increments
qcom,cl-max-decrement-deciperc	Maximum percentage that the capacity can fall as the result of a single charge cycle; this property corresponds to 0.1% decrements
qcom,cl-max-temp-decidegc	Above this temperature, capacity learning is canceled
qcom,cl-mix-temp-decidegc	Below this temperature, capacity learning is canceled
qcom,cl-max-start-capacity	Battery SOC must be below this value at the start of a charge cycle for capacity learning to be run
qcom,capacity-estimation-on	Boolean property that makes the fuel gauge driver attempt to estimate battery capacity using the battery resistance algorithm for equivalent series resistance (ESR)
qcom,aging-eval-current-ma	Current used to evaluate battery aging; this value must be near the steady state current drawn from the battery when the phone is low on battery **Note**: This field is only used if the qcom,capacity-estimation-on field is used

2.2.23 Charge cycle counter

NOTE: Numerous changes were made in this section.

The charge cycle counters provide the number of times the battery is charged, and are separated by *bucket* or charging cycle ranges.

There are eight buckets allocated for the battery SOC (0-100%). For example, bucket ID 1 corresponds to charge cycle from 0 up to 12.5%, ID 2 is 12.5 up to 25%, and so on.

The bucket count is defined as follows:

```
#define BUCKET_COUNT    8
```

The bucket count is defined as follows:

```
#define BUCKET_SOC_PCT    (256 / BUCKET_COUNT)
```

The bucket design enables increments the cycle counters based on multiple charging buckets. This enables studying the charging pattern, because the user charges the battery at random levels.

- The cycle counter is exposed through the cycle_count property,

  POWER_SUPPLY_PROP_CYCLE_COUNT
- The cycle count corresponds to the POWER_SUPPLY_PROP_CYCLE_COUNT_ID
- Device tree parameter settings for the charge cycle counter are as follows:

Device tree parameter settings	Description
qcom,cycle-counter-en	Boolean property that enables the cycle counter feature; if this property is present, define the other properties in this table to specify low and high SOC thresholds

3 Fuel gauge driver

3.1 Location

- Driver source code – `kernel/drivers/power/qpnp-fg.c`
- Device tree configuration – `kernel/arch/arm/boot/dts/qcom/msm-pmi8994.dtsi`
- Document – `kernel/Documentation/devicetree/bindings/power/qpnp-fg.txt`

3.2 Configuration example

```
pmi8994_fg: qcom,fg {
     spmi-dev-container;
     compatible = "qcom,qpnp-fg";
     #address-cells = <1>;
     #size-cells = <1>;
     qcom,resume-soc = <95>;
     status = "okay";
     qcom,bcl-lm-threshold-ma = <127>;
     qcom,bcl-mh-threshold-ma = <405>;
     qcom,fg-iterm-ma = <125>;
     qcom,fg-chg-iterm-ma = <100>;
     qcom,cycle-counter-en;
     qcom,capacity-learning-on;
     qcom,fg-cc-cv-threshold-mv = <4340>;
     qcom,pmic-revid = <&pmi8994_revid>;

     qcom,fg-soc@4000 {
          status = "okay";
          reg = <0x4000 0x100>;
          interrupts =   <0x2 0x40 0x0>,
                    <0x2 0x40 0x1>,
                    <0x2 0x40 0x2>,
                    <0x2 0x40 0x3>,
                    <0x2 0x40 0x4>,
                    <0x2 0x40 0x5>,
                    <0x2 0x40 0x6>,
                    <0x2 0x40 0x7>;
```

```
        interrupt-names =  "high-soc",
                           "low-soc",
                           "full-soc",
                           "empty-soc",
                           "delta-soc",
                           "first-est-done",
                           "sw-fallbk-ocv",
                           "sw-fallbk-new-battrt-sts",
                           "fg-soc-irq-count";
    };

    qcom,fg-batt@4100 {
        reg = <0x4100 0x100>;
        interrupts =  <0x2 0x41 0x0>,
                      <0x2 0x41 0x1>,
                      <0x2 0x41 0x2>,
                      <0x2 0x41 0x3>,
                      <0x2 0x41 0x4>,
                      <0x2 0x41 0x5>,
                      <0x2 0x41 0x6>,
                      <0x2 0x41 0x7>;

        interrupt-names =  "soft-cold",
                           "soft-hot",
                           "vbatt-low",
                           "batt-ided",
                           "batt-id-req",
                           "batt-unknown",
                           "batt-missing",
                           "batt-match";
    };

    qcom,fg-adc-vbat@4254 {
        reg = <0x4254 0x1>;
    };

    qcom,fg-adc-ibat@4255 {
        reg = <0x4255 0x1>;
    };

    qcom,revid-tp-rev@1f1 {
        reg = <0x1f1 0x1>;
    };
```

```
            qcom,fg-memif@4400 {
                 status = "okay";
                 reg = <0x4400 0x100>;
                 interrupts =  <0x2 0x44 0x0>,
                           <0x2 0x44 0x1>;

                 interrupt-names =  "mem-avail",
                               "data-rcvry-sug";
            };
        };
    };
```

3.3 Interrupts

Interrupt	Peripheral	Description
JEITA_SOFT_COLD_RT_ STS	FG_BATT	Interrupt to notify that the battery temperature < JEITA_soft_cold threshold; JEITA_SOFT_HOT < threshold < 288.15 K
JEITA_SOFT_HOT_RT_STS	FG_BATT	Interrupt to notify that the battery temperature > JEITA_soft_hot threshold; 253.15 K < threshold < JEITA_SOFT_COLD
VBATT_LOW_RT_STS	FG_BATT	Interrupt to notify that the battery voltage < IRQ_volt_min threshold, digital comparison on the ADC value
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 must identify software 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
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 Monotonic SOC > High SOC Threshold
LOW_SOC_RT_STS	FG_SOC	IRQ to notify that Monotonic SOC < Low SOC Threshold
FULL_SOC_RT_STS	FG_SOC	IRQ to notify that Monotonic SOC = 100%
EMPTY_SOC_RT_STS	FG_SOC	IRQ to notify that Monotonic SOC = 0%
DELTA_SOC_RT_STS	FG_SOC	IRQ to notify that Monotonic SOC change exceeded the specified delta SOC threshold; the reasonable range is 0.039% to 12.55%
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 = software allowed to access FG memory
DATA_RCVRY_SUG_RT_ STS	FG_MEMIF	1 = software allowed to update FG RAM contents to assist in boot
VBAT_LT_THR_INT_RT_ STS	FG_ADC_ USR	IRQ indicating that VBAT < threshold defined in VBAT_INT__THR register
VBAT_LT_THR_INT_RT_ STSFG_ADC	FG_ADC_ MDM	IRQ indicating that VBAT < threshold defined in VBAT_INT__THR register
IBAT_GT_THR_RT_STS	FG_ADC_ USR	IRQ indicating that IBAT > threshold defined in IBAT_INT__THR register
IBAT_GT_THR_RT_STS	FG_ADC_ USR	IRQ indicating that IBAT > threshold defined in IBAT_INT__THR register

3.4 Power supply interfaces

NOTE: Numerous changes were made in this section.

```
POWER_SUPPLY_PROP_CAPACITY        /* MONOTONIC SOC of battery read only*/
POWER_SUPPLY_PROP_CAPACITY_RAW    /* SOC of BATTERY read only*/
POWER_SUPPLY_PROP_CURRENT_NOW     /* Data CURRENT read only */
POWER_SUPPLY_PROP_VOLTAGE_NOW     /* Data VOLTAGE read only */
POWER_SUPPLY_PROP_VOLTAGE_OCV     /* Data OCV read only */
POWER_SUPPLY_PROP_VOLTAGE_MAX_DESIGN /* Max rated voltage of the battery */
POWER_SUPPLY_PROP_CHARGE_NOW  /* Battery SOC Coulomb Counter */
POWER_SUPPLY_PROP_CHARGE_NOW_RAW /* SOC Coulomb Count Raw */
POWER_SUPPLY_PROP_CHARGE_NOW_ERROR /* SOC Correction applied from Voltage Mode */
POWER_SUPPLY_PROP_CHARGE_FULL     /* Real time (aged) battery capacity */
POWER_SUPPLY_PROP_CHARGE_FULL_DESIGN /* Nominal battery capacity */
POWER_SUPPLY_PROP_TEMP            /* Current TEMP read only */
POWER_SUPPLY_PROP_COOL_TEMP       /* SOFT COLD read and write */
POWER_SUPPLY_PROP_WARM_TEMP       /* SOFT HOT read and write */
POWER_SUPPLY_PROP_RESISTANCE /* BATT ESR read only */
POWER_SUPPLY_PROP_RESISTANCE_ID /* BATT ID read only */
POWER_SUPPLY_PROP_BATTERY_TYPE    /* from qcom,battery-type */

POWER_SUPPLY_PROP_UPDATE_NOW /* force data nodes updating (write only)*/
POWER_SUPPLY_PROP_ESR_COUNT       /* Counter of last update of Battery ESR */
POWER_SUPPLY_PROP_VOLTAGE_MIN   /* VBATT Low status (below IRQ_Volt_Min)*/
POWER_SUPPLY_PROP_CYCLE_COUNT     /* Cycle counter of the bucket ID */
POWER_SUPPLY_PROP_CYCLE_COUNT_ID /*Bucket id of battery charging cycles */
```

3.5 Battery profile

A battery profile includes all the information necessary to allow the FG to have the best possible SOC estimate. From the software design point, the customer has two options to choose for the battery profile setting listed here. The second option is the normal design option, and the first one is for reference only.

Option 1 – Force one profile

- For testing or customer with only one intended profile
- In device tree, specify only one battery profile:
  - `qcom,use-otp-profile` – Specify this flag to avoid RAM loading any battery profile
  - Remove `qcom,batt-id-range-pct`

Option 2 – Battery ID-based software loading

- Specify `qcom,batt-id-range-pct` in battery data node (DTSI)
- `#include` other battery profile .dtsi files (see Example – Battery ID-based software loading)

The selected profile loads into SRAM at bootup and overwrites the generic profile.

NOTE: Submit all intended batteries to QTI for battery characterization.

Example – Battery ID-based software loading

The following is a sample battery profile for option 2.

```
qcom,itech-3000mah {
   /* #Itech_B00826LF_3000mAh_Feb24th_Averaged*/
   qcom,max-voltage-uv = <4350000>;
   qcom,v-cutoff-uv = <3400000>;
   qcom,chg-term-ua = <100000>;
   qcom,batt-id-kohm = <100>;
   qcom,battery-type = "itech_3000mah";
   qcom,chg-rslow-comp-c1 = <4365000>;
   qcom,chg-rslow-comp-c2 = <8609000>;
   qcom,chg-rslow-comp-thr = <0xBE>;
   qcom,chg-rs-to-rslow = <761000>;
   qcom,fastchg-current-ma = <2000>;
   qcom,fg-cc-cv-threshold-mv = <4340>;
   qcom,checksum = <0x0B7C>;     /* checksum for fg-profile-data only*/
   qcom,fg-profile-data = [
        F0 83 6B 7D
        66 81 EC 77
        … … … …;
};
```


NOTE: `qcom,fg-profile-data` is internal propriety data (context meanings unexposed to customer) which is loaded into SRAM by the software for the FG algorithm.

3.6 SRAM

Most of the FG registers are accessible using the indirect addressing controls located in the fg_memif peripheral. The following table is an SRAM partitioning map from *PMI8994 Fuel Gauge HW/SW Control* (80-VT310-123).

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

3.7 Debug FG

3.7.1 FG MEM_INTF access

The following commands show how to read FG MEM_INTF access.

```
adb shell "echo 0xXXX > /sys/kernel/debug/fg_memif/address"
adb shell "echo 0xXX > /sys/kernel/debug/fg_memif/count"
adb shell "cat /sys/kernel/debug/fg_memif/data"
```

The following commands show how to write FG MEM_INTF access.

```
adb shell "echo 0xXXX > /sys/kernel/debug/fg_memif/address"
adb shell "echo 0xXX > /sys/kernel/debug/fg_memif/count"
adb shell "echo 0xXX > /sys/kernel/debug/fg_memif/data"
```

NOTE: These commands must write 4 bytes at a time, or FG does not update the SRAM.

3.7.2 FG SRAM dump

To read the FG data periodically:

1. Obtain the latest sample script (`dumper.sh`) from your local PMIC CE support team or see *Fuel Gauge Data Collector* (80-NU716-1 A).
2. Once the sample script is obtained, using ADB, push the script onto the test device.

```
adb root
adb wait-for-devices
adb push \\<your fg_dump script folder>\dumper.sh /data/
adb shell chmod 777 /data/dumper.sh
```

3. Run the following command for the serial window or via ADB shell.

```
/data/dumper.sh > /data/kmsg.txt &
```

NOTE: Insert a space before the & character in the command.

4. After the test completes, run the following commands.

```
adb root
adb wait-for-devices
adb pull /data/kmsg.txt
```

3.7.3 FG debug logging

NOTE: Numerous changes were made in this section.

Use the debug_mask module parameter at compile time to enable FG debug logging.

For a definition of the debug flag for each bit, refer to the following:

https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/tree/drivers/power/qpnp-fg.c?h=msm-3.10#n83

To enable debug logging for each bit and specify a debug log file, issue the following commands at runtime:

```
echo 0xff > /sys/module/qpnp_fg/parameters/debug_mask
echo 8 > /proc/sys/kernel/printk

dmesg > debug_log_filename
```

NOTE: In the first command, `0xff` turns on debug logging for all eight bits.

A References

A.1 Related documents

Documents	
Qualcomm Technologies, Inc.	
PM8994/PM8996 and PMI8994/PMI8996 Power Management ICS Design Guidelines	80-NJ117-5
PMI8994 Power Management IC Device Specification (Advance Information)	80-NJ118-1
MSM8994.LA Charger Software	80-NM328-56
PMI8994 Fuel Gauge HW/SW Control	80-VT310-123

A.2 Acronyms and terms

Term	Definition
ADC	Analog-to-digital converter
BSI	Battery serial interface
CC_to_CV	Constant charge to constant voltage
DTSI	Linux device tree include file
EOC	End of charge
ESR	Equivalent series resistance
FG	Fuel gauge
LA	Linux Android
SBL	Secondary boot loader
SOC	State of charge
SRAM	Static random access memory