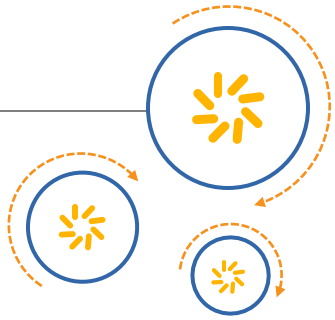
Qualcomm Technologies, Inc.



Linux Android PMIC Fuel Gauge User Guide

80-NV610-41 C

December 22, 2015

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

Revision history

Revision	Date	Description
A	April 2015	Initial release
B	July 2015	Updated Sections 2.2.7, 2.2.8, 2.2.9, 2.2.10, 3.2, 3.3, 3.5, and 3.7.3; removed USB-ID detection with FG ADC; updated parameters
C	December 2015	Updated the applicable chipsets in Section 1.1.

Contents

Figures

1 Introduction

1.1 Purpose

This document describes the programmable features for the Fuel Gauge (FG) and the software driver configuration for the PMI895x. It is intended for customers using the PMI895x chip. To properly configure the charging parameters in the device tree file, users must have a complete understanding of the hardware specifications; see *PMI8952 Power Management IC Design Guidelines* (80-NT391-3).

This document is also applicable to MSM8952, MSM8956/MSM8976, and MSM8937 chipsets.

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 FG module offers a hardware implementation of a new algorithm that can accurately estimate the battery state-of-charge (SoC) by mixing current monitoring with the voltage-based technique. This ensures both excellent short-term linearity and long-term high accuracy. Furthermore, full battery cycling or zero current load conditions are not required to maintain the accuracy.

When precise measurements of voltage, current, and temperature are used, an accurate SoC 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 SoC estimate throughout the battery life.

The FG allows for measuring the battery pack temperature. This is achieved through an external thermistor. Detection of a missing battery is also incorporated to monitor for battery insertion and removal situations, while properly updating 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 of time
- Disable USB on-the-go (OTG) functionality when battery SoC falls below a programmable threshold

NOTE: Use this document concurrently with the *PMI8994 Fuel Gauge HW/SW Control Application Note* (80-VT310-123) as it focuses on software aspects and support for the PMI895x FG solution.

2.2 Functional description

The FG monitor is integrated 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 the following:

- Both Coulomb counting and voltage-based techniques
- Ensuring short-term linearity
- 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 then used in the algorithm to detect the condition of the battery being used and to provide an estimation of the residual capacity, according to both the calculation of the total charge present in the battery (Coulomb count) and its variation with respect to the return 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 can be accessed 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 can be accessed 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 pulled up internally.

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 smart battery should be identified by the software (Battery Serial Interface (BSI) module).

The BAT_ID can be accessed 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. As a default option, the RBIAS and THERM pins are to be connected as shown in Figure 2-1, with a thermistor placed externally and in close proximity to the battery pack.

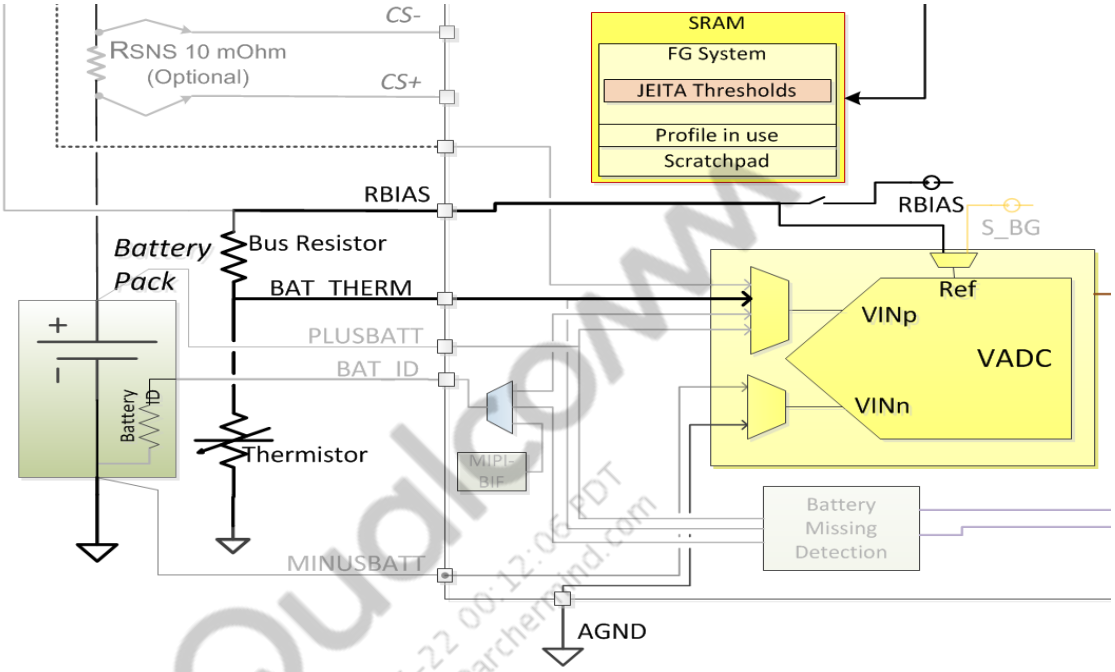


Figure 2-1 Temperature monitoring architecture

This configuration accurately measures the battery temperature. For 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 operated the same way 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 Temperature monitoring

2.2.5.1 Battery temperature sensing

The PMI895x FG monitors battery temperature through the dedicated THERM pin. If the battery pack is provided with an integrated thermistor, the THERM pin is connected to the related terminal on the battery case. Otherwise, it is connected to a thermistor placed on the PCB, possibly in close proximity to the battery.

The information about the battery temperature is used by the PMI895x for the following 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 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 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 PMI895x, this beta value is handled by the software and loaded to static random access memory (SRAM).

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

For HLOS, the software setting is in the form of a kernel device tree parameter:

- qcom,thermal-coefficients – Byte array of thermal coefficients for reading battery thermistor. This should be exactly 6 bytes in length:

```
Example:
Match the B value with the beta ccoefficients as quoted from the
above document:
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]
```

- Secondary Boot Loader (SBL) can also handle this task, however, this adds 1.5 sec to boot time due to SRAM access. By default, this feature is not enabled.

  ...\boot_images\core\systemdrivers\pmic\config\msm895x\pm_config_target.c

- Edit the 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 }, //thremistor_c1_coeff:  default = 0x86D8;
    { 0x0448,     0x50F1,   0,          2,
PM_DISABLE_CONFIG }, //thremistor_c2_coeff:  default = 0x50F1;
    { 0x0448,     0x3C11,   2,          2,          PM_DISABLE_CONFIG }
//thremistor_c3_coeff:  default = 0x3C11;
};
```

2.2.5.2 Soft thermal monitor (JEITA)

The FG and CHARGER modules are provided with a soft thermal monitor designed to be compliant with the latest JIS8714 and JEITA standard safety requirements. The battery temperature information is used to modulate the battery charging voltage and current when the temperature is in between programmable ranges (see Figure 2-2).

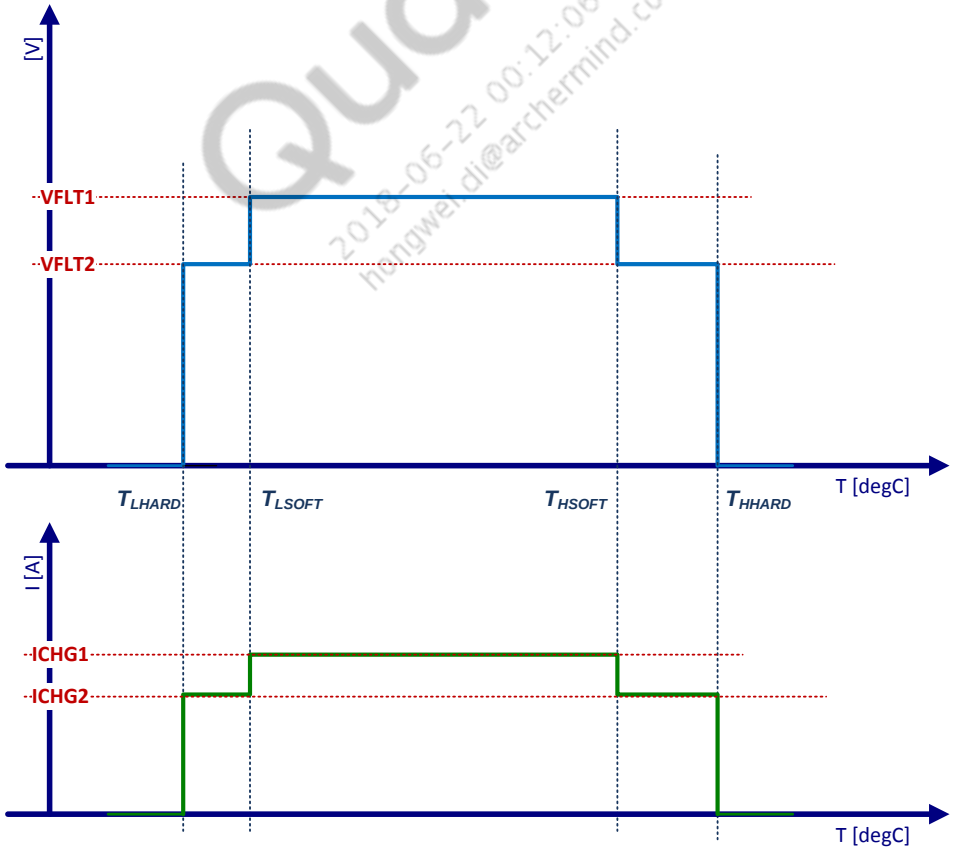


Figure 2-2 JEITA

Together with the JEITA_hard_hot and JEITA_hard_cold thresholds, the FG module is also provided with dedicated registers for JEITA_soft_hot and JEITA_soft_cold thresholds respectively. Any time the battery temperature is inside the range between the JEITA_soft_hot and JEITA_soft_cold thresholds, the battery is allowed to be charged with default charging current ICHG and floating voltage VFLT, as described in the *MSM8952 Linux Android Charger Software User Guide* (80-NV610-44).

If at any time the battery 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 ICHG1 and VFLT1. Both ICHG1 and VFLT1 are user-programmable (see the *MSM8952 Linux Android Charger Software User Guide* (80-NV610-44)).

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

File	Description
qcom,warm-bat-decidegc	Warm battery temperature in decidegC
qcom,cool-bat-decidegc	Cool battery temperature in decidegC
qcom,hot-bat-decidegc	Hot battery temperature in decidegC
qcom,cold-bat-decidegc	Cold battery temperature in decidegC

- JEITA_soft_cold and JEITA_soft_hot are also accessible (read/write) through the power supply framework as follows:
  - POWER_SUPPLY_PROP_WARM_TEMP
  - POWER_SUPPLY_PROP_COOL_TEMP
- Use SBL software to set up this data by editing the following file:
  - ...\boot_images\core\systemdrivers\pmic\config\msm895x\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: Hex data here is temperature offset added to 243 Kelvin. Therefore, 0x00 = 243 K = 243 K; 0x1E = 243 K + 30 = 273 K; 0x7F = 243 K + 128 = 371 K LSB = 1 K.

2.2.6 Battery Equivalent Series Resistance (ESR) estimation

The value of ESR shows strong variation together with temperature, and it is also influenced by the battery residual capacity (SoC) and the battery model. To guarantee an accurate SoC estimation, the algorithm must rely on a real estimate of the actual ESR. The PMI895x FG monitors the ESR variation by sampling valid synchronous readings of the battery voltage and current. The data collected is postprocessed 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 justified because the ESR strongly varies with the cell 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 can raise the rock bottom current draw of the device. Because the ESR measurements are unnecessary in low current conditions, the software currently disables them when going into suspend.

2.2.7 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 file (.dtsi) is used if the customer wants to change behavior when 100% is reported vs. when charging actually stops (known as end of charge (EoC) set by qcom,fg-chg-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 */
```

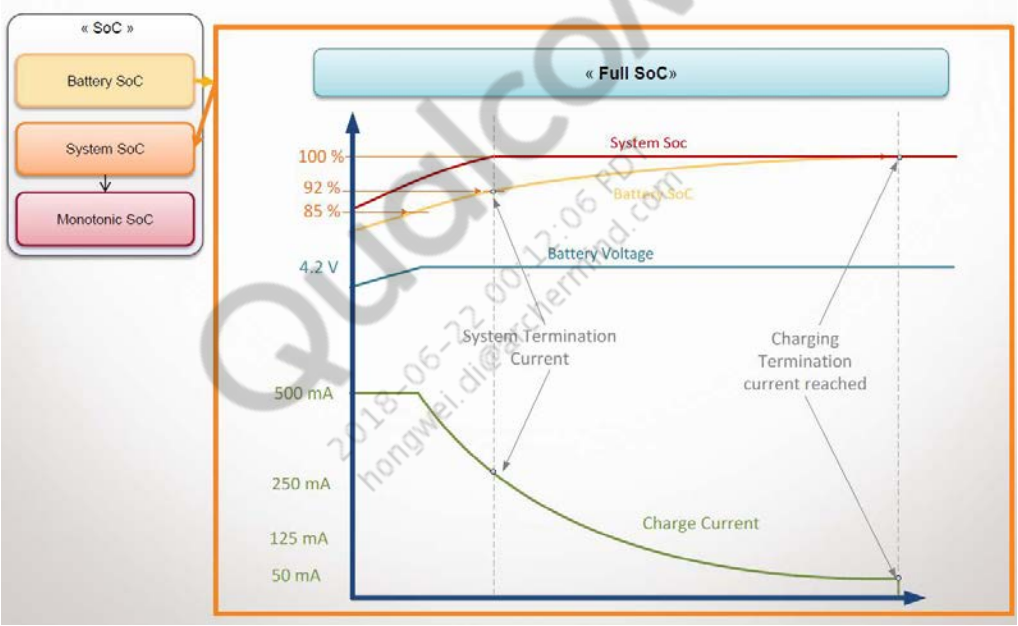


Figure 2-3 System full SoC example

2.2.8 FG termination current

This feature is used to terminate charging when the source of the detection is the FG ADC (not the analog charger terminate current). If the charger is using the FG ADC current to determine end of charge, configure the termination current in the FG.

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

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

2.2.9 Battery profile terminate current

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

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

2.2.10 System cutoff voltage

This is the set point that affects the estimate of the 0% SoC estimate (3.0 V to 3.8 V).

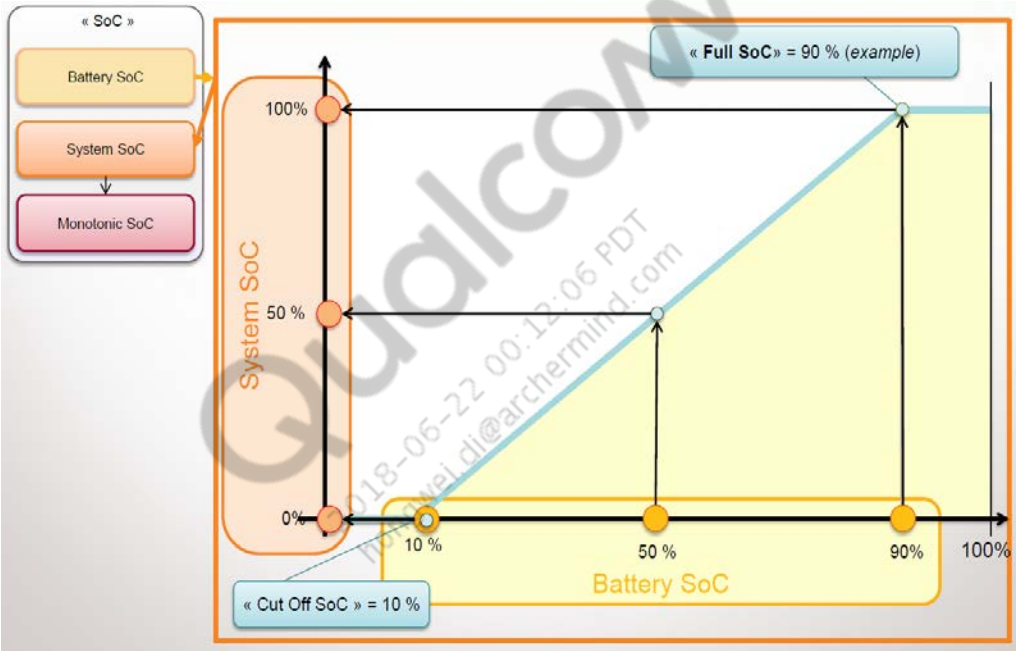


Figure 2-4 System full and cutoff SoC example

The configuration for this data is in the .dtsi file as follows:

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

This configuration must always be set higher or equal to the empty SoC voltage (IRQ_volt_empty).

2.2.11 Estimate voltage at bootup

If the estimated voltage based on SoC and battery current or resistance differs from the actual voltage by more than this amount, the FG redoes the first SoC estimate when the driver probes.

Reload the profile for a better SoC estimate if the vbat_current_predicted differs from the actual vbat by more than a set threshold.

The configurable parameter in the device tree file (.dtsi) is, for example:

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

2.2.12 CC_to_CV threshold set point

The CC_to_CV voltage point is used as a possible source other than the charger for when to detect entering into Taper or CV mode. When the float voltage adjustment algorithm is active, the CC_CV (Constant Charge to Constant Voltage) set point needs to be set closer to the float voltage.

A configurable device tree data file (.dtsi) is as follows:

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

Specify the CC_CV set point for PMI895x FG to 4340 mV (currently default), which is 10 mV lower than the float voltage configured (currently default is 4350 mV). This is needed for EoC to be notified properly.

NOTE: Specify this property only if the qcom,autoadjust-vfloat property is specified in the charger driver to ensure a proper operation.

2.2.13 Resume charging based on SoC

Resume SoC can be configured for automatic recharge in FG. This, along with the recharge threshold source configuration in the charger, makes the recharging based on FG.

Enable the parameter resume-soc to resume charging set in percentage unit in the .dtsi as follows:

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

2.2.14 Standby current

Standby current is an additional parameter used for the cutoff SoC (0%), which is estimated during low current discharge. The purpose of this parameter is to provide a current minimum used by the FG hardware in calculating the SoC. This derates the SoC and is used to make sure the device has enough capacity left at a low % SoC to run in Active mode for a small amount of time before shutting down. The following is an additional Linux Device Tree Include File (.dtsi) added for this current:

```
qcom, fg-ibatt-standby-ma
```

2.2.15 External/internal current sense

The running parameter is required for parallel charging. The switch needs to be done before parallel charging is turned on. The following is an additional DTSI setting added for this feature:

`qcom,ext-sense-type`: Current sense channel used by the FG; set this to use external rsense

2.2.16 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 (may need more headroom than 50 mV value, depending on the system)
- Software will force a 0% on an empty interrupt – User space is notified via the power supply framework; user space reads 0% SoC and immediately shuts down to prevent Under Voltage Lockout (UVLO)
- Configurable as device tree data – `qcom, irq-volt-empty-mv`

2.2.17 Battery age detection

The following are the battery age algorithms:

- Battery Capacity using ESR algorithm − Estimates the battery capacity with the measured battery ESR
- Battery Learning Capacity algorithm − Learns the battery capacity by doing software Coulomb counting

NOTE: The battery age algorithms are QTI intellectual property, so 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 takes into account the following:

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

2.2.17.1 Device tree parameter settings for battery age detection

Device tree parameter settings	Description
qcom,capacity-learning-on	Boolean property to have the FG 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 percent that the capacity can rise as the result of a single charge cycle; this property corresponds to .1% increments
qcom,cl-max-decrement-deciperc	Maximum percent that the capacity can fall as the result of a single charge cycle; this property corresponds to .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-soc	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 to have the FG driver attempt to estimate the battery capacity using battery resistance algorithm (ESR)

Device tree parameter settings	Description
qcom,aging-eval-current-ma	Current used to evaluate battery aging; this value should be around the steady state current drawn from the battery when the phone is low on battery

2.2.17.1 Charge cycle count

The charge cycle counter provides the number of times the battery is charged between the specified thresholds. The counter is incremented only in the following conditions:

- When the battery starts charging and the battery SoC is below the specified low SoC threshold, the counting algorithm starts. When the battery continues charging and the battery SoC reaches above the specified high SoC threshold, the counter is incremented.
- If the charger is disconnected before the battery, the SoC threshold reaches a high SoC or the charging happens between low and high SoC thresholds, the counter is not incremented. Also, the counter stored in the FG SRAM is cleared in the event of battery removal.
- The cycle counter is exposed through the cycle_count property, POWER_SUPPLY_PROP_ CYCLE_COUNT.
- The following are the device tree parameter settings for the charge cycle counter:

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
qcom,cycle-counter-low-soc	Low SoC threshold that is compared against the battery SoC before starting the cycle counter algorithm
qcom,cycle-counter-high-soc	High SoC threshold that is compared against the battery SoC before incrementing the cycle counter

3 PMI895x FG driver

3.1 Location

- Driver source code – kernel/drivers/power/qpnp-fg.c
- Device tree configuration – kernel/arch/arm/boot/dts/qcom/msm-pmi895x.dtsi
- Document – kernel/Documentation/devicetree/bindings/power/qpnp-fg.txt

3.2 Configuration example

```
pmi8950_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 = <150>;
            qcom,fg-chg-iterm-ma = <100>;
            qcom,pmic-revid = <&pmi8950_revid>;
            qcom,cycle-counter-en;
            qcom,cycle-counter-low-soc = <15>;
            qcom,cycle-counter-high-soc = <85>;
            qcom,capacity-learning-on;

            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>;

                 interrupt-names =  "high-soc",
                              "low-soc",
```

```
                        "full-soc",
                        "empty-soc",
                        "delta-soc",
                        "first-est-done",
                        "update-soc";
        };

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

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

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

3.3 Interrupts (for PMI895x only)

Interrupt	Peripheral	Description
JEITA_SOFT_COLD_RT_STS	FG_BATT	Interrupt to notify that the battery temperature is lower than <JEITA_soft_cold threshold>
JEITA_SOFT_HOT_RT_STS	FG_BATT	Interrupt to notify that the battery temperature is higher <JEITA_soft_hot threshold>
VBATT_LOW_RT_STS	FG_BATT	Interrupt to notify that the battery voltage is lower <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 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
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 the Monotonic SoC is greater than the High SoC Threshold
LOW_SOC_RT_STS	FG_SOC	IRQ to notify that the Monotonic SoC is below the Low SoC threshold
FULL_SOC_RT_STS	FG_SOC	IRQ to notify that the Monotonic SoC = 100 %
EMPTY_SOC_RT_STS	FG_SOC	IRQ to notify that the Monotonic SoC = 0 %; must always be set lower or equal to the system
DELTA_SOC_RT_STS	FG_SOC	IRQ to notify that the Monotonic SoC change exceeds the specified delta SoC 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
UPDATE_SOC_RT_STS	FG_SOC	IRQ to notify when the SoC is updated (occurs every FG cycle; approximately 1.47 sec)
FG_MEM_RT_STS	FG_MEMIF	▪ For CMA: ▫ 1 – Conventional access logic indicating memory is available ▪ For IMA: ▫ 1 – Ready/end of transaction
IMA_XCP_RT_STS	FG_MEMIF	Used only with IMA SRAM access 1 – Exception thrown
DATA_RCVRY_SUG_RT_STS	FG_MEMIF	1 – Indicates that the software allowed to update the FG RAM contents to assist in boot
IACS_INTR	FG_MEMIF	1 – Interleave memory access transaction complete
VBAT_LT_THR_INT_RT_STS	FG_ADC_USR	IRQ indicating that VBAT is less than the threshold defined in VBAT_INT__THR register
VBAT_LT_THR_INT_RT_STS	FG_ADC_MDM	IRQ indicating that VBAT is less than the threshold defined in VBAT_INT__THR register
IBAT_GT_THR_RT_STS	FG_ADC_USR	IRQ indicating that IBAT is greater than the threshold defined in IBAT_INT__THR register
IBAT_GT_THR_RT_STS	FG_ADC_USR	RQ indicating that IBAT is greater than the threshold defined in IBAT_INT__THR register

3.4 Power supply interfaces

```
POWER_SUPPLY_PROP_CAPACITY      /* 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_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 */
```

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 the following options to choose for the battery profile setting listed here:

- Option 1 (reference only) – Force one profile
  - For testing or customer with only one profile desired
  - In the device tree specify only one battery profile
  - Specify `qcom,use-otp-profile` to avoid RAM loading any battery profile
  - Remove `qcom,batt-id-range-pct`
- Option 2 (normal design option) – Battery ID-based software loading
  - Specify `qcom,batt-id-range-pct` in battery data node (DTSI)
  - `#include` other battery profile .dtsi files (see sample below)

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

NOTE: It is important that the customer submit all batteries intended for use to QTI for battery characterization.

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

Access most of the FG registers using the indirect addressing controls located in the fg_memif peripheral. The following is an SRAM partitioning map from *PMI8994 Fuel Gauge HW/SW Control Application Note* (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: The commands above should write 4 bytes at a time or FG will 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 *Application Note: Fuel Gauge Data Collector* (80-NU716-1).
2. Use ADB to 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. In the serial window or with the ADB shell, run the following command:

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

4. After the test is completed, run the following commands:

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

3.7.3 FG debug logging

- At compile time use the debug_mask module parameter.
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/tree/drivers/power/qpnp-fg.c?h=msm-3.10#n83
- Issue the following commands at runtime:

/*all debug types bit turned on */

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

dmesg > debug_output_filename
```

A References

A.1 Related documents

Documents	
Qualcomm Technologies, Inc.	
PMI8952 Power Management IC Design Guidelines	80-NT391-3
PMI8994 Fuel Gauge HW/SW Control Application Note	80-VT310-123
MSM8952 Linux Android Charger Software User Guide	80-NV610-44
Application Note: Fuel Gauge Data Collector	80-NU716-1

A.2 Acronyms and terms

Term	Definition
ADC	Analog to Digital Converter
BSI	Battery Serial Interface
DTSI	Linux device tree include file
EoC	end of charge
ESR	Equivalent Series Resistance
FG	Fuel Gauge
OTG	on-the-go
SBL	Secondary Boot Loader
SoC	state-of-charge
SRAM	static random access memory