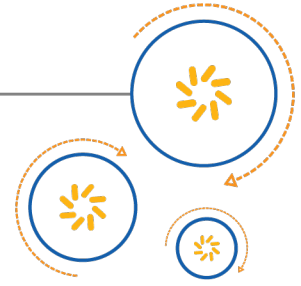




Qualcomm Technologies, Inc.



Linux Android PMIC Charger Software

User Guide

80-P2484-77 Rev. D

January 29, 2018

Confidential and Proprietary - Qualcomm Technologies, Inc.

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or websites to DocCtrlAgent@qualcomm.com.

Restricted Distribution: Not to be distributed to anyone who is not an employee of either Qualcomm Technologies, Inc. or its affiliated companies without the express approval of Qualcomm Configuration Management.

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies, Inc.

Qualcomm Quick Charge is a product of Qualcomm Technologies, Inc. Qualcomm WiPower wireless charging technology is licensed by Qualcomm Incorporated. Other Qualcomm products referenced herein are products of Qualcomm Technologies, Inc.

Qualcomm and WiPower are trademarks of Qualcomm Incorporated, registered in the United States and other countries. Quick Charge is a trademark of Qualcomm Incorporated. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

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

Revision history

Revision	Date	Description
A	September 2016	Initial release
B	April 2017	Updated the document title and references
C	August 2017	Removed a note from Section USB PD .
D	January 2018	Added Section 4.7, Section 10.4, and Chapters 5 through 9.

Contents

Revision history	2
1 Introduction	9
1.1 Purpose	9
1.2 Conventions	9
1.3 Technical assistance	9
2 Hardware overview	10
2.1 Charge cycle	13
2.1.1 Prequalification	13
2.1.2 Charging phases	13
2.1.3 Automatic recharging	15
2.1.4 Charger inhibit	15
2.1.5 Battery over-voltage protection	16
2.1.6 Battery missing detection	16
2.1.7 Charge conditions	17
2.2 Charger detection	18
2.3 Input current limits	20
2.3.1 Type-C	20
2.3.2 Automatic input current limiting	20
2.3.3 Step charging	21
2.4 Battery current limit	21
2.4.1 JEITA	22
2.5 Parallel charging	23
2.5.1 Quick Charge	25
2.5.2 USB PD	25
2.6 USB OTG	26
2.6.1 Simultaneous WiPower charging and OTG	27
3 Software architecture and directory structure	29
3.1 Battery	30
3.2 USB	30

3.3 Parameters	32
3.4 Device tree settings	32
4 Software drivers	33
4.1 BMD	33
4.2 Charger detection	33
4.3 Input current limit	34
4.3.1 qcom,suspend-input	34
4.3.2 qcom,usb-icl-ua	35
4.3.3 qcom,dc-icl-ua	35
4.3.4 Type-C	36
4.4 Battery current limit	37
4.4.1 qcom,fcc-max-ua	37
4.5 Parallel charging	37
4.6 OTG and VCONN	38
4.6.1 Example device tree	38
4.6.2 VBUS and VCONN enablement	38
4.7 UEFI charger app	39
4.7.1 QTI charger app	40
4.7.2 PmicDxe driver	41
4.7.3 ChargerLib	41
5 Charger configuration	43
5.1 Multilevel JEITA charging	43
5.2 Step charging	44
6 Dead battery recovery	46
6.1 Dead battery recovery in XBL	46
6.2 Dead battery recovery in UEFI	50
6.2.1 Configuration quick reference table	51
7 UEFI QTI charger app configuration	56
7.1 QTI charger app configuration file	56
7.2 Threshold charging configuration	56
7.2.1 SocOrVoltageBaseBoot	56
7.2.2 LoadBatteryProfile	57
7.2.3 DispSignOfLifeMaxThresholdMv	57
7.2.4 FgCondRestart	57
7.2.5 ChargerLedConfig	57
7.2.6 EnShipMode	57
7.3 Debug configuration	58

7.3.1 PrintChargerAppDbgMsg	58
7.3.2 PrintChargerAppDbgMsgToFile	58
7.3.3 FileLoggingDbgLevelMask	58
7.3.4 EnableChargerFGDump	59
7.4 FG and FG debug configuration	59
7.4.1 BatteryIdTolerance	59
7.4.2 DumpSram	59
7.4.3 DumpSramStartAddr	59
7.4.4 DumpSramEndAddr	59
7.4.5 DumpSramDuration	59
7.5 Initial required charger configuration	59
7.5.1 BootToHLOSThresholdInMv	60
7.5.2 OsStandardBootSocThreshold	60
7.5.3 BattVoltLimHighDelta	60
7.5.4 ChgFvMax	60
7.5.5 ChgFccMax	60
7.5.6 ChargingTermCurrent	60
7.5.7 ConservChgFvDelta	60
7.5.8 BATT_THERM coefficients	61
7.5.9 BATT_THERM configs	61
7.5.10 AUX_THERM coefficients	62
7.5.11 AUX_THERM configs	62
7.5.12 Device skin and charger hot thresholds	63
7.5.13 Charger_THERM source configs	63
7.5.14 EmergencyShutdownVbatt	63
7.5.15 EnableChargerWdog	63
7.5.16 VBtEmpty threshold	63
7.5.17 VBattEstDiffThreshold	64
7.5.18 RConn compensation resistance	64
7.6 Battery error handling configuration	64
7.6.1 DebugBoardBatteryIdMin and DebugBoardBatteryIdMax	64
7.6.2 SmartBatteryIdMin and SmartBatteryIdMax	64
7.6.3 RegularBatteryIdMin and RegularBatteryIdMax	64
7.6.4 UnknownBatteryBehavior	65
7.6.5 DebugBoardBehavior	65
7.6.6 BattMissingCfg	65
7.7 Jeita configuration	65
7.7.1 Jeita zones	65

7.7.2 JeitaCcCompCfg	66
7.7.3 JeitaFvCompCfg	66
7.7.4 NoChargeAndWait	66
7.8 WiPower configuration	66
7.8.1 WiPowerSupported	66
7.8.2 DCInBootToHLOSThresholdInMv	66
7.8.3 SuspendDCIn	67
7.9 Thermal configuration	67
7.9.1 SWThermalMitigationEnable	67
7.9.2 TsensTimeoutMins	67
7.9.3 Tsens limits or zone	67
7.10 QTI charger app configuration file for LA (example)	67
8 Off mode charging	72
8.1 Enable Off mode charging	72
8.2 Enable or disable charging in fastboot	73
8.3 Off mode call flow	73
9 Voter	74
9.1 struct votable	74
9.2 Initialization functions	74
9.2.1 create_votable()	75
9.2.2 destroy_votable()	75
9.2.3 lock_votable()	75
9.2.4 unlock_votable()	76
9.3 Votable functions	76
9.3.1 vote()	76
9.3.2 rerun_election()	77
9.3.3 find_votable()	77
9.4 Get functions	78
9.4.1 get_client_vote()	78
9.4.2 get_client_vote_locked()	78
9.4.3 get_effective_result()	78
9.4.4 get_effective_result_locked()	79
9.4.5 get_effective_client()	79
9.4.6 get_effective_client_locked()	80
9.4.7 is_client_vote_enabled()	80
9.4.8 is_client_vote_enabled_locked()	81
9.5 Voter call flows	81

10 Debug	84
10.1 Use ADB over Wi-Fi to capture log dumps while charging	84
10.2 Enable charger logs when providing kernel logs	85
10.3 Guidelines for creating a support request	85
10.4 Debug overwrite feature	85
10.4.1 Enable the debug overwrite feature with Flash tools FV	85
11 FAQs	88
A Input current limits	89
B References	95
B.1 Related documents	95
B.2 Acronyms and terms	95

Qualcomm
2018-08-13 19:33:26 PDT
songpeng2@huagin.com

Figures

Figure 2-1: Hardware architecture.....	12
Figure 2-2: Charger cycle plot.....	17
Figure 2-3: BC 1.2 algorithm flow.....	19
Figure 2-4: Float voltage compensation based on battery temperature conditions.....	22
Figure 2-5: Charge current compensation based on battery temperature conditions.....	22
Figure 2-6: JEITA flowchart.....	23
Figure 2-7: Parallel charging application with SMB1381.....	24
Figure 2-8: Reverse boost (OTG/HDMI/MHL) functional flowchart.....	27
Figure 3-1: Charger software architecture block diagram.....	29
Figure 4-1: Charger presence detection flowchart.....	34
Figure 4-2: Type-C detection flowchart.....	36
Figure 4-3: Battery condition detection flowchart.....	37

1 Introduction

1.1 Purpose

This document describes the PMI8998 charger concept and objectives. Details provided include software requirements, design, and verification of the charging feature.

This document is intended for software developers implementing the charging feature to comply with feature and software design requirements. This document is also intended for test engineers to design test plans for charging.

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:.`

If you are viewing this document using a color monitor, or if you print this document to a color printer, **red boldface** indicates code that is to be **added**, and ~~blue strikethrough~~ indicates code that is to be replaced or removed.

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 Hardware overview

The PMI8998 is a switch-mode Li-ion battery charger and an input and output power controller for portable devices. The device is used with systems using single-cell Li-ion and Li-polymer battery packs.

The device is fully programmable via the SPMI interface. Nonvolatile configuration registers enable device configuration in different power-up states and are used to reconfigure those settings during operation.

Input selection and arbitration

PMI8998 uses the D+ and D- USB lines to determine the type of charger connected to the handheld device. Input current automatically adjusts based on the power source type.

The device accepts either a general DC input (DC_IN) or a USB VBUS DC input (USB_IN) on separate input paths. Both inputs have programmable input current limiting to protect upstream charging sources (such as USB). The PMI8998 automatically detects and chooses the best source. The priority between DC_IN and USB_IN, if both inputs are present, is selectable.

Note: The software is coded assuming that USBIN is at a higher priority than DCIN. Changing this code might cause issues.

The PMI8998 can use the battery as the input source and provide power to the peripherals compliant with the USB On-the-Go (OTG) specification. To enable OTG mode, reverse the internal path and use the USB_IN input as an output, providing 5.0 V and up to 1500 mA.

The OTG_EN bit (register 0x1140[0]) is set via the regulator framework software. This bit is controlled by the USB software for OTG Host mode support and Type-C downstream-facing port (DFP) mode to supply VBUS.

System output supply and control

PMI8998 automatically maximizes charge current for a given power source level by detecting the maximum stable output current of the AC/DC adapter, which allows for current level optimization between the power adapter and the portable device.

The device also has an integrated output current path manager. PMI8998 manages the current from the USB_IN or DC_IN inputs and routes it to one of two separate outputs – battery or system. Because the two outputs are independently switched and PMI8998 is programmable, the charge currents are adjusted during the operation.

A missing/disconnected/faulty battery does not block the current path of the system. If enough current is available, the system operates from the DC_IN/USB_IN regardless of the battery state.

Battery charging

PMI8998 is compatible with the Alliance for Wireless Power (A4WP) version 1.2 Qualcomm® WiPower™ wireless charging specification. When configured in this mode, the DC_IN input limits the minimum programmable input impedance and maximum input power to 5 W per the A4WP specification.

For the WiPower applications, the PMI8998 pairs with the Stark divide-by-two charge pump, which communicates the power transmitting unit (PTU) state and charge pump divide-by-two/pass-through mode through the CHG_OK and DIV2_EN pins.

IC protection

In addition to safety timers and cell temperature protection, PMI8998 includes a wide range of IC protection features as follows:

- Input and output over-voltage/over-current protection
- Thermal shutdown
- Charger current or float voltage compensation
- Automatic battery discharge
- Battery missing detection

An open-drain STAT output indicates various status and fault conditions, which are selected via the STAT output configuration register.

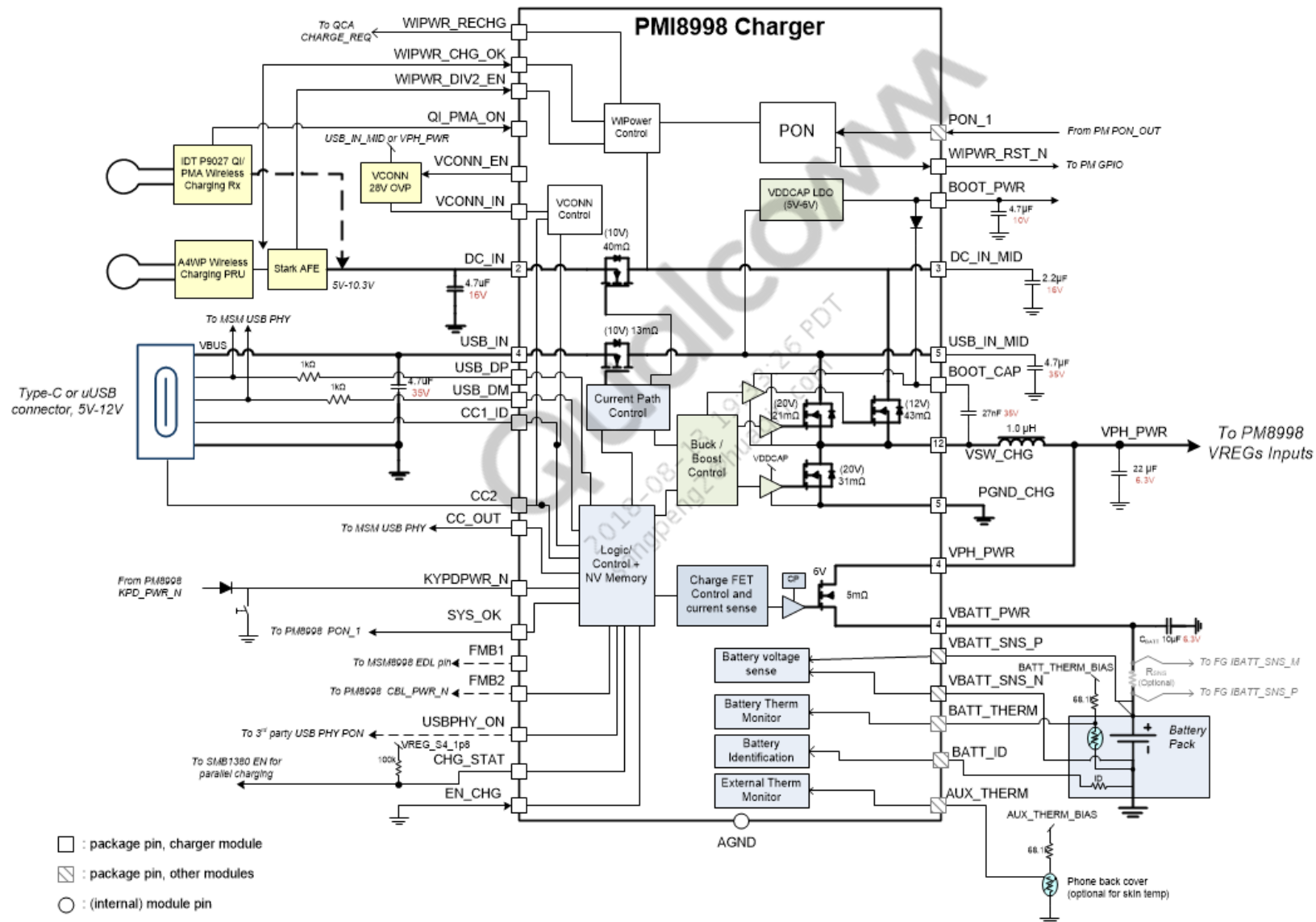


Figure 2-1 Hardware architecture

2.1 Charge cycle

2.1.1 Prequalification

When an external wall adaptor or a USB cable is connected to the corresponding input, the charger runs a series of prequalification tests before initiating the first charge cycle. The input voltage level must be higher than the UVLO threshold, lower than the OVLO threshold, 100 mV (DCIN or USBIN rising) greater than the battery voltage, and the appropriate charger ENABLE register SPMI command or EN_ENABLE input must be asserted.

If enabled, the battery voltage must be below the battery OV threshold and battery temperature must be below the cold/hot hard thermal limits. Prequalification parameters are continuously monitored and the charge cycle is suspended if one of the parameters is beyond its programmed limits.

2.1.2 Charging phases

PMI8998 provides the following charging phases:

- Trickle-charge
- Preconditioning (precharge)
- Constant current (fast charge)
- Constant voltage (taper charge)

All charger phases (except trickle charge) are fully programmable, allowing various battery charging algorithms that support multiple system designs and new battery technologies. The following parameters are programmable:

- Precharge current
- Fast-charge current
- Termination current
- Float voltage
- Automatic recharge threshold
- Precharge voltage threshold
- Low battery voltage threshold
- Charger inhibit level
- Input current limit (DC_IN/USB_IN)
- Safety timer duration
- Battery thermal limits
- Die thermal limits
- Interrupt conditions
- Missing battery detection
- Automatic float voltage compensation level (battery IR compensation)

Trickle charging phase

When all prequalification conditions are met, the device checks the battery voltage to determine if trickle-charging is required. If the battery voltage is below approximately 2.1 V a charging current of 45 mA. This allows battery pack protection circuit reset and increasing battery voltage level without compromising safety. The trickle-charging current source is powered from V_{sys} to minimize power dissipation when charging from higher voltage inputs.

Precharging phase

When the battery voltage crosses the 2.1 V level, the charger precharges the battery to safely charge the deeply discharged cells.

The precharge constant current is programmable from 0 mA to 1575 mA in 25 mA steps in PRE_CHARGE_CURRENT_CFG. The CHARGER remains in this mode until the battery voltage reaches the precharge to fast-charge voltage threshold programmed in PRE_TO_FAST_CHARGE_THRESHOLD.

If the precharge to fast-charge voltage threshold is not exceeded before the precharge timer expires, the charge cycle is terminated, and a corresponding timeout fault signal is asserted.

The precharge current setting is typically programmed to C/10, but can vary by battery pack.

Fast charging phase

When the battery voltage exceeds the precharge to fast-charge voltage threshold, the device enters the Fast Charge mode. During this mode, the fast charge current level is set by the fast charge register.

The fast charge current is programmable from 0 mA to 4500 mA with 25 mA steps in FAST_CHARGE_CURRENT_CFG.

Fast charge current level (output) is limited by the following conditions:

- Input current limit setting
- Load being drawn from the V_{sys} output
- Thermal environment of the device

The fast-charge current setting is typically programmed to 1C-2C, but can vary by battery pack.

Taper charging phase

The following occurs when the battery voltage reaches the predefined float voltage:

- Fast charge current starts decreasing exponentially
- Remaining battery capacity is filled
- Battery voltage remains constant

The charger spends a significant amount of time in constant voltage mode depending on the following factors:

- Battery size
- Internal pack and chemical resistance
- Selected charge rate

The float voltage is programmable from +3.4875 V to +4.92 V in 25 mV steps in `FLOAT_VOLTAGE_CFG`. The higher float voltage settings of the charger enable charging modern battery packs with a required float voltage of 4.35 V and higher. Dynamically adjusting the float voltage enables implementing sophisticated battery charging and control algorithms.

During the Taper Charging phase, the charger has two options (selected in `CHGR_CFG2`) that determine when to terminate charging.

- Fuel gauge – Current threshold is programmed into the fuel gauge peripheral. The fuel gauge monitors battery current using its IADC and indicates to the charger internally when the programmed threshold has been crossed.
- Charger – Value is set in `TCCC_CHARGE_CURRENT_TERMINATION` and determined by a digital algorithm. The algorithm detects the battery current by maintaining the `ICHG` reference close to the actual battery current.

The programmed float voltage setting must be greater than the programmed minimum system regulation voltage setting in `VSYS_MIN_SEL_CFG`.

2.1.3 Automatic recharging

The charger allows the battery to be automatically recharged when the fuel gauge indicates that the battery must be topped off. To determine when to recharge the battery, the fuel gauge has an option to use state-of-charge (SoC) or battery voltage. This option and its associated thresholds are programmed in the fuel gauge module. Refer to *Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74).

A new charging cycle initiates if the following conditions are met:

- Input power supply is present
- Charging remains enabled (SPMI command)
- All prequalification parameters are met

The battery capacity remains high without having to manually restart a charging cycle.

If the charge inhibit function is enabled, the automatic recharge threshold is overridden to the (higher) charge inhibit voltage threshold.

2.1.4 Charger inhibit

Unless the battery voltage is 50 mV, 100 mV, 200 mV, or 300 mV below the float voltage (register `0x1072[1:0]`), the charger inhibit option prevents charging initiation upon charger insertion, power cycling, or charge enabling/disabling. This option prevents over-stressing the battery via continuous charging cycles in systems with short run times and frequent power cycling (input power connect/disconnect). Additional charging cycles degrade cell life at a faster rate, which results in a poor user experience.

This feature is only active during the power cycling and manual charge enabling. If the device enters and exits Suspend mode, charging continues if the battery is above the charge inhibit voltage threshold.

When this feature is enabled, the automatic recharge threshold is overridden to the same threshold if it is set higher than the inhibit threshold.

2.1.5 Battery over-voltage protection

The PMI8998 charger features an analog comparator that monitors the battery voltage and compares it to the float voltage setting. The battery overvoltage IRQ triggers and charging is automatically disabled within 175 ms if each of the following are true:

- Charging is enabled
- Battery voltage exceeds the programmed float voltage set in the register `FLOAT_VOLTAGE_CFG` by ≥ 100 mV
- Register `CHGR_CFG2` is set to 1

This feature provides the following safety functionality:

- Does not allow the battery to charge above the float voltage
- Keeps the charger buck output (VPH) regulated when an over-voltage battery is connected
- Prevents damage to downstream devices that cannot handle higher voltage transients

2.1.6 Battery missing detection

The PMI8998 battery missing detection (BMD) options include the battery pack thermal monitor and battery identification pin.

Battery pack thermal monitor

The first option uses the battery pack thermal monitor IO (`BATT_THERM`). The device checks if there is a resistor to ground on the third terminal of the battery (T) by comparing the `BATT_THERM` pin voltage to the `BATT_THERM_BIAS` pull-up supply voltage. If the `BATT_THERM` pin voltage is above 97.5% of the `BATT_THERM_BIAS` voltage, then the battery is determined to be missing.

When no battery is detected, corresponding IRQ and status bits are asserted and charging is suspended. After the battery is reconnected, charging is automatically initiated, assuming that input power is present, all qualification parameters are met and charging is enabled. When not used, `BATT_THERM` must be connected to GND for normal operation.

Battery identification pin

The battery has a resistance to ground from this pin and that value is programmed into the fuel gauge memory. When the battery is present, its ID pin connects to the PMI's `BATT_ID` pin, which is self-biased by an internal current source.

If no battery is connected, the current source changes the `BATT_ID` pin to its supply voltage and the battery is determined to be missing by the fuel gauge. After the battery is reconnected, charging is automatically initiated if the following are true:

- Input power is present
- All qualification parameters are met
- Charging is enabled

Note: When not used for ID detection or MIPI BIF, the ID must be connected to GND for normal operation.

2.1.7 Charge conditions

- Valid charger connection
 - Input voltage < OVLO
 - Input voltage > UVLO
 - Input voltage > VBAT + 0.180 V
- Trickle-charge
 - VBAT < 2.1 V
 - IBAT 45 mA
- Precharge
 - VBAT from 2.4 V to 3.0 V
 - IBAT from 100 to 1500 mA
- Constant current charge
 - IBAT from 0 mA to 4500 mA
 - Constant voltage charge
 - VFLOAT from 3.6 V to 4.79 V
- Charger cycle plot – The programmable charging algorithm is shown in the following figure:

Charge Algorithm (CC-CV) vs. Time

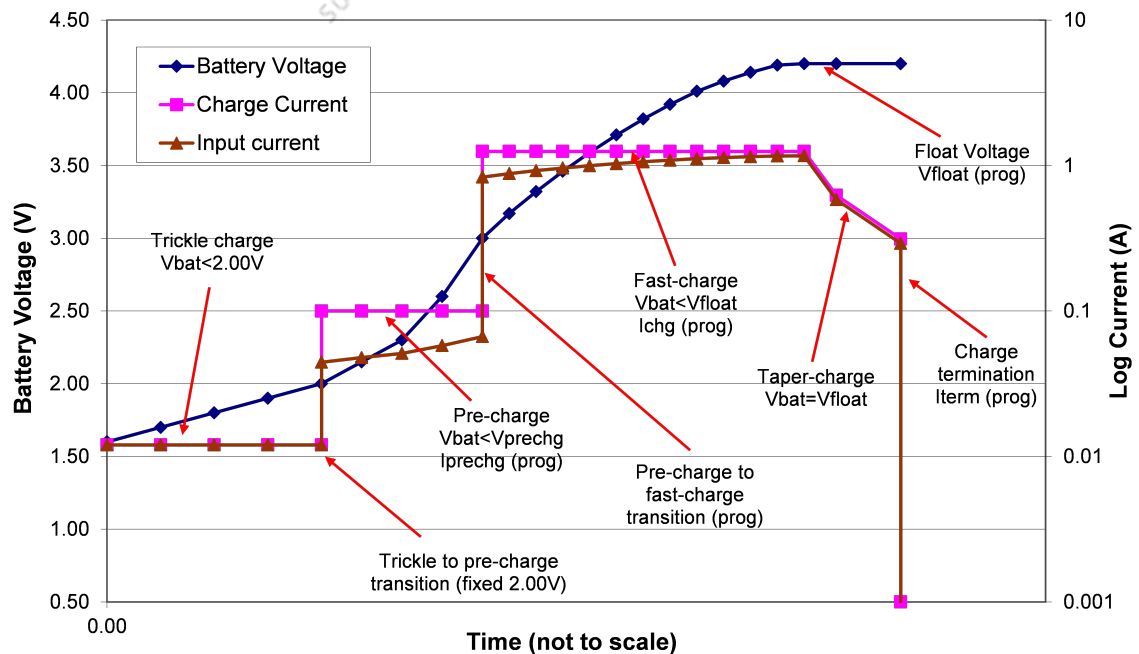


Figure 2-2 Charger cycle plot

2.2 Charger detection

The BC 1.2 detection algorithm is programmed to run as soon as a valid VBUS voltage is detected or after Type-C CC detection has completed. The first step of the BC 1.2 algorithm is to initiate the data contact detect (DCD) timeout, which is programmable to either 300 ms or 600 ms. Per the BC 1.2 specification, DCD is implemented to detect when the data lines have made contact, or to implement a forced DCD timeout to allow for enough time to insert the connector.

The PMI8998 has a dedicated USBIN_PLUGIN charger presence interrupt. The PMI's hardware autonomous power source detection (APSD) algorithm ensures compliance with the USB Type-C 1.2 and USB BC 1.2 specifications and authenticates Qualcomm® Quick Charge™ 2.0/3.0 adapters. If VBUS is present during APSD, BC 1.2, detection runs autonomously in parallel to Type-C detection hardware. Power source types detected are shown in the following table:

Power source type	Description
Standard downstream port (SDP)	This is a computer USB port, capable of USB 1.1 (100 mA), USB 2.0 (100 mA/500 mA), or USB 3.0 (150 mA/900 mA). D+ and D- are independently pulled down in the host with a resistance of 14.25 kΩ to 24.8 kΩ.
Charging downstream port (CDP)	Powered USB hub capable of a maximum of 1.5 A.
Dedicated charging port (DCP)	Standard wall charger capable of at least 500 mA and a maximum of 5 A. D+ and D- are shorted in the wall adapter with a maximum resistance of 200 Ω. BC 1.2 and USB Type-C specifications allow up to 1.5 A to be drawn from a DCP through a legacy cable without using proprietary charging methods.
Other charging port (OCP)	Nonstandard charger with a proprietary D+/D- configuration for charging ports not covered by the USB charging specification 1.2. These chargers have similar current capability to normal DCPs. D+ and D- are connected to VBUS via a resistor divider, which gives fixed voltage levels.
Floating charger	Nonstandard charger with D+ and D- floating and is not compliant with the BC 1.2 specification. This type of charger is allowed by the Type-C specification as long as it either has a Type-C receptacle or a captive cable.

The USBIN_SOURCE_CHANGE interrupt triggers if there is a change in APSD status, HVDCP2 presence status, or HVDCP3 authentication. When one of these conditions changes and the

USBIN_SOURCE_CHANGE interrupt fires, the software checks the real-time status of APSD, HVDCP2, and HVDCP3. The following algorithm flowchart illustrates this procedure:

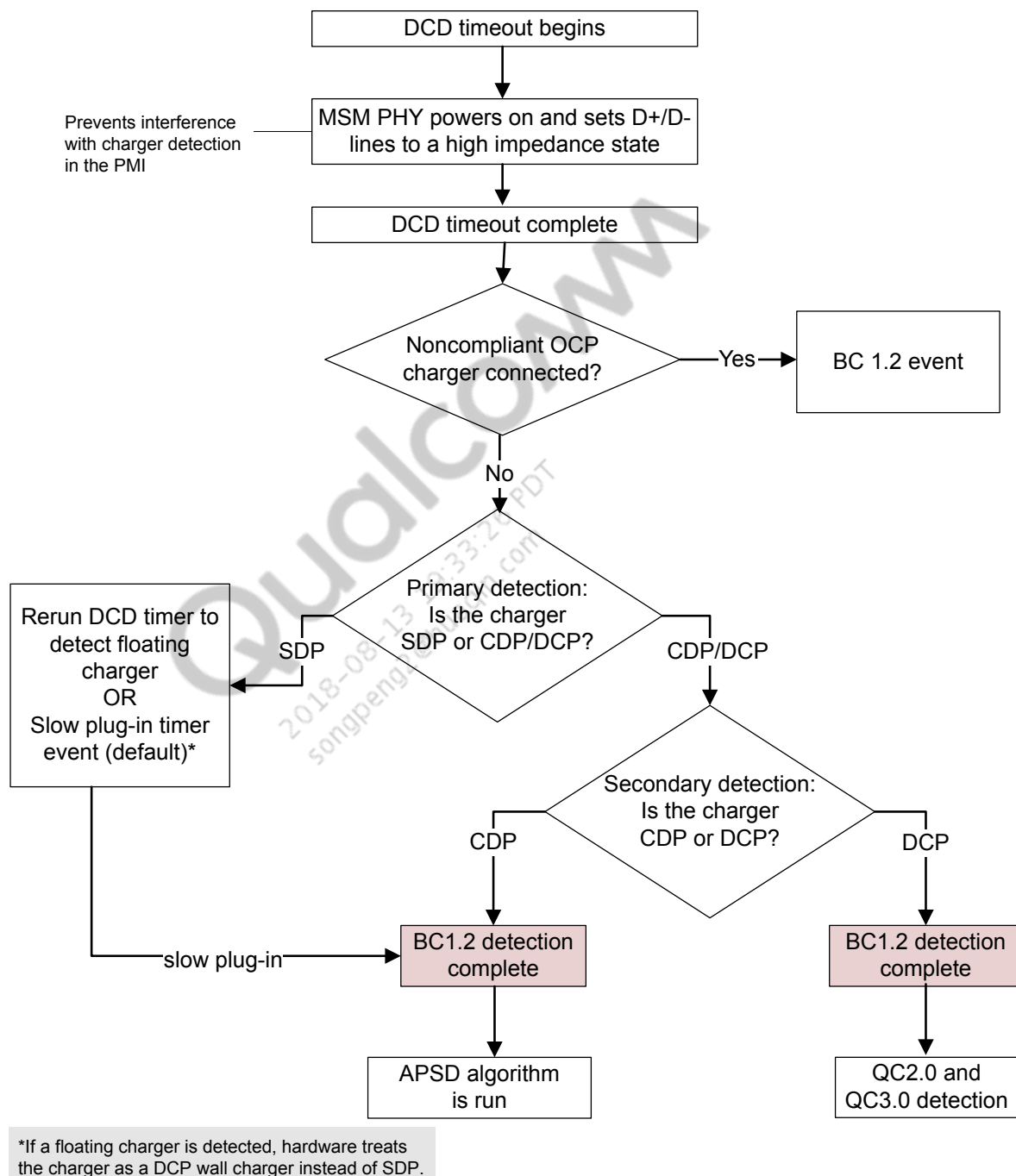


Figure 2-3 BC 1.2 algorithm flow

Refer to PM8998, PM8005, and PMI8998 Power Management ICS Design Guidelines (80-P1086-5).

2.3 Input current limits

For a list of input current limits by charger type, refer to [Input current limits](#).

2.3.1 Type-C

When programmed to operate in USB Type-C mode, the PMI8998 charger is fully compliant with USB Type-C as defined in the *USB Battery Charging 1.2 Compliance Plan Specification* (www.usb.org).

PMI8998 operates in the following modes:

- Dual Role Power (DRP) mode
- Power Sink mode (up to 3 A input current, 4.8 A with USB Power Delivery (PD))
- Power Source mode (up to 1.5 A output current, 2.0 A with USB PD)

PMI8998 supports the following:

- Cable orientation detection for SuperSpeed USB (using the CC_OUT pin and register, analog audio adapters, and debug mode (Rp/Rp only))
- USB PD 3.0 (conditional FR_SWAP support)
- Vconn supply as required for active cables, AMAs, and passive e-marked cables

In addition to USB Type-C compliance, the charger includes Factory mode detection that leverages the USB Type-C CC interface to enter a Factory Configuration mode.

2.3.2 Automatic input current limiting

An automatic input current limiting (AICL) algorithm allows PMI8998 to automatically maximize the current drawn from an AC/DC adapter connected to one of the following:

- USB_IN input
- Wireless charger connected to the DC_IN input

The AICL algorithm is initiated when a valid input is connected to either USB_IN or DC_IN, and the switcher is running.

As input current increases, one of the following two states occur:

- Output current reaches the limit of the AC/DC adapter – Adapter's output voltage begins to collapse until it reaches the programmed V_AICL threshold

When the V_AICL threshold is reached, the charger begins a programmable deglitch time of 30 μ s to 30 ms (1.1 silicon is fixed 5 ms, 2.0 is programmable 30 μ s to 30 ms) and reduces the input current draw one step at a time until the voltage recovers above V_AICL.

- Maximum programmed input current reaches the limit value in one of the following registers:
 - 0x1370[7:0] (USB_IN) – Maximum value (ICL_MAX) depends on APSD results
 - 0x1470[7:0] (DC_IN) – Maximum value depends on whether the part is configured in standard DC_IN mode or WiPower mode (register 0x1495[0]=1 for WiPower mode)

USB_IN input

The AICL algorithm sets the maximum input current limit to 500 mA for USB_IN connections. The charger must wait for BC 1.2 detection to complete before the power source is confirmed to be capable of more than 500 mA (maximum for a USB 2.0 SDP and the worst case scenario). See [Charger detection](#) for information on the BC 1.2 algorithm.

During BC 1.2 detection, the AICL algorithm remains enabled if one of the following occurs:

- BC 1.2 detects a noncompliant charger (current limit < 500 mA)
- A high-resistance cable is connected to USB_IN, which causes a large DC voltage drop

After BC 1.2 detection is completed, the new maximum input current limit is set to the value programmed in high current register 0x1370[7:0] depending on the detection results.

Wireless charger connected to DC_IN

For DCN_IN connections, the AICL algorithm sets the maximum input current limit to one of the following:

- Non-WiPower configurations – The value programmed in register 0x1470[7:0]
- WiPower configurations – Registers 0x1492 to 0x1498[7:0], in which the input current limit depends on the DCIN voltage, DIV2_EN pin, and CHG_OK pin

2.3.3 Step charging

Step charging is a hardware-autonomous algorithm that enables a staircase charging profile for lithium-ion battery packs. This algorithm allows programmable charge current values based on programmable thresholds on battery voltage or SoC.

This implementation relies on battery characterization data from battery cell manufacturers to determine the charging profile to be used. Software interaction for step charging is not currently available.

2.4 Battery current limit

Battery current limit is configured via the registers listed in the following table for precharge and fastcharge phases. For battery current limit configuration, see [Device tree settings](#).

Current limit	Register	Address	Range
Precharging	PRE_CHARGE_CURRENT_CFG	0x1060	0 mA to 1575 mA (25 mA steps)
Fastcharging	FAST_CHARGE_CURRENT_CFG[7:0]	0x1061	0 mA to 4500 mA (25 mA steps)

Outside of these register configurations, the soft JEITA conditions alter the battery charging current limitation. Refer to *PM8998*, *PM8005*, and *PMI8998 Power Management ICS Design Guidelines* (80-P1086-5).

2.4.1 JEITA

The PMI8998 fuel gauge manages all battery thermal monitoring. The fuel gauge takes automatic battery temperature measurements every 1.5 sec. If the battery temperature exceeds a preprogrammed soft or hard thermal threshold, an IRQ is generated and sent to the charger module. There is no built-in hysteresis in the thresholds, but this hysteresis is added by independent adjustment in software.

The temperature sensing I/O (THERM) is used for adaptive charge current or float voltage control (that is, charge current and float voltage compensation). These features enable PMI8998 to charge the battery at a reduced current and voltage when the battery pack temperature is between the soft and hard over-/under-temperature limits, making the device compatible with the latest JIS8714 and JEITA standards.

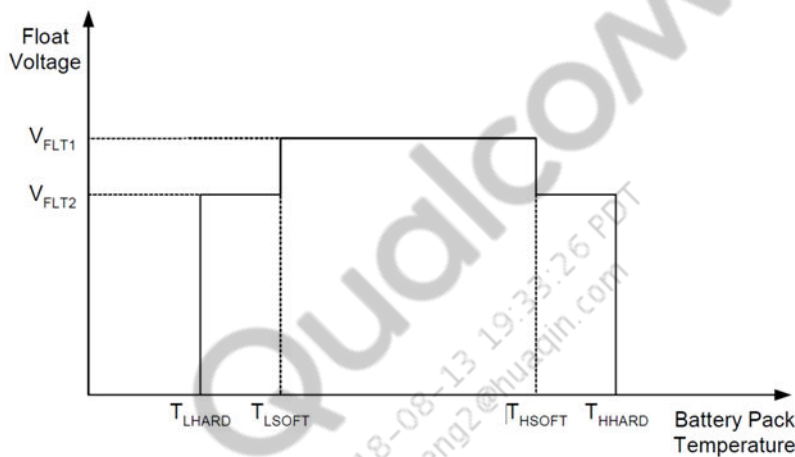


Figure 2-4 Float voltage compensation based on battery temperature conditions

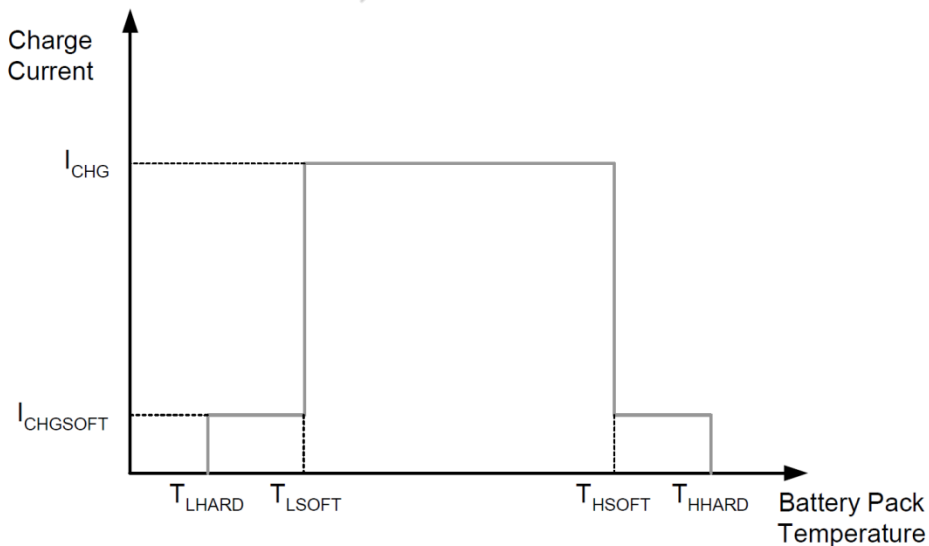


Figure 2-5 Charge current compensation based on battery temperature conditions

The soft hot and cold temperature limits are programmable in the fuel gauge module (refer to *Linux Android PMIC Fuel Gauge Software User Guide* (80-P2484-74)). The corresponding soft charge current and soft float voltage settings are also programmable.

If the battery voltage is greater than the compensated float voltage value, charging is disabled because the termination current has been reached.

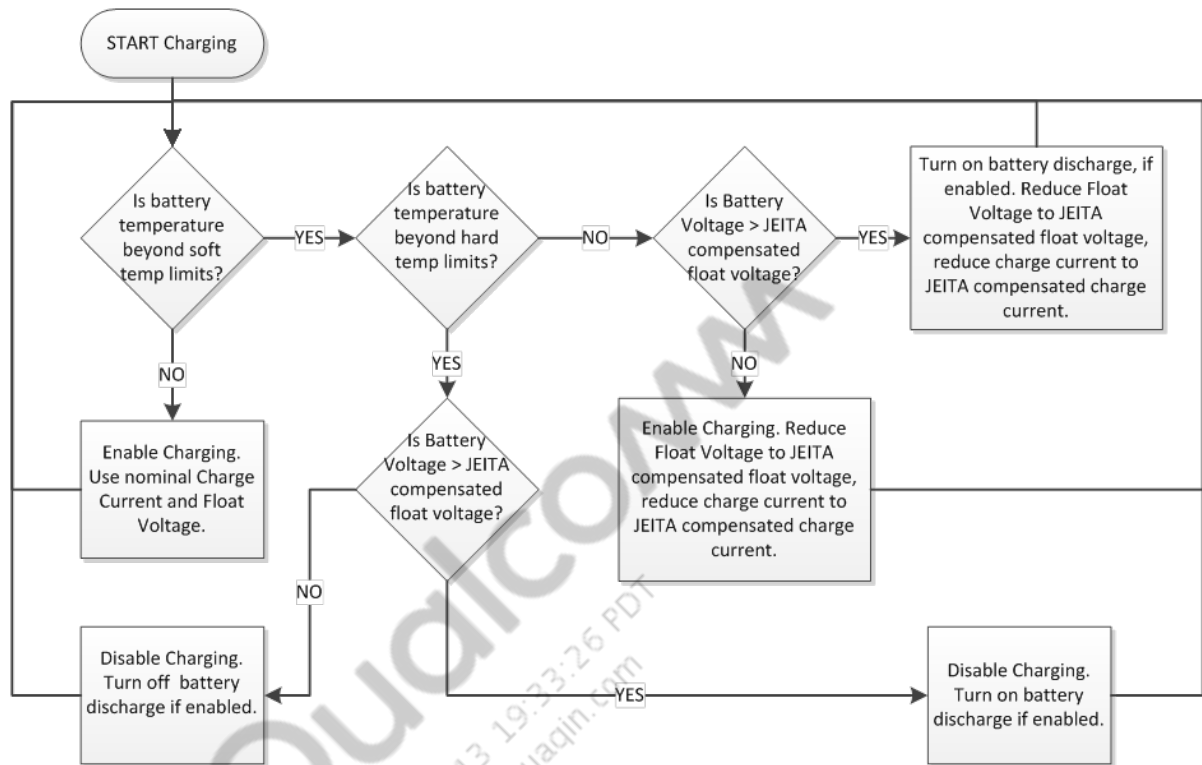


Figure 2-6 JEITA flowchart

After the battery temperature returns to the normal range, charging is only re-enabled if the battery voltage is less than the automatic recharge voltage threshold.

If the programmed soft-limit charge current reduction value is less than the programmed termination current value, a false termination event occurs during a JEITA soft-limit condition. This occurs because the charge current reduces to less than the termination current value. To avoid this event, ensure that the termination current setting is always higher than the soft-limit current reduction setting.

I/O (THERM) also prevents excessive battery temperatures during charging. If the temperature hard limits are exceeded and the JEITA hard-limit function is enabled, battery charging is suspended while safety timers maintain their values but are paused. During this mode, the FET between the battery and the system is turned on only for the system to be powered by the battery, but not for charging. The corresponding cold hard-limit status and IRQ or hot hard-limit status and IRQ is asserted.

After the temperature level returns to the safe operating range (with hysteresis), charging is automatically re-enabled, the corresponding fault bit is reset, and safety timers continue counting. The corresponding status bit is not latched and changes in real time (with hysteresis) as the temperature moves in or out of the temperature limits.

2.5 Parallel charging

The following solutions enable maintaining the power dissipation at charge currents greater than 4.0 without increasing the phone case or skin temperature phone beyond an undesirable value:

- PMI8998 is paired with an external SMB1381 charger
- The two ICs charge the battery in parallel during the Constant Current mode

When both chargers are used in parallel, the current in each charger is reduced in half, so the total power savings due to conduction loss is reduced. The heat generated spreads throughout the two ICs in different areas of the PCB for uniform thermal distribution across the phone case.

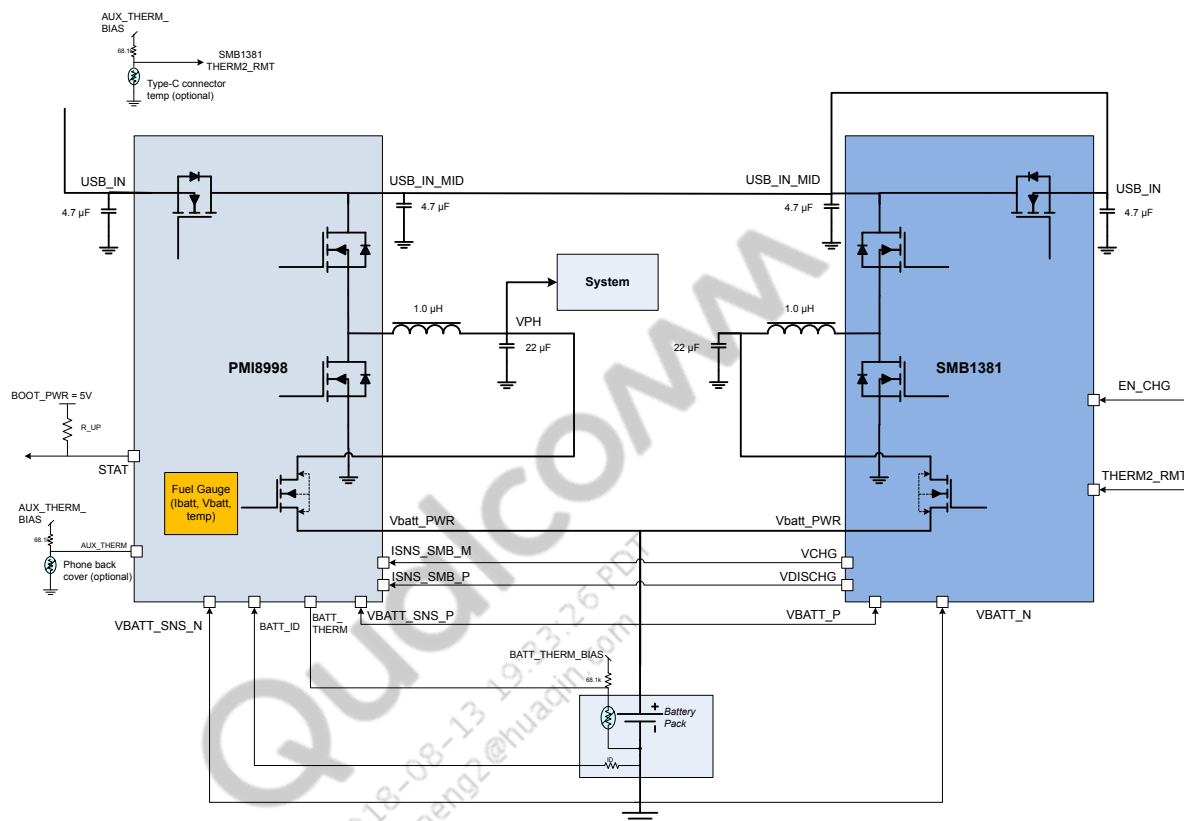
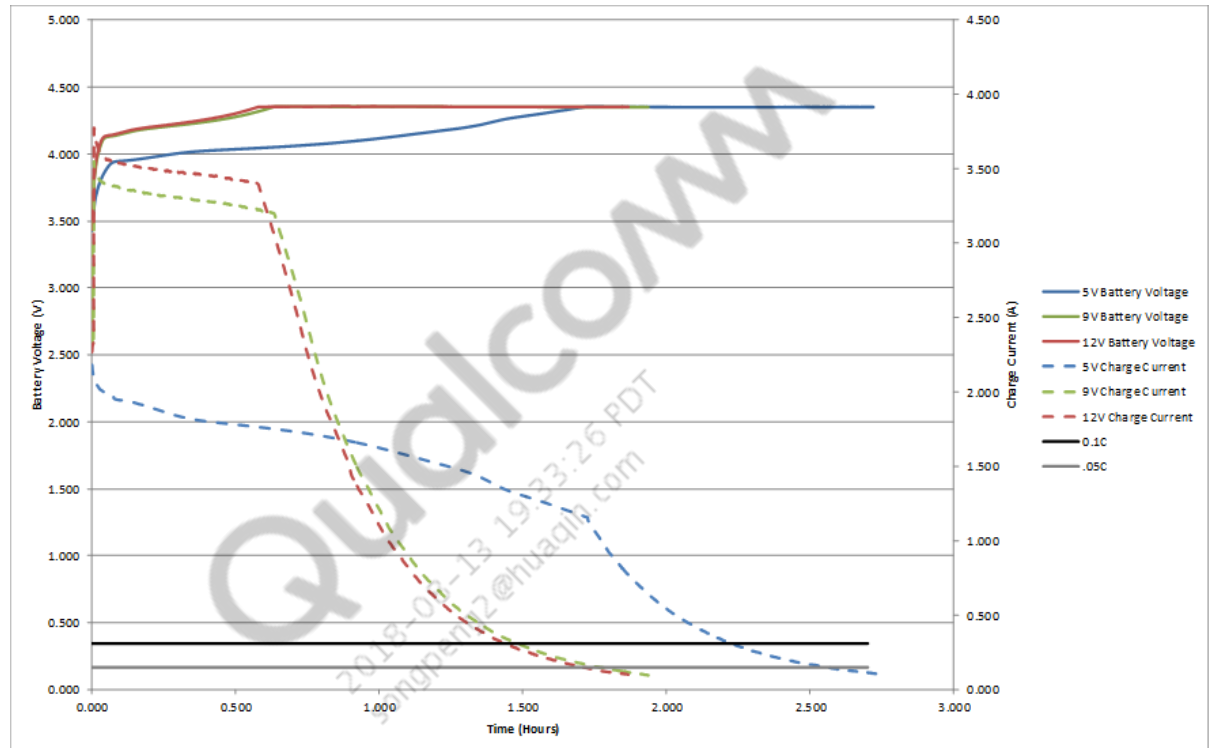


Figure 2-7 Parallel charging application with SMB1381

For details on the hardware operation of this feature, refer to *PMI8998 Power Management ICS Design Guidelines* (80-P1086-5).

2.5.1 Quick Charge

As shown in the figure, Quick Charge authentication occurs after secondary detection has determined that a DCP is connected. This detection flow is stopped by disabling the HVDCP detection, in which case only a DCP is detected. An interrupt is issued after BC 1.2 detection is complete and, if enabled, after HVDCP detection is complete. The following figure also provides a 9 V and 5 V charging time comparison:



2.5.2 USB PD

The PMI8998 features a USB PD PHY. The PD PHY is used in both Power Source and Sink modes to communicate over a single CC line with a connected PD-capable device. Because the rate of PD communication is faster than the debounce the PMI8998 charger uses to detect a CC voltage change, this communication co-exists with Rp and Rd. USB PD communication is initiated as soon as the CC lines are debounced and an Rp value of medium/high current is detected on either CC line.

The PMI8998 supports the following USB PD use cases.

PR_SWAP

Power role swap (PR_SWAP) makes the charger the power source if it was originally the power sink, and vice versa. When swapping power roles, the Rp and Rd current sources/resistors swap to indicate the change of power role. Additionally, if Vconn is being supplied, the source of Vconn must not change to avoid disruption in power.

An example use case is a dongle with a charging port connected to a PMI8998 device. In this case, the PMI8998 would provide Vbus and Vconn initially, but the dongle would initiate a PR_SWAP if a user plugs in a charger to its charging port.

VCONN_SWAP

Vconn swap (VCONN_SWAP) changes the device providing Vconn power, either the Source or the Sink. An example use case is an active cable connected to a powered USB hub. Because the hub always has power, the phone requests the hub to power Vconn.

DR_SWAP

Data role swap (DR_SWAP) is the legacy USB equivalent of changing host and device roles. In USB Type-C and PD, this is called the upstream facing port (device or UFP) and downstream facing port (Host or DFP).

This command only swaps the UFP and DFP roles – DR_SWAP has nothing to do with the power flow direction.

FR_SWAP

Fast role swap (FR_SWAP) is conditionally supported by the PMI8998 because the timing requirement allows little time to execute the swap. If the current power source is suddenly removed, this command quickly swaps power roles.

The following table shows that if a powered USB hub provides Vbus and Vconn to all connected devices, FR_SWAP forces a connected phone to provide both of these power rails within 150 μ s if a user suddenly removes the power. This prevents interruption in data and video when the power removal event occurs. The following table summarizes USB PD swapping port behaviour:

Swap	Host/data roles	Rp/Rd	Vbus source/sink	Vconn source
PR_Swap	Unchanged	Swapped	Swapped	Unchanged
DR_Swap	Swapped	Unchanged	Unchanged	Unchanged
VCONN_Swap	Unchanged	Unchanged	Unchanged	Swapped*
* Swapping of VCONN source port				

2.6 USB OTG

In the USB OTG mode, the PMI8998 uses the battery as the input source and provides power to peripherals compliant with the USB OTG specification.

This feature is enabled by reversing the internal path and using the USB_IN input as an output, providing 5.0 V and up to 1500 mA.

This mode is hardware autonomous on PMI8998, and detection occurs in the Fuel Gauge module. Refer to *Linux PMIC Fuel Gauge Software User Guide (80-P2484-74)*.

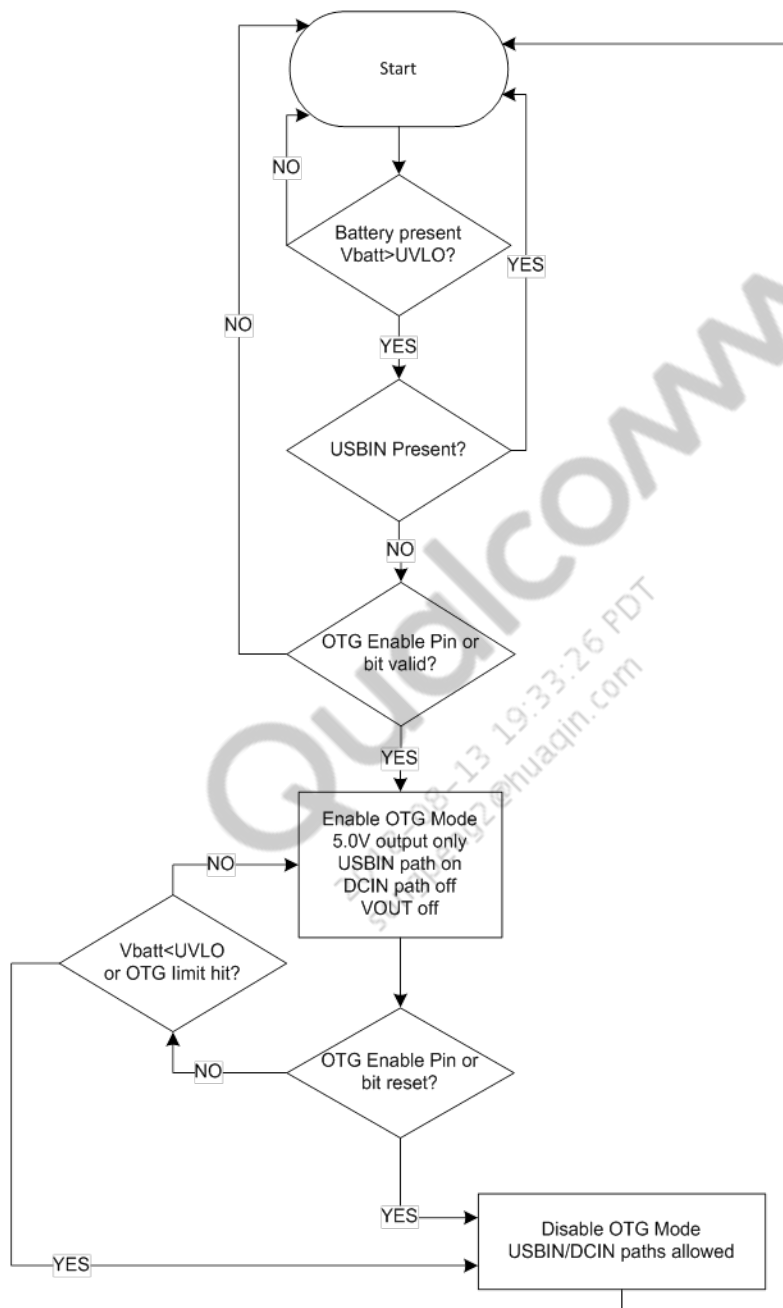


Figure 2-8 Reverse boost (OTG/HDMI/MHL) functional flowchart

2.6.1 Simultaneous WiPower charging and OTG

The PMI8998 supports OTG and charging operation from the USB_IN path, and charging only through the DC_IN path. Because PMI8998 uses the same existing power train for both OTG and charging, it is not possible to charge the battery and provide OTG power at the same time.

Because the PMI8998 includes a 5 V, 1 A boost converter to supply power to audio amplifiers, this is repurposed for OTG.

When the audio boost is configured for OTG operation, an external OVP/OCP chip must be used to protect the converter and provide current limiting. The configuration in the following figure enables simultaneous wireless (DC_IN) charging and OTG support, for applications that do not require the audio boost.

Software handles the following use cases:

- Interrupt issued when there is a change in Rid on PMI8998.
- OTG device detected, initiating the following sequence:
 - a. Enablement of the USB Suspend mode in PMI8998 which prevents the switcher from enabling when USB_IN is present.
 - b. Enablement of the audio boost.
 - c. Enablement of USB3OTG_VBUS_EN (GPIO5) to turn on the external OVP switch.
- Removal of OTG device detected, initiating the following sequence:
 - a. Disablement of USB3OTG_VBUS_EN (GPIO5) to turn off the external OVP switch.
 - b. Disablement of the audio boost.
 - c. Disablement the USB Suspend mode in PMI8998.

3 Software architecture and directory structure

The following charger software architecture block diagram shows interfaces between the user space, kernel, and hardware layers:

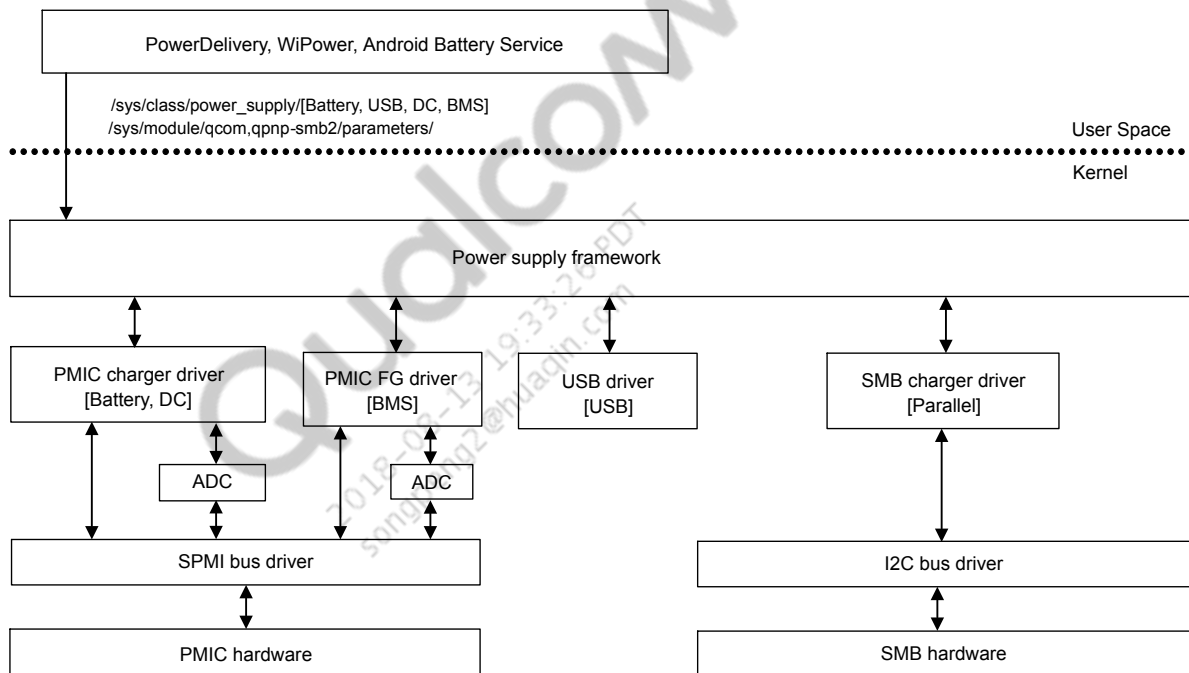


Figure 3-1 Charger software architecture block diagram

The Linux kernel charger driver resides in the following location:

```
/drivers/power/qcom-charger/qnp-smb2.c
```

The Linux kernel charger driver references the following SMB library that contains most of the base functions:

```
/drivers/power/qcom-charger/smb-lib.c
```

The software is modified via the device tree structure. The charger device tree node, `pmicobalt_charger`, is located in the following location:

```
/arch/arm/boot/dts/qcom/msm-pmicobalt.dtsi
```

3.1 Battery

The battery processes are available via sysfs at `/sys/class/power_supply/battery`.

Process	Description
status	Current charging status of the battery: <ul style="list-style-type: none"> ■ <code>POWER_SUPPLY_STATUS_DISCHARGING</code> – Battery is discharging ■ <code>POWER_SUPPLY_STATUS_CHARGING</code> – Battery is charging ■ <code>POWER_SUPPLY_STATUS_FULL</code> – Battery has reached a full charge
health	SDP online status: <ul style="list-style-type: none"> ■ <code>POWER_SUPPLY_HEALTH_OVERVOLTAGE</code> – Battery OVP is active ■ <code>POWER_SUPPLY_HEALTH_COLD</code> – Battery temperature is in the JEITA hard cold range and charging is disabled ■ <code>POWER_SUPPLY_HEALTH_COOL</code> – Battery temperature is in the JEITA soft cold range, JEITA float voltage and current compensation active if enabled ■ <code>POWER_SUPPLY_HEALTH_OVERHEAT</code> – Battery temperature is in the JEITA hard hot range and charging is disabled ■ <code>POWER_SUPPLY_HEALTH_WARM</code> – Battery temperature is in the JEITA soft hot range, JEITA float voltage and current compensation active if enabled ■ <code>POWER_SUPPLY_HEALTH_GOOD</code> – Battery temperature is in the normal range and is not overvoltage
present	Battery presence status
input_suspend	Suspend command (or suspend status) of USBIN charge path
charge_type	Current charging phase: <ul style="list-style-type: none"> ■ <code>POWER_SUPPLY_CHARGE_TYPE_NONE</code> – Battery is not charging ■ <code>POWER_SUPPLY_CHARGE_TYPE_TRICKLE</code> – Battery is charging in the precharge phase ■ <code>POWER_SUPPLY_CHARGE_TYPE_FAST</code> – Battery is charging in the Fastcharge phase ■ <code>POWER_SUPPLY_CHARGE_TYPE_TAPER</code> – Battery is charging in the taper charge phase
capacity	Current battery capacity state of charge (%SoC) as reported by the fuel gauge

3.2 USB

The USB processes are available at `/sys/class/power_supply/usb`.

Process	Description
present	VBUS charger presence status on USBIN
online	SDP online status
voltage_min	Lowest allowed VBUS voltage when USB Power Delivery is being used
voltage_max	Highest allowed VBUS voltage when USB Power Delivery is being used
voltage_now	Current VBUS voltage reading
current_max	Current USBIN input current limitation imposed by the <code>usb_icl_votable</code> vote aggregation

Process	Description
type	<p>Type of charger adapter detected (if present)</p> <ul style="list-style-type: none"> POWER_SUPPLY_TYPE_UNKNOWN – No charger POWER_SUPPLY_TYPE_USB – SDP POWER_SUPPLY_TYPE_USB_CDP – CDP POWER_SUPPLY_TYPE_USB_DCP – OCP is treated as DCP POWER_SUPPLY_TYPE_USB_DCP – Floated wall chargers treated as DCP POWER_SUPPLY_TYPE_USB_DCP – DCP POWER_SUPPLY_TYPE_USB_HVDCP – HVDCP2/QuickCharge 2.0 POWER_SUPPLY_TYPE_USB_HVDCP_3 – HVDCP3/QuickCharge 3.0
typec_mode	<p>Current USB Type-C mode for UFP or DFP of the PMI8998 (if Type-C is detected)</p> <p>PMI8998 in UFP mode:</p> <ul style="list-style-type: none"> POWER_SUPPLY_TYPEC_NONE – No Type-C present POWER_SUPPLY_TYPEC_SOURCE_DEFAULT – PMI8998 presenting UFP with a default power Type-C current capability detected POWER_SUPPLY_TYPEC_SOURCE_MEDIUM – PMI8998 presenting UFP with a Medium power 1.5 A Type-C current capability detected POWER_SUPPLY_TYPEC_SOURCE_HIGH – PMI8998 presenting UFP with a High power 3 A Type-C current capability detected <p>PMI8998 in DFP mode:</p> <ul style="list-style-type: none"> POWER_SUPPLY_TYPEC_NONE – No Type-C present POWER_SUPPLY_TYPEC_SINK_AUDIO_ADAPTER – PMI8998 presenting DFP for an audio adapter POWER_SUPPLY_TYPEC_SINK_DEBUG_ACCESSORY – PMI8998 presenting DFP for a debug accessory POWER_SUPPLY_TYPEC_SINK_POWERED_CABLE – PMI8998 presenting DFP for a powered cable POWER_SUPPLY_TYPEC_SINK – PMI8998 presenting DFP for an attached UFP Type-C device POWER_SUPPLY_TYPEC_SINK_POWERED_CABLE_ONLY – PMI8998 presenting DFP for an attached powered cable with no Type-C UFP device attached on the other cable end
typec_power_role	<p>VBUS charger presence status on:</p> <ul style="list-style-type: none"> USBIN POWER_SUPPLY_TYPEC_PR_NONE – No type-C present USBIN POWER_SUPPLY_TYPEC_PR_DUAL – PMI8998 is DFP USBIN POWER_SUPPLY_TYPEC_PR_SINK – PMI8998 is UFP USBIN POWER_SUPPLY_TYPEC_PR_SOURCE – PMI8998 is DFPF
typec_cc_orientation	<p>Indicates the orientation of the Type-C cable (if present) based on the PMI8998 CC_OUT pin:</p> <ul style="list-style-type: none"> 0 – No Type-C attached 1 – CC1 attached 2 – CC2 attached
pd_allowed	USB power delivery allowed status (not allowed if Quick Charge is detected because QuickCharge takes priority)
pd_active	

3.3 Parameters

The parameter processes are available at `/sys/module/qcom,qnp-smb2/parameters`.

Process	Description
<code>__debug_mask</code>	Enables debug message configuration. It is recommended to set this to value to 0xFF, which enables all debug messages.

3.4 Device tree settings

The device tree settings are available at `/documentation/devicetree/bindings/power/qcom-charger/qnp-smb2.txt`.

Setting	Description
<code>qcom,fcc-max-ua</code>	Specifies the maximum fast charge current in micro-amps (u32 value)
<code>qcom,usb-icl-ua</code>	Specifies the USB input current limit in micro-amps (u32 value)
<code>qcom,dc-icl-ua</code>	Specifies the DC input current limit in micro-amps (u32 value)
<code>qcom,suspend-input</code>	Boolean flag indicating that the charger must not draw current from any of its input sources (USB, DC). The value type is empty.

NOTE The `qcom,suspend-input` setting is in the default software device tree. This setting prevents all charging. To enable charging, remove this setting from the device tree.

4 Software drivers

4.1 BMD

PMI8998 provides three options for BMD (address – 70, register: BM_CFG[2:0]):

- BATT_THERM – Battery pack thermal monitor
- BATT_ID – Battery identification pin
- BMA – Algorithm that checks for battery terminal pin presence

When not used, BATT_THERM must be connected to GND for normal operation.

When not used (for ID detection or MIPI BIF), BATT_ID must be connected to GND for normal operation.

See [Battery missing detection](#) for more information.

4.2 Charger detection

Charger detection includes features active upon insertion and removal.

The PMI8998 has a dedicated USBIN_PLUGIN charger presence interrupt.

Upon charger insertion or removal detection, the `smblib_handle_usb_plugin()` interrupt handler runs. Within the VBUS presence detection interrupt handler, the `regulator_enable()` function requests USB_PHY to float D+/D- pins. This setting enables APSD detection, HVDCP detection, and authentication to be completed without interference, as shown in the figure.

Upon charger removal, the `smblib_handle_usb_plugin()` event handler calls `regulator_disable()` function to notify USB_PHY that the D+/D- pins can be managed. The charger presence status is stored in the `chg->vbus_present` parameter (see figure).

The USBIN_SOURCE_CHANGE interrupt triggers if there is a change in one of the following:

- APSD status
- HVDCP2 presence status
- HVDCP3 authentication

When the USBIN_SOURCE_CHANGE interrupt fires, the `smblib_handle_usb_source_change()` interrupt handler runs the `smblib_handle_apsd_done()` function to recheck and update the charger

type in software. The USB_PHY power supply is not updated until type detection is completed. See [Input current limits](#) for more information.

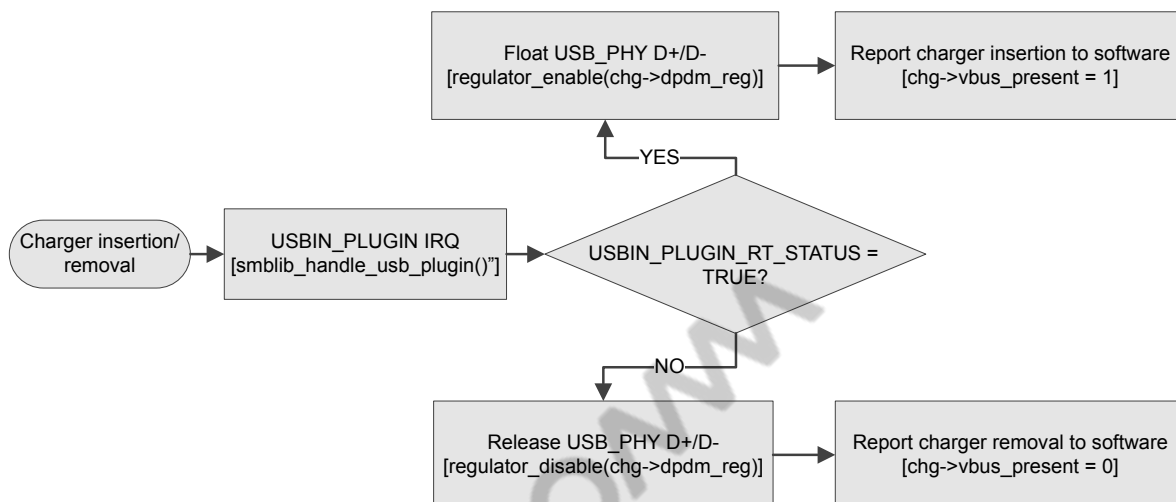


Figure 4-1 Charger presence detection flowchart

4.3 Input current limit

The USB and DC input current limits are configured from the device tree. This input current is lowered during runtime depending on a number of conditions, but the current is never allowed to exceed these device tree settings.

4.3.1 qcom,suspend-input

Boolean flag indicating that the charger must not draw current from its input sources (USB, DC).

Usage	Optional
Value type	<empty>

Example:

```

/arch/arm/boot/dts/qcom/msm-pmicobalt.dtsi
    pmicobalt_charger: qcom,qnp-smb2 {
        compatible = "qcom,qnp-smb2";
        #address-cells = <1>;
        #size-cells = <1>;
        qcom,pmic-revid = <&pmicobalt_revid>;
        /* do not draw current from USB or DC */
        qcom,suspend-input;
        dpdm-supply = <&qusb_phy0>;
    }
  
```

The qcom,suspend-input setting is in the default software device tree for pre-ES2. This setting prevents all charging, to enable charging, remove this from the device tree.

4.3.2 qcom,usb-icl-ua

Specifies the USB input current limit in micro-amps.

Usage	Optional
Value type	<u32>

Example:

```
/arch/arm/boot/dts/qcom/msm-pmicobalt.dtsi
    pmicobalt_charger: qcom,qnp-smb2 {
        compatible = "qcom,qnp-smb2";
        #address-cells = <1>;
        #size-cells = <1>;
        qcom,pmic-revid = <&pmicobalt_revid>;
        /* do not draw current from USB or DC */
        qcom,usb-icl-ua = <3000000>;//sets the USBIN limit to 3A
        dpdm-supply = <&qusb_phy0>;
```

4.3.3 qcom,dc-icl-ua

Specifies the DC input current limit in micro-amps.

Usage	Optional
Value type	<u32>

Example:

```
/arch/arm/boot/dts/qcom/msm-pmicobalt.dtsi
    pmicobalt_charger: qcom,qnp-smb2 {
        compatible = "qcom,qnp-smb2";
        #address-cells = <1>;
        #size-cells = <1>;
        qcom,pmic-revid = <&pmicobalt_revid>;
        /* do not draw current from USB or DC */
        qcom,dc-icl-ua = <3000000>;//sets the DCIN limit to 3A
        dpdm-supply = <&qusb_phy0>;
```

4.3.4 Type-C

Input current limitations based on Type-C cable capability detection is done completely in hardware. The type-c-change interrupt notifies software of Type-C detection. The `smblib_handle_usb_typec_change()` interrupt handler runs if there is a change in Type-C presence detection. The following figure outlines the code flow for Type-C detection. Hardware limits current.

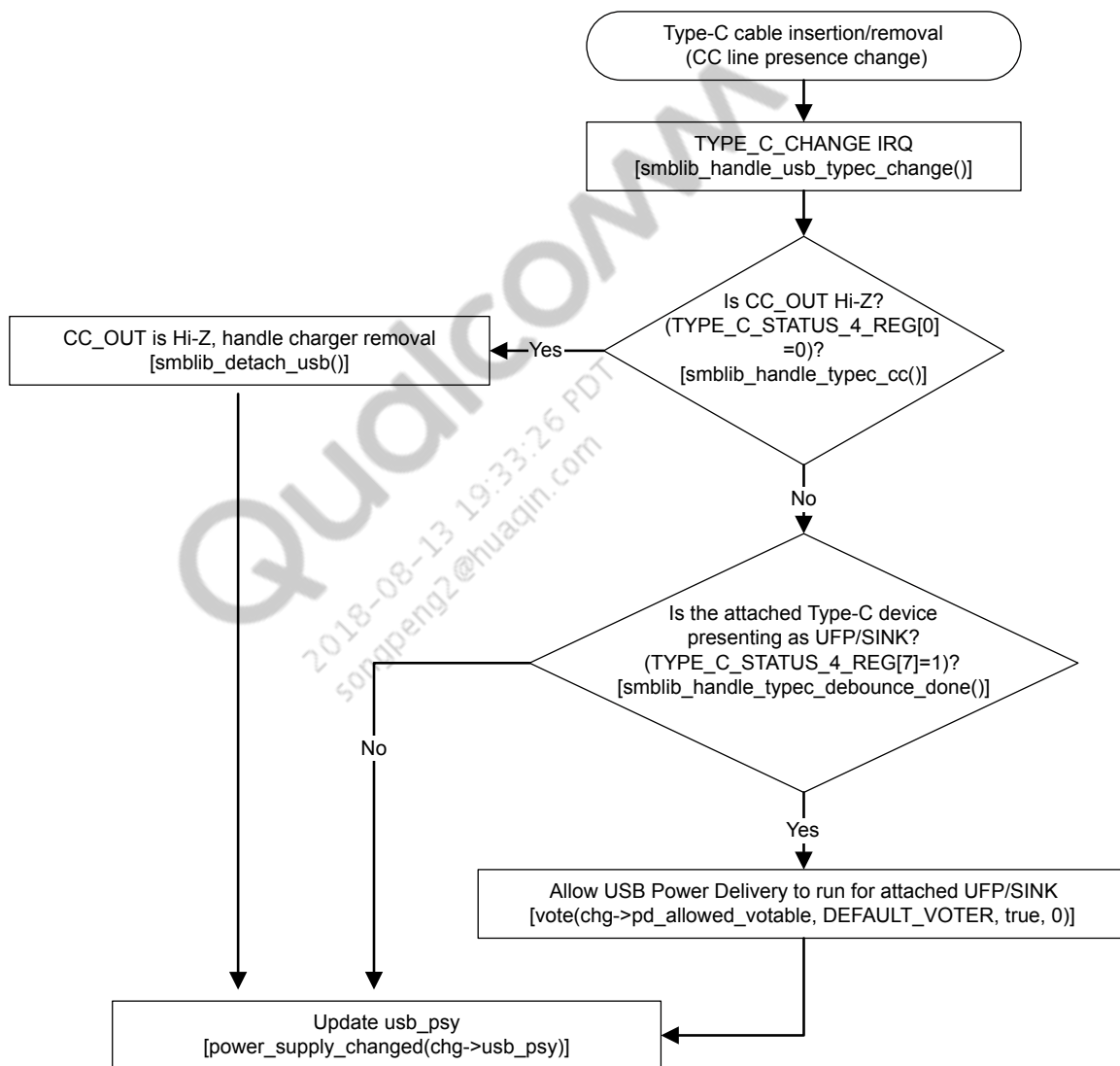


Figure 4-2 Type-C detection flowchart

4.4 Battery current limit

The `smblib_handle_batt_psy_changed()` interrupt handler handles each battery condition interrupt. The `smblib_handle_batt_psy_changed()` interrupt handler pokes the battery power supply to recheck its properties and update accordingly.

Interrupt	Description
bat-temp	JEITA soft or hard event
bat-ocp	Battery over current condition
bat-ov	Battery over voltage condition
bat-low	Low battery condition
bat-therm-or-id-missing	Battery missing detection, pin based
bat-terminal-missing	Battery missing detection, terminal based

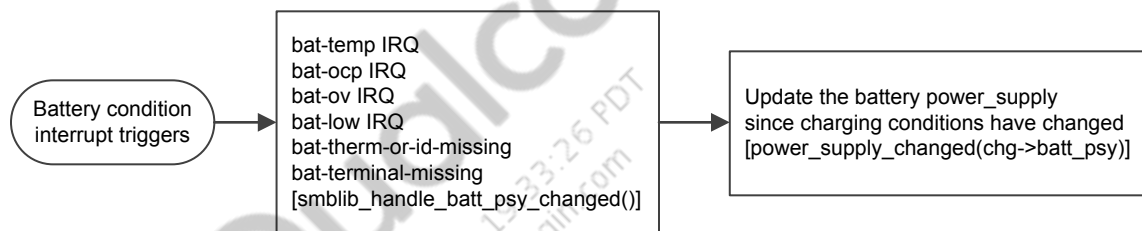


Figure 4-3 Battery condition detection flowchart

4.4.1 qcom,fcc-max-ua

Specifies the maximum fast charge current in micro-amps.

Usage	Optional
Value type	<u32>

Example:

```

/arch/arm/boot/dts/qcom/msm-pmicobalt.dtsi
    pmicobalt_charger: qcom,qnp-smb2 {
        compatible = "qcom,qnp-smb2";
        #address-cells = <1>;
        #size-cells = <1>;
        qcom,pmic-revid = <&pmicobalt_revid>;
        /* do not draw current from USB or DC */
        qcom,fcc-max-ua = <3000000>; //sets the FCC limit to 3A
        dpdm-supply = <&qusb_phy0>;
    }
  
```

4.5 Parallel charging

Parallel charging between PMI8998 and SMB1381 is mostly hardware autonomous. Software is responsible for splitting the FCC between the two charger ICs.

4.6 OTG and VCONN

The qnp-smb2.c charger driver for the PMI8998 provides the following regulator structures for supplying VBUS for OTG or for Type-C when a UFP is attached, and for supplying VCONN if needed for Type-C.

- VBUS regulator – qcom,smb2-vbus
- VCONN regulator – qcom,smb2-vconn

4.6.1 Example device tree

```
/arch/arm/boot/dts/qcom/msmcobalt-regulator.dtsi
&pmicobalt_charger {
    smb2_vbus: qcom,smb2-vbus {
        regulator-name = "smb2-vbus";
    };
    smb2_vconn: qcom,smb2-vconn {
        regulator-name = "smb2-vconn";
    };
}

/arch/arm/boot/dts/qcom/msm-pmicobalt.dtsi
pmicobalt_pdphy: qcom,usb-pdphy@1700 {
    compatible = "qcom,qnp-pdphy";
    reg = <0x1700 0x100>;
    vdd-pdphy-supply = <&pmicobalt_124>;
    vbus-supply = <&smb2_vbus>;
    vcon-supply = <&smb2_vconn>;
}
```

For information on Linux kernel regulator controls, refer to *MSM8998.LA Linux Voltage Regulator Software User Guide* (80-P2484-79).

4.6.2 VBUS and VCONN enablement

The functions in this section run when VBUS or VCONN are enabled to be supplied by the PMI8998 for OTG or Type-C. Because these features are largely hardware-agnostic, the software configures the VBUS or VCONN enable bits.

Enable or disable the VBUS supply for OTG or Type-C (PMI8998 as DFP)

```
smblib_vbus_regulator_enable()
```

Enables the OTG_EN bit in the OTG_CMD_OTG register (0x1140[0]=1).

```
smblib_vbus_regulator_disable()
```

Disables the OTG_EN bit in the OTG_CMD_OTG register (0x1140[0]=0).

Enable or disable VCONN Supply for Type-C

```
smblib_vconn_regulator_enable()
```

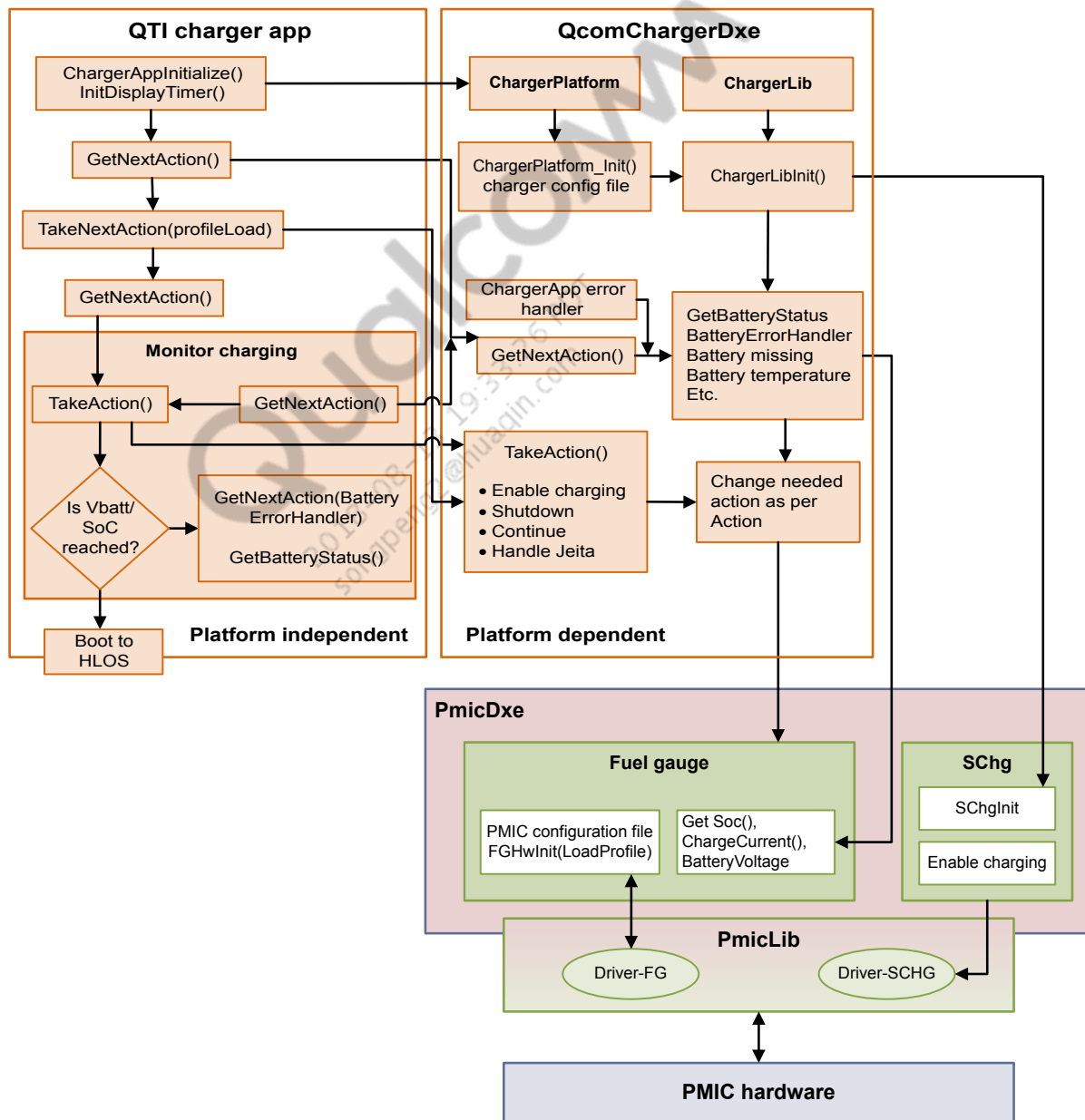
Enables the VCONN_EN bit in the TYPE_C_INTRPT_ENB_SOFTWARE_CTRL register (0x1368[3]=1).

```
smblib_vconn_regulator_disable()
```

Disables the VCONN_EN bit in the TYPE_C_INTRPT_ENB_SOFTWARE_CTRL register (0x1368[3]=0).

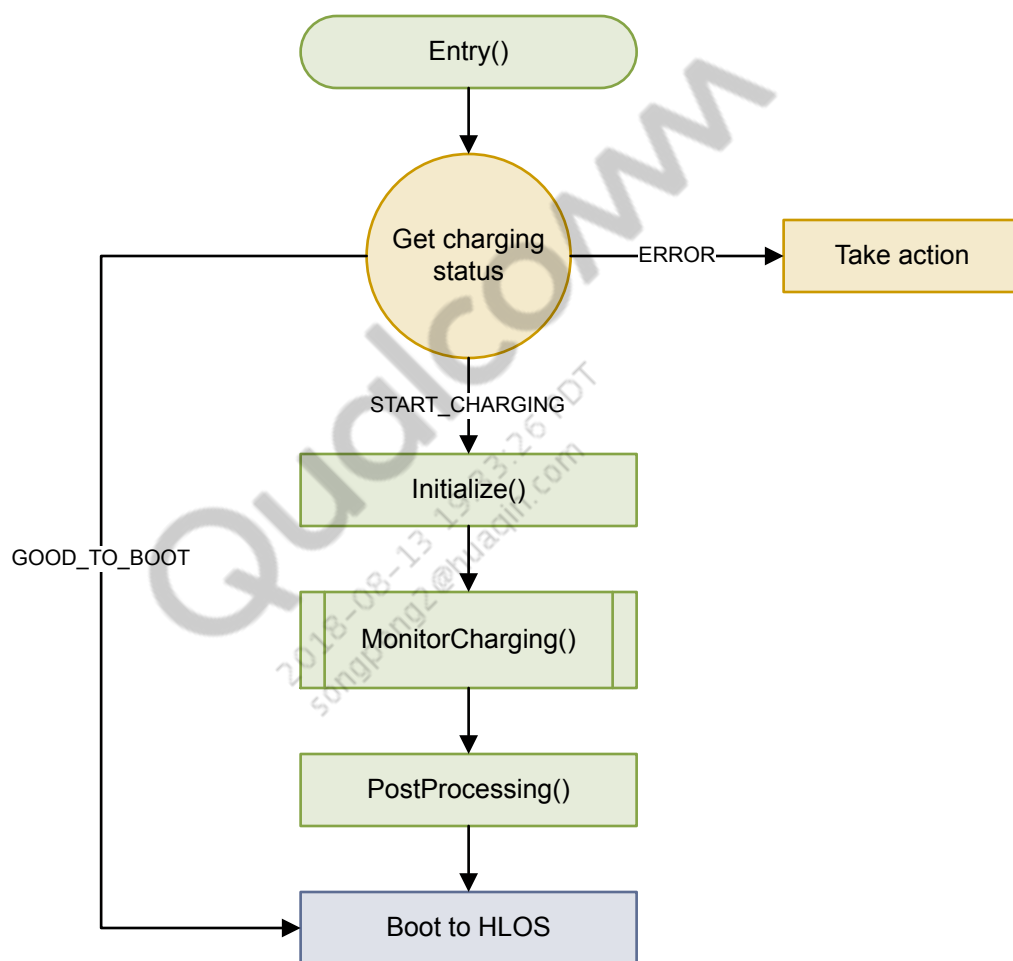
4.7 UEFI charger app

The following is a diagram of the UEFI charger app design:



4.7.1 QTI charger app

The QTI charger app supports voltage-based and SoC-based threshold charging. The QTI charger app calls ChargerLib to determine if it is OK to boot to HLOS or whether to stay in UEFI charging until the threshold is reached. The QTI charger app has one main loop that keeps the application running, checking if it is OK to exit. The following figure shows the QTI charger app flow:



The charger has a timer that performs the following tasks:

- Turns the display ON/OFF
- Displays the charging/battery icon
- Runs the following in a loop every 3 sec until threshold is reached:
 - Check battery status
 - Manage charging
 - Error handling

Additional support includes safety features, such as thermal mitigation, WiPower, etc.

The following table describes the QTI charger app functions:

Source file	Function	Description
QcomChargerApp.c	Entry()	Initializes the QTI charger app, starts the charging loop, and cleans up after exiting the charging loop
	Initialize()	Sets the charging parameters and starts the display timer
	MonitorCharging()	Charging loop that monitors battery charging status until battery status is at a good enough charge to boot to HLOS
	PostProcessing()	Called after exiting the charging loop; closes all events and timers, and decides whether to boot to HLOS, stay in UEFI, or shut down
	DeInitialize()	Performs required exit actions before leaving the app
QcomChargerApp-EventHandler.c	KeyPressEventHandler()	Turns on the display, starts the display timer, and exits LPM mode when the volume up/down, power key, or home key is pressed
	KeyPressControl()	Registers or unregisters volume up, volume down, and home key press event callback with the keypad driver
	HandleLPMClock()	Signals the clock driver to enter or exit LPM
	HandleLPMDisplay()	Signals the display driver to enter or exit LPM to turn the display ON or OFF
	EnterLPM()	Called when the display is turned off; sends an enter LPM signal to clock and display drivers
	ExitLPM()	Called when the display is turned on by key press; sends an exit LPM signal to the clock and display drivers
	AnimImgTimer()	Animates image display during charging (display ON)
	DisplayTimerEvent()	Displays the battery image during charging (display ON)

4.7.2 PmicDxe driver

The PmicDxe driver implements functions to program the charger and fuel gauge (FG) hardware. It also implements such functions as parsing/load battery profile, FG boot sequence, AICL rerun, and so on.

4.7.3 ChargerLib

The platform-dependent ChargerLib provides the following functions:

- Enables/disables charging
- Sets charger maximum battery current
- Float voltage (FV)
- Checks charger source

- Gets battery state of charge (SoC), voltage, and current
- Handles battery errors and software Jeita

Qualcomm
2018-08-13 19:33:26 PDT
songpeng2@huawei.com

5 Charger configuration

5.1 Multilevel JEITA charging

The multilevel JEITA software solution samples the battery temperature periodically for JEITA compensation. If the software-based JEITA solution is enabled, hardware-based JEITA compensation is disabled.

To configure JEITA charging, update the table based on the battery profile provided by the battery vendor.

- Temperature ranges are programmed in the hardware.
- Range data must be in increasing ranges and must not overlap.
- It is acceptable if the boundaries of the ranges do not coincide, as long as the ranges do not overlap.

To enable multilevel JEITA charging, set the Boolean flag `qcom,sw-jeita-enable` to enabled. Additionally, modify the `jeita_fcc_cfg` and `jeita_fv_cfg` structs.

struct jeita_fcc_cfg

This structure is used to the FCC in different JEITA regions.

Type	Parameter	Description
u32	psy_prop	Defines the power supply property to be read
char	*prop_name	Name of the property
int	hysteresis	Enables switching for falling temperature
struct range_data	fcc_cfg	Maximum number of levels supported as defined by the <code>MAX_STEP_CHG_ENTRIES</code> macro

Example – Defining JEITA ranges

```
static struct jeita_fcc_cfg jeita_fcc_config = {
    .psy_prop    = POWER_SUPPLY_PROP_TEMP,
    .prop_name    = "BATT_TEMP",
    .hysteresis   = 10, /* 1degC hysteresis */
    .fcc_cfg      = {
        /* TEMP_LOW      TEMP_HIGH      FCC */
        {0, 100, 600000}, // 0 -10 degC
        {101, 200, 2000000}, // 10.1 - 20 degC
        {201, 450, 3000000}, // 20.1 - 45 degC
    }
}
```

```

        {451, 550, 600000}, // 45.1 - 55 degC
    },
};

```

struct jeita_fv_cfg

This structure is used to control Float voltage in various JEITA regions.

Type	Parameter	Description
u32	psy_prop	Defines the power supply property to be read
char	*prop_name	Name of the property
int	hysteresis	Enables switching for falling temperature
struct range_data	fv_cfg	Maximum number of levels supported are defined by the MAX_STEP_CHG_ENTRIES macro

Example – Setting JEITA float voltage

```

static struct jeita_fv_cfg jeita_fv_config = {
    .psy_prop    = POWER_SUPPLY_PROP_TEMP,
    .prop_name   = "BATT_TEMP",
    .hysteresis  = 10, /* 1degC hysteresis */
    .fv_cfg      = {
        /* TEMP_LOW    TEMP_HIGH    FV */
        {0, 100, 4200000}, // 0 -10 degC
        {101, 450, 4400000}, // 10.1 - 45 degC
        {451, 550, 4200000}, }, // 45.1 - 55 degC
    },
};

```

5.2 Step charging

SoC-based step charging is supported by default. To enable voltage-based step charging, set the Boolean flag `qcom, step-charging-enable` to `TRUE`.

struct step_chg_cfg

This structure is used to control the FCC in different step charging regions defined by SoC or battery voltage thresholds.

Type	Parameter	Description
u32	psy_prop	Defines the power supply property to be read
char	*prop_name	Name of the property
int	hysteresis	Enables switching for falling temperature
struct range_data	fcc_cfg	Maximum number of levels supported are defined by the MAX_STEP_CHG_ENTRIES macro

Example – Vbat-based step charging

```
static struct step_chg_cfg step_chg_config = {
    .psy_prop      = POWER_SUPPLY_PROP_VOLTAGE_NOW,
    .prop_name     = "VBATT",
    .hysteresis    = 100000, /* 100mV */
    .fcc_cfg       = {
        /* VBAT_LOW      VBAT_HIGH      FCC */
        {3600000,      4000000,      3000000}, // 3.6V - 4.0 V
        {4001000,      4200000,      2800000}, // 4.1V - 4.2 V
        {4201000,      4400000,      2000000}, // 4.2V - 4.4 V
    },
};
```

Example – SoC-based step charging

```
static struct step_chg_cfg step_chg_config = {
    .psy_prop      = POWER_SUPPLY_PROP_CAPACITY,
    .prop_name     = "SOC",
    .fcc_cfg       = {
        //SOC_LOW      SOC_HIGH      FCC
        {20,          70,          3000000}, // 20% - 70%
        {70,          90,          2750000}, // 70% - 90%
        {90,          100,         2500000}, // 90% - 100%
    },
};
```

6 Dead battery recovery

PMI8998 enables portable devices to start the charging when the battery voltage is below a low (dead) battery threshold without booting the system to HLOS. The battery must have enough charge to handle the peak currents during the power up, which results in a power cycle.

Dead battery recovery takes place in two stages – XBL and UEFI.

XBL

- Battery charges to the bootup threshold
- Sign of life is LED
- Bootup battery threshold must be chosen to power up to UEFI where the display is ON

UEFI

- Battery charges to the power-up threshold
- Signs of life are display and LED
- Power up threshold must be chosen to handle peak current during power up

6.1 Dead battery recovery in XBL

Relevant files for XBL (new)

- Common driver – pmic\driver\schg\
- SBL charging app driver APIs – pmic\app\chg\

XBL configuration parameters

The configuration parameters for dead battery recovery in XBL are defined in pm_config_target.c.

Parameter	Description
sbl_schg_specific_data	<p>The following parameters are defined as charger-specific data:</p> <ul style="list-style-type: none"> ▪ Vlowbatt threshold ▪ APSD rerun Vlowbatt threshold ▪ FG skin hot threshold – Valid range -30 to 97 °C ▪ FG skin too hot threshold – Valid range -30 to 97 °C ▪ FG charge hot threshold – Valid range -30 to 97 °C ▪ FG charge too hot threshold – Valid range -30 to 97 °C ▪ Use BATID and/or THERM pin for battery missing detection ▪ Enable or disable and timeout watchdog configuration ▪ FAST charging current ▪ PRE charge current ▪ Float voltage ▪ USBIN input current limit ▪ DCIN input current limit ▪ Battery thermistor ground selection ▪ Aux thermistor ground selection ▪ Debug board detect – BATT_ID pulldown resistor ADC Min/Max read value range in Ohms ▪ Bootup battery threshold (mV) ▪ WiPower bootup battery threshold (mV) ▪ APSD reset threshold (mV) – APSD reset only applicable if initial the Vbat level is less than this threshold ▪ APSD reset threshold (no_uvlo in mV) – Used if the last reset reason is NOT UVLO ▪ APSD reset threshold (mV) – Used if last reset reason is UVLO ▪ Enable or disable JEITA hard temperature limit check in SBL ▪ DBC USB 500 mode ▪ Verbose SBL CHG UART logging
sbl_schg_wipower_specific_data	<ul style="list-style-type: none"> ▪ Enable or disable WiPower charger power – If enabled, valid range in uWatt = 5000000 to 20000000 ▪ Enable or disable the wait time <ul style="list-style-type: none"> □ If disabled, use the wait time in which the default minimum value is 3500 ms and has no maximum value □ If enabled, wait forever

Parameter	Description
<code>sbl_schg_jeita_threshold</code>	<ul style="list-style-type: none"> ■ Enable or disable JEITA hard cold threshold – If enabled, default = 0x3C, but configurable ■ Enable or disable JEITA soft cold threshold – If enabled, default = 0x50, but configurable ■ Enable or disable JEITA soft hot threshold – If enabled, default = 0x96, but configurable ■ Enable or disable JEITA hard hot threshold – If enabled, default = 0xAA, but configurable
<code>pm_sbl_schg_batt_therm_type</code>	<p>If enabled:</p> <ul style="list-style-type: none"> ■ Battery critical low temperature limit -20 °C ■ Battery critical high temperature limit 80 °C ■ Thermistor_c1_coeff – Default = 0xA1 ■ Thermistor_c2_coeff – Default = 0x50 ■ Thermistor_c3_coeff – Default = 0xFF ■ Thermistor_c1_coeff – Default = 0xBF ■ Thermistor_c2_coeff – Default = 0x36 ■ Thermistor_c3_coeff – Default = 0xFF

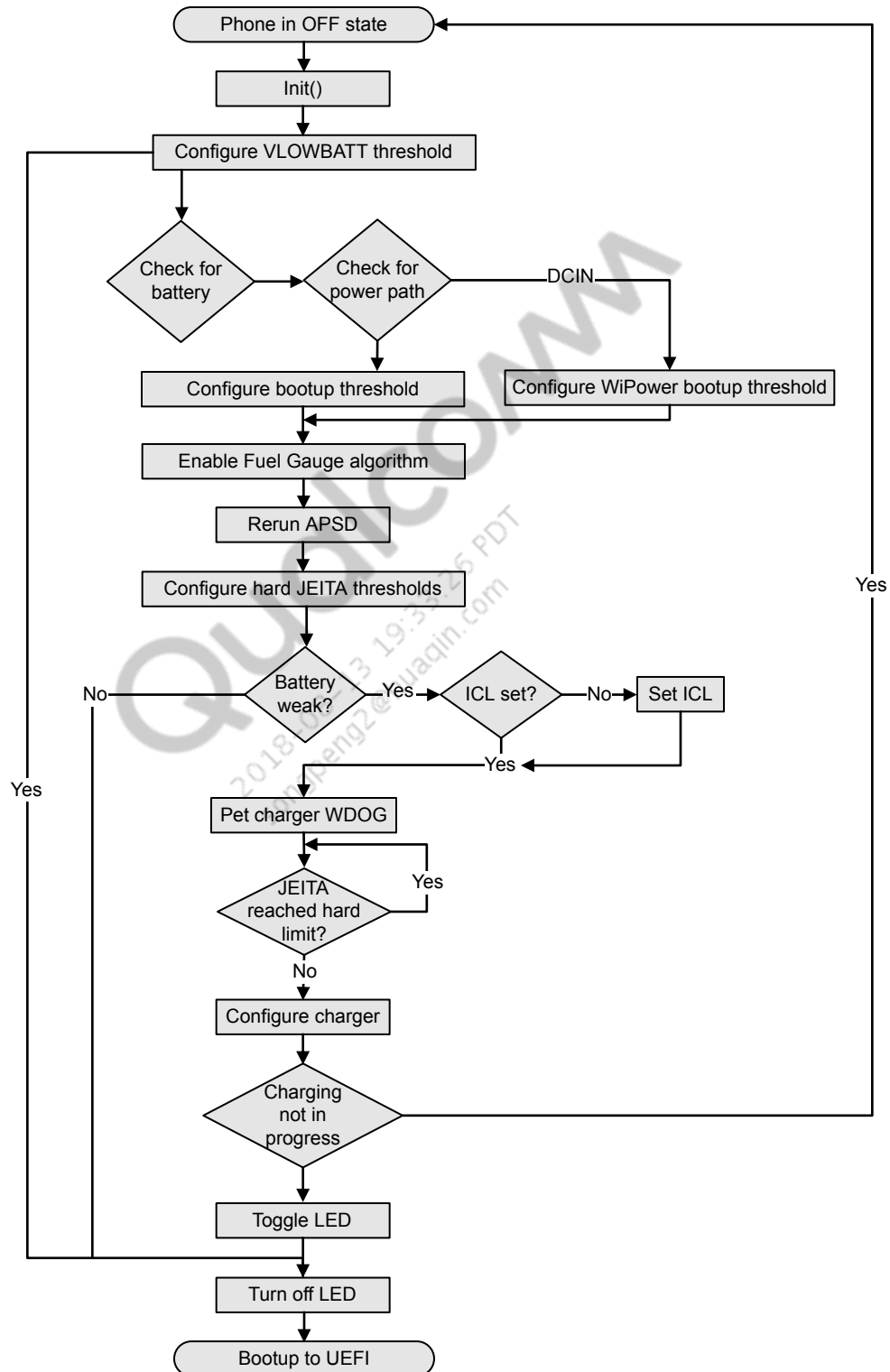
SBL charging flow

SBL charging check starts on `pm_sbl_chg_init()` which is defined in `pm_sbl_boot.c`.

The following are executed in the PMIC SBL charging initialization:

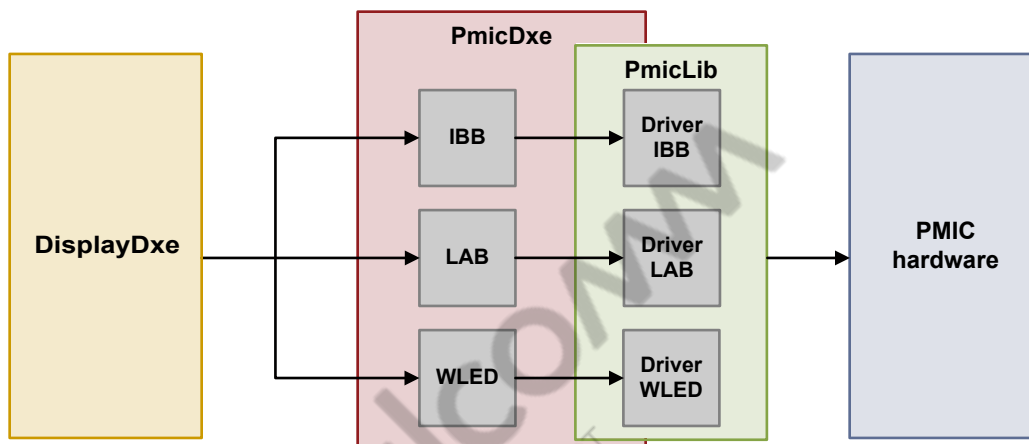
1. Checks if PMIC is supported.
2. Executes the PMIC SBL charging preinitialization.
3. Checks for auto power on.
4. Enables BMD and disable battery ID conversion.
5. Configures the fake battery ID detection range.
6. Checks if the debug board is in use. If the debug board is in use, enable temperature monitoring.
7. Configures charging parameters.
8. Configures JEITA parameters:
 - Thermistor coefficient
 - Charger and skin temperature thresholds
 - BATT thermistor and AUX thermistor ground selection

Upon initialization, SBL checks for battery charge status as shown in the following diagram:



6.2 Dead battery recovery in UEFI

UEFI for dead battery recovery enables using the display as the sign of life. The following diagram shows driver interaction in UEFI for the display:



For more information, refer to *UEFI PMIC Software User Guide* (80-P2484-42).

QcomChargerApp

QComChargerApp manages the following tasks:

- Charging the battery to a specific level
- Determining whether to boot to HLOS – If the battery is at a specific level, dead battery recovery in UEFI can be skipped

QComChargerApp displays sign of life to indicate the following:

- Battery status indicating charging status
- Low or normal battery level

Additional features can be supported in UEFI dead battery recovery by configuring safety features such as thermal mitigation.

RELATED INFORMATION

[“QTI charger app” on page 40](#)

[“Configuration quick reference table” on page 51](#)

6.2.1 Configuration quick reference table

The following table describes configuration parameters:

Parameter	Description	Default	Dependency	Data format
Threshold charging				
SocOrVoltageBaseBoot	Sets the thresholds in QTI charger app to allow boot when the configured threshold is met	FALSE	LoadBatteryProfile	Boolean
LoadBatteryProfile	Loads the battery profile for battery SoC accuracy	TRUE	DispSignOfLifeMaxThresholdMv	Boolean
DispSignOfLifeMaxThresholdMv	Displays the sign of life during battery profile load, providing a user indication for cold boot	3700 mV	LoadBatteryProfile	mV
FgCondRestart	Decides if the FG must be restarted to allow battery accuracy	TRUE	LoadBatteryProfile	Boolean
ChargerLedConfig	Configures the charger LED status	1	None	Decimal
EnShipMode	If enabled, ChargerApp gets variable value and, if found set, puts device into ShipMode at UEFI level	FALSE	'ShipModeVarStr = ShipMode' Needs UEFI variable service to be working upon reboot	Boolean
Debug				
PrintChargerAppDbgMsg	If TRUE, enables QTI charger app logs and respective pmic.dxe charger logs, and displays them on UART	FALSE	None	Boolean
PrintChargerAppDbgMsgToFile	If TRUE, enables QTI charger app file logs and respective pmic.dxe charger logs, and saves them to a file	FALSE	LogFS partition availability	Boolean
FileLoggingDbgLevelMask	Debug level mask (in hex) for file logging	80000042	None	Bit flag
EnableChargerFGDump	If TRUE, enables PMIC charger and FG peripheral dumps	FALSE	PrintChargerAppDbgMsg PrintChargerAppDbgMsgToFile	Boolean

Parameter	Description	Default	Dependency	Data format
FG and FG debug				
BatteryIdTolerance	Reads the battery ID to load the battery profile and set a tolerance limit	8%	LoadBatteryProfile	8% ± on current battery ID reading
DumpSram	Enables QTI charger app FG SRAM dumps	FALSE	PrintChargerAppDbgMsg PrintChargerAppDbgMsgToFile	Boolean
DumpSramStartAddr	SRAM dump start address (values in decimal)	0	DumpSram	Decimal
DumpSramEndAddr	SRAM dump end address (values in decimal)	124	DumpSram	Decimal
DumpSramDuration	Dump SRAM contents timer duration in seconds	90 sec	DumpSram	Seconds
Required initially				
BootToHLOSThresholdInMv	Configures the QTI charger app threshold to allow boot	3600 mV	None	mV
OsStandardBootSocThreshold	Configures the QTI charger app minimum threshold to allow boot	7	LoadBatteryProfile	1 to 100% SoC
BattVoltLimHighDelta	Enables delta FV and current limit to charge the battery	30 mV	None	mV
ChgFvMax	Enables battery FV	4350 mV	None	mV
ChgFccMax	Enables fast charging current	2000 mA	None	mA
ChargingTermCurrent	Enables charger termination current to declare 100% SoC	200 mA	None	mA
ConservChgFvDelta	Enables maximum charger delta FV maximum for unknown battery configurations	200 mV	None	mV
BATT_THERM coefficients	Enables battery thermal coefficients to read the battery temperature accuracy	BATT_THERM coefficients	ProgramBattThermCoeffs=TRUE	Hexidecimal

Parameter	Description	Default	Dependency	Data format
BATT_THERM configs	Enables battery thermal bias wait and ground select configs; supported configs parameters are: <ul style="list-style-type: none"> ▪ BattThermBiasWait ▪ BattThermGndSel 	BATT_THERM configs	None	BATT_THERM configs
AUX_THERM coefficients	Enables auxiliary thermal coefficients to adjust voltage to the temperature mapping	AUX_THERM coefficients	ProgramAuxThermCoeffs=TRUE	Hexidecimal
AUX_THERM configs	Enables battery auxiliary thermal bias wait and ground select configs; supported configs parameters are: <ul style="list-style-type: none"> ▪ AuxThermBiasWait ▪ AuxThermGndSel 	AUX_THERM configs	None	AUX_THERM configs
Device skin and charger hot thresholds	Enables configuring device skin and charger hot thresholds	Device skin and charger hot thresholds	ProgramSkinAndChargerHot Threshold = TRUE	Hexidecimal
Charger_THERM source configs	Enables/disables charger thermal source	<ul style="list-style-type: none"> ▪ SkinTempSrc = TRUE ▪ DieTempSrc = TRUE ▪ DieTempCompSrc = TRUE 	None	BOOLEAN
EmergencyShutdownVbatt	Configures the device emergency shutdown limit	3200 mV	None	mV
EnableChargerWdog	Enables the charger watchdog to safeguard unintentional charging if the software gets stuck	TRUE	None	Boolean
VBtEmpty threshold	Configures the low battery voltage threshold for the SoC empty interrupt	2800 mV	None	mV
VBattEstDiffThreshold	Configures the estimated voltage difference threshold to restart the FG if the threshold difference is higher	30 mV	FgCondRestart	mV
RConn compensation resistance	Configures RConn compensation resistance; value in mOhms - range is ± 100 mOhms	0 ohms	None	Ohms

Parameter	Description	Default	Dependency	Data format
Battery error handling				
DebugBoardBatteryIdMin and DebugBoardBatteryIdMax	Specifies the debug board battery ID range	2000 to 14000	DebugBoardBehavior	Ohms
SmartBatteryIdMin and SmartBatteryIdMax	Specifies the smart battery ID range	240000 to 450000	None	Ohms
RegularBatteryIdMin and RegularBatteryIdMax	Specifies the regular battery ID range	15000 to 137000	None	Ohms
UnknownBatteryBehavior	Defines unknown battery behavior detects if the battery ID is within the specified range	BattID < 2000 and ID > 450000	Battery error handling configuration	Ohms
DebugBoardBehavior	Defines debug board battery behavior	2	Battery error handling configuration	Decimal
BattMissingCfg	Configures the battery missing detection behavior	0	None	Decimal
Jeita				
Jeita zones	Enables specified Jeita zones	Jeita configuration	None	Celsius
JeitaCcCompCfg	Enables configuring Jeita charge current compensation when device is within the battery temperature soft limit for hardware Jeita	1000 mA	None	mA
JeitaFvCompCfg	Enables configuring Jeita charge voltage compensation when device is in battery temperature soft limit for hardware Jeita	105 mV	None	mV
NoChargeAndWait	Configures device behavior for temperatures outside of the charging range but within the operating range	TRUE	None	Boolean
WiPower				
WiPowerSupported	Enables configuring WiPower support for the QTI charger app charging device	TRUE	None	Boolean
DCInBootToHLOSThresholdInMv	Enables configuring the WiPower threshold for QTI charger app charging	3600 mV	None	mV

Parameter	Description	Default	Dependency	Data format
SuspendDCIn	Enables configuring suspended DCIN behavior when WiPower is enabled	FALSE	None	Boolean
Thermal				
SWThermalMitigationEnable	Configures thermal safety mitigation in the QTI charger app charging device	FALSE	None	Boolean
TsensTimeoutMins	Configures a thermal safety timer when the device is in the thermal zone and not in charger wait state	30 min	None	Minutes
Tsens limits or zone	Configures thermal safety zones or limits	Tsens limits or zone	None	Celsius
See QTI charger app configuration file for LA (example) for a QTI charger app LA example file.				

7 UEFI QTI charger app configuration

The UEFI QTI charger app handles the following:

- Charges the battery to a specified level
- Determines if the battery is OK to boot to HLOS without UEFI charging

See [QTI charger app](#) for an overview of the QTI charger app.

7.1 QTI charger app configuration file

The QTI charger app includes a build-time configuration file with parameters for charger/FG management.

See [QTI charger app configuration file for LA \(example\)](#) for an example file.

The configuration file is located in the following:

`/QcomPkg/Drivers/QcomChargerDxe/QcomChargerConfig_VbattTh_<chipset>.cfg`
<chipset> corresponds to the applicable chipset product. For example: `/QcomPkg/Drivers/QcomChargerDxe/QcomChargerConfig_VbattTh_SDM845.cfg`.

The configuration file must be renamed `QcomChargerCfg.cfg` to be used.

7.2 Threshold charging configuration

The following sections describe threshold charging configuration parameters. See [QTI charger app configuration file for LA \(example\)](#) for a QTI charger app LA example file.

7.2.1 SocOrVoltageBaseBoot

This parameter sets the thresholds in the QTI charger app to allow boot when the configured threshold is met. The threshold can be voltage- or SoC-based.

- Default value – FALSE (indicates the voltage-based thresholds charging)
- To allow accurate SoC estimates during charging, this parameter must be set to the [LoadBatteryProfile](#) configuration.

7.2.2 LoadBatteryProfile

This parameter loads the battery profile for battery SoC accuracy. If enabled, the QTI charger app loads profile data to the FG first.

- Default value – TRUE (indicates voltage-based thresholds charging)
- This configuration might add a delay of ~1.5 sec to boot while the profile loads and the SoC estimate is calculated. [DispSignOfLifeMaxThresholdMv](#) configuration shows sign of life while loading the battery profile. This delay/wait only applies to cold boot or battery removal.
- The FG SRAM profile integrity status dedicated register contains the profile load and FG restart status.

7.2.3 DispSignOfLifeMaxThresholdMv

This parameter displays the sign of life during battery profile load, providing a user indication for cold boot. If the [LoadBatteryProfile](#) parameter is enabled, the QTI charger app shows sign of life and loads the profile first.

- Default value – 3700 mV (used to decide on displaying image)
- Dependency – [LoadBatteryProfile](#) must be set for an accurate SoC estimate during charging.

7.2.4 FgCondRestart

This parameter decides if the fuel gauge must be restarted to allow battery accuracy for the following condition:

```
If abs(Vbatt_Estimate_diff ) > Vbatt_Estimate_diff_threshold
```

- Default value – TRUE
- Dependency – [LoadBatteryProfile](#) must be set.

7.2.5 ChargerLedConfig

This parameter configures the charger LED status. If enabled, the LED turns off after threshold charging is complete, that is, when the device boots to HLOS. Supported values are:

- 0 – Disable
- 1 (default) – Solid during charging
- 2 – LED blinks during charging

7.2.6 EnShipMode

If this parameter is enabled based on 'ShipModeVarStr = ShipMode', ChargerApp gets the variable value and, if found set, puts the device into ShipMode at the UEFI level.

- Default value – FALSE
- Dependency – ShipModeVarStr must be set to ShipMode by HLOS or other tool/module used by OEM to set the device to ShipMode in factory. Once set, the device is put into ShipMode when

rebooted (until end user restarts in mission mode). This parameter needs UEFI variable service to be working upon reboot.

- Use ShipMode string to query variable status.

7.3 Debug configuration

The following sections describe debug configuration parameters, which only work on nonproduction builds. See [QTI charger app configuration file for LA \(example\)](#) for a QTI charger app LA example file.

7.3.1 PrintChargerAppDbgMsg

If TRUE, this parameter enables QTI charger app logs and respective pmic.dxe charger logs, and displays them on UART.

- Default value – FALSE (excludes extensive charging logs and only displays battery status information on UART for general users)
- Example charger log

```
-----
- 0x09C0AD000 [16004] QcomChargerApp.efi
QcomChargerApp:: QcomChargerApp_MonitorCharging
TimeStamp, StateOfCharge, Voltage, ChargeCurrent, Temp
16, ChargerApp:: Battery Status 1,3536,206,25
Waiting for 3 sec
0x2B, ChargerApp: Battery Status 17,3732,-1072,24
Waiting for 3 sec
```

7.3.2 PrintChargerAppDbgMsgToFile

If TRUE, this parameter enables QTI charger app file logs and respective pmic.dxe charger logs, and saves them to a file.

- Default value – FALSE (indicates excludes extensive charging logs and only displays battery status information on UART for general users)
- If file log configuration is enabled, the chargerlog.txt and pmiclog.txt files are generated in the debug LogFS 8 MB partition. When the file size reaches its limit, this configuration acts as a circular buffer and starts overwriting old logs.
- The chargerlog.txt file has the charger and boot configuration. The pmiclog.txt file has FG and charger-related information, for example, SRAM dumps.

7.3.3 FileLoggingDbgLevelMask

This parameter provides a debug level mask (in hex) for file logging.

- Default value – 80000042
- Refer to DebugLib.h

7.3.4 EnableChargerFGDump

If TRUE, This parameter enables PMIC charger and FG peripheral dumps. Dumps also occur during charger app initialization and exit. Dumps are initiated if any key is pressed while charging.

- Default value – FALSE
- Dependency – [PrintChargerAppDbgMsg](#) charger logs must be enabled.

7.4 FG and FG debug configuration

The following sections describe FG and FG debug configuration parameters. See [QTI charger app configuration file for LA \(example\)](#) for a QTI charger app LA example file.

7.4.1 BatteryIdTolerance

This parameter reads the battery ID to load the battery profile and set a tolerance limit. If the battery ID falls into tolerance range, only the respective profile with that battery ID is flashed to the FG. Otherwise, the default profile is loaded from the battery profile file, which is the first profile.

- Default value – 8%
- QTI charger app includes eight profile supports

7.4.2 DumpSram

This parameter enables QTI charger app FG SRAM dumps. If enabled (TRUE), the QTI charger app periodically dumps SRAM to get debug information from hardware FG algorithms.

- Default value – FALSE
- Dependency – [PrintChargerAppDbgMsg](#) charger logs must be enabled.

7.4.3 DumpSramStartAddr

This parameter is the SRAM dump start address (values in decimal).

7.4.4 DumpSramEndAddr

This parameter is the SRAM dump end address (values in decimal).

7.4.5 DumpSramDuration

This parameter is the dump SRAM contents timer duration in seconds. The default value is 90 sec.

7.5 Initial required charger configuration

The following sections describe initial required charger configuration parameters. See [QTI charger app configuration file for LA \(example\)](#) for a QTI charger app LA example file.

7.5.1 BootToHLOSThresholdInMv

This parameter configures the QTI charger app threshold to allow boot.

- Default value – 3600 mV
- This threshold is also used for unsupported batteries or battery emulators.

7.5.2 OsStandardBootSocThreshold

This parameter configures the QTI charger app minimum threshold to allow boot.

- Default value – 7 % SoC
- Dependencies
 - [SocOrVoltageBaseBoot](#) = TRUE
 - [LoadBatteryProfile](#) = TRUE
- This threshold is not used in unsupported batteries or battery emulators.

7.5.3 BattVoltLimHighDelta

This parameter enables delta FV to charge the battery. The default value is 30 mV.

7.5.4 ChgFvMax

This parameter enables battery FV. The default value is 4350 mV.

7.5.5 ChgFccMax

This parameter enables fast charging current. The default value is 2000 mA.

7.5.6 ChargingTermCurrent

This parameter enables charger termination current to declare 100% SoC. The default value is 200 mA.

7.5.7 ConservChgFvDelta

This parameter enables maximum charger delta FV maximum for unknown battery configurations. The default value is 200 mV.

7.5.8 BATT_THERM coefficients

This parameter enables battery thermal coefficients to read the battery temperature accurately. This is picked up as per ThermBias value per device/battery and initial values given are formatted with half-float encoding. If these values must be updated, open a Salesforce case with Qualcomm to ensure the correct coefficients are used. Refer to the BATT_THERM master beta coefficient table in *Understanding PMI8998 Fuel Gauge* (80-VT310-138).

- Default values
 - BattThermC1 = A1
 - BattThermC2 = 50
 - BattThermC3 = FF
- Values are based on the following ThermBias and pull-up resistor value:
`BattThermHalfRangeInC = 25`
- Dependency – ProgramBattThermCoeffs = TRUE

7.5.9 BATT_THERM configs

This parameter enables battery thermal bias wait and ground select configs.

- Supported values
 - BattThermBiasWait
 - 0 = 0 ms
 - 1 = 1 ms
 - 2 = 4 ms
 - 3 = 12 ms
 - 4 = 20 ms
 - 5 = 40 ms
 - 6 = 60 ms
 - 7 = 80 ms
 - BattThermGndSel
 - TRUE = Thermistor is located on the battery pack
 - FALSE = Thermistor on the PCB (skin temp)
- Default values
 - BattThermBiasWait = 4
 - BattThermGndSel = TRUE
- Dependency – None

7.5.10 AUX_THERM coefficients

This parameter enables auxiliary thermal coefficients to adjust voltage to the temperature mapping. Temperature mapping is based on the beta 3435 of the thermistor in use and the associated temperature to 50% ratio (dependent on the pull-up value). Refer to the AUX_THERM master beta coefficient table in *Understanding PMI8998 Fuel Gauge* (80-VT310-138).

- Default values
 - AuxThermC1 = BF
 - AuxThermC2 = 36
 - AuxThermC3 = FF
- Values are based on the following ThermBias and pull-up resistor value:
AuxThermHalfRangeInC = 25
- Dependency – ProgramAuxThermCoeffs = TRUE

7.5.11 AUX_THERM configs

This parameter enables battery auxiliary thermal bias wait and ground select configs.

- Supported values
 - AuxThermBiasWait
 - 0 = 0 ms
 - 1 = 1 ms
 - 2 = 4 ms
 - 3 = 12 ms
 - 4 = 20 ms
 - 5 = 40 ms
 - 6 = 60 ms
 - 7 = 80 ms
 - AuxThermGndSel
 - TRUE = Thermistor is located on the battery pack
 - FALSE = Thermistor on the PCB (skin temp)
- Default value
 - AuxThermBiasWait = 4
 - AuxThermGndSel = FALSE
- Dependency – None

7.5.12 Device skin and charger hot thresholds

This parameter enables device skin and charger hot threshold configuration. Device skin temperature is usually on device display. Charger hot thresholds are used for thermal mitigation via intelligent negotiation for optimum voltage (INOV).

- Default values
 - DeviceSkinHotInC = 80
 - DeviceSkinTooHotInC = 90
 - ChargerHotInC = 80
 - ChargerTooHotInC = 90
- Dependency – ProgramSkinAndChargerHotThreshold must be TRUE.

7.5.13 Charger_THERM source configs

This parameter enables/disables the charger thermal source.

- Default values
 - SkinTempSrc = TRUE
 - DieTempSrc = TRUE
 - DieTempCompSrc = TRUE
- Dependency – None

7.5.14 EmergencyShutdownVbatt

This parameter configures the device emergency shutdown limit. The QTI charger app monitors charge current (less than 0 mA) and battery voltage (less than 3.2) for three consecutive reads before initiating emergency shutdown to safeguard the battery. The default value is 3200 mV.

7.5.15 EnableChargerWdog

This parameter enables the charger watchdog to safeguard unintentional charging if the software gets stuck. Charging becomes disabled if the watchdog is configured. Software must pet the watchdog based on its set expiration limit. Supported values are:

- 0 – Do not enable the charger watchdog
- 1 (Default) – Enable the charger watchdog during charging and disable before exiting
- 2 – Enable the charger watchdog during charging and leave enabled when exiting

7.5.16 VBtEmpty threshold

This parameter configures the low battery voltage threshold for the SoC empty interrupt. The default value is 2800 mV.

VBtEmpty = 2800

7.5.17 VBattEstDiffThreshold

This parameter configures the estimated voltage difference threshold to restart the FG if the threshold difference is higher.

- Default value – 30 mV (Vbatt_Estimate_diff_threshold)
- Dependency – [FgCondRestart](#)

7.5.18 RConn compensation resistance

This parameter configures the Rconn compensation resistance.

- Value – In mOhms - range is ± 100 mOhms
- Default value – RconnComp = 0
- Dependency – None

7.6 Battery error handling configuration

The following sections describe battery error handling configuration parameters. See [QTI charger app configuration file for LA \(example\)](#) for a QTI charger app LA example file.

7.6.1 DebugBoardBatteryIdMin and DebugBoardBatteryIdMax

This parameter specifies the debug board battery ID range. If the voltage associated with the battery ID falls below the range, the QTI charger app handles the battery as a debug board based on Vbatt > boot threshold (3.6 V) allow boot to HLOS; otherwise, shut down.

- Default value – 2000 to 14000 Ohms
- Dependency – [DebugBoardBehavior](#)
- See also [BootToHLOSThresholdInMv](#).

7.6.2 SmartBatteryIdMin and SmartBatteryIdMax

This parameter specifies the smart battery ID range. The QTI charger app handles the battery as a smart battery based on Vbatt > 3.6 V allow boot to HLOS; otherwise, continue charging until the configured threshold is reached.

- Default value – 240000 to 450000 Ohms (smart battery ID range)
- See also [BootToHLOSThresholdInMv](#).

7.6.3 RegularBatteryIdMin and RegularBatteryIdMax

This parameter specifies the regular battery ID range. The QTI charger app handles the battery as a regular board based on Vbatt > 3.6 V allow boot to HLOS; otherwise, continue charging until the configured threshold is reached. Default value is 15000 to 137000 Ohms (regular battery ID range).

7.6.4 UnknownBatteryBehavior

This parameter defines unknown battery behavior. It detects if the battery ID is within the specified range. Supported values are:

- 0 – Shut down the device
- 1 – Boot to HLOS if battery is more than threshold, else shut down
- 2 – Conservative charging
- 3 (Default) – Regular charging

7.6.5 DebugBoardBehavior

This parameter defines debug board battery behavior. Supported values are:

- 0 – Show low battery icon, disable PON1/USBIN trigger to prevent reboot and shutdown
- 1 (Default) – Show low battery icon and stay on until device is turned off by user
- 2 – Boot to HLOS

7.6.6 BattMissingCfg

This parameter configures the battery missing detection behavior. Supported values are:

- 0 (Default) – Use battery ID
- 1 – Use battery thermistor
- 2 – Use battery thermistor and ID

7.7 Jeita configuration

The following sections describe Jeita configuration parameters. See [QTI charger app configuration file for LA \(example\)](#) for a QTI charger app LA example file.

7.7.1 Jeita zones

These parameters enable specified Jeita zones. For negative values, use the negative symbol, for example, -30.

Parameter	Default value (°C)
JeitaCriticalTempLowLimit	-20
JeitaHardColdLimit	0
JeitaSoftColdLimit	10
JeitaSoftHotLimit	45
JeitaHardHotLimit	60
JeitaCriticalTempHighLimit	70

7.7.2 JeitaCcCompCfg

This parameter enables Jeita charge current compensation configuration when the device is within the battery temperature soft limit for the hardware Jeita.

- Jeita compensation values
 - Minimum value – 0 mA
 - Maximum value – 1575 mA
 - Step size – 25 mA
- Default value – 1000 mA

7.7.3 JeitaFvCompCfg

This parameter enables Jeita charge voltage compensation configuration when the device is in battery temperature soft limit for hardware Jeita.

- Jeita compensation values
 - Minimum value – 0 mV
 - Maximum value – 472.5 mV
 - Step size – 7.5 mV
- Default value – 105 mV

7.7.4 NoChargeAndWait

This parameter configures device behavior for temperatures outside of the charging range but within the operating range. Supported values are:

- TRUE (Default) – Disable charging and wait
- FALSE – Shut down the device if the temperature is outside of the charging range

7.8 WiPower configuration

The following sections describe WiPower configuration parameters. See [QTI charger app configuration file for LA \(example\)](#) for a QTI charger app LA example file.

7.8.1 WiPowerSupported

This parameter enables WiPower support configuration for the QTI charger app charging device. The default value is TRUE.

7.8.2 DCInBootToHLOSThresholdInMv

This parameter enables WiPower threshold configuration for QTI charger app charging. The default value is 3600 mV.

7.8.3 SuspendDCIn

This parameter enables suspended DCIN behavior configuration when WiPower is enabled in QTI charger app charging. The default value is FALSE.

7.9 Thermal configuration

The following sections describe thermal configuration parameters. See [QTI charger app configuration file for LA \(example\)](#) for a QTI charger app LA example file.

7.9.1 SWThermalMitigationEnable

This parameter enables thermal safety mitigation configuration in the QTI charger app charging device. Mitigation is based on the MSM or SDM Tsens maximum average temperature reading. The default value is FALSE.

7.9.2 TsensTimeoutMins

This parameter enables thermal safety timer configuration when the device is in the thermal zone and not in charger wait state. The device waits for the configured wait time and initiates shutdown if thermal conditions do not normalize.

- Default value – 30 min
- Give-up time in thermal wait for battery disconnect – Max 60 min

7.9.3 Tsens limits or zone

These parameters enable configuration of thermal safety zones or limits. If the temperature is above the extreme temperature limit, the device performs automatic fault protection (AFP).

If the Tsens temperature limit is within the specified wait range (for example, 75 to 90 min), the QTI charger app waits 30 min (polling every 3 sec) to allow the device to cool down. The device performs AFP after the 30 min expiration.

Parameter	Default value (°C)	Description
TsensHighTemp	85	High temperature limit for thermal wait
TsensExtremeTemp	90	High temperature limit for battery and device safety on battery disconnect
TsensLowTemp	75	Low temperature limit for end of thermal wait

7.10 QTI charger app configuration file for LA (example)

```
#
# Default Charger App Config settings
#
[CHARGER Config]
#
# Version/Information:
```

```

# file ChargerApp_VbattTh_SDM845.cfg
#
# Implements the Qualcomm's Charger application config parameters
#
# Copyright (c) 2016, Qualcomm Technologies Inc. All rights reserved.
#
#     1 : Initial revision
#     2 : Deleting not needed config params and removing dummy battery2
support
#     3: Adding Jeita Compensation params
#     4 : Adding parameters for different battery types and QC 3.0 and QC
2.0 chargergergers
#     5 : Added parameter to support enabling watchdog when charging is
enabled
#     6 : Adding parameters for Aux Coffes, SkinHot and Charger Hot settings
#     7 : Update for Battery profile load
#     8 : Added SupportHostMode
#     9 : Adding Thermal configs
#    10 : Adding support for Charger Fg Peripheral dumps
#    11 : Adding HVDCP Enable control
#    12 : Adding WIPower configs
#    13 : Removed config item for setting IUSB_MAX in case of SDP
#    14 : Adding Restarting FG flag
#    15 : Adding Charger led indication config, rasing skin hot to 70-80C,
disabling watchdog as default
#    16 : Added changes for supporting different platforms, MTP, QRD, etc.
CfgVersion = 17
--- Threshold Charging Configurations ---
#Use Battery SOC or voltage as threshold charging criteria
#Voltage is default
SocOrVoltageBaseBoot = FALSE
#Load Fuel Gauge Battery Profile profile for SOC estimation and accuracy
LoadBatteryProfile = FALSE
#Below VBAT threshold is used to decide on showing sign of life first before
FG Module Initialization and continuing with threshold charging
DispSignOfLifeMaxThresholdMv = 3700
# FG Conditional Restart on Device reset
FgCondRestart = TRUE
# Charging status indication via led
# 0 = Disable 1 = solid during charging 2 = led blinks during charging
# if turned on LED will be turned off after threhsold charging is completed
i.e. when device boot to HLOS
ChargerLedConfig = 1
--- Threshold Charging Configurations End ---
--- Debug Configurations ---
# Print Charger DEBUG Messages
PrintChargerAppDbgMsg = FALSE

```

```

#Print Charger DEBUG Messages to ULOG File..Default is false
PrintChargerAppDbgMsgToFile = FALSE
#Enable/disable Charger/FG Dump support
EnableChargerFgDump = FALSE
--- ----Debug FG Configurations ----
#dump SRAM contents Default value - FALSE
DumpSram = FALSE
#dump SRAM Start and End Address in Hex Format
#SRAM Block      SRAM Address
#System          0x00 - 0x17
#Profile          0x18 - 0x3C
#Scratchpad      0x50 - 0x7C
#values in decimal
DumpSramStartAddr = 0
#values in decimal
DumpSramEndAddr   = 124
#dump SRAM contents timer Duration in s
DumpSramDuration = 90
--- ---- Debug FG Configurations End ----
--- Debug Configurations End ---
--- Battery Error Handling Configurations ---
#Battery ID Tolerance Percentage 8%
BatteryIdTolerance = 8
#Debug board ID range, value in Ohms
DebugBoardBatteryIdMin = 0
DebugBoardBatteryIdMax = 14000
#Regular battery ID range, value in Ohms
RegularBatteryIdMin = 15000
RegularBatteryIdMax = 137000
#Smart battery ID range, value in Ohms
SmartBatteryIdMin = 240000
SmartBatteryIdMax = 450000
#Support unknown battery charging behavior
# 0: Shuts down device,      1: Boot to HLOS if battery more than threshold
else shutdown
# 2: Conservative Charging 3: Regular charging
UnknownBatteryBehavior = 3
#Debug board behavior
# 0: Show low battery icon, disable PON1/USBIN trigger to prevent reboot and
shutdown
# 1: Show low battery icon and stay on until device is turned off by user.
# 2: Boot to HLOS
DebugBoardBehavior = 2
#Battery missing config
# 0 = using batt id 1 = using batt therm 2 = both
BattMissingCfg = 0
--- Battery Error Handling Configurations End ---

```

```
--- Jeita Configurations ---
# Configure limits for Battery Temperature (For negative values, use negative
sign. Ex: -30)
JeitaCriticalTempLowLimit = -20
JeitaHardColdLimit = 0
JeitaSoftColdLimit = 10
JeitaSoftHotLimit = 45
JeitaHardHotLimit = 60
JeitaCriticalTempHighLimit = 70
#JEITA Charge Current Compensation when in battery temperature soft-limit
#JEITA CC = min is 0 ma and max is 1575 ma - step size is 25mA
JeitaCcCompCfg = 1000
#JEITA Float Voltage Compensation when in battery temperature soft-limit
#min is 0 and max .4725 V step size is 7.5 mV - unit is in mV
JeitaFvCompCfg = 105
#device behaviour if temp is outside charging range but within operational
range
# 1= Disable charging and wait. 0 = Shutdown device is temp outside
NoChargeAndWait = TRUE
--- Jeita Configurations End ---
--- Initial Configurations ---
#Boot device to HLOS in case of unsupported battery or battery emulator. In
millivolt*/
BootToHLOSThresholdInMv = 3600
#Minimum SOC Threshold before allowing to boot to HLOS
#below param is considered only when SocOrVoltageBaseBoot = TRUE and
LoadBatteryProfile = TRUE
OsStandardBootSocThreshold = 7
# Configure Battery Voltage and Current limit
BattVoltLimHighDelta = 30
# Configure VddMax and IbatMax values
# Set to 0 to configure through API
ChgFvMax = 4350
ChgFccMax = 2000
#Charging termination current in milliamps
ChargingTermCurrent = 200
# Voltage (in mV) to be reduced from FV_MAX during conservative charging
ConservChgFvDelta = 200
#Program THERM coeffs ..
#Picked up as per ThermB value per device/battery and initial values are
given in HALF encoded
ProgramAuxThermCoeffs = TRUE
AuxThermC1 = A0
AuxThermC2 = 4F
AuxThermC3 = CF
#based on ThermB and pull up resistor value
AuxThermHalfRangeInC = 25
```

```
#Program device Skin and Charger Hot thresholds
ProgramSkinAndChargerHotThreshold = TRUE
DeviceSkinHotInC      = 50
DeviceSkinTooHotInC   = 60
ChargerHotInC         = 80
ChargerTooHotInC      = 90
#Lowest Voltage at which device should shutdown gracefully
#value in mV
EmergencyShutdownVbatt = 3200
#Charger WDOG Support options
# 0: Do not enable Charger WDOG
# 1: Enable Charger WDOG during charging and Disable before exiting
# 2: Enable Charger WDOG during charging and leave enabled when exiting
EnableChargerWdog = 1
#Vbat Empty threshold in mv
VBtEmpty = 2800
--- Initial Configurations End ---
--- Thermal Configurations ---
#Enable SW thermal mitigation during charging by default FALSE
# Mitigation is based on MSM Tsens max avg temp reading
SWThermalMitigationEnable = FALSE
## TSENS ##
#High Temperature limit for thermal wait
TsensHighTemp = 85
#High Temperature limit for battery and device safety (battery disconnect)
TsensExtremeTemp = 90
#Low Temperature limit for end of thermal wait
TsensLowTemp = 75
# Give up time in thermal wait for battery disconnect - support up to 60min
TsensTimeoutMins = 30
--- Thermal Configurations End ---
--- WiPower Configurations ---
#support wipower or not
WiPowerSupported = FALSE
#Boot device to HLOS in case of wipower charging. In millivolt
DCInBootToHLOSThresholdInMv = 3600
#suspend DCIn or not after exiting UEFI
SuspendDCIn = TRUE
--- WiPower Configurations End ---
#
# End of config
# Blank line needed after the last config
#
```

8 Off mode charging

In Off mode, APPSBL can boot Android in a mode that displays a battery animation instead of the Android idle screen. Off mode occurs after dead battery recovery for XBL and UEFI.

Off mode is also known by the following names:

- Charger mode
- Off state
- Power off (POFF)
- Charge-only mode

Off mode charging uses the Linux kernel charger drivers with a reduced version of the HLOS by passing `androidboot.mode=charger` the kernel command line.

Passing `androidboot.mode=normal` to the kernel command line powers up to the Android idle screen.

8.1 Enable Off mode charging

Off mode charging is not enabled by default. Make the following code change to keep Off mode charging enabled:

1. Access the UpdateCmdLine.c file in the following location:

```
/bootable/bootloader/edk2/QcomModulePkg/Library/BootLib
```

2. Modify the following code in the UpdateCmdLine() function:

```
CmdLineLen += AsciiStrLen(AndroidBootMode);
CmdLineLen += AsciiStrLen(FfbmStr);
/* reduce kernel console messages to speed-up boot */
CmdLineLen += AsciiStrLen(LogLevel);
} else if (BatteryStatus && IsChargingScreenEnable()) {
} else if (BatteryStatus) {
    DEBUG((EFI_D_INFO, "Device will boot into off mode charging mode\n"));
    PauseAtBootUp = 1;
    CmdLineLen += AsciiStrLen(BatteryChgPause);
```


8.2 Enable or disable charging in fastboot

Setting the Charge mode in fastboot stores the configuration in persistent storage for APPSBL. Memory structures are updated in `device.charger_screen_enabled`.

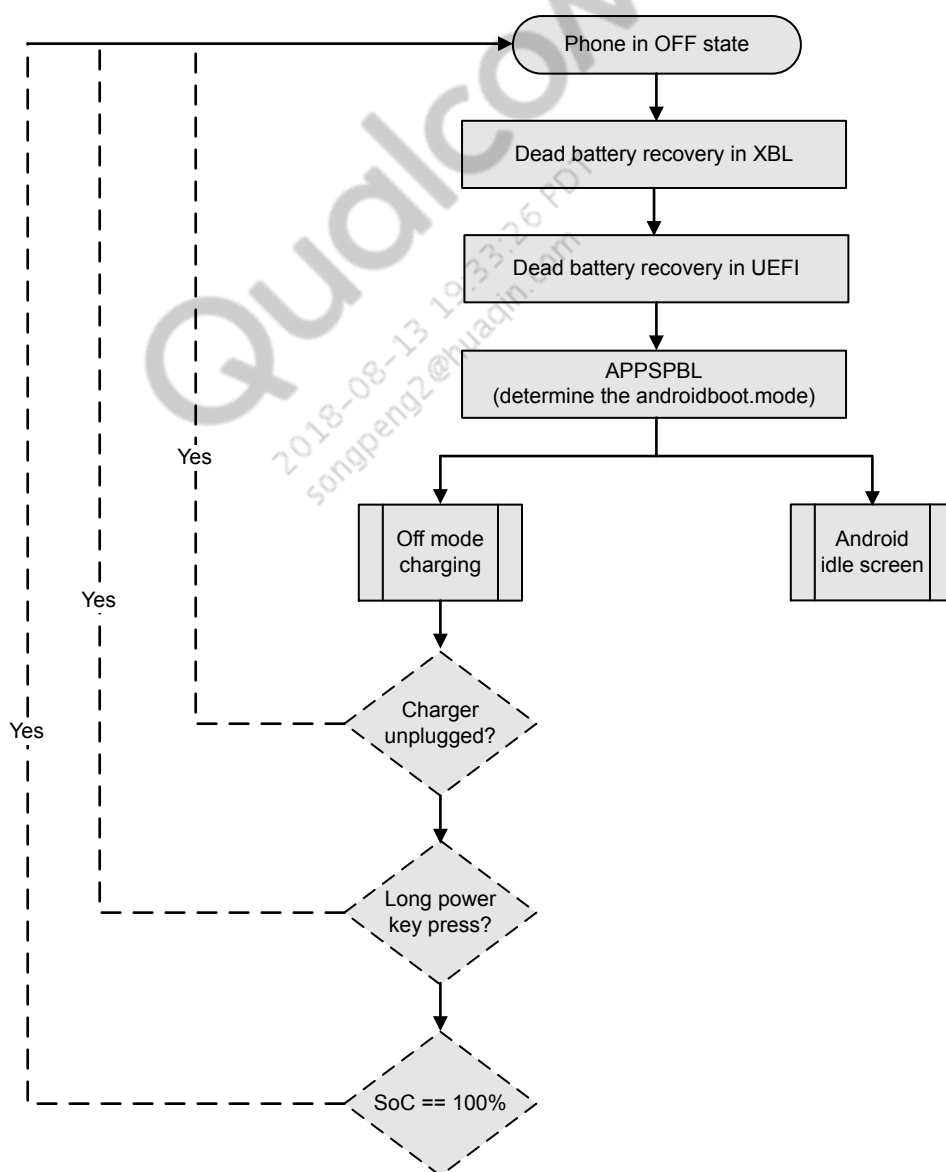
1. Use the following command to enable Charge mode:

```
fastboot oem enable-charger-screen
```

2. Use the following command to disable Charge mode:

```
fastboot oem disable-charger-screen
```

8.3 Off mode call flow



9 Voter

The Voter API enables client voting to share common resources by implementing a voting scheme to vote on the resource values.

Votable types are as follows:

- `VOTE_ANY` – If one client votes TRUE, the result is TRUE. If any client votes FALSE, the result is FALSE.
- `VOTE_MIN` – The result is the minimum of all the values.
- `VOTE_MAX` – The result is the maximum of all the values.

Objects sent to the `votable` struct must be passed to Voter APIs.

9.1 struct votable

This structure is used to pass information to Voter APIs.

Type	Parameter	Description
const char	name	Votable name
struct list_head	list	Votable list
struct client_vote	votes	Maximum number of clients for a votable
int	num_clients	Current number of clients
int	type	Votable type
int	effective_client_id	ID of the client for which the vote is currently applied
int	effective_result	Effective value of the current vote
struct mutex	vote_lock	Mutex to lock the vote
void	*data	Private data to be passed
int	(*callback)	Callback function executed when the vote is applied
char	*client_strs[NUM_MAX_CLIENTS]	Names of the clients for this votable
bool	voted_on	Indicates if any vote has been applied to this votable
u32	force_val	Forced votable value by the user
bool	force_active	Write 1 to activate the forced vote and 0 to deactivate

9.2 Initialization functions

This section describes the Voter API functions used during initialization.

9.2.1 create_votable()

This function is used to create new votables.

Prototype

```
struct votable *create_votable(const char *name, int votable_type,
                              int (*callback)(struct votable *votable, void *data,
                              int effective_result, const char *effective_client),
                              void *data);
```

Parameters

in	name	Name of the votable
in	votable_type	Votable type. Possible values: <ul style="list-style-type: none"> ▪ VOTE_MIN ▪ VOTE_MAX ▪ VOTE_SET_ANY
in	callback	Reference to executable code in which the callback is called in one of the following conditions: <ul style="list-style-type: none"> ▪ Change in election results ▪ Voting takes place for the first time
in	data	Private data to be passed

Returns

Votable object.

9.2.2 destroy_votable()

This function removes votables.

Prototype

```
void destroy_votable(struct votable *votable);
```

Parameters

in	votable	Votable object
----	---------	----------------

Returns

None

9.2.3 lock_votable()

This function is a locks votables to prevent synchronization issues.

Prototype

```
void lock_votable(struct votable *votable);
```

Parameters

in	votable	Votable object
----	---------	----------------

Returns

None

9.2.4 unlock_votable()

This function is a unlocks votables to prevent synchronization issues.

Prototype

```
void unlock_votable(struct votable *votable);
```

Parameters

in	votable	Votable object
----	---------	----------------

Returns

None

9.3 Votable functions

The Voter API functions in this section enable the voting process.

9.3.1 vote()

This function is used to cast new votes.

Prototype

```
int vote(struct votable *votable, const char *client_str, bool enabled, int val)
```

Parameters

in	votable	Votable object
in	client_str	Intended client
in	enabled	Enabled or disabled. Setting to FALSE (disabled) excludes the client from an election.
in	val	Vote value

Detailed description

The `val` parameter is only considered if the `enabled` parameter is `TRUE`. Because there is no concept of abstaining from election for `SET_ANY` votables, the `val` parameter is ignored for `SET_ANY` votable types.

For `MIN` and `MAX` votable types, the value of the `val` parameter is used as follows:

- If the `enabled` parameter = `TRUE`, clients use the vote value
- If the `enabled` parameter = `FALSE`, this value is ignored

The callback is called only in the following conditions:

- There is a change in the election results
- It is the first time a client is voting

Returns

The value the client voted for if successful, `-EINVAL` for errors.

9.3.2 `rerun_election()`

This function reruns all casted votes.

Prototype

```
int rerun_election(struct votable *votable);
```

Parameters

in	votable	Votable object
----	---------	----------------

Returns

The value the client voted for if successful, `-EINVAL` for errors.

9.3.3 `find_votable()`

This function finds the votable to get the votable handle.

Prototype

```
struct votable *find_votable(const char *name);
```

Parameters

in	name	Name of the votable.
----	------	----------------------

Returns

Votable object.

9.4 Get functions

The Voter API get functions return voter information.

9.4.1 `get_client_vote()`

This function gets the unlocked variant of the effective client among all the enabled voters.

Prototype

```
int get_client_vote(struct votable *votable, const char *client_str);
```

Parameters

in	votable	Votable object
in	client_str	Intended client

Returns

The value the client voted for if successful, -EINVAL for errors.

9.4.2 `get_client_vote_locked()`

This function gets the locked variant of the effective client among all the enabled voters.

Prototype

```
int get_client_vote_locked(struct votable *votable, const char *client_str);
```

Parameters

in	votable	Votable object
in	client_str	Intended client

Returns

The value the client voted for if successful, -EINVAL for errors.

9.4.3 `get_effective_result()`

This function gets the unlocked variant of the effective value among all of the enabled voters.

Prototype

```
int get_effective_result(struct votable *votable);
```

Parameters

in	votable	Votable object
----	---------	----------------

Returns

Effective result, one of the following otherwise:

- MIN and MAX votables – Returns -EINVAL if the votable object has been created but no clients have casted their votes or the last enabled client disables its vote.
- SET_ANY votables – Returns 0 if no clients have casted their votes. Because there is no concept of abstaining from election for SET_ANY votables, votes for all the clients of the SET_ANY votable default to FALSE.

9.4.4 get_effective_result_locked()

This function gets the locked variant of the effective value among all of the enabled voters.

Prototype

```
int get_effective_result_locked(struct votable *votable);
```

Parameters

in	votable	Votable object
----	---------	----------------

Returns

Effective result, one of the following otherwise:

- MIN and MAX votables – Returns -EINVAL if the votable object has been created but no clients have casted their votes or the last enabled client disables its vote.
- SET_ANY votables – Returns 0 if no clients have casted their votes. Because there is no concept of abstaining from election for SET_ANY votables, votes for all the clients of the SET_ANY votable default to FALSE.

9.4.5 get_effective_client()

This function gets the unlocked client among all of the enabled voters.

Prototype

```
const char *get_effective_client(struct votable *votable);
```

Parameters

in	votable	Votable object
----	---------	----------------

Returns

Votable object or NULL.

- MIN and MAX votables – Returns NULL if the votable object has been created but no clients have casted their votes or the last enabled client disables its vote.
- SET_ANY votable – Returns NULL if no clients have casted their votes. Because there is no concept of abstaining from election for SET_ANY votables, SET_ONLY votables only return the client that casts a vote or the client that caused the result to change.

9.4.6 get_effective_client_locked()

This function gets the locked client among all of the enabled voters.

Prototype

```
const char *get_effective_client_locked(struct votable *votable);
```

Parameters

in	votable	Votable object
----	---------	----------------

Returns

Votable object or NULL.

- MIN and MAX votables – Returns NULL if the votable object has been created but no clients have casted their votes or the last enabled client disables its vote.
- SET_ANY votable – Returns NULL if no clients have casted their votes. Because there is no concept of abstaining from election for SET_ANY votables, SET_ONLY votables only return the client that casts a vote or the client that caused the result to change.

9.4.7 is_client_vote_enabled()

This function gets unlocked variants specifying whether a client's vote is enabled.

Prototype

```
bool is_client_vote_enabled(struct votable *votable, const char *client_str);
```

Parameters

in	votable	Votable object
in	client_str	Intended client

Returns

TRUE if the client's vote is enabled; FALSE otherwise.

9.4.8 is_client_vote_enabled_locked()

This function gets locked variants specifying whether a client's vote is enabled.

Prototype

```
bool is_client_vote_enabled_locked(struct votable *votable,  
                                const char *client_str);
```

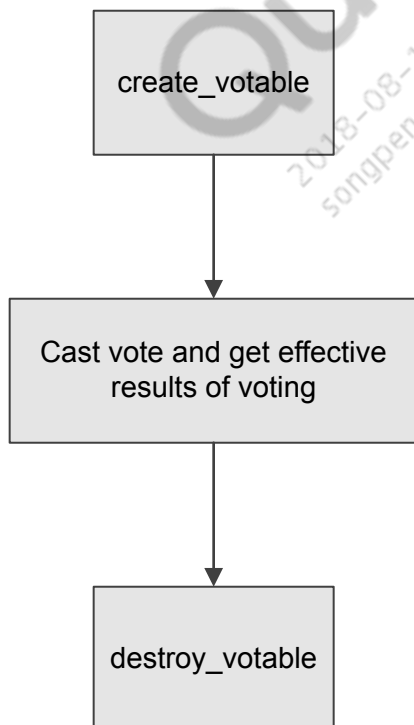
in	votable	Votable object
in	client_str	Intended client

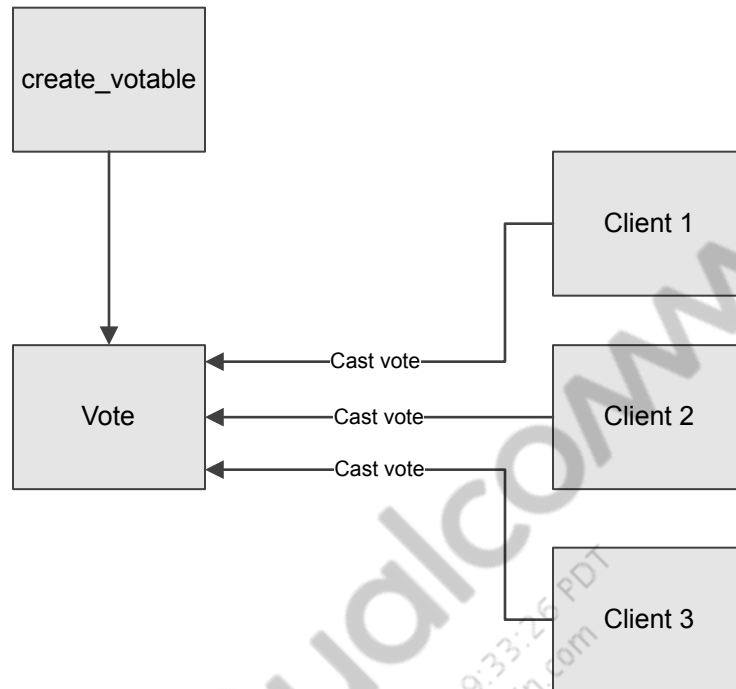
Returns

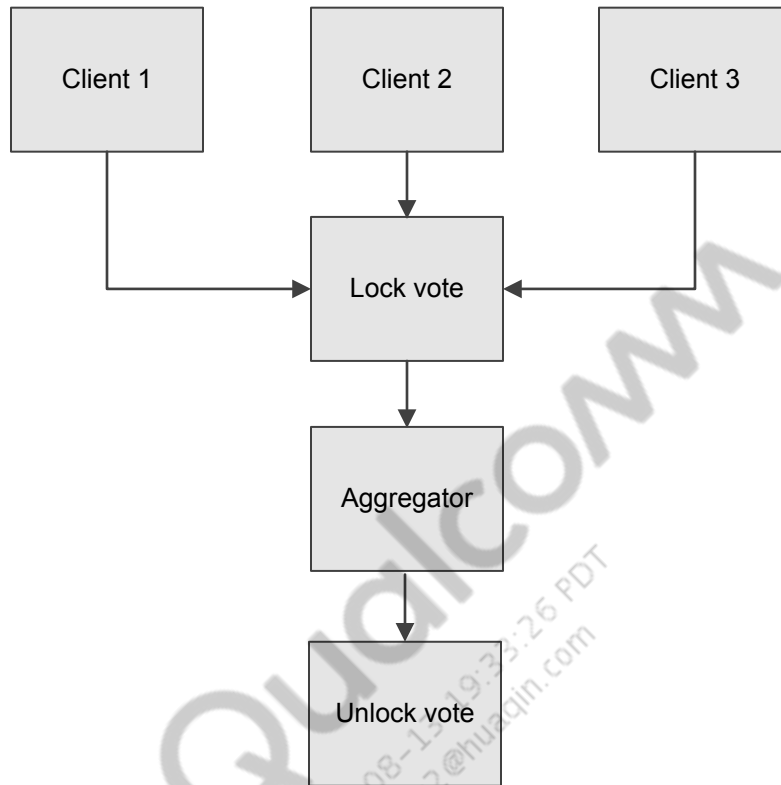
TRUE if the client's vote is enabled; FALSE otherwise.

9.5 Voter call flows

Voter lifecycle



Voter cast

Voter lock and unlock

10 Debug

10.1 Use ADB over Wi-Fi to capture log dumps while charging

Use the Android debug bridge (adb) shell to capture dumps from the following sources:

- PMI8998 register
 - USB
 - Battery power_supply sysfs nodes
1. Connect the device to the same Wi-Fi network as the PC is connected to (PC must be connected to Wi-Fi).
 2. In Settings, Wi-Fi/WLAN, select the network name to get the device IP address. In this example, the IP address of the phone is **192.168.68.167**.
 3. Verify that the PC is connected to same wireless network as the device.
 4. Connect the device to the PC via USB and run the following adb commands:

```
adb shell "setprop persist.adb.tcp.port 5555"
adb tcpip 5555
```

5. Disconnect the device from the PC USB and run the following commands (replace 192.168.68.167 with the device IP address):

```
adb connect 192.168.68.167
adb shell
cat /sys/class/power_supply/battery/*
cat /sys/class/power_supply/usb/*
cd /sys/kernel/debug/omapreg/spmi0-02
echo 0x100 > count
echo 0x21000 > address
cat data
echo 0x21100 > address
cat data
echo 0x21200 > address
cat data
echo 0x21300 > address
cat data
echo 0x21400 > address
cat data
echo 0x21500 > address
cat data
```

```
echo 0x21600 > address
cat data
echo 0x21700 > address
cat data
```

10.2 Enable charger logs when providing kernel logs

1. Use one of the following methods to enable the PMI logs for a charger-related issue:

- For software driver hardcoded change:

/drivers/power/qcom-charger/qnp-smb2.c

Add the text shown in red and remove the text shown in ~~blue strikethrough~~:

```
static int __debug_mask;
static int __debug_mask = 0xFF;
```

- Run the following commands in the adb shell (must be reissued every power cycle):

```
adb root && adb wait-for-device
adb shell
echo 0xFF > /sys/module/qcom,qnp-smb2/parameters/__debug_mask
```

2. Collect the kernel logs using the following adb shell command:

```
adb shell dmesg > klog.txt
```

10.3 Guidelines for creating a support request

When requesting support, always provide the following information:

- Boot kernel logs (which show the configuration process)
- Charger kernel logs
- Register dumps capturing the issue
- If it is a charger detection issue, always provide waveforms of D+, D-, VBUS, CC1, CC2, VCONN, and IUSB current capturing the issue.

10.4 Debug overwrite feature

The debug overwrite feature enables debug logs and overwrites UEFI configuration at runtime. Overwriting configuration at runtime makes it unnecessary to rebuild the core. On debug builds, the UEFI QTI charger app is built during initialization.

This feature only works on debug builds.

10.4.1 Enable the debug overwrite feature with Flash tools FV

The QComChargerCfg.cfg file can be copied via mass storage application from the UEFI boot device selection (BDS) menu to mount the LogFS drive on which to copy the file. The FV tool must be flashed to enable the UEFI BDS menu.

The tools.fv file is located in the following metabuild folder:

```
..\boot_images\QcomPkg\QcomToolsPkg\Bin\QcomTools\DEBUG
```


1. Run the following command:

```
fastboot flash toolsfv ..\boot_images\QcomPkg\QcomToolsPkg\Bin\QcomTools\DEBUG\tools.fv
```

2. During boot, hold down the **volume** key to display the BDS menu.
3. Use the **volume down** key to scroll down the menu, and press the **Home/Power** key to select Fastboot.

```
KeyMap=> Up: Vol+, Down: Vol-, Sel: Camera/Home/Pwr Exit: Esc
BDS Menu:
```


```
-----
0  Exit BDS Menu
1  Enable Secure Boot
2  Disable Secure Boot
3  Enable Debug Policy
4  Disable Debug Policy
5  Config PPI display
6  Provision RPMB
7  Enter Shell
8  Boot USB First
9  MassStorage
10 Reboot
11 USB Menu
12 PMIC Menu
13 UEFI Menu
14 EDL Mode
-> 15 Fastboot
```



```
KeyMap=> Up: Vol+, Down: Vol-, Sel: Camera/Home/Pwr Exit: Esc - 0x09532D000 [32517] I
astboot.efi
Fastboot Build Info: Nov 17 2016 19:03:10
DALLOG Device [0x2000145]: Unable to turn ON clock: gcub_h_ieckVDVDOSBOE
DALLOG Device [0x2000145]: Unable to turn ON clock: gcub_h_ieckVDVDOSBOE
Fastboot: Initializing...
Fastboot: Processing commands
32, PmicDxe:: EFI_PmicSchgGetChargerPortType APSD done status: 1
```

4. Reboot the device, press and hold down the **volume** key to open the UEFI BDS menu, and press the **volume down** key to select MassStorage.

```
-----
0  Exit BDS Menu
1  Enable Secure Boot
2  Disable Secure Boot
3  Enable Debug Policy
4  Disable Debug Policy
5  Config PPI display
6  Provision RPMB
7  Enter Shell
8  Boot USB First
-> 9  MassStorage
10 Reboot
11 USB Menu
12 PMIC Menu
13 UEFI Menu
14 EDL Mode
15 Fastboot
```



5. Press the **Home/Power** key to launch the UEFI MassStorage app.

6. Mount the LogFS partition.

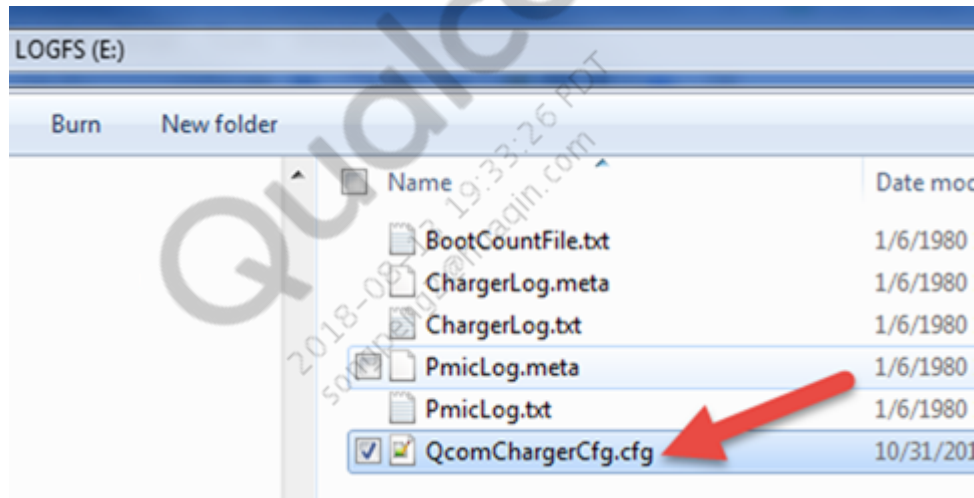
```

54 limits (4)
55 toolsfv (1024)
->**56 logfs (8192)
57 st1 (2048)
58 devcfg (128)
59 storsec (128)
60 storsecbak (128)
61 UFS-LUN 5 (1572864)
62 modemst1 (2048)
63 modemst2 (2048)
64 fsc (4)
67 RPMB (16384)

68 <Mount Partitions>
KeyAction: Mounting Drives...
Unmount error: Not Found
MountSelection: Mounting logfs on Lun 0
MountSelection: Press Any Key To Unmount

```

7. Copy the QcomChargerCfg.cfg file to the LogFS partition root level with the modified configuration.



8. Remove LogFS from the task bar.
9. Reboot the device.

The new configuration takes effect and the modified configuration is listed in the LogFS configuration file, that is, the configuration file changes to enable the charger.

11 FAQs

How can charging be enabled on the pre-ES2 software?

Remove qcom, suspend-input; from the pmicobalt_charger device tree node by removing the text shown in ~~blue strikethrough~~:

```
/arch/arm/boot/dts/qcom/msm-pmicobalt.dtsi
pmicobalt_charger: qcom,qnp-smb2 {
    compatible = "qcom,qnp-smb2";
    #address-cells = <1>;
    #size-cells = <1>;
    qcom,pmic-revid = <&pmicobalt_revid>;
    /* do not draw current from USB or DC */
qcom,suspend-input;
    dpdm-supply = <&qusb_phy0>;
```


A Input current limits

The input current limits by charger type in the following table only applies to 2.0 silicon for PMI8998 (not 1.0 silicon).

BC1.2 result	Type-C result	Legacy cable?	Use BC1.2 for DCP, OCP, or floating?	ICL min (mA)	ICL start (mA)	ICL max (mA)	Charger type
SDP_100 / USB100 (disabled)	Standard current	Yes	N/A	25	100/150	100/150	Legacy charger with Type-A to Type-C cable
SDP_100 / USB100 (disabled)	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	Type-C 1.5 A charger with Type-C to Type-C cable
SDP_100 / USB100 (disabled)	Medium current	Yes	N/A	25	100/150	100/150	SDP charger with noncompliant legacy Type-A to Type-C cable
SDP_100 / USB100 (disabled)	High current	No	N/A	25	500	max(ICL_CFG, 3000 mA)	Type-C 3.0 A charger with Type-C to Type-C cable
SDP_100 / USB100 (disabled)	High current	Yes	N/A	25	100/150	100/150	SDP charger with noncompliant legacy Type-A to Type-C cable
SDP_500 / USB500 (disabled)	Standard current	Yes	N/A	25	500	500/900	Legacy charger with Type-A to Type-C cable
SDP_500 / USB500 (disabled)	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	Type-C 1.5 A charger with Type-C to Type-C cable
SDP_500 / USB500 (disabled)	Medium current	Yes	N/A	25	500	500/900	SDP charger with noncompliant legacy Type-A to Type-C cable
SDP_500 / USB500 (disabled)	High current	No	N/A	25	500	max(ICL_CFG, 3000 mA)	Type-C 3.0 A charger with Type-C to Type-C cable
SDP_500 / USB500 (disabled)	High current	Yes	N/A	25	500	500/900	SDP charger with noncompliant legacy Type-A to Type-C cable
USBAC (disabled)	Standard current	Yes	N/A	25	500	max(ICL_CFG, 1500 mA)	Legacy charger with Type-A to Type-C cable

BC1.2 result	Type-C result	Legacy cable?	Use BC1.2 for DCP, OCP, or floating?	ICL min (mA)	ICL start (mA)	ICL max (mA)	Charger type
USBAC (disabled)	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	Type-C 1.5 A charger with Type-C to Type-C cable
USBAC (disabled)	Medium current	Yes	N/A	25	500	max(ICL_CFG, 1500 mA)	Unknown charger with noncompliant legacy Type-A to Type-C cable or captive 1.5 A Type-C charger
USBAC (disabled)	High current	No	N/A	25	500	max(ICL_CFG, 3000 mA)	Type-C 3.0 A charger with Type-C to Type-C cable
USBAC (disabled)	High current	Yes	N/A	25	500	max(ICL_CFG, 3000 mA)	Unknown charger with noncompliant legacy Type-A to Type-C cable or captive 3.0 A Type-C charger
CDP	Standard current	Yes	N/A	25	500	min(ICL_CFG, 1500 mA)	CDP charger with legacy Type-A to Type-C cable
CDP	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	CDP Type-C 1.5 A charger with Type-C to Type-C cable
CDP	Medium current	Yes	N/A	25	500	min(ICL_CFG, 1500 mA)	CDP charger with noncompliant legacy Type-A to Type-C cable
CDP	High current	No	N/A	25	500	max(ICL_CFG, 3000 mA)	CDP Type-C 3.0 A charger with Type-C to Type-C cable
CDP	High current	Yes	N/A	25	500	min(ICL_CFG, 1500 mA)	CDP charger with noncompliant legacy Type-A to Type-C cable
Floating_100	Standard current	Yes	N/A	25	100/150	100/150	Floating charger with legacy Type-A to Type-C cable
Floating_100	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	Floating Type-C 1.5 A charger with Type-C to Type-C cable
Floating_100	Medium current	Yes	N/A	25	100/150	100/150	Floating charger with noncompliant legacy Type-A to Type-C cable or captive 1.5 A Type-C charger
Floating_100	High current	No	N/A	25	500	max(ICL_CFG, 3000 mA)	Floating Type-C 3.0 A charger with Type-C to Type-C cable
Floating_100	High current	Yes	N/A	25	100/150	100/150	Floating charger with noncompliant legacy Type-A to Type-C cable or captive 3.0 A Type-C charger

BC1.2 result	Type-C result	Legacy cable?	Use BC1.2 for DCP, OCP, or floating?	ICL min (mA)	ICL start (mA)	ICL max (mA)	Charger type
Floating_500	Standard current	Yes	N/A	25	500	500/900	Floating charger with legacy Type-A to Type-C cable
Floating_500	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	Floating Type-C 1.5 A charger with Type-C to Type-C cable
Floating_500	Medium current	Yes	N/A	25	500	500/900	Floating charger with noncompliant legacy Type-A to Type-C cable or captive 1.5 A Type-C charger
Floating_500	High current	No	N/A	25	500	max(ICL_CFG, 3000 mA)	Floating Type-C 3.0 A charger with Type-C to Type-C cable
Floating_500	High current	Yes	N/A	25	500	500/900	Floating charger with noncompliant legacy Type-A to Type-C cable or captive 3.0 A Type-C charger
Floating_HC	Standard current	Yes	N/A	25	500	ICL_CFG	Floating charger with legacy Type-A to Type-C cable
Floating_HC	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	Floating Type-C 1.5 A charger with Type-C to Type-C cable
Floating_HC	Medium current	Yes	No	25	500	max(ICL_CFG, 1500 mA)	Floating charger with noncompliant legacy Type-A to Type-C cable or captive 1.5 A Type-C charger
Floating_HC	Medium current	Yes	Yes	25	500	ICL_CFG	Floating charger with noncompliant legacy Type-A to Type-C cable or captive 1.5 A Type-C charger
Floating_HC	High current	No	N/A	25	500	max(ICL_CFG, 3000 mA)	Floating Type-C 3.0 A charger with Type-C to Type-C cable
Floating_HC	High current	Yes	No	25	500	max(ICL_CFG, 3000 mA)	Floating charger with noncompliant legacy Type-A to Type-C cable or captive 3.0 A Type-C charger
Floating_HC	High current	Yes	Yes	25	500	ICL_CFG	Floating charger with noncompliant legacy Type-A to Type-C cable or captive 3.0 A Type-C charger
OCP_500	Standard current	Yes	N/A	25	500	500	Proprietary charger with legacy Type-A to Type-C cable

BC1.2 result	Type-C result	Legacy cable?	Use BC1.2 for DCP, OCP, or floating?	ICL min (mA)	ICL start (mA)	ICL max (mA)	Charger type
OCP_500	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	Proprietary Type-C 1.5 A charger with Type-C to Type-C cable
OCP_500	Medium current	Yes	N/A	25	500	500	Proprietary charger with noncompliant legacy Type-A to Type-C cable
OCP_500	High current	No	N/A	25	500	max(ICL_CFG, 3000 mA)	Proprietary Type-C 3.0 A charger with Type-C to Type-C cable
OCP_500	High current	Yes	N/A	25	500	500	Proprietary charger with noncompliant legacy Type-A to Type-C cable
OCP_HC	Standard current	Yes	N/A	25	500	ICL_CFG	Proprietary charger with legacy Type-A to Type-C cable
OCP_HC	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	Proprietary Type-C 1.5 A charger with Type-C to Type-C cable
OCP_HC	Medium current	Yes	No	25	500	max(ICL_CFG, 1500 mA)	Proprietary charger with noncompliant legacy Type-A to Type-C cable
OCP_HC	Medium current	Yes	Yes	25	500	ICL_CFG	Proprietary charger with noncompliant legacy Type-A to Type-C cable
OCP_HC	High current	No		25	500	max(ICL_CFG, 3000 mA)	Proprietary Type-C 3.0 A charger with Type-C to Type-C cable
OCP_HC	High current	Yes	No	25	500	max(ICL_CFG, 3000 mA)	Proprietary charger with noncompliant legacy Type-A to Type-C cable
OCP_HC	High current	Yes	Yes	25	500	ICL_CFG	Proprietary charger with noncompliant legacy Type-A to Type-C cable
DCP	Standard current	Yes	N/A	25	500	ICL_CFG	DCP charger with legacy Type-A to Type-C cable
DCP	Medium current	No	N/A	25	500	max(ICL_CFG, 1500 mA)	DCP Type-C 1.5 A charger with Type-C to Type-C cable
DCP	Medium current	Yes	No	25	500	max(ICL_CFG, 1500 mA)	DCP charger with noncompliant legacy Type-A to Type-C cable or captive 1.5 A Type-C charger
DCP	Medium current	Yes	Yes	25	500	ICL_CFG	DCP charger with noncompliant legacy Type-A to Type-C cable or captive 1.5 A Type-C charger

BC1.2 result	Type-C result	Legacy cable?	Use BC1.2 for DCP, OCP, or floating?	ICL min (mA)	ICL start (mA)	ICL max (mA)	Charger type
DCP	High current	No	N/A	25	500	max(ICL_CFG, 3000 mA)	DCP Type-C 3.0 A charger with Type-C to Type-C cable
DCP	High current	Yes	No	25	500	max(ICL_CFG, 3000 mA)	DCP charger with noncompliant legacy Type-A to Type-C cable or captive 3.0 A Type-C charger
DCP	High current	Yes	Yes	25	500	ICL_CFG	DCP charger with noncompliant legacy Type-A to Type-C cable or captive 3.0 A Type-C charger
HVDCP	Standard current	Yes	N/A	25	500	3000	HVDCP charger with legacy Type-A to Type-C cable
HVDCP	Medium current	No	N/A	25	500	3000	HVDCP Type-C 1.5 A charger with Type-C to Type-C cable
HVDCP	Medium current	Yes	N/A	25	500	3000	HVDCP charger with noncompliant legacy Type-A to Type-C cable
HVDCP	High current	No	N/A	25	500	3000	HVDCP Type-C 3.0 A charger with Type-C to Type-C cable
HVDCP	High current	Yes	N/A	25	500	3000	HVDCP charger with noncompliant legacy Type-A to Type-C cable
Powered audio	Ra	N/A	N/A	25	500	500	Powered audio adapter
SDP_100 / USB100 (disabled)	Disabled	N/A	N/A	25	100/150	100/150	USB 1.0 SDP charger
SDP_500 / USB500 (disabled)	Disabled	N/A	N/A	25	500	500/900	USB 2.0 SDP charger
USBAC (disabled)	Disabled	N/A	N/A	25	500	ICL_CFG	Unknown charger
CDP	Disabled	N/A	N/A	25	500	max(ICL_CFG, 1500 mA)	CDP charger
Floating_100	Disabled	N/A	N/A	25	100/150	100/150	Noncompliant floating charger
Floating_500	Disabled	N/A	N/A	25	500	500/900	Noncompliant floating charger
Floating_HC	Disabled	N/A	N/A	25	500	ICL_CFG	Noncompliant floating charger
OCP_500	Disabled	N/A	N/A	25	500	500	Noncompliant proprietary charger
OCP_HC	Disabled	N/A	N/A	25	500	ICL_CFG	Noncompliant proprietary charger

BC1.2 result	Type-C result	Legacy cable?	Use BC1.2 for DCP, OCP, or floating?	ICL min (mA)	ICL start (mA)	ICL max (mA)	Charger type
DCP	Disabled	N/A	N/A	25	500	ICL_CFG	DCP charger
HVDCP	Disabled	N/A	N/A	25	500	ICL_CFG	HVDCP charger
SDP_100 / USB100 (disabled)	Floating	N/A	N/A	25	100/150	100/150	SDP charger with noncompliant type-C charger or cable
SDP_500 / USB500 (disabled)	Floating	N/A	N/A	25	500	500/900	SDP charger with noncompliant type-C charger or cable
USBAC (disabled)	Floating	N/A	N/A	25	500	ICL_CFG	Unknown charger with noncompliant type-C charger or cable
CDP	Floating	N/A	N/A	25	500	min(ICL_CFG, 1500 mA)	CDP charger with noncompliant type-C charger or cable
Floating_100	Floating	N/A	N/A	25	100/150	100/150	Floating charger with noncompliant type-C charger or cable
Floating_500	Floating	N/A	N/A	25	500	500/900	Floating charger with noncompliant type-C charger or cable
Floating_HC	Floating	N/A	N/A	25	500	ICL_CFG	Floating charger with noncompliant type-C charger or cable
OCP_500	Floating	N/A	N/A	25	500	500	Proprietary charger with noncompliant type-C charger or cable
OCP_HC	Floating	N/A	N/A	25	500	ICL_CFG	Proprietary charger with noncompliant type-C charger or cable
DCP	Floating	N/A	N/A	25	500	ICL_CFG	DCP charger with noncompliant type-C charger or cable
HVDCP	Floating	N/A	N/A	25	500	ICL_CFG	HVDCP charger with noncompliant type-C charger or cable
9 V/12V charger	N/A	N/A	N/A	25	500	ICL_CFG	High-voltage charger
Type-C FMB	FMB	N/A	N/A	25	1500	3000 or 4000	Type-C factory mode boot
uUSB FMB	FMB	N/A	N/A	25	1500	3000 or 4000	Type-C factory mode boot
DCIN Qi	N/A	N/A	N/A	25	25 or 500	ICL_CFG	Qi charger likely chooses 25 mA for ICL_START
DCIN WiPower	N/A	N/A	N/A	25	25 or 500	ICL_CFG	WiPower charger likely chooses ADC method for AICLL

B References

B.1 Related documents

Title	Number
Qualcomm Technologies, Inc.	
<i>PM8998, PM8005, and PMI8998 Power Management ICs Design Guidelines</i>	80-P1086-5
<i>PMI8998 Hardware Register Description</i>	80-P1087-2X
<i>UEFI PMIC Software User Guide</i>	80-P2484-42
<i>Linux Android PMIC Fuel Gauge (Gen 3) Software User Guide</i>	80-P2484-74
<i>MSM8998.LA Linux Voltage Regulator Software User Guide</i>	80-P2484-79
<i>SDM845 Linux Android PMIC Voltage Regulator and Clock Software User Guide</i>	80-P9301-79
Resources	
USB Battery Charging 1.2 Compliance Plan	www.usb.org

B.2 Acronyms and terms

Acronym or term	Definition
BC1.2	Battery charging specification 1.2
CDP	Charging downstream port
CV	Constant voltage
DCD	Data contact detect
DCP	Dedicated charging port
DFP	Downstream facing port
ICHG	Battery current
ICL	Input current limit
OCP	Other charging port
SDP	Standard downstream port
SoC	State of charge
UFP	Upstream facing port