Qualcomm Technologies, Inc.

PMI632 Battery Temperature Sensing and Compensation Configuration

80-P3255-39 B

July 20, 2018

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

Revision history

Revision	Date	Description
A	May 2018	Initial release
B	July	Updated section 2.6

Contents

1 Introduction

1.1 Purpose

This document helps to configure the ADC parameters for Battery temperature sensing and compensation based on the customer-specific battery NTC specification.

Use the attached spreadsheet to generate the conversion table and JEITA ADC codes.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `cp armcc armcpp`.

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

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

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

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

1.3 Technical assistance

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

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

2 Overview

The Qualcomm® Battery Gauge™ module monitors battery temperature using a dedicated BATT_THERM pin via VADC. Information about the battery's temperature is used by the Qualcomm Battery Gauge for the following purposes:

- Improve SoC accuracy by adjusting the Qualcomm Battery Gauge models based on the temperature
- Accommodate charger operation with respect to JEITA requirements

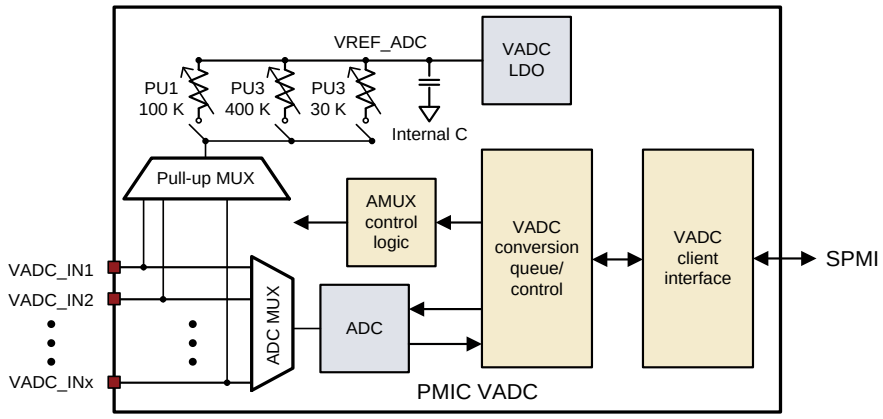
Previous scheme

- Thermistors require a type of bias to generate a voltage
- Past designs have used external dedicated or shared pull-ups
- Adds a pull-up for each thermistor
- Requires package pin for LDO output
- Consumes power-on all thermistors whenever LDO is on

PMI632 scheme

Optimizing cost and performance, PMI632 features internal pull-up resistors.

- Three pull-up values (30K, 100K, 400K)
- Trimmed to within 0.5% accuracy – Better tolerance than discrete R
- Low temperature coefficient (< 100 ppm/C) – Better than discrete R
- Bias source is a calibrated internal LDO – No package pin
- LDO is automatically controlled by ADC hardware
- Each thermistor is only biased when measured – Saves power



2.1 Functionality

The Qualcomm Battery Gauge uses a ratiometric conversion. The advantage of this design is that the pull-up used to bias the thermistor is the same regulator internal to the PMIC that supplies the ADC. This design allows a more robust measurement scheme and simplifies thermistor modeling. The BATT_THERM_BIAS control is also automatically handled by hardware.

2.2 BATT_THERM_RBIAS pull-up selection

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

2.3 Conversion table and thermistor selection

It is recommended to choose a thermistor with a beta value on the lower end of the supported range. This selection improves accuracy over the entire temperature range as a lower beta value varies less over temperature.

To convert from voltage measurements to temperature readings, the PMIC uses a linear approximation. This approximation uses the conversion table, which is an array of a mapping of {mV, °C}.

To function correctly, the conversion table must be programmed into the Qualcomm Battery Gauge for the thermistor value selected. Failure to do this can result in inaccurate temperature measurement.

2.4 Conversion table creation

1. Download the attached spreadsheet.
2. Enter NTC resistor/beta values based on the battery datasheet.

Thermistor					
NTC Resistor =	100	kΩ	1%		enter NTC Resistor /Beta based on Battery datasheet
@	25	°C			
NTC Beta =	4250	K	1%		

3. Chose the pull-up value closest to the NTC resistor.

Temperature (°C)	100K pull-up	400K pull-up	30K pull-up
	Vtherm (V)	Vtherm (V)	Vtherm (V)
-40	1.840	1.744	1.864
-38	1.835	1.724	1.863
-36	1.828	1.701	1.861
-34	1.821	1.676	1.858
-32	1.813	1.648	1.856
-30	1.803	1.618	1.853

Temperature (°C)	100K pull-up	400K pull-up	30K pull-up
	Vtherm (V)	Vtherm (V)	Vtherm (V)
-28	1.793	1.584	1.850
-26	1.781	1.548	1.846
:	:	:	:
80	0.184	0.050	0.498
82	0.173	0.046	0.474
84	0.163	0.043	0.451
86	0.153	0.041	0.428
88	0.144	0.038	0.407
90	0.136	0.036	0.387
92	0.128	0.034	0.367
94	0.120	0.032	0.349
96	0.114	0.030	0.332
98	0.107	0.028	0.315

4. Enter the JEITA threshold to generate the corresponding ADC code for each pull-up.

JEITA threshold	Temp (°C)**	ADC code*		
		100K pull-up	400K pull-up	30K pull-up
JEITA hot soft	40	25E3	CA8	46D4
JEITA hot hard	45	20B8	A73	4111
JEITA cold soft	5	5314	2E5B	65EB
JEITA cold hard	0	58CD	3624	6865
	* Choose the ADC code for the same pull-up chose for BAT_THERM_LUT			
	** Enter the JEITA threshold to generate the corresponding ADC code for each pull-up.			

2.5 Relevant files

ADC driver source	kernel/drivers/hwmon/qpnp-adc-common.c kernel/drivers/hwmon/qpnp-adc-voltage.c kernel/include/linux/qpnp/qpnp-adc.h kernel/drivers/power/supply/qcom/qg-util.c
Charger driver source	kernel/drivers/power/supply/qcom/smb5-lib.c
DTSI documentation	For the latest DTSI parameter information, refer to the DTSI binding documentation: kernel/Documentation/devicetree/bindings/hwmon/qpnp-adc-voltage.txt kernel/ Documentation/devicetree/bindings/batterydata/batterydata.txt
DTSI files	kernel/arch/arm64/boot/dts/qcom/pmi632.dtsi
SBL	boot_images/core/systemdrivers/pmic/target/msm8953_pm8953_pmi8950/system/src/pm_sbl_boot_oem.c
Battery profile dtsi	kernel/arch/arm64/boot/dts/qcom/qg-batterydata-mlp356477-2800mah.dtsi

2.6 Example – Update interpolation table for BATT_THERM

1. Add the interpolation table and the new scale function in qpnp-adc-common.c.

```
/* Voltage to temperature */
static const struct qpnp_vadc_map_pt adcmap_batt_therm_pu100_b4250[] = {
     {1840,    -400},
     {1835,    -380},
     {1828,    -360},
     {1821,    -340},
     {1813,    -320},
     {1803,    -300},
     {1793,    -280},
     {1781,    -260},
     {1768,    -240},
     {1753,    -220},
     {1737,    -200},
     {1719,    -180},
     {1700,    -160},
     {1679,    -140},
     {1655,    -120},
     {1630,    -100},
     {1603,    -80},
     {1574,    -60},
     {1543,    -40},
     {1510,    -20},
     {1475,    00},
     {1438,    20},
     {1400,    40},
     {1360,    60},
     {1318,    80},
     {1276,    100},
     {1232,    120},
     {1187,    140},
     {1142,    160},
     {1097,    180},
     {1051,    200},
     {1005,    220},
     {960,     240},
     {915,     260},
     {871,     280},
     {828,     300},
     {786,     320},
     {745,     340},
     {705,     360},
     {666,     380},
     {629,     400},
     {594,     420},
```

```
        {560,     440},
        {527,     460},
        {497,     480},
        {467,     500},
        {439,     520},
        {413,     540},
        {388,     560},
        {365,     580},
        {343,     600},
        {322,     620},
        {302,     640},
        {284,     660},
        {267,     680},
        {251,     700},
        {235,     720},
        {221,     740},
        {208,     760},
        {195,     780},
        {184,     800},
        {173,     820},
        {163,     840},
        {153,     860},
        {144,     880},
        {136,     900},
        {128,     920},
        {120,     940},
        {114,     960},
        {107,     980}
};

int32_t qpnp_adc_batt_therm_B4250_pu100 (struct qpnp_vadc_chip *chip,
                int32_t adc_code,
                const struct qpnp_adc_properties *adc_properties,
                const struct qpnp_vadc_chan_properties *chan_properties,
                struct qpnp_vadc_result *adc_chan_result)
{
        int64_t batt_thm_voltage = 0;

        if (!chan_properties || !chan_properties->offset_gain_numerator
||
                !chan_properties->offset_gain_denominator
|| !adc_properties
                || !adc_chan_result)
                return -EINVAL;

        /* (code * vref_vadc (1.875V) * 1000) / (scale_code * 1000) */
        if (adc_code > QPNP_VADC_HC_MAX_CODE)
                adc_code = 0;
```

```
        batt_thm_voltage = (int64_t) adc_code;
        batt_thm_voltage *= (adc_properties->adc_vdd_reference
                                                * 1000);
        batt_thm_voltage = div64_s64(batt_thm_voltage,
                        adc_properties->full_scale_code * 1000);
        qpnp_adc_map_voltage_temp(adcmap_batt_therm_pu100_b4250,
                ARRAY_SIZE(adcmap_batt_therm_pu100_b4250),
                batt_thm_voltage, &adc_chan_result->physical);
        return 0;
}
EXPORT_SYMBOL(qpnp_adc_batt_therm_B4250_pu100);
```

2. Update the enum corresponding to the table in qpnp-adc.h and add a new scale function declaration.

```
enum qpnp_adc_scale_fn_type {
SCALE_DEFAULT = 0,
…
SCALE_BATT_THERM_B4250_PU100,
SCALE_NONE,
}


/**
 * qpnp_adc_batt_therm_B4250_pu100() - Scales the pre-calibrated digital
output
 *             of an ADC to the ADC reference and compensates for the
 *             gain and offset. Returns the temperature in decidegC.
 *             It uses a mapping table computed for a 400K pull-up.
 * @dev:       Structure device for qpnp vadc
 * @adc_code:  pre-calibrated digital output of the ADC.
 * @adc_prop:  adc properties of the adc such as bit resolution,
 *             reference voltage.
 * @chan_prop: individual channel properties to compensate the i/p
scaling,
 *             slope and offset.
 * @chan_rslt: physical result to be stored.
 */
int32_t qpnp_adc_batt_therm_B4250_pu100 (struct qpnp_vadc_chip *dev,
                        int32_t adc_code,
                        const struct qpnp_adc_properties *adc_prop,
                        const struct qpnp_vadc_chan_properties
*chan_prop,
                        struct qpnp_vadc_result *chan_rslt);
```

3. Add a new entry to the mapping table of qpnp-adc-voltage.c and equate it to the enum in qpnp-adc-common.c and qpnp-adc.h.

```
static struct qpnp_vadc_scale_fn vadc_scale_fn[] = {
        [SCALE_USBIN_I] = {qpnp_adc_scale_usbin_curr},
        [SCALE_BATT_THERM_TEMP_QRD] = {qpnp_adc_batt_therm_qrd},
        [SCALE_SMB1390_DIE_TEMP] = {qpnp_adc_scale_die_temp_1390},
+       [SCALE_BATT_THERM_B4250_PU100] =
{ qpnp_adc_batt_therm_B4250_pu100},

 };
```

4. Add or modify the VADC channel node in the PMI632.dtsi based on the pull-up value. Select the ADC hardware channel number based on the pull-up.

Internal-pull up	100K PU	400K PU	30K PU	No-Pull up
Hardware channel	0x4a	0x6a	0x2a	0xa

```
chan@4a {
         label = "bat_therm_PU100_B4250";
         reg = <0x4a>;  → ADC HW channel number based on pull up
         qcom,decimation = <2>;
         qcom,pre-div-channel-scaling = <0>;
         qcom,calibration-type = "ratiometric";
         qcom,scale-function = <24>; → must match
qpnp_adc_scale_fn_type enum
         qcom,hw-settle-time = <0>;
         qcom,fast-avg-setup = <0>;
         qcom,cal-val = <0>;
                 };
```

5. Update the pull in the charger peripheral register in the boot_images/core/ systemdrivers/pmic/config/msm8953/pmi632/ pm_config_target_sbl_sequence.c file.

```
INTERNAL_PULL_UP_BAT_THM_NO_PULLUP 0
INTERNAL_PULL_UP_BAT_THM_30K_PULLUP 1
INTERNAL_PULL_UP_BAT_THM_100K_PULLUP 2
INTERNAL_PULL_UP_BAT_THM_400K_PULLUP 3

// MODE - SCHG_Configs : 35
   //sid   data                        register Mask     reg
op                   cond start             cond end

{ 2,                     0x02,                     0x1286,
   0xFF,     PM_SBL_WRITE,              0,                              0}
,           // Line 43     Source:
Write(Nebula.SCHG_BATIF.ADC_INTERNAL_PULL_UP, 0x02)  →  default code
++{ 2,                     0x01,                     0x1286,
     0xFF,     PM_SBL_WRITE,              0,
0},           // Line 43     Source:
```

```
Write(Nebula.SCHG_BATIF.ADC_INTERNAL_PULL_UP, 0x01)  ➔ add new line to
modify as 0x1 for 30k pull up
```

6. Update the JEITA threshold in the battery profile dtsi - qg-batterydata-mlp356477-2800mah.dtsi

```
qcom,mlp356477_2800mah {
     /* mlp356477_2800mah_averaged_MasterSlave_Aug14th2017 */
     qcom,max-voltage-uv = <4400000>;
     qcom,fg-cc-cv-threshold-mv = <4390>;
     qcom,fastchg-current-ma = <4200>;
     qcom,batt-id-kohm = <82>;
     qcom,battery-beta = <4250>;
     qcom,battery-therm-kohm = <32>;
     qcom,battery-type =
          "mlp356477_2800mah_averaged_MasterSlave_Aug14th2017";
     qcom,qg-batt-profile-ver = <100>;

     qcom,jeita-fcc-ranges = <0   150   560000
                    151 450  4200000
                    451 550  2380000>;
     qcom,jeita-fv-ranges =  <0   150  4150000
                    151 450  4400000
                    451 550  4150000>;
+        qcom,jeita-soft-thresholds = <0x5314 0x25e3>;
+        qcom,jeita-hard-thresholds = <0x58cd 0x20b8>;
```

7. Change the channel based on pull up in drivers/power/supply/qcom/qg-util.c.

/* No pull up */	VADC_BAT_THERM	0xa
/* PU1 is 30K pull up */	VADC_BAT_THERM_PU1	0x2a
/* PU2 is 100K pull up */	VADC_BAT_THERM_PU2	0x4a
/* PU3 is 400K pull up */	VADC_BAT_THERM_PU3	0x6a

```
--- a/drivers/power/supply/qcom/qg-util.c
+++ b/drivers/power/supply/qcom/qg-util.c
@@ -303,14 +303,14 @@ int qg_get_battery_temp(struct qpnp_qg *chip, int
*temp)
          return 0;
     }

-    rc = qpnp_vadc_read(chip->vadc_dev, VADC_BAT_THERM_PU2, &result);
+    rc = qpnp_vadc_read(chip->vadc_dev, VADC_BAT_THERM_PU1, &result);
```

A References

A.1 Related documents

Title	Number
Qualcomm Technologies, Inc.	
PMIC Dump and Register Access Overview	80-NL708-1
SDM632 + PMI632-5 + PM8953 + PM8004-1 Reference Schematic	80-PF078-41
PM8953 + PMI632 Power Management IC Design Guidelines	80-PF457-5A

A.2 Acronyms and terms

Acronym or term	Definition
ADC	Analog to digital converter
DTSI	Device tree source include