KBA-180523214236

[Description]:

How to configure JEITA threshold for PMI632?

[Platform]:

SDM632, SDM450

[Solution]:

First, need below patch for configuring the JEITA threshold through DTSI.
https://source.codeaurora.org/quic/la/kernel/msm-4.9/commit/?id=
f05bd55755f9b37caac850564cd5600ff831e0c6&h=msm-4.9
https://source.codeaurora.org/quic/la/kernel/msm-4.9/commit/?id=
589939b6f21db8ef44b49ae62377a8645f429f3b&h=msm-4.9

Qualcomm
2018-12-20 23:36:22 PST

For PMI632 the HW jeita threshold is filled by adc code, but not specific temperature, that's to say it need to be configured based on the actual ntc voltage-temp mapping table.
For example below table <adcmap_batt_therm>, if we need jeita soft hot threshold to be 45C, according to the table the ntc voltage need to be 465mV, then the soft hot threshold will be
adc_code=voltage*full_scale/vdd_ref=465*0x70e4/1875=0x1BFF

```
+ qcom,jeita-soft-thresholds = <0x3ecc 0x1bff>;
+ qcom,jeita-hard-thresholds = <0x4aff 0x15aa>;
```

full_scale & vdd_ref refer to below vadc node.

```
pmi632_vadc: vadc@3100 {
qcom,adc-vdd-reference = <1875>;
qcom,adc-full-scale-code = <0x70e4>;
```

```
/* Voltage to temperature */
static const struct qpnp_vadc_map_pt adcmap_batt_therm[] = {
.........................
.........................
{555, 380},
{528, 400},
{502, 420},
{477, 440},
{453, 460},
{430, 480},
{409, 500},
.........................
.........................
};
```

In PMI632, besides JEITA soft & hard limit, there're also JEITA TooHot AFP threshold & TooCold AFP threshold, which need to be configured higher/lower than the TooHot/TooCold threshold.
For these two thresholds, we can configure in SBL. Also from below SBL code we can configure all the JEITA settings.
@boot_images/core/systemdrivers/pmic/config/{platform target}/pm_config_target_sbl_sequence.c

```
{ 2, 0x1B, 0x1094, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 48 Source: Write(
Nebula.SCHG_CHGR.JEITA_HOT_THRESHOLD_MSB, 0x1B)
{ 2, 0xFF, 0x1095, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 49 Source: Write(
Nebula.SCHG_CHGR.JEITA_HOT_THRESHOLD_LSB, 0xFF)
{ 2, 0x44, 0x1096, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 50 Source: Write(
Nebula.SCHG_CHGR.JEITA_COLD_THRESHOLD_MSB, 0x44)
{ 2, 0xC7, 0x1097, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 51 Source: Write(
Nebula.SCHG_CHGR.JEITA_COLD_THRESHOLD_LSB, 0xC7)
{ 2, 0x15, 0x1098, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 52 Source: Write(
Nebula.SCHG_CHGR.JEITA_THOT_THRESHOLD_MSB, 0x15)
{ 2, 0xAA, 0x1099, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 53 Source: Write(
Nebula.SCHG_CHGR.JEITA_THOT_THRESHOLD_LSB, 0xAA)
{ 2, 0x4A, 0x109A, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 54 Source: Write(
Nebula.SCHG_CHGR.JEITA_TCOLD_THRESHOLD_MSB, 0x4A)
{ 2, 0xFF, 0x109B, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 55 Source: Write(
Nebula.SCHG_CHGR.JEITA_TCOLD_THRESHOLD_LSB, 0xFF)
{ 2, 0x13, 0x109C, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 56 Source: Write(Nebula.SCHG_CHGR.
JEITA_THOT_AFP_THRESHOLD_MSB, 0x13)
{ 2, 0x0A, 0x109D, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 57 Source: Write(Nebula.SCHG_CHGR.
JEITA_THOT_AFP_THRESHOLD_LSB, 0x0A)
{ 2, 0x55, 0x109E, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 58 Source: Write(Nebula.SCHG_CHGR.
JEITA_TCOLD_AFP_THRESHOLD_MSB, 0x55)
{ 2, 0xCE, 0x109F, 0xFF, PM_SBL_WRITE, 0, 0}, // Line 59 Source: Write(Nebula.SCHG_CHGR.
JEITA_TCOLD_AFP_THRESHOLD_LSB, 0xCE)
```

Qualcomm