[Description]:

Some customers want to use PMIC GPIO as ADC, but on 8150 and 6150 platforms, the API and method is changed.

[Platform]: SM8150, SM6150

[Solution]: PMIC GPIOs can be set as ADC at kernel side

Detail method:

Please confirm the HW circuit is correct and check the ADC change at PMIC device spec, such as PM6150, 80-PH856-1.

Below is one sample which set PM6150 GPIO9 as ADC.

1. Please check whether GPIO can control at kernel side first, if no access, please check KBA-181120001409
2. Ensure that the GPIO is disabled and that it is not being used by other processors, such as in project-pinctrl.c .


```
&pm6150_gpios {

gpio9_adc {

gpio9_adc_default: gpio9_adc_default {

pins = "gpio9"; /* GPIO 9 */

function = "normal"; /* normal */

bias-high-impedance; /* DISABLE GPIO9 for ADC*/

};

};

};
```

3. Add pinctrl-names and pinctrl-0 in the consumer node. For the pinctrl framework to configure GPIO at runtime,
The consumer node must use the default GPIO node

```
iclient_node {
```

……

pinctrl-names = "default";

pinctrl-0 = <&gpio9_adc_default>;

};

Such as you use it in charger module:

&pm6150_charger {

……

pinctrl-names = "default";

pinctrl-0 = <&gpio9_adc_default>;

}

4. Add GPIO ADC at adc_chans_pmic5[ADC_MAX_CHANNEL] table

Such as use 100k pull up

Change

[ADC_GPIO3_PU2] = ADC_CHAN_TEMP("gpio3_pu2", 1,

SCALE_HW_CALIB_THERM_100K_PULLUP)

To

[ADC_GPIO3_PU2] = ADC_CHAN_VOLT("gpio3_pu2", 1,

SCALE_HW_CALIB_DEFAULT)

5. Add the ADC in according dtsi

Such as sm6150.dtsi

&pm6150_vadc {

gpio3_pu2 {

reg = <ADC_GPIO3_PU2>;

label = "gpio3_pu2";

qcom,pre-scaling = <1 1>;

};

}

6. Add the ADC in in the consumer node, please check 80-PF777-72 3.3.4

client_node {

……

io-channels = <&pm6150_vadc ADC_GPIO3_PU2>;

io-channel-names = "gpio9_voltage";

};

Such as,

&pm6150_charger {

io-channels = <&pm6150_vadc ADC_USB_IN_V_16>,

<&pm6150_vadc ADC_USB_IN_I>,

<&pm6150_vadc ADC_CHG_TEMP>,

<&pm6150_vadc ADC_DIE_TEMP>,

<&pm6150_vadc ADC_AMUX_THM4_PU2>,

<&pm6150_vadc ADC_SBUx>,

<&pm6150_vadc ADC_VPH_PWR>,

<&pm6150_vadc ADC_GPIO3_PU2>;

io-channel-names = "usb_in_voltage",

"usb_in_current",

"chg_temp",

"die_temp",

"conn_temp",

"sbux_res",

"vph_voltage",

"gpio9_voltage";

7. Driver: Associate the client with the corresponding device and get the device structure in the driver.

/* Get the ADC device instance (one time) */

struct iio_channel *gpio9_voltage_chan;

gpio9_voltage_cha = iio_channel_get(chg->dev, "gpio9_voltage");

8. Driver: Use the IIO API to read the ADC channel (vph_pwr_uv contains the result in microvolts).

int rc, gpio9_uv = -EINVAL;

if (!gpio9_voltage_cha)

return -EINVAL;

rc = iio_read_channel_processed(gpio9_voltage_cha, &gpio9_uv);

if (rc < 0) {

pr_err("Error in reading gpio9_voltage channel, rc:%d\n", rc);

return rc;

}