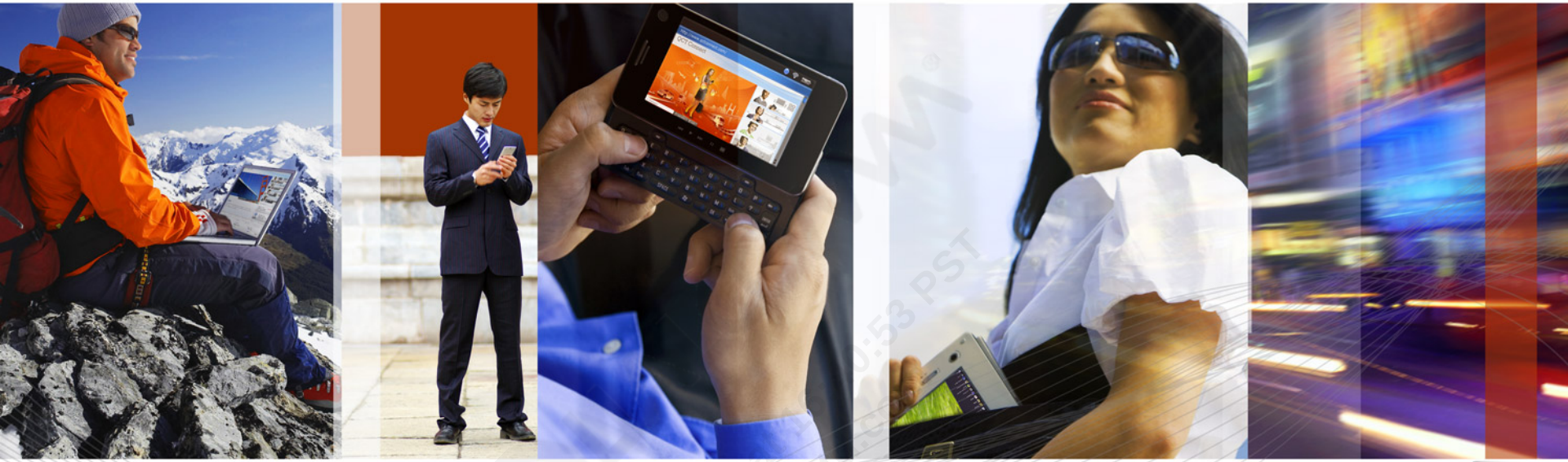


QUALCOMM® CDMA Technologies

REDEFINING MOBILITY

MSM8960™/PM8921™ ADC Drivers Overview

80-N8212-1 A

Qualcomm Confidential and Proprietary

Restricted Distribution. Not to be distributed to anyone who is not an employee of either Qualcomm or a subsidiary of Qualcomm without the express approval of Qualcomm's Configuration Management.

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

Qualcomm reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed for any damages arising directly or indirectly by their use or application. The information provided in this document is provided on an "as is" basis.

This document contains Qualcomm confidential and proprietary information and must be shredded when discarded.

QUALCOMM is a registered trademark of QUALCOMM Incorporated in the United States and may be registered in other countries. Other product and brand names may be trademarks or registered trademarks of their respective owners. CDMA2000 is a registered certification mark of the Telecommunications Industry Association, used under license. ARM is a registered trademark of ARM Limited. QDSP is a registered trademark of QUALCOMM Incorporated in the United States and other countries.

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 Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
U.S.A.

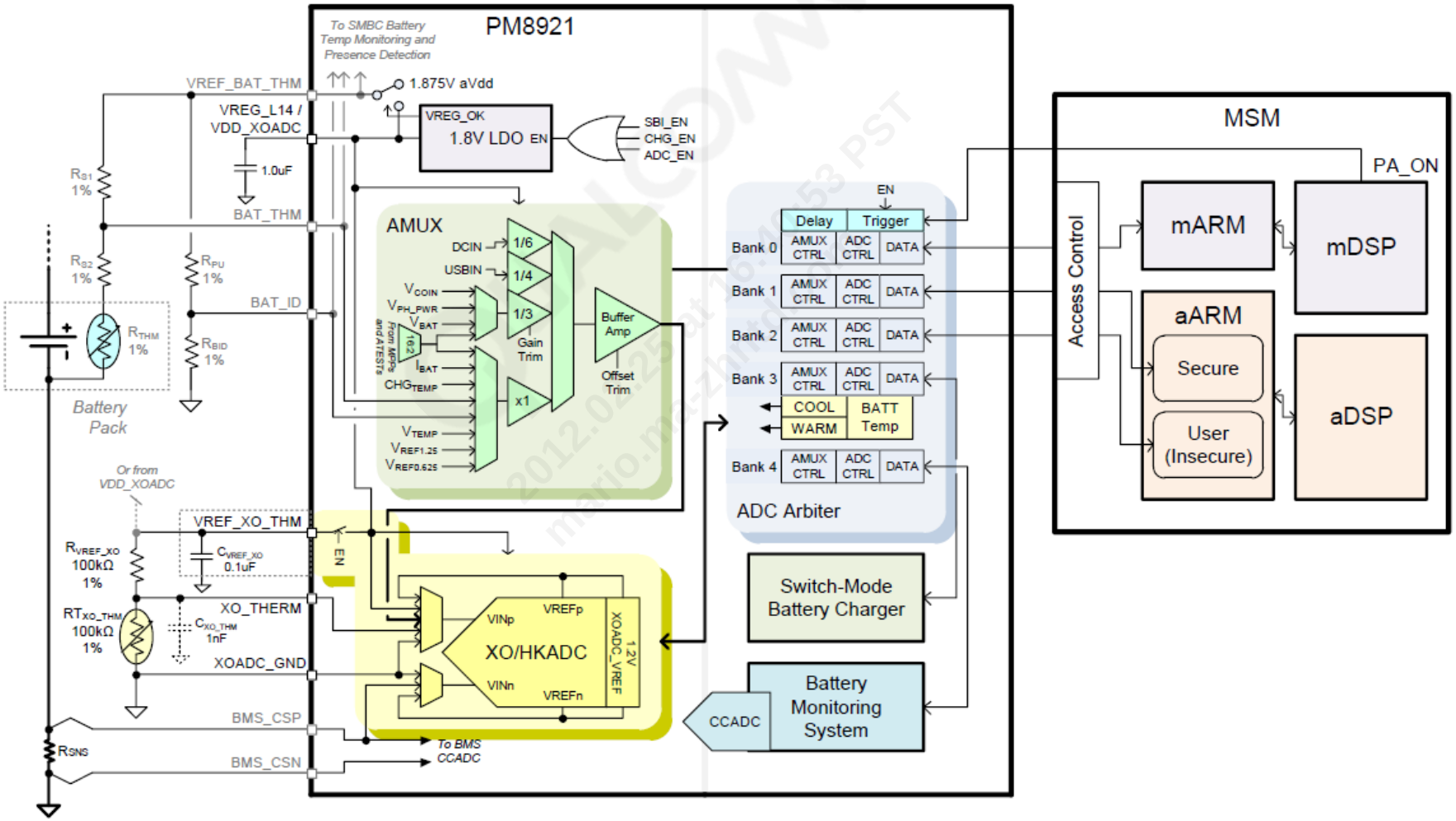
Revision History

Version	Date	Description
A	Nov 2011	Initial release

Contents

PM8921™ ADC Overview

PM8921 ADC Subsystem Diagram

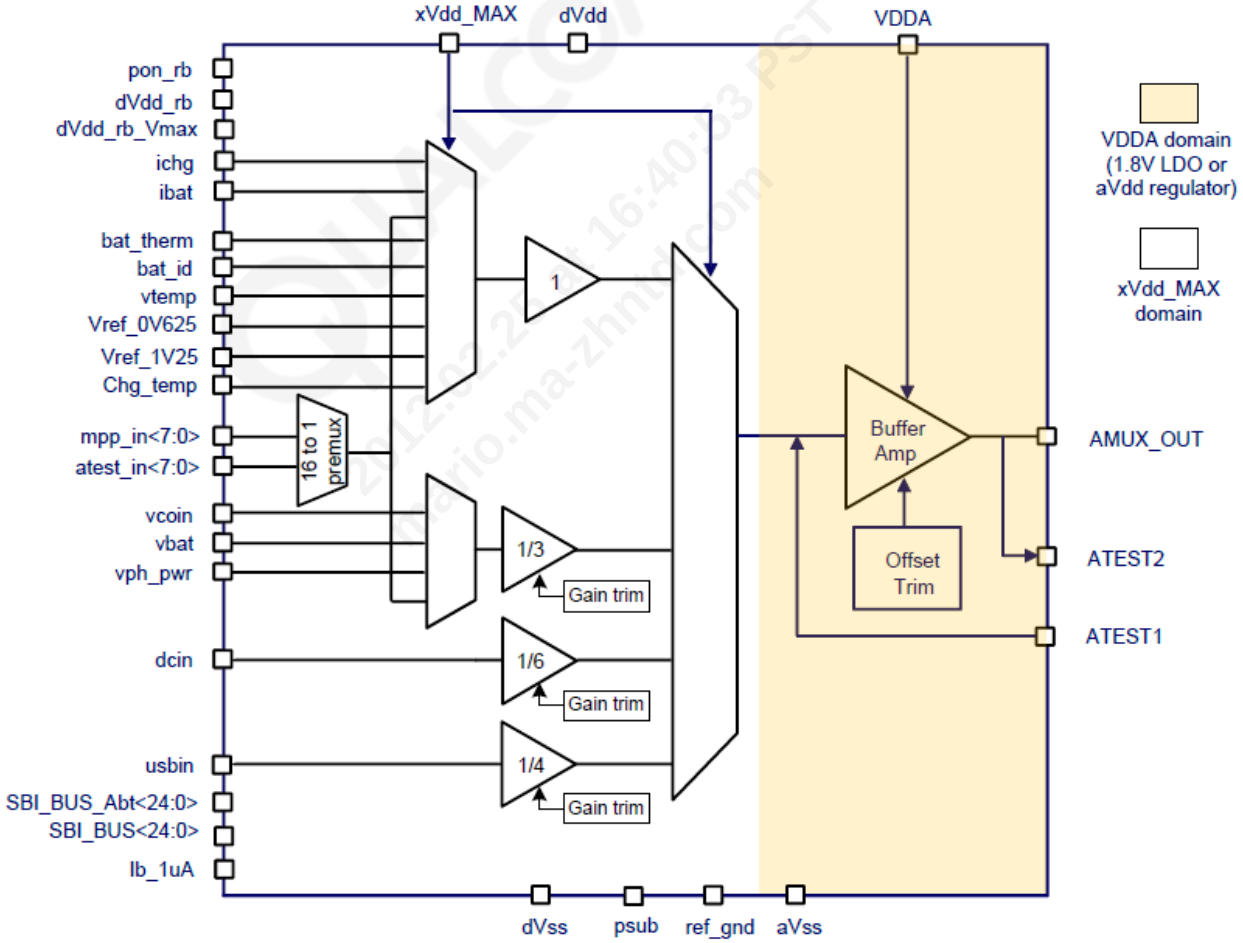


PM8921 ADC Arbiter

- ADC conversions can be requested by configuring up to five arbiter register banks:
  - ADC_ARB_SECP
  - ADC_ARB_USRP
  - ADC_ARB_MP
  - ADC_ARB_BMS
  - ADC_ARB_BTM
- When a bank requests a conversion, its mirrored Amux/ADC register contents are written to the corresponding Amux/ADC register.
- The converted ADC value is stored in the bank's data registers when the conversion is completed.
- With arbitration taken care of automatically in hardware, each client can control the Amux and ADC as if it has its own Amux and ADC.

PM8921 Main Mux and Premux Channels

- PM8921 ADC has support to read 16 Amux input signals and a 16:1 premux to the input Amux-MPP channels.



PM8921 Amux Channels

Main mux

	AMUX_CNTRL	Signal	Scaling
	0x00	VCOIN	x1/3
	0x10	VBAT	x1/3
	0x20	DCIN	x1/6
	0x30	ICHG	x1
	0x40	VPH_PWR	x1/3
	0x50	IBAT	x1
Main-mux-6	0xM4	MPP or ATEST	x1
Main-mux-7	0xM8	MPP or ATEST	x1/3
	0x80	BAT_THERM	x1
	0x90	BAT_ID	x1
	0xA0	USBIN	x1/4
	0xB0	DIE_TEMP	x1
	0xC0	0.625 V V_{REF}/2	x1
	0xD0	1.25 V V_{REF}	x1
	0xE0	CHG_TEMP	x1
	0xF0	AMUX OFF	N/A

Premux

M	Signal
0x0	ATEST<8>
0x1	USBIN_SNS_DIV20
0x2	DCIN_SNS_DIV20
0x3	AMUX3 PA_THERM
0x4	AMUX4 PM_HARDWARE_ID
0x5	MPP-AMUX5
0x6	MPP-AMUX6
0x7	MPP-AMUX7
0x8	MPP-AMUX8
0x9	ATEST<1>
0xA	ATEST<2>
0xB	ATEST<3>
0xC	ATEST<4>
0xD	ATEST<5>
0xE	ATEST<6>
0xF	ATEST<7>

PM8921 Amux Input Range

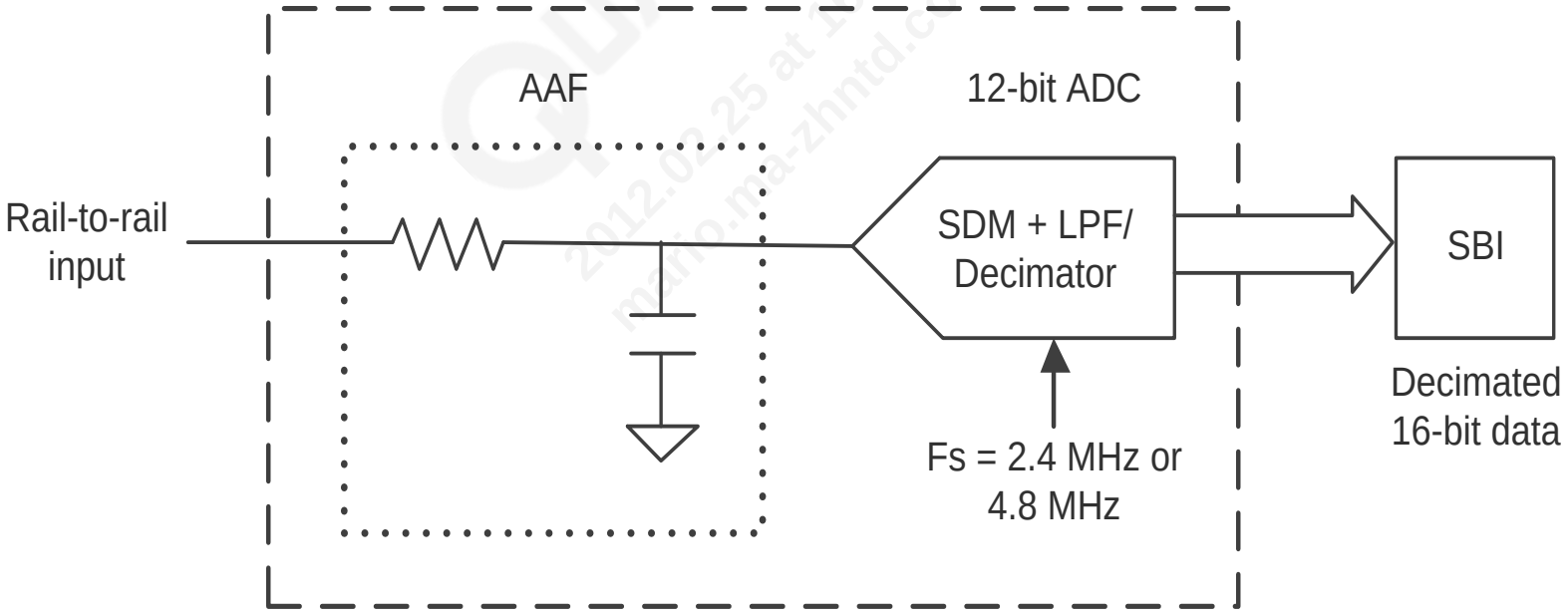
- Due to the saturation of the output buffer, Amux loses its accuracy drastically when its output goes below 0.05 V or above 1.75 V.
- Avoid using MPP and Amux to measure signals that cause the Amux output to go out of this region.
- When customers encounter ADC accuracy issues:
  - Check if the Amux input is within the range.
  - If not,
    - Move the input to another Amux channel that can tolerate the input voltage.
    - Use the voltage divider to lower the input voltage.

Amux Channel and Input Range

Channel #	Description	Typical input range (V)	Automatic scaling	Typical output range (V)
0	VCOIN pin	2.0 – 3.25	1/3	0.67 – 1.08
1	VBAT pin	2.5 – 4.5	1/3	0.83 – 1.5
2	DCIN pin (over-voltage protected)	4.5 – 9.5	1/6	0.75 – 1.58
3	Change-current monitor	0.05 – (VDDA-0.05)	1	0.05 – (VDDA-0.05)
4	VPH_PWR	2.5 – 4.5	1/3	0.83 – 1.5
5	IBAT – Battery charge current	0.3 – 1.5	1	0.3 – 1.5
6	Selected input from MPP, ATEST	0.05 – (VDDA-0.05)	1	0.05 – (VDDA-0.05)
7	Selected input from MPP, ATEST	0.15 – 3*(VDDA-0.05)	1/3	0.05 – (VDDA-0.05)
8	BAT_THERM	0.05 – (VDDA-0.05)	1	0.05 – (VDDA-0.05)
9	BAT_ID	0.05 – (VDDA-0.05)	1	0.05 – (VDDA-0.05)
10	USBIN pin (over-voltage protected)	4.35 – 6.5	1/4	1.09 – 1.63
11	Die-temperature monitor	0.4 – 0.9	1	0.4 – 0.9
12	0.625 V reference voltage	0.625	1	0.625
13	1.25 V reference voltage	1.25	1	1.25
14	CHG_Temp – Charger temperature	0.05 – (VDDA-0.05)	1	0.05 – (VDDA-0.05)
15	Module power off	—	—	—

PM8921 ADC Configuration

- PM8921 ADC can be configured based on different sampling rates and decimation ratios.
- Decimation ratio is a downsampling technique.
- Decimation ratio M means taking one sample for every M samples.



PM8921 ADC Configuration (cont.)

- XOADC conversion time is dependent on XOADC clock and decimation ratio
  - $F_{DATA\_UPDATE} = F_{CLK}/Decimation\_Ratio$
- Conversion time under different XOADC clock rate and decimation ratio

Decimation filter configuration – sinc1 and sinc2 are enabled			
XOADC clock rate	**Decimation ratio**	**Data update rate (kHz)**	**Conversion time (µs)**
2.4 MHz (TCXO/8)	512	4.69	213
	1024	2.34	427
	2048	1.17	853
	4096	0.59	1707
4.8 MHz (TCXO/4)	512	9.38	107
	1024	4.69	213
	2048	2.34	427
	4096	1.17	853
9.6 MHz (TCXO/2)	512	19.76	53
	1024	9.38	107
	2048	4.69	213
	4096	2.34	427

PM8921 ADC Configuration (cont.)

- The output of ADC conversion is 24 bits, but only 16 bits are meaningful.
- Two 8-bit registers XOADC_DATA0 and XOADC_DATA1 are used to store the ADC result.
- The table shows how XOADC performs data truncation under different decimation ratios.

Data truncation	Decimation ratio			
	512	1024	2048	4096
	Ignore last 4 LSBs	Ignore last 6 LSBs	Ignore last 8 LSBs	Ignore last 10 LSBs

PM8921 Amux Channel Contention

- Does QCT have a mechanism to avoid two processors mapping different MPP pins to the same Amux channel?
  - On PM8921, there is a second mux called the premux; MPP channels get routed to the premux channels 5 through 8; output of the premux goes to the input of amux_7 or amux_6.
  - Premux channel 5 is reserved for dynamic mapping of Amux → MPP channels by the modem processor.
  - Premux channel 6 is reserved for dynamic mapping of Amux → MPP channels by the apps processor.
  - Premux channel 7 is reserved for static mapping of Amux → MPP.
  - Premux channel 8 is reserved for static mapping of Amux → MPP.
  - MPPs that are dynamically mapped must be owned by either the apps or the modem processor and cannot be read by the nonowner.
  - The two statically mapped MPPs can be read by both the modem and apps processors. QCT has reserved these two static mappings for PA_THERM1 (MPP8 routed to PreMUX8) and a possible third PA thermistor routed to PreMUX7.

PM8921 Modem ADC APIs

Path from MPP to ADC

- Two types of Amux channels in PM8921 are:
  - Dedicated channels for VCOIN, VBAT, USBIN, etc.
  - Configurable channels for MPP pins
- MPP pins provide customers flexibility to measure signals in which they are interested.
- In PM8921, no MPP configuration is needed before performing the ADC reading.
  - The ADC driver automatically calls pm_mpp_config_analog_input to map the MPP pin to the appropriate Amux and then configures the MPP back to its default low power mode after the ADC read is complete.
- To measure MPP signals using ADC through the Amux channel:
  - Create the mapping of the Amux channel to the logical channel.
  - Call the API to request an ADC conversion.

PM8921 Amux Channel Configuration

- In XoADcBsp.c

```
static const XoAdcChannelConfigType xoAdcChannels[] = {
…
/* 10: CHG_TEMP          */ {14,PM_ADC_XOADC_INPUT_PMIC_IN_XOADC_GND,
PM_ADC_AMUX_MAIN,    PM_MPP_INVALID, PM_MPP__AIN__CH_INVALID,
XOADC_CONFIG_AMUX_INPUT, FALSE, XoAdcScalePmicTherm,
XOADC_CAL_METHOD_ABSOLUTE},
/* 11: PA_THERM0         */ {3, PM_ADC_XOADC_INPUT_PMIC_IN_XOADC_GND,
PM_ADC_PREMUX_TO_CH7,PM_MPP_INVALID, PM_MPP__AIN__CH_INVALID,
XOADC_CONFIG_AMUX_INPUT, FALSE, XoAdcScalePaTherm,
XOADC_CAL_METHOD_RATIOMETRIC},
/* 12: PM_HARDWARE_ID    */ {4, PM_ADC_XOADC_INPUT_PMIC_IN_XOADC_GND,
PM_ADC_PREMUX_TO_CH7,PM_MPP_INVALID, PM_MPP__AIN__CH_INVALID,
XOADC_CONFIG_AMUX_INPUT, FALSE, NULL, XOADC_CAL_METHOD_ABSOLUTE},
/* 13: MPP1              */ {5, PM_ADC_XOADC_INPUT_PMIC_IN_XOADC_GND,
PM_ADC_PREMUX_TO_CH7,PM_MPP_1,       PM_MPP__AIN__CH_AMUX5,
XOADC_CONFIG_AMUX_INPUT, FALSE, NULL, XOADC_CAL_METHOD_ABSOLUTE},
…
};
```

Mapping of Logical Channel to Physical Channel

- In Adc_props.xml, two steps show for the physical to logical mapping.
- Use the config tab representing the ADC device ID and channel ID.

```
<var_seq name="vcoin_config" type=DALPROP_DATA_TYPE_UINT32_SEQ>
       0, 0, end
      </var_seq>
<var_seq name="vbatt_config" type=DALPROP_DATA_TYPE_UINT32_SEQ>
       0, 1, end          // 0=XOADC, 1 =main mux channel 1
      </var_seq>
…
```

Mapping of Logical Channel to Physical Channel (cont.)

- Map the logical channel to the physical channel through the config tab.

```
/*** Define analog input names ***/
      <props name=ADC_INPUT_VCOIN // the logical channel name
  type=DALPROP_ATTR_TYPE_UINT32_SEQ_PTR>
         vcoin_config
      </props>
      <props name=ADC_INPUT_VBATT type=DALPROP_ATTR_TYPE_UINT32_SEQ_PTR>
         vbatt_config
      </props>
```

Mapping of Logical Channel to Physical Channel (cont.)

- ADC logical channel names are defined in AdcStdInputs.h.

```
** VBATT - nPhysical units are in millivolts          */
#define ADC_INPUT_VBATT              "VBATT“

/** VCHG - nPhysical units are in millivolts           */
#define ADC_INPUT_VCHG               "VCHG"

/** VCOIN - nPhysical units are in millivolts          */
#define ADC_INPUT_VCOIN              "VCOIN“
```

- Customers can define a new logical channel name using Adc_props.xml and AdcStdInputs.h.

Read ADC Value Using DAL-ADC Framework

- Get the ADC properties from adc_props.xml based on the DAL DeviceID.
  - DALSYS_PROPERTY_HANDLE_DECLARE(phAdcProp);
  - DALSYS_GetDALPropertyHandle((DALDEVICEID)DALDEVICEID_ADC, phAdcProp);
- Get the AdcInputProps given a logical channel name.
  - AdcInputPropertiesType AdcInputProps
  - DalAdc_GetInputProperties(phAdcProp, channel_name, &AdcInputProps);
- Read the ADC result.
  - AdcClientBlockingReadObjType *pObj
  - AdcRequestParametersType adcParams;
  - adcParams.nDeviceIdx = AdcInputProps.nDeviceIdx;
  - adcParams.nChannelIdx = AdcInputProps.nChannelIdx;
  - DalAdc_RequestConversion(pObj→phDev, &adcParams, NULL);
  - Result is stored in pObj→pAdcResult

ADC Conversion Result

```
typedef struct
{
   AdcResultStatusType eStatus;  /* status of the conversion */
   uint32 nToken;        /* token which identifies this conversion */
   uint32 nDeviceIdx;    /* the device index for this conversion */
   uint32 nChannelIdx;   /* the channel index for this conversion */
   int32 nPhysical;      /* result in physical units. Units depend on BSP */
   uint32 nPercent;      /* result as percentage of reference voltage used
                          * for conversion. 0 = 0%, 65535 = 100% */
   uint32 nMicrovolts;   /* result in microvolts */
   uint32 nReserved;     /* reserved for internal use */
} AdcResultType;
```

AMSS DAL-ADC Framework

- ADC_READ can be implemented in different patterns using DAL-ADC APIs.
  - Blocking read
  - Buffered read
  - Hybrid read
- For DAL-ADC framework API details, see [Q2].

PM8921 Linux ADC APIs

PM8921 Amux Channel in board-msm8960.c

- In pm8921-adc.h, logical channels are defined as:

```
enum pm8921_adc_channels {
CHANNEL_VCOIN = 0,
CHANNEL_VBAT,
CHANNEL_DCIN,
CHANNEL_ICHG,
CHANNEL_VPH_PWR,
CHANNEL_IBAT,
CHANNEL_MPP_1,              (AMUX6 for MPP Pins)
CHANNEL_MPP_2,              (AMUX7 for MPP Pins)
CHANNEL_BATT_THERM,
CHANNEL_BATT_ID,
CHANNEL_USBIN,
CHANNEL_DIE_TEMP,
CHANNEL_625MV,
CHANNEL_125V,
CHANNEL_CHG_TEMP,
CHANNEL_MUXOFF,
CHANNEL_NONE,
};
```

PM8921 Amux Configuration in board-msm8960.c

```
static struct pm8921_adc_amux pm8921_adc_channels_data[] = {
    {"vcoin", CHANNEL_VCOIN, CHAN_PATH_SCALING2, AMUX_RSV1,
        ADC_DECIMATION_TYPE2, ADC_SCALE_DEFAULT},
    {"vbat", CHANNEL_VBAT, CHAN_PATH_SCALING2, AMUX_RSV1,
        ADC_DECIMATION_TYPE2, ADC_SCALE_DEFAULT},
    {"dcin", CHANNEL_DCIN, CHAN_PATH_SCALING4, AMUX_RSV1,
        ADC_DECIMATION_TYPE2, ADC_SCALE_DEFAULT},
    {"ichg", CHANNEL_ICHG, CHAN_PATH_SCALING1, AMUX_RSV1,
        ADC_DECIMATION_TYPE2, ADC_SCALE_DEFAULT},
    {"vph_pwr", CHANNEL_VPH_PWR, CHAN_PATH_SCALING2, AMUX_RSV1,
        ADC_DECIMATION_TYPE2, ADC_SCALE_DEFAULT},
    {"ibat", CHANNEL_IBAT, CHAN_PATH_SCALING1, AMUX_RSV1,
        ADC_DECIMATION_TYPE2, ADC_SCALE_DEFAULT},
    {"batt_therm", CHANNEL_BATT_THERM, CHAN_PATH_SCALING1, AMUX_RSV2,
        ADC_DECIMATION_TYPE2, ADC_SCALE_BATT_THERM},
    {"batt_id", CHANNEL_BATT_ID, CHAN_PATH_SCALING1, AMUX_RSV1,
        ADC_DECIMATION_TYPE2, ADC_SCALE_DEFAULT},
    {"usbin", CHANNEL_USBIN, CHAN_PATH_SCALING3, AMUX_RSV1,
        ADC_DECIMATION_TYPE2, ADC_SCALE_DEFAULT},
      ….
};
```

PM8921 ADC Read

- Location – kernel/drivers/mfd/pm8921-adc.c
- Header file – kernel/include/linux/mfd/pm8921-adc.h
- API reads the ADC value of dedicated pins
  - pm8921_adc_read(enum pm8921_adc_channels channel, struct pm8921_adc_chan_result *result)
  - For example, the customer needs to know the “end of charging” based on the charging current.
    - pm8921_adc_read(CHANNEL_ICHG, &result)
- API reads the ADC value of an MPP pin
  - pm8921_adc_mpp_config_read(uint32_t mpp_num, enum pm8921_adc_channels channel, struct pm8921_adc_chan_result *result);
  - For example, the customer connects the main_therm pin to mpp_7 pin and does not know its voltage value from the apps side (main_mux 6).
    - pm8921_adc_mpp_config_read(PM8921_AMUX_MPP_7, ADC_MPP_1_AMUX6, &result)

PM8921 ADC Conversion Result

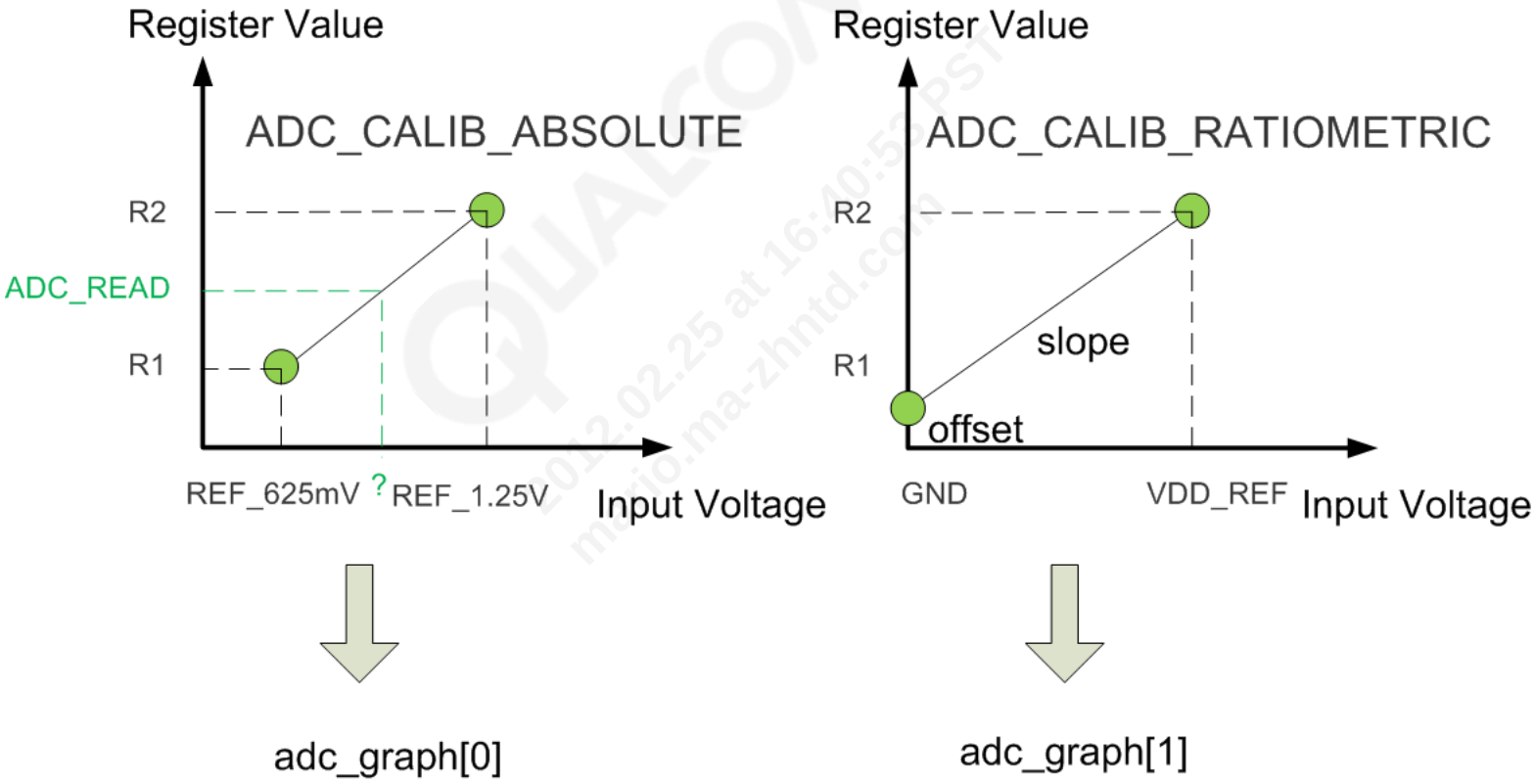
- `struct pm8921_adc_chan_result {`
  `uint32_t chan;`
  `int32_t  adc_code;`
  `int64_t  measurement;`
  `int64_t  physical;`
  `};`
  - chan – Channel number of the requested conversion
  - adc_code – Precalibrated digital output of a given ADC relative to the ADC reference
  - measurement – In units specific for a given ADC; most ADCs use reference voltage but some ADCs use reference current; this measurement is a number relative to a reference of a given ADC
  - physical – Data meaningful for each individual channel whether it is voltage, current, temperature, etc.

PM8921 ADC Calibration

- PM8921 has two ADC calibration types
  - ADC_CALIB_ABSOLUTE
  - ADC_CALIB_RATIOMETRIC
- ADC_CALIB_ABSOLUTE – Use 625 mV and 1.25 V reference channels
- ADC_CALIB_RATIOMETRIC – Use reference voltage/GND
- pm8921_adc_calib_device()
  - API for ADC calibration
  - Called at once when the ADC driver is initiated
  - ADC_CALIB_ABSOLUTE and ADC_CALIB_RATIOMETRIC are performed
  - Both calibration outcomes are stored in pm8921_adc_linear_graph

PM8921 Calibration Algorithm

- Given two points, the offset and slope of the line can be obtained.



PM8921 Channel Calibration Types

Signal	AMUX channel	Scaling	ADC speed	Calibration type	Calibration data	Conversion trigger
VCOIN	0x00	x1/3	Normal	Absolute	Historical	SBI
VBAT	0x10	x1/3	Normal	Absolute	Historical	SBI
			Fast			Synchronous, XOADC conversion sequence
			Fast			Synchronous, XOADC conversion sequence
DCIN	0x20	x1/6	Normal	Absolute	Historical	SBI
ICHG	0x30	x1	Normal	Absolute	Historical	SBI
VPH_PWR	0x40	x1/3	Normal	Absolute	Historical	SBI
IBAT	0x50	x1	Normal	Absolute	Historical	SBI
MPP/ATEST	0xM4	x1	TBD	TBD	TBD	TBD
MPP/ATEST	0xM8	x1/3	TBD	TBD	TBD	TBD
BAT_THERM	0x80	x1	Normal	Ratiometric	Historical	Periodic, arbiter
BAT_ID	0x90	x1	Normal	Ratiometric	Historical	—
USBIN	0xA0	x1/4	Normal	Absolute	Historical	SBI
DIE_TEMP	0xB0	x1	Normal	Absolute	Historical	SBI
0.625 V V_{REF}/2	0xC0	x1	Normal	N/A	N/A	N/A
1.25 V V_{REF}	0xD0	x1	Normal	N/A	N/A	N/A
CHG_TEMP	0xE0	x1	Normal	Absolute	Historical	SBI
XO_THERM	N/A	N/A	Slow	Ratiometric	Fresh	SBI

PM8921 Linux ADC Debug Interface

- Linux ADC driver support debugfs for debugging using adb shell command
  - mount -t debugfs none /sys/kernel/debug
  - cd /sys/kernel/debug
  - cd pm8921_adc
- Example
  - Read battery voltage – cat vbat
  - Read die temperature – cat die_temp
  - Read battery temperature – cat batt_therm
  - Read battery ID – cat batt_id
  - Read battery current – cat ibat
- ADC debugfs is read-only

References

Ref.	Document	
Qualcomm		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Application Note: Configuration of MPP to ADC Path*	80-N1153-1

Questions?

https://support.cdmatech.com