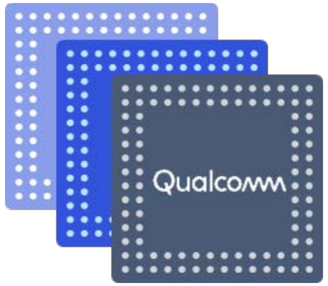
Qualcomm Technologies, Inc.

SDM670/SDM710/SDM712 Non-HLOS PMIC Software Overview

80-PD126-24 Rev. C



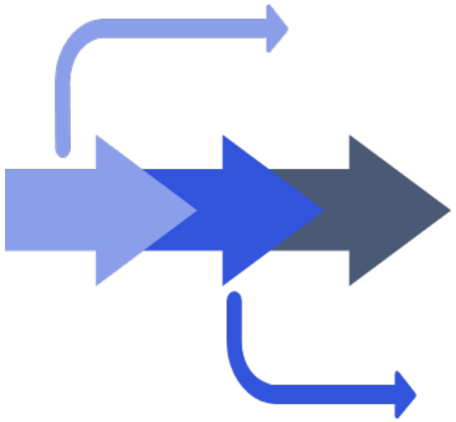
Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	August 2017	Initial release
B	April 2018	Updated title and other minor SDM710 changes
C	October 2018	Updated for SDM712

Contents

- Introduction
- Boot Architecture
- PMIC Software Drivers
- Output Power Management
- VREG
- Clocks
- ADC
- Reset Architecture
- SIM Card Removal Support – Battery UICC Alarm
- Coin cell
- GPIO
- Communication Interface – SPMI
- Debug Tools
- References
- Questions?

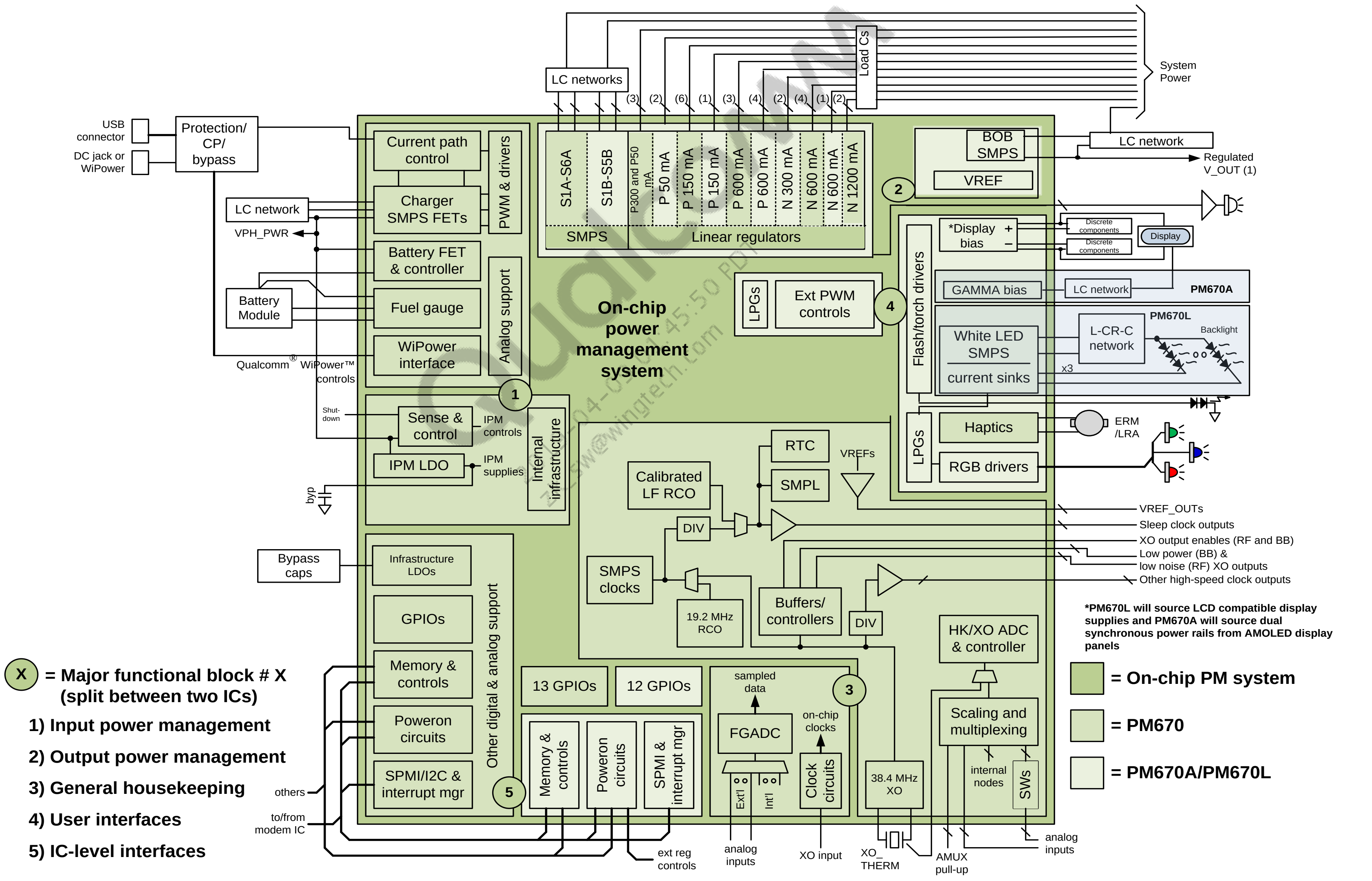


Introduction

Scope

- This document provides an overview of the software drivers for PM670 and PM670A/PM670L across all non-HLOS images
- See *SDM670/SDM710/SDM712 Linux Android PMIC Software Overview* (80-PD126-4) for similar HLOS content

PM670 Block Diagram



*PM670L will source LCD compatible display supplies and PM670A will source dual synchronous power rails from AMOLED display panels

X = Major functional block # X (split between two ICs)

1) Input power management

2) Output power management

3) General housekeeping

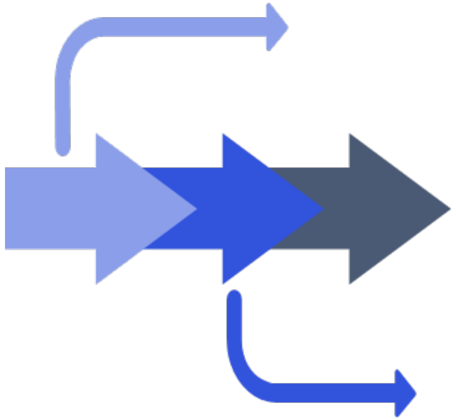
4) User interfaces

5) IC-level interfaces

🟩 = On-chip PM system

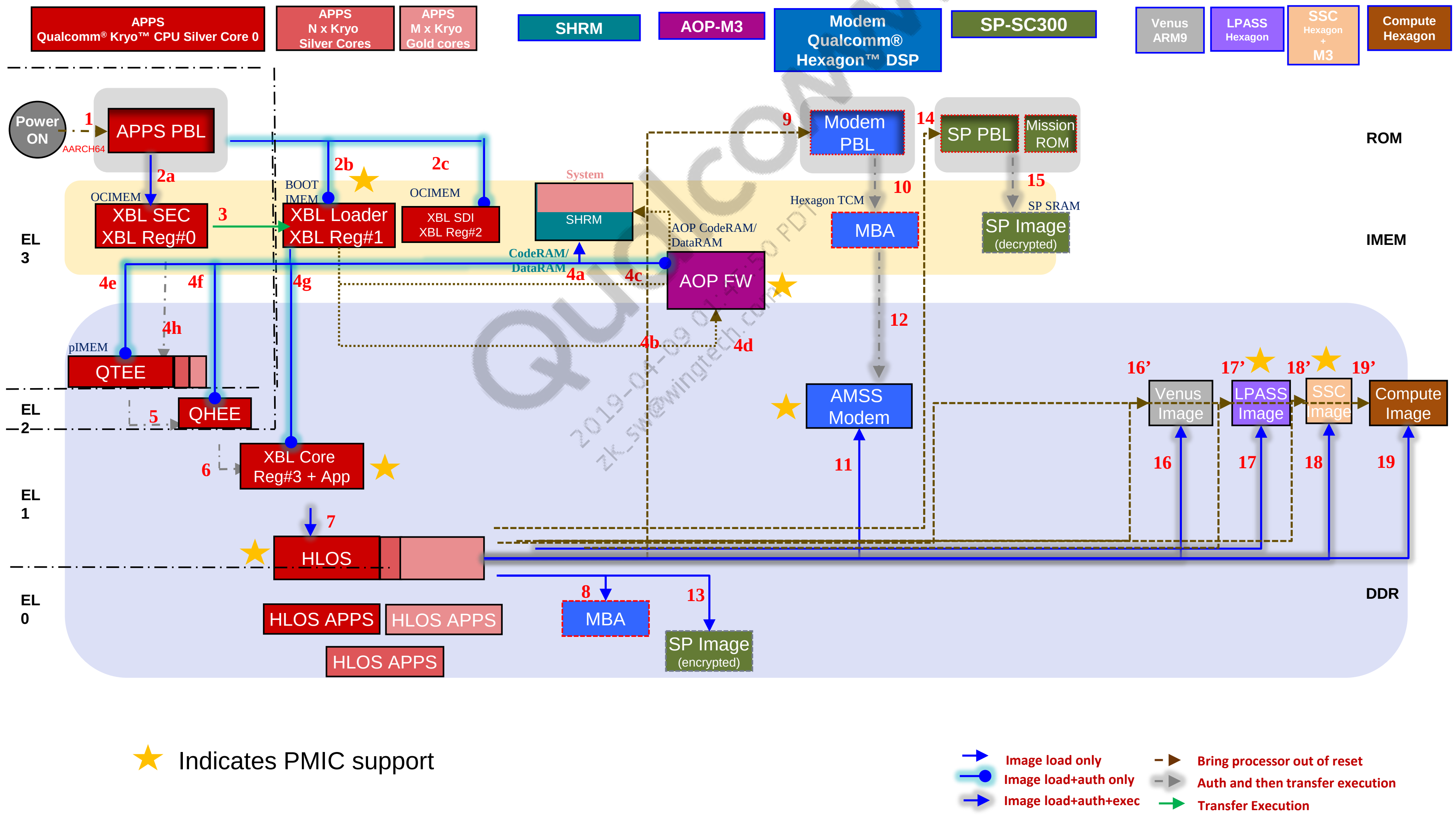
🟩 = PM670

🟩 = PM670A/PM670L



Boot Architecture

PMIC Software Drivers Support



★ Indicates PMIC support

PMIC Software Drivers

PMIC Software Drivers – XBL Loader

- XBL loader (SBL) PMIC driver primary roles:
  - PMIC hardware register and PBS RAM configuration
    - PMIC.elf filename has been changed to XBLConfig.elf
  - Dead battery recovery
- SBL also offers API support for customizing most PMIC resources
- API
  - boot_images\QcomPkg\Include\api\pmic
  - boot_images\QcomPkg\Include\DDIAdc.h
- Settings
  - boot_images\QcomPkg\SDM670Pkg\Settings\ADC\core\VAdcSettings.c
  - boot_images\QcomPkg\SDM670Pkg\Settings\PMIC\pm_config_target.c
- SPMI permissions
  - SPMI permission configuration for PMIC registers has moved from TZ to boot build
  - boot_images\QcomPkg\SDM670Pkg\Settings\PMIC\loader\pm_spmi_config.c
- Customization
  - All OEM customization must be defined within or after the pm_driver_post_init() function definition
  - boot_images\QcomPkg\Library\PmicLib\target\sdm670_pm670_pm670l\system\src\pm_sbl_boot_oem.c

PMIC Software Drivers – XBL Core (UEFI)

- XBL core driver main roles:
  - Weak battery charging
  - Battery profile loading (during FG initialization)
  - Display initialization support
- API
  - boot_images\QcomPkg\Include\Protocol\EFIPmicxxx.h
  - boot_images\QcomPkg\Include\DDIAdc.h
- Settings and Configuration
  - boot_images\QcomPkg\SDM670Pkg\Settings\PMIC\core\pm_config_pam.c
  - boot_images\QcomPkg\Drivers\QcomChargerDxe
  - boot_images\QcomPkg\SDM670Pkg\Settings\ADC\core\VAdcSettings.c
- Customization
  - All OEM customization must be defined within or after the pm_post_pmic_initialization() function definition
    - boot_images\QcomPkg\SDM670Pkg\Library\PmicLib\core\la\pm_core.c

Note: See *UEFI PMIC Software User Guide* (80-P2484-42).

PMIC Software Drivers – AOP

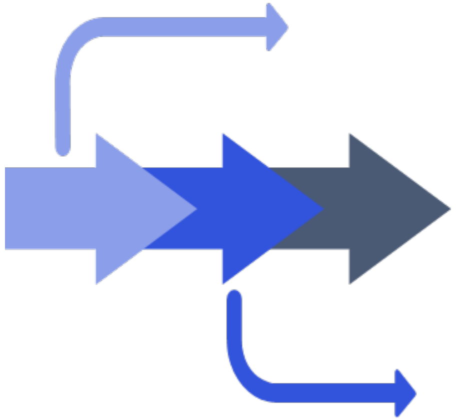
- Always on processor (AOP) driver handles the following subset of tasks performed by the RPM driver on previous platforms:
  - Parent-child regulator dependencies
  - Workarounds
  - Complex aggregation
- API
  - aop_proc\core\api\pmic\pm
- For more information, see *RPM Hardening Overview and Debug* (80-P9301-16)

PMIC Software Drivers – MPSS

- Main purpose of the MPSS driver is to request PMIC regulator or clock resources used by modem software clients
- MPSS driver also provides APIs required for customization
- API
  - modem_proc\core\api\pmic
  - modem_proc\core\api\hwengines\adc.h
- Drivers
  - modem_proc\core\settings\pmic\pm\config\sdm670\pm_config_pam.c
  - modem_proc\core\settings\hwengines\adc\config\670\VAdcSettings.c

PMIC Software Drivers – aDSP, CDSP, SLPI

- Main purpose of these PMIC drivers is to request PMIC regulator or clock resources used by aDSP or CDSP and SLPI clients
- There is no SPMI connection to the PMIC from these subsystems – Support is limited to VREG or CLK resource voting through the RPMh interface
- API
  - adsp_proc\core\api\pmic\pm
  - slpi_proc\core\api\pmic\pm
- Settings
  - adsp_proc\core\settings\pmic\pm\config
  - slpi_proc\core\settings\pmic\pm\config



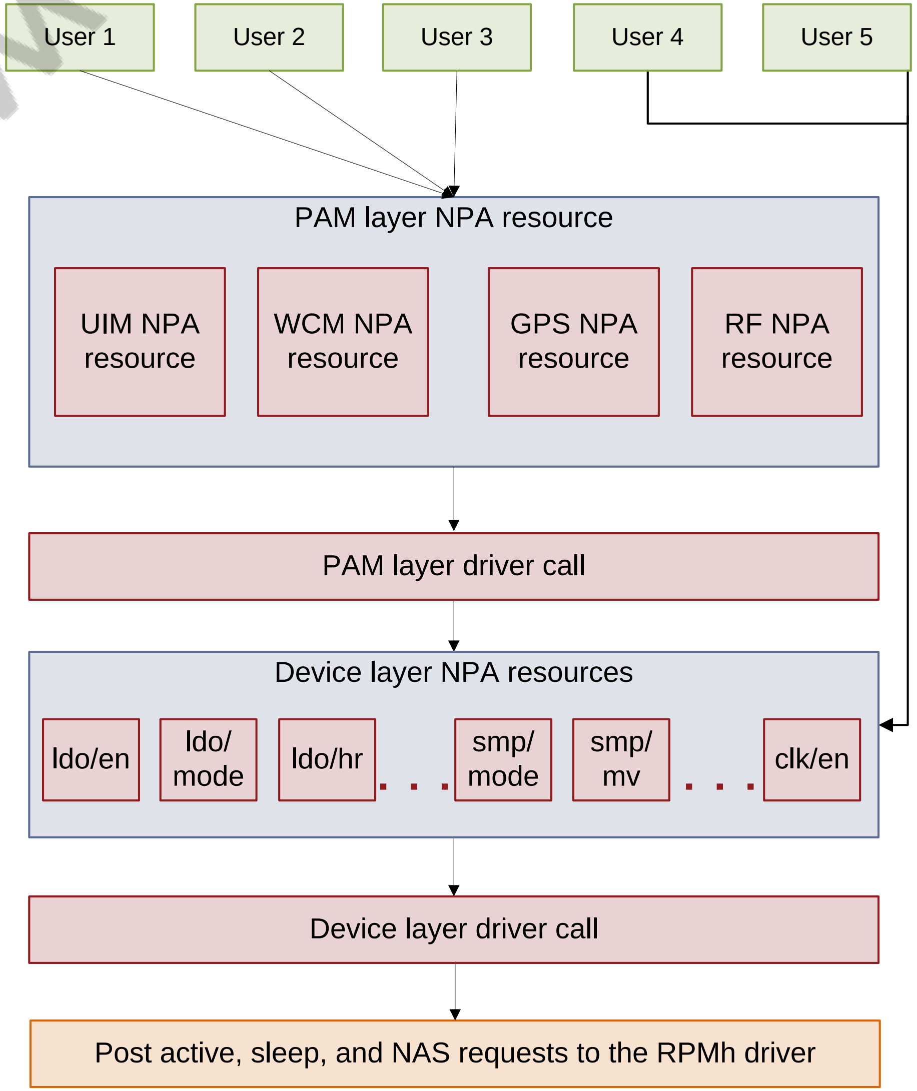
Output Power Management

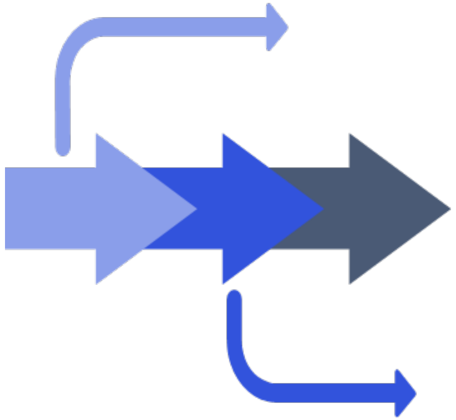
Output Power Management – Software Support

Image	Support
XBL loader	Basic APIs for initialization
UEFI	PMIC resource manager (PRM)
AOP	Parent-child dependencies, complex aggregation
MPSS	PRM
SLPI	PRM
aDSP	PRM
HLOS	RPMh regulator framework

PRM Framework

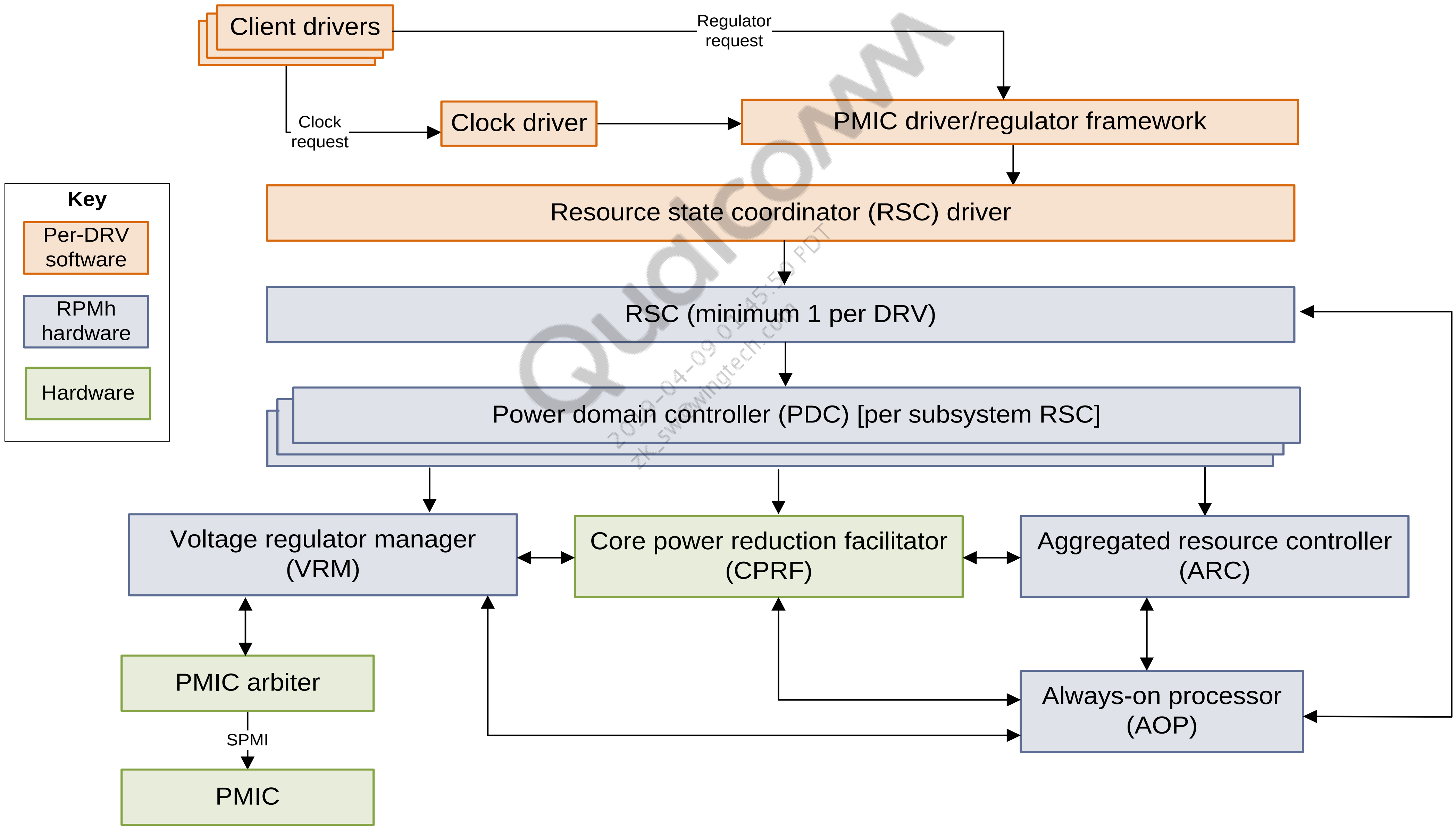
- PRM replaces the PMIC node power architecture (NPA) framework in previous chipsets
- No more key value pairs – Only resource states (values) in PMIC arbitration matrix (PAM) data
- Supports en, mode, volt, and headroom for VREG
- Only supports en for CLK (XOB and XO)
- PMIC does not handle aggregation, NPA does
- Query support for client drivers to ensure that the request was honored





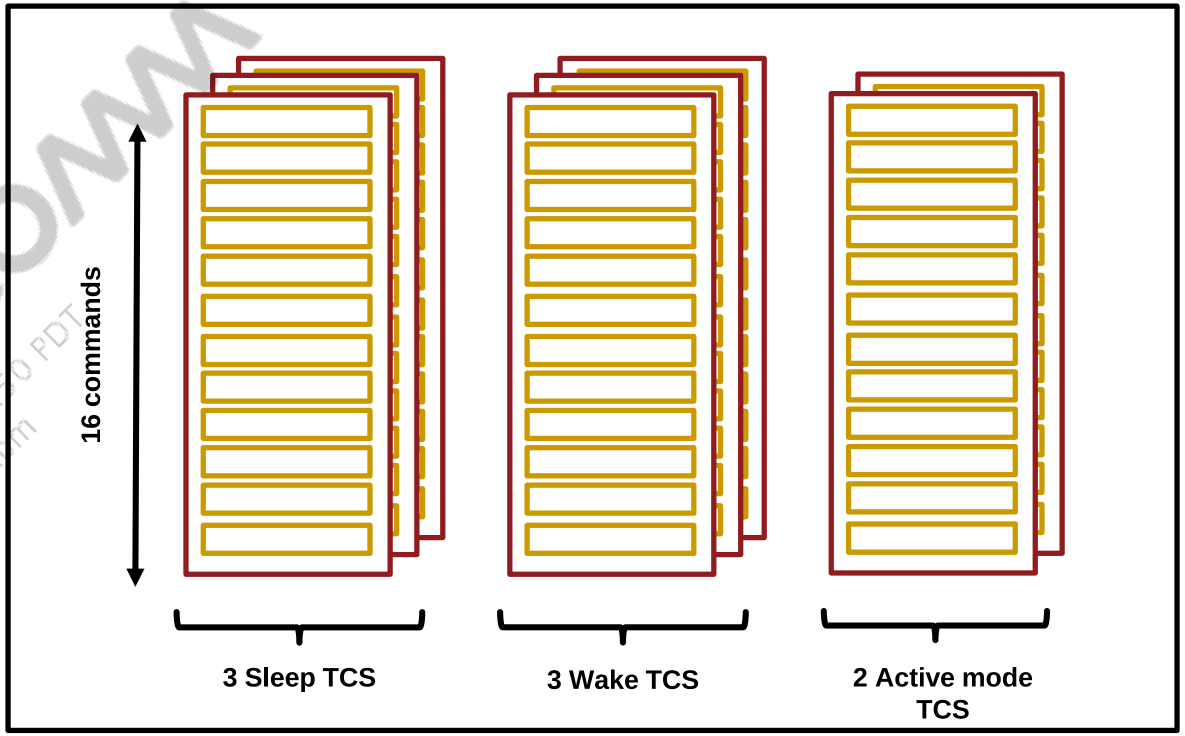
VREG

VREG Control Block Diagram


Client drivers
Regulator request
Clock request
Clock driver
PMIC driver/regulator framework
Key
Per-DRV software
RPMh hardware
Hardware
Resource state coordinator (RSC) driver
RSC (minimum 1 per DRV)
Power domain controller (PDC) [per subsystem RSC]
Voltage regulator manager (VRM)
Core power reduction facilitator (CPRF)
Aggregated resource controller (ARC)
PMIC arbiter
SPMI
PMIC
Always-on processor (AOP)

VREG Control Flow – RSC

- Each subsystem direct resource voter (DRV) has a resource state coordinator (RSC)
- Each RSC has several trigger command set (TCS) blocks to relay subsystem requests to RPMh
- A TCS may be hardwired to trigger for subsystem sleep entry or exit
- All TCS blocks can be configured to send requests immediately in active mode configuration (AMC)

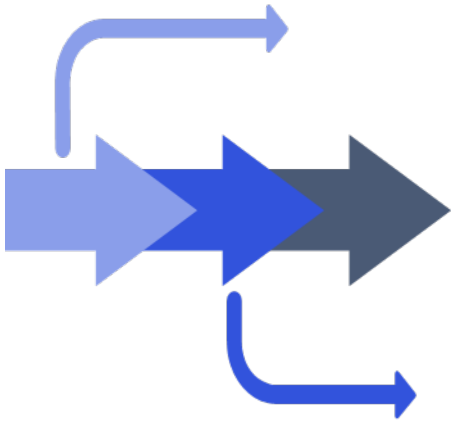


VREG Control Flow – RPMh

- Voltage regulator manager (VRM)
  - Manages PMIC regulators
  - Supports voting on the following regulator parameters:
    - Enable – 0 = off, 1 = on
    - Voltage – Output voltage in microvolts
    - Mode – Raw PMIC regulator mode control register value
    - Headroom voltage – Parent-child minimum voltage delta in microvolts
  - All parameters are aggregated by applying a MAX function
- Aggregated resource controller (ARC)
  - Handles aggregation of CPR-managed PMIC regulators
  - Supports voting on the following voltage-level regulator parameters:
    - Voltage level values range from 0 to 15 and are mapped to meaningful levels like off, retention, SVS, Nominal, and Turbo via command DB
    - For collapsible domains, voltage level 0 corresponds to off
  - The single parameter is aggregated by applying a MAX function

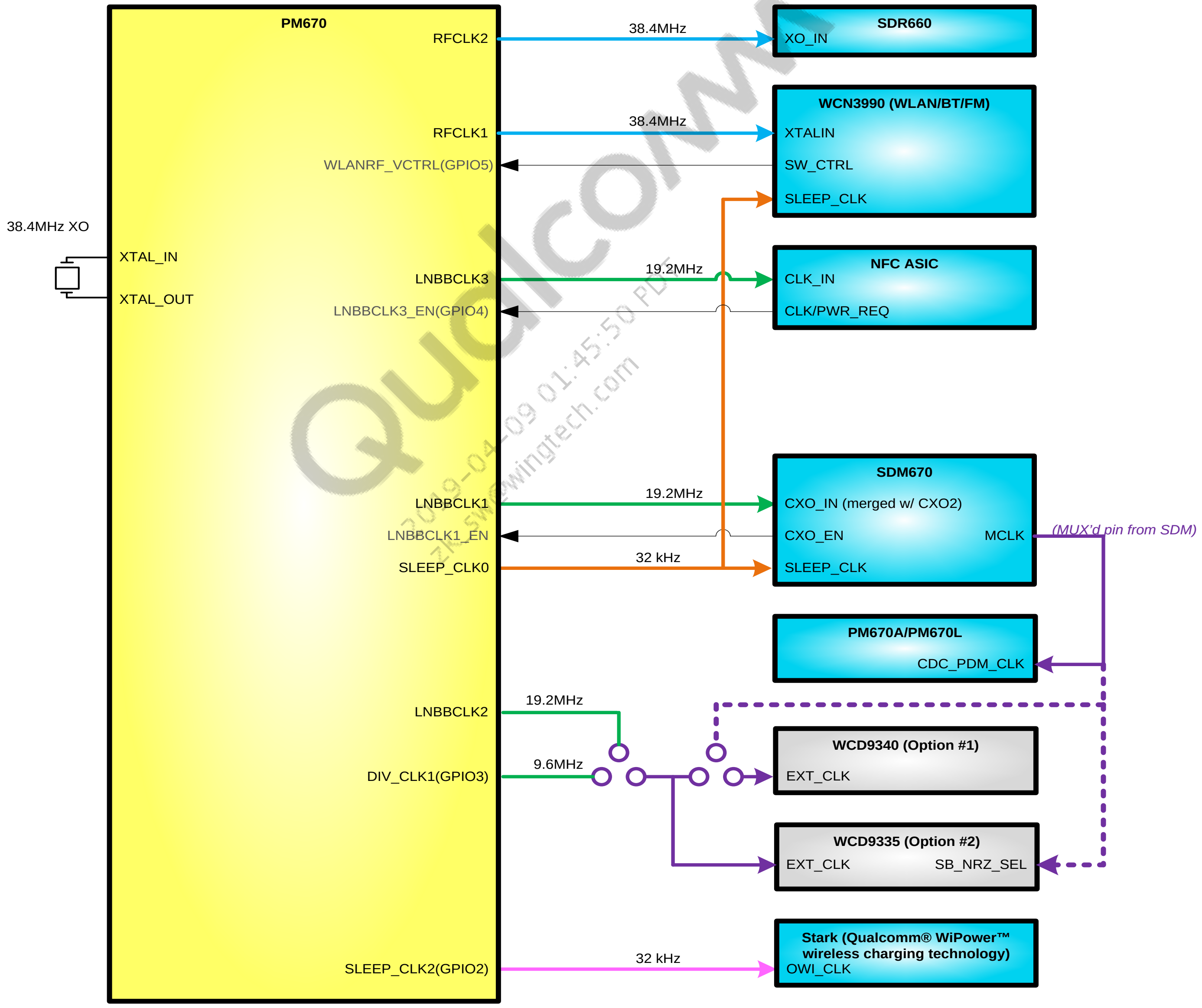
VREG Control Flow – Command DB

- Software must know how to map a given physical PMIC resource to an RPMh slave address
- For ARC resources, software must know how to map from a given voltage level (SVS, Nominal, Turbo, and so on) to an integer ARC operating level
- Command DB contains entries that satisfy both of these requirements
- Command DB is stored SMEM memory
- For more information, see *Non-HLOS PMIC Voltage Regulator and Clock Software User Guide* (80-P9301-78)



Clocks

Clocks – Hardware Block Diagram

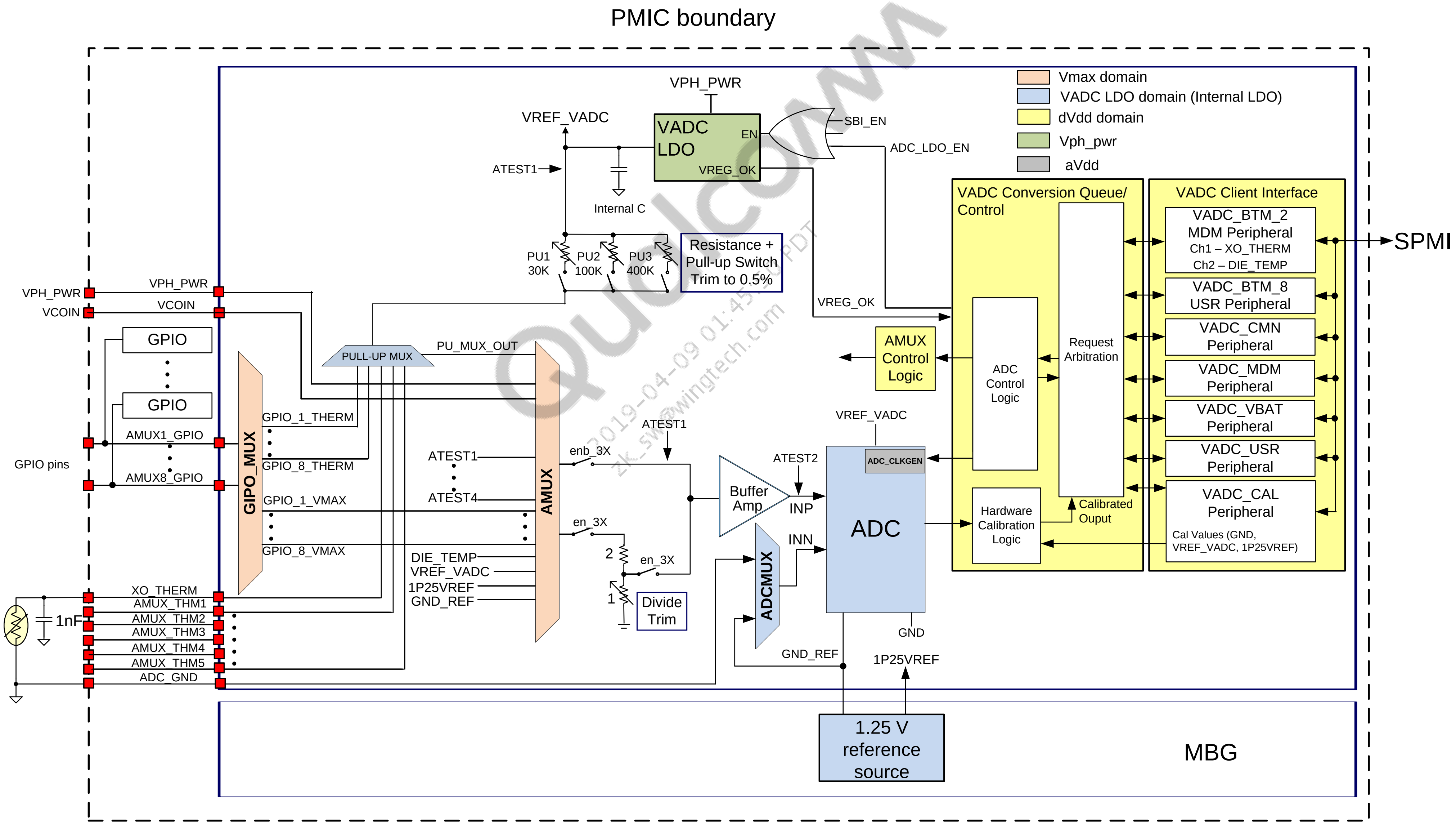


Clocks – Software Support

- Software block diagram – Voting mechanism works same as regulator voting mechanism through PRM
- For more information, see *Non-HLOS PMIC Voltage Regulator and Clock Software User Guide* (80-P9301-78)

ADC

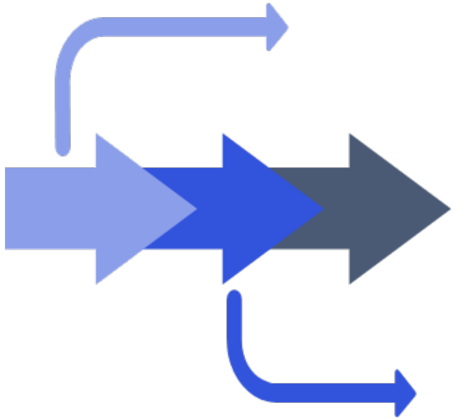
ADC Hardware



ADC — Software Support

ADC peripheral	Use	Subsystem owner
VADC_HC1_USR	Execute polled measurements on any channel	APSS
VADC_HC2_MDM	Execute polled measurements on any channel	MPSS
VADC_HC7_BTM8	Execute recurring measurements to provide an interrupt when defined thresholds are crossed	APSS
VADC_HC5_VBAT	Execute recurring measurements on Vbatt or Vphpwr channel, using hardware settling delay between each measurement in an average to capture minimum levels	APSS
VADC_HC4_CAL	Measure REF_GND, 1P25V, VREF_VADC (1.875 V) to use with calibration hardware and provide hardware calibration for all VADC results	APSS
VADC_HC9_BTM_2	Execute recurring measurements to provide an interrupt when defined thresholds are crossed	MPSS
VADC_HC10_CMN	Configure universal ADC parameters	APSS

Note: For more information, see *Non-HLOS PMIC ADC Software User Guide* (80-P2484-71).



Reset Architecture

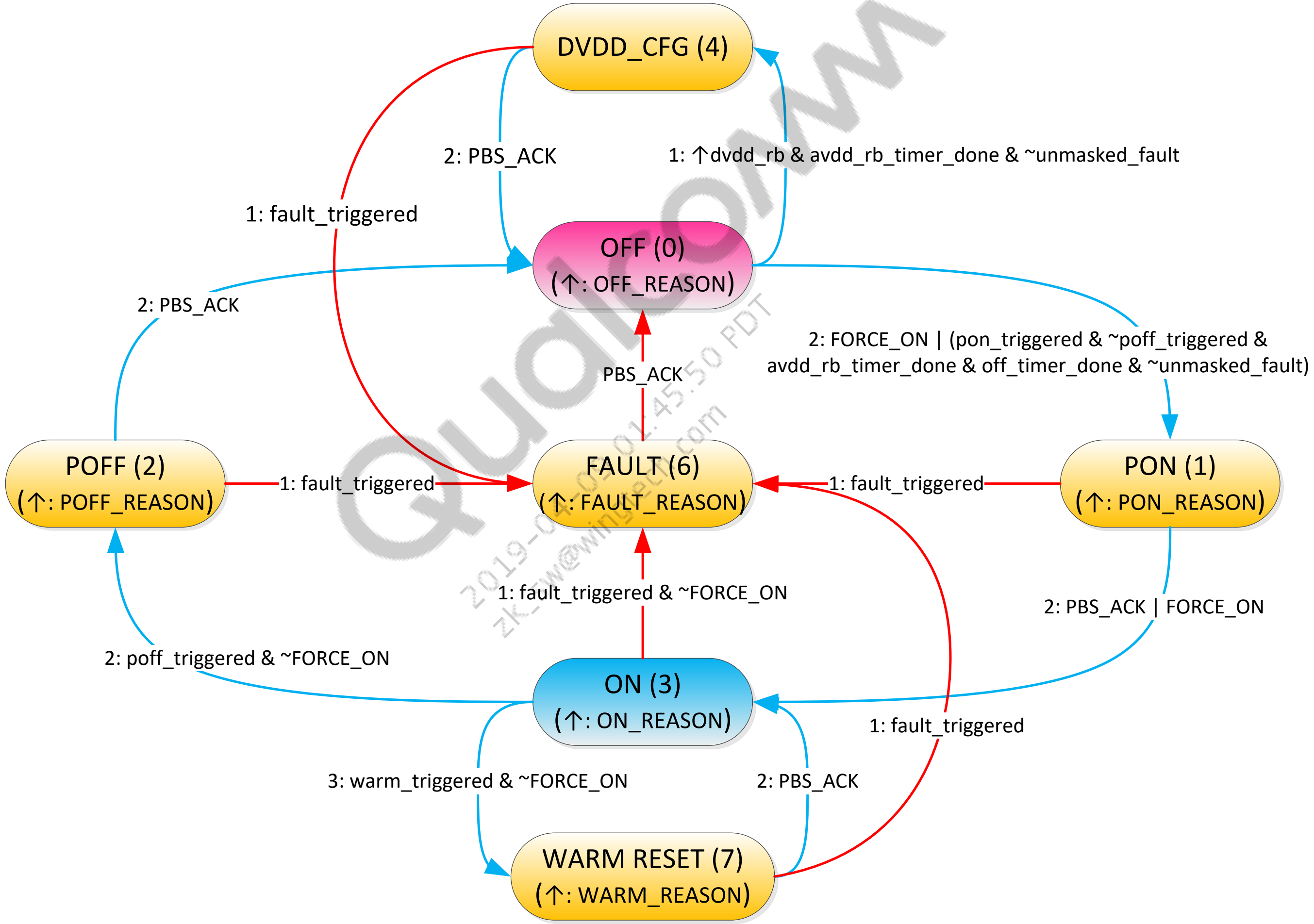
Reset Architecture – Overview

- PMIC PON controls system power on, off, and reset
- PMIC states
  - OFF
  - ON
  - Fault
  - Warm reset
  - DVDD CFG

- System reset triggers
  - Power key
  - GP1 (external reset trigger available for other applications)
  - SDM PS_HOLD
  - RESIN (Volume down)
  - Key combination (power key + volume down)
  - Over temperature
  - SMPL
  - UVLO
  - OVLO

Note: For more information, see *Linux Android PMIC Power ON Software User Guide* (80-P2484-40).

PON State Machine



Note:
1. The index at each arc indicates its priority. The smaller the index, the higher the priority.
2. There is hidden arc to OFF state. When dvdd_rb occurs, the PON FSM will be reset to OFF state.

🟪 = OFF   🟦 = ON   🟨 = PBS trigger

Reset Architecture – Software Support

System-level reset trigger	Software API	Configuration	PM (S1, S2, type)	PMI (S1, S2, type)	Software support
SDM PS_HOLD	Warm reset	PS HOLD	WARM_RST	NO_CFG_NEEDED	Yes
		GP1	NO_CFG_NEEDED	NO_CFG_NEEDED	
	Hard reset	PS HOLD	HARD_RST	NO_CFG_NEEDED	Yes
		GP1	NO_CFG_NEEDED	WARM_RST_AND_SHDN	
	Shutdown	PS HOLD	SHDN	NO_CFG_NEEDED	Yes
		GP1	NO_CFG_NEEDED	WARM_RST_AND_SHDN	
	DVDD hard reset	PS HOLD	DVDD_HARD_RST	NO_CFG_NEEDED	No
		GP1	NO_CFG_NEEDED	NO_CFG_NEEDED	
	DVDD shutdown	PS HOLD	DVDD_SHDN	NO_CFG_NEEDED	No
		GP1	NO_CFG_NEEDED	NO_CFG_NEEDED	

Reset Architecture – Software Support (cont.)

System-level reset trigger	Software API	Configuration	PM (S1, S2, type)	PMI (S1, S2, type)	Software support
Power key, resin or key combo (power key + volume down)	Warm reset	KPDPWR_S2	WARM_RST	NO_CFG_NEEDED	Yes
	Hard reset	KPDPWR_S2	HARD_RST	NO_CFG_NEEDED	No
	Shutdown	KPDPWR_S2	SHDN	NO_CFG_NEEDED	No
	DVDD hard reset	KPDPWR_S2	DVDD_HARD_RST	NO_CFG_NEEDED	Yes
	DVDD shutdown	KPDPWR_S2	DVDD_SHDN	NO_CFG_NEEDED	Yes

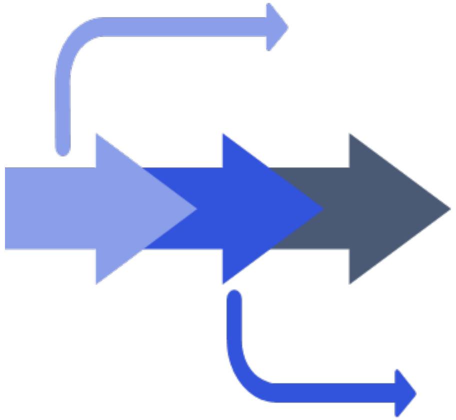
Reset Architecture – Software Support (cont.)

System-level reset trigger	Software API	Configuration	PM (S1, S2, type)	PMI (S1, S2, type)	Software support
Fail-safe (S3) reset	KPDPWR_N	–	Timer = X	Timer = X (PMI can only trigger S3 when BATFET is open Enters Fault state when Fault_N is pulled low)	Yes
	RESIN_N	–	Timer = X	Timer = X	Yes
	KPDPWR_ AND_RESIN	–	Timer = X	Timer = X	Yes
	KPDPWR_ OR_RESIN	–	Timer = X	Timer = X	Yes

Reset Architecture – Software Support (cont.)

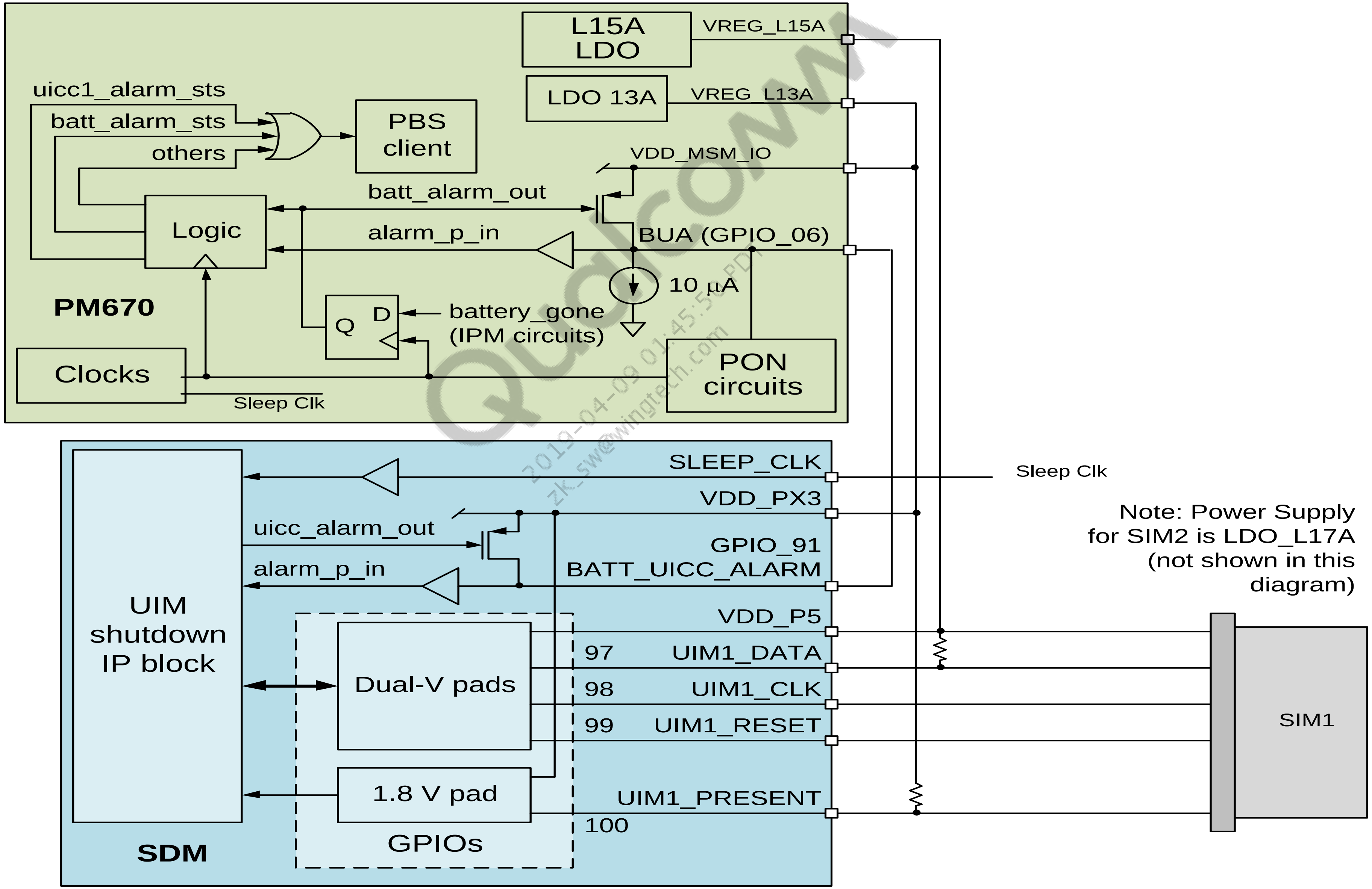
System-level reset trigger	Software API	Configuration	PM (S1, S2, type)	PMI (S1, S2, type)	Software support
Configuration at boot[1]	–	–	PS_HOLD = dVdd_HR KPD S2 = dVdd HR S2 resets configured as below	GP1 S2 = WR then dVdd_SD All other S2 reset disabled.	Yes
Configured at boot[2]	SDM WD configuration API	–	PS_HOLD = WR	GP1 S2 = WR then dVdd_SD All other S2 resets disabled.	Yes
PM overtemperature reset (internal)	No API	–	PM enters Fault state and forces all slave PMICs to enter fault state by pulsing FAULT_N	Enters Fault state when FAULT_N is pulsed	No
PMI overtemperature reset (internal)	No API	–	Enters Fault state when FAULT_N is pulsed	PMI enters Fault state and forces all other PMICs to enter fault state by pulsing FAULT_N	No
PMI AFP reset (internal)	No API	–	–	–	No

1 – Configured by SBL settings for production
2 – Configured by software for the SDM WD.device_post_init() function



SIM Card Removal Support – Battery UICC Alarm

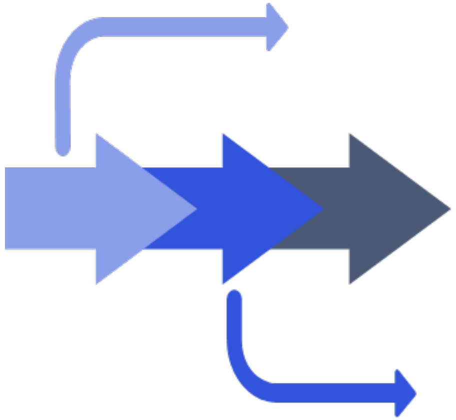
BUA – Hardware Block Diagram



Note: Power Supply
for SIM2 is LDO_L17A
(not shown in this
diagram)

BUA – Overview

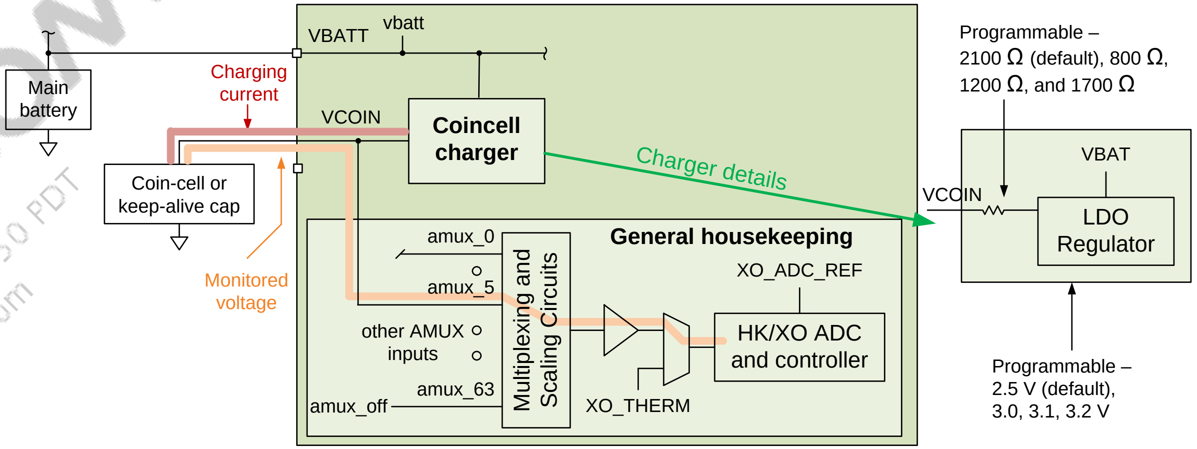
- PMIC supports UIM with the following modules:
  - UICC LDOs – UIM voltage is supplied from the PMIC
  - BUA – Battery UICC alarm is a bidirectional interface between the PMIC and the SDM that allows for a controlled power-down of UICC when either the battery or the UICC is removed
  - PBS – Turns OFF LDO
- Following software drivers are supported for UIM power regulation:
  - UIM drivers – High-level driver
  - PMIC drivers – Low-level drivers
    - UICC application
    - BUA
    - PBS
    - LDO
  - PMIC NPA driver – PMIC NPA framework for hardware resource requests

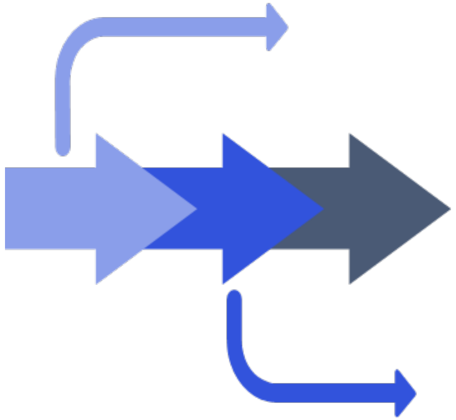


Coin cell

Coin cell Overview

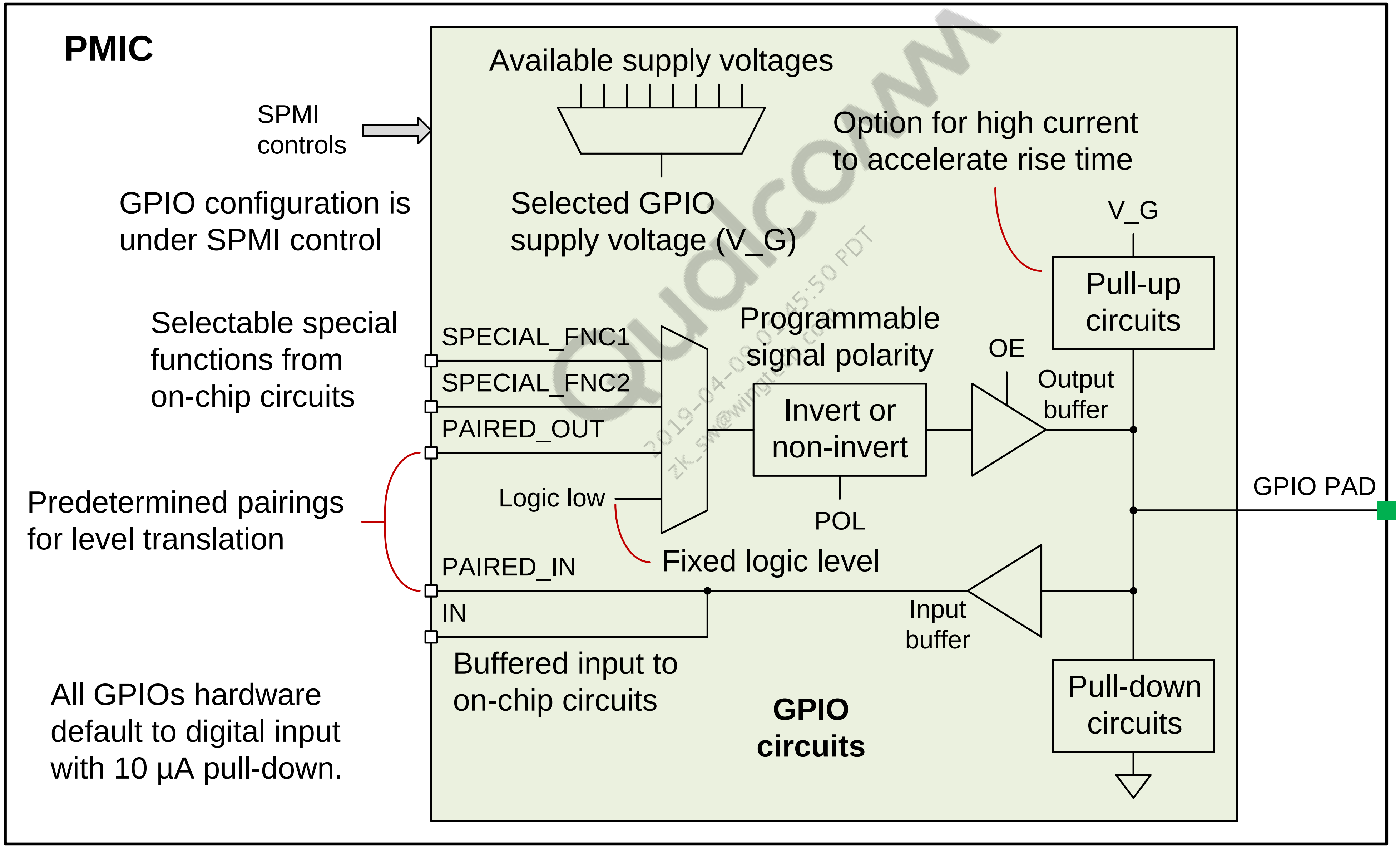
- Coin cell module supports the following features on SDM670/SDM710/SDM712:
  - RTC
  - SMPL
  - Coin cell backed PMIC register settings
  - Coin cell charging
- XBL
  - boot_images\QcomPkg\Include\api\pmic\pm_coincell.h
  - boot_images\QcomPkg\Library\PmicLib\drivers\coincell\src





GPIO

GPIO – Hardware Block Diagram



GPIO configuration is under SPMI control

Selectable special functions from on-chip circuits

Predetermined pairings for level translation

All GPIOs hardware default to digital input with 10 µA pull-down.

Buffered input to on-chip circuits

Externally accessible GPIO pad

GPIO Software

- Number of GPIOs available to each PMIC

PMIC	GPIOs
PM670	13
PM670A/PM670L	12

- Files and locations

File type	Location
Header	boot_images\QcomPkg\Include\api\pmic\pm\pm_gpio.h
Source	boot_images\QcomPkg\Library\PmicLib\drivers\gpio\src\pm_gpio.c boot_images\QcomPkg\Library\PmicLib\drivers\gpio\src\pm_gpio_driver.c

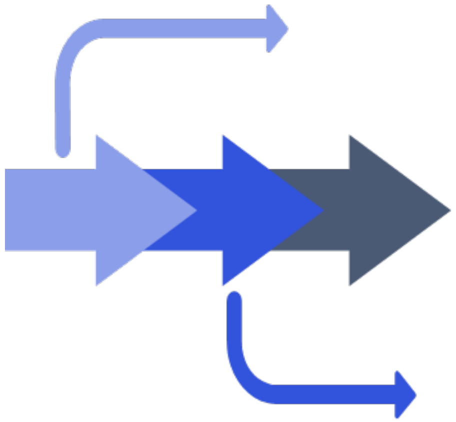
- For more information, see *Linux Android PMIC GPIO Software User Guide* (80-P9301-88)

GPIO – Software Support PM670

GPIO	Special function	Type	AMUX?	SDM	Comments
1	–	MV	–	Option 1	Reserved for PM670 PON options
2	SLEEP_CLK2	LV	Y	Stark - OWI_CLK (32 kHz)	Sleep clock for Stark OWI
3	DIV_CLK1	LV	Y	External codec clock 9.6 MHz	Can be reused when not used for an external codec
4	–	MV	Y	NFC - CLK_EN	NFC clock request
5	–	LV	Y	WCN3990 – WLANRF_VCTRL	RFCLK1_EN for WLAN Pin control for LDO6A, LDO9A, and LDO19A
6	SLEEP_CLK2	LV	–	PMK SLEEP_CLK_IN	Can be reused if PMK is not needed
7	–	LV	Y	BUA	Battery UIMM alarm
8	–	MV	–	SLB	Used for PON handshaking between PM670 and PM670A/PM670L
9	–	MV	Y	TYPEC_UUSB_SEL Hi-Z/Pull high = Type-C Pull low = Micro USB	Read before boot so charger can be configured for µUSB if needed
10	–	LV	Y	SDM – WCSS_PWR_REQ	WLAN power request
11	–	LV	–	Home key	–
12	SLEEP_CLK2	LV	Y	WIPWR_DIV2_EN	Can be reused for other purposes if WiPower is not used
13	Qnovo FET discharge	LV	–	Qnovo FET discharge	Can be used for other purposes if Qnovo is not used

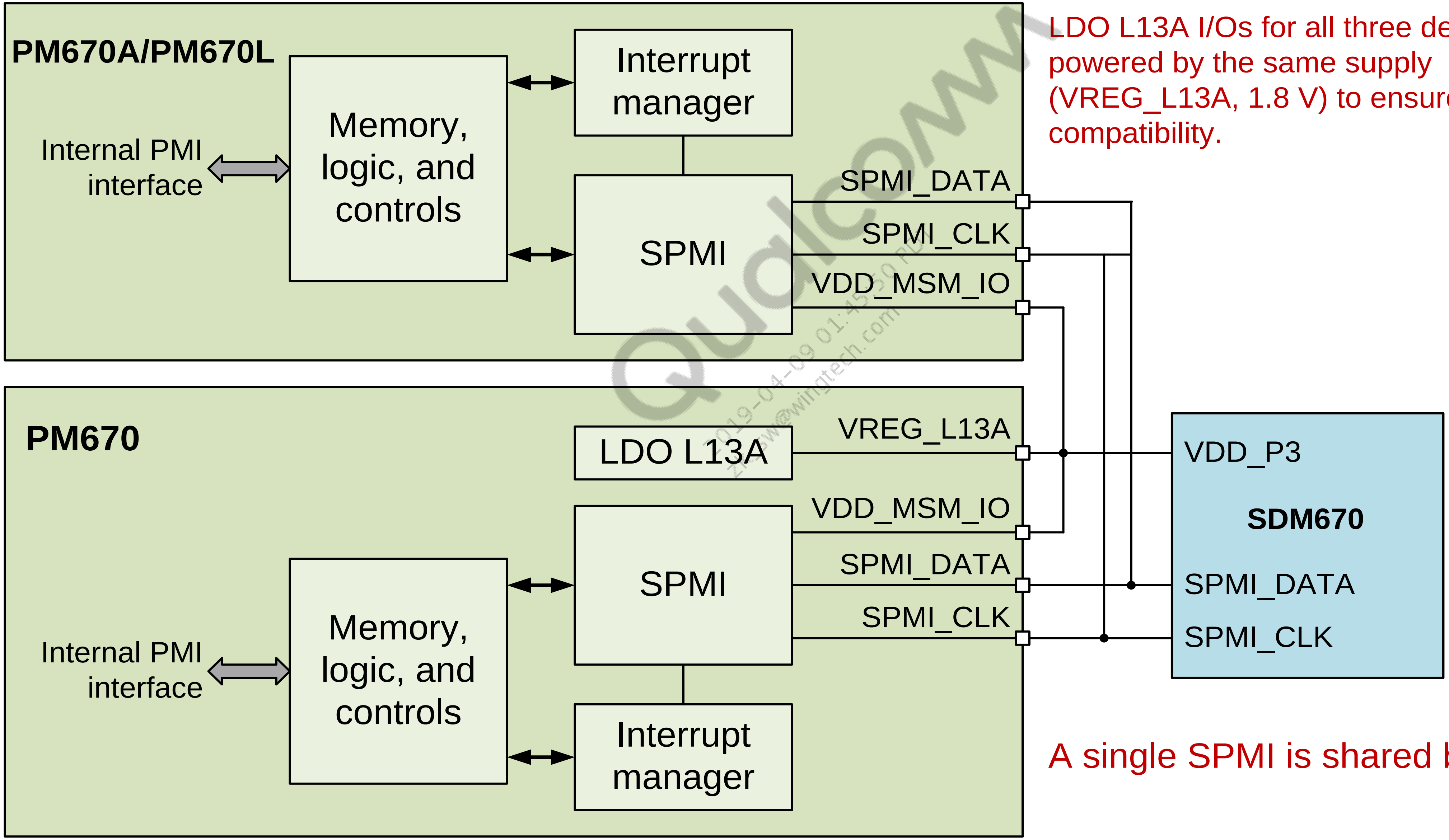
GPIO – Software Support PM670A/PM670L

GPIO	Special function	Type	SDM	Comments
1	–	MV	Option 2	Reserved for PBS during PON to determine configuration support
2	–	LV	–	–
3	–	LV	CAM_1P2_DVDD_LDO_EN	Enable for camera LDO1
4	–	LV	CAM_1P0_DVDD_LDO_EN	Enable for camera LDO1
5	–	LV	VBUS_OVP_EN_N	External OVP IC control pin
6	LPG channel 4	MV	LED_DRV	White LED driver/PWM
7	–	MV	KYPD_VOLP_N	Key interrupt to SDM
8	–	LV	EUD	Embedded USB debug
9	–	LV	WCSS pin ctrl	Pin control for WSSC voltage requests (also goes to PM670 GPIO 10)
10	–	MV	SLB	Single-line bus for PMIC to PMIC handshake during PON
11	–	LV	LP4x_REG_EN	LPDDR4X option detect and regulator control
12	–	LV	LP4x _MODE	Signal to LP4x regulator to force PWM mode



Communication Interface – SPMI

SPMI – Hardware Block Diagram



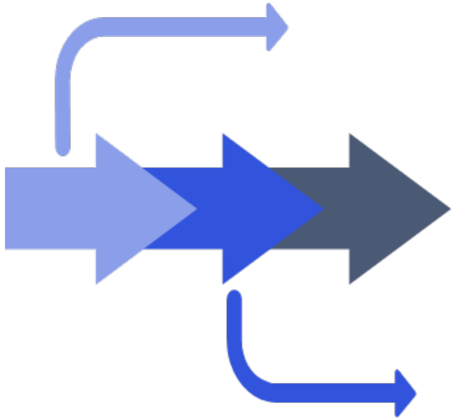
LDO L13A I/Os for all three devices are powered by the same supply (VREG_L13A, 1.8 V) to ensure logic compatibility.

A single SPMI is shared by all PMICs.

SPMI – Software Support

PMIC module	XBL	AOP	MPSS	TZ	aDSP	SLPI	HLOS
SPMI	Y	Y	Y	Y	N	N	Y

Note: See *System Drivers PMIC SPMI IRQ Overview* (80-NN989-1) and *Linux PMIC SPMI IRQ Overview* (80-NN990-1).



Debug Tools

PMIC Register Dump

- To run the tool, type the following command on any TRACE32 window (Apps or AOP TRACE32 are recommended):

```
"do PMICDump.cmm"
```

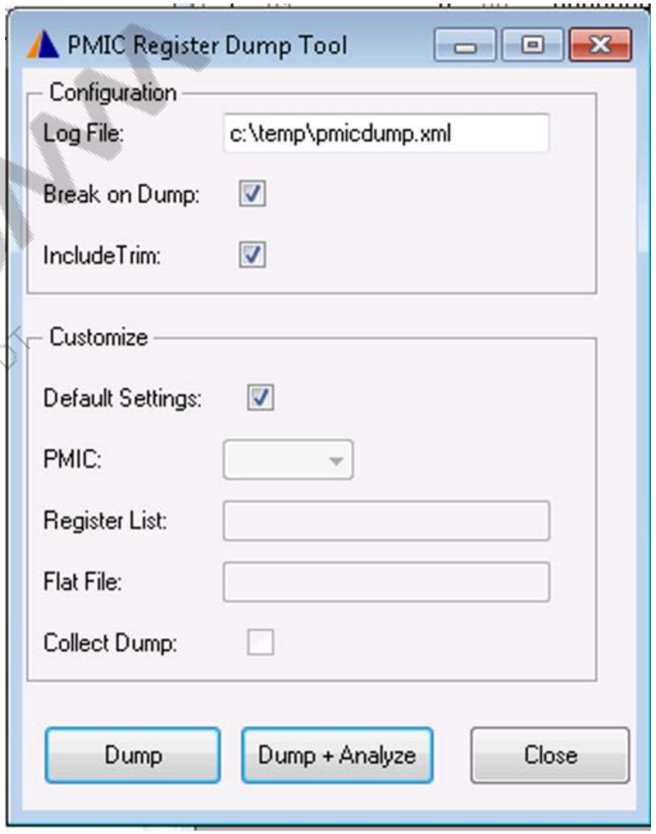
- GLUE build location:
  - \Common\Core\tools\tools\systemdrivers\pmic\hoya

Note: To run the script from the AOP TRACE32 window, first attach to the Apps TRACE32 window via sys.m.a command.

- It is unnecessary to modify the following settings irrespective of the TRACE32 window the script is being run from:
  - Owner – "DEFAULT"
  - AccessMode – "EZAXI"
- If CX is collapsed or in retention during sleep, modify the following:
  - Owner – "AOP"
  - AccessMode – "A"

PMIC Register Dump (cont.)

- Click **Dump** for a full dump from all PMICs on the target
- Click **Dump + Analyze** for a dump from all PMICs on the target and parsed PMIC dump output
- Optional dump configuration:
  - Select **Log File** to save logs to the default or custom location
  - Deselect **Break on Dump** to collect PMIC dump while TRACE32 is running
- To collect the dump:
  1. Deselect **Default Settings**
  2. Select the **PMIC** option (PMIC A, PMIC B, and so on) for which the dump must be collected
  3. Select **Collect Dump**



Log File: c:\temp\pmicdump.xml

Break on Dump: ☑

IncludeTrim: ☑

Default Settings: ☑

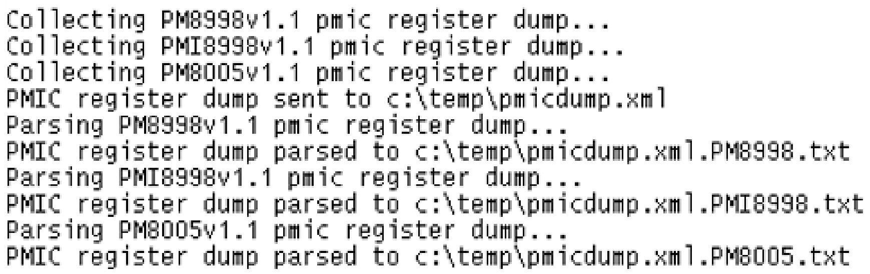
PMIC:

Register List:

Flat File:

Collect Dump: □

- Output example



```
Collecting PM8998v1.1 pmic register dump...
Collecting PMI8998v1.1 pmic register dump...
Collecting PM8005v1.1 pmic register dump...
PMIC register dump sent to c:\temp\pmicdump.xml
Parsing PM8998v1.1 pmic register dump...
PMIC register dump parsed to c:\temp\pmicdump.xml.PM8998.txt
Parsing PMI8998v1.1 pmic register dump...
PMIC register dump parsed to c:\temp\pmicdump.xml.PMI8998.txt
Parsing PM8005v1.1 pmic register dump...
PMIC register dump parsed to c:\temp\pmicdump.xml.PM8005.txt
```

PMIC Peek or Poke

- To run the tool, type the following command on any TRACE32 window (Apps or AOP TRACE32 are recommended):

```
"do PMICPeek.cmm"
```

- GLUE build location:
  - \Common\Core\tools\tools\systemdrivers\pmic\hoya

Note: To run the script from the AOP TRACE32 window, first attach to the Apps TRACE32 window via sys.m.a command.

- Fill the address to peek or poke
  - To read from the secondary slave ID address, use 0x1xxxx
- Command mode syntax:

```
do PMICPeek.cmm <address> <pmic_index> <data>
```

- Example – Peek address 0x11A04 from PMIC A:

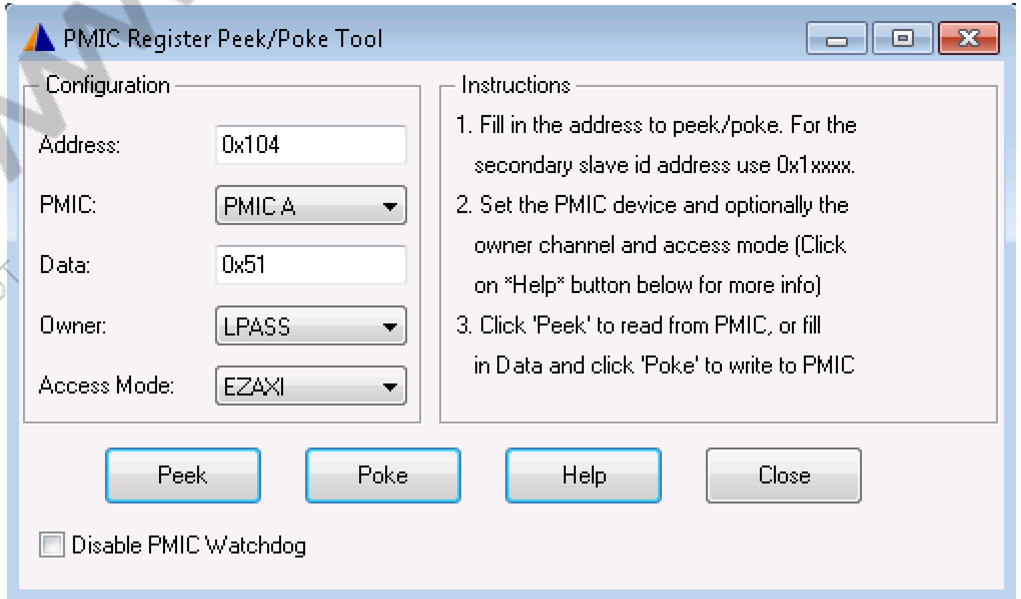
```
do PMICPeek.cmm 0x11A04 0
```

- Example – Poke address 0x11446 from PMIC B with data 0x80:

```
do PMICPeek.cmm 0x11446 1 0x80
```

PMIC Peek or Poke (cont.)

- Select the **PMIC** option (PMIC A, PMIC B, and so on.)
- It is unnecessary to modify the following settings irrespective of the TRACE32 window the script is being run from:
  - Owner – "DEFAULT"
  - AccessMode – "EZAXI“
- If CX is collapsed or in retention during sleep, modify the following:
  - Owner – "AOP"
  - AccessMode – "A"
- Click **Peek** to read from the PMIC address
- Fill the **Data** field and click **Poke** to write to the PMIC address



PMIC Register Peek/Poke Tool

Configuration

Address: 0x104

PMIC: PMIC A

Data: 0x51

Owner: LPASS

Access Mode: EZAXI

Instructions

1. Fill in the address to peek/poke. For the secondary slave id address use 0x1xxxx.
2. Set the PMIC device and optionally the owner channel and access mode (Click on "Help" button below for more info)
3. Click 'Peek' to read from PMIC, or fill in Data and click 'Poke' to write to PMIC

□ Disable PMIC Watchdog

PMIC PBUS Logger

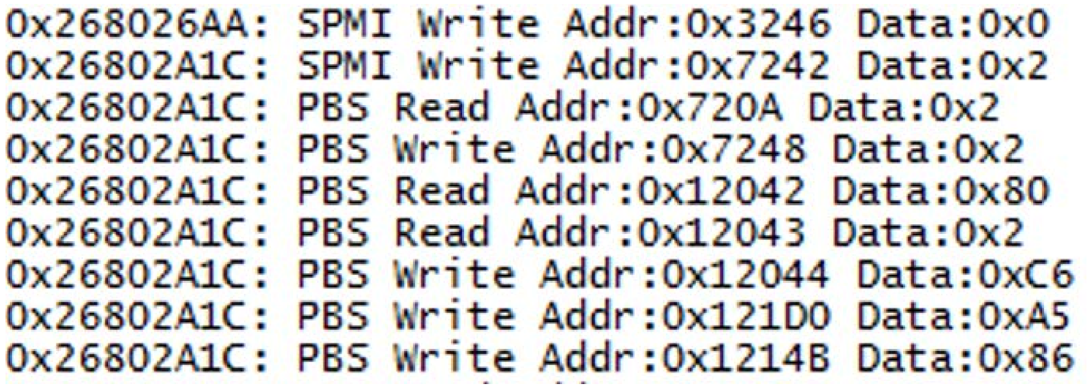
- To run the tool, type the following command on any TRACE32 window (Apps or AOP TRACE32 are recommended):

```
"do PBUSLogger.cmm"
```

- GLUE build location:
  - \Common\Core\tools\tools\systemdrivers\pmic\hoya

Note: To run the script from the AOP TRACE32 window, first attach to the Apps TRACE32 window via sys.m.a command.

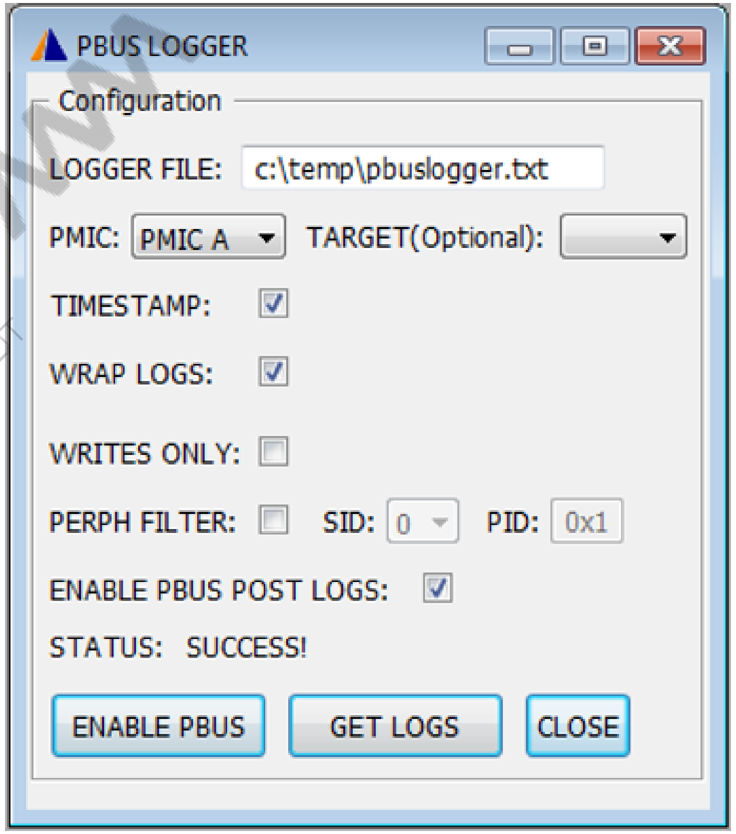
- Output example



```
0x268026AA: SPMI Write Addr:0x3246 Data:0x0
0x26802A1C: SPMI Write Addr:0x7242 Data:0x2
0x26802A1C: PBS Read Addr:0x720A Data:0x2
0x26802A1C: PBS Write Addr:0x7248 Data:0x2
0x26802A1C: PBS Read Addr:0x12042 Data:0x80
0x26802A1C: PBS Read Addr:0x12043 Data:0x2
0x26802A1C: PBS Write Addr:0x12044 Data:0xC6
0x26802A1C: PBS Write Addr:0x121D0 Data:0xA5
0x26802A1C: PBS Write Addr:0x1214B Data:0x86
```

PMIC PBUS Logger (cont.)

- LOGGER FILE specifies the location to save logs to be saved or use the default location. PBUS Logger creates a unique file when the tool is opened
- Select the PMIC option (PMIC A, PMIC B, and so on.)
- If SPMI has not already been initialized, select a TARGET option
- TIMESTAMP, WRAP LOGS, WRITES ONLY, and PERPH FILTER are optional
- Click **ENABLE PBUS** to collect PMIC transactions
- Click **GET LOGS** to get the transactions logged from the point at which the logger was enabled
- Select **ENABLE PBUS POST LOGS** to keep the PBUS logger enabled after getting the logs, or deselect to disable the logger after getting the logs



LOGGER FILE: c:\temp\pbuslogger.txt

PMIC: PMIC A TARGET(Optional):

TIMESTAMP: ☑

WRAP LOGS: ☑

WRITES ONLY: □

PERPH FILTER: □ SID: 0 PID: 0x1

ENABLE PBUS POST LOGS: ☑

STATUS: SUCCESS!

PMIC PBUS Logger (cont.)

- Example – To collect the write transactions to the LDO9 peripheral (sid: 0x1 base: 0x4800):
  - Select **PERIPH FILTER** with SID: 1 and PID: 0x48
  - Select **WRITES ONLY** to write transactions

- For the logger buffer to remain during shutdown and reset:

1. To ensure PBUS logger peripheral does not follow any shutdown or reset, in the intended PMIC:
   a. Write 0xA5 to the 0x4D0 register to unlock it.
   b. Write 0x0 to secured register 0x4DA.
2. To get the logs:
   a. Wait for the phone to reset.
   b. Click **GET LOGS** to get the logs.

PMIC Dashboard

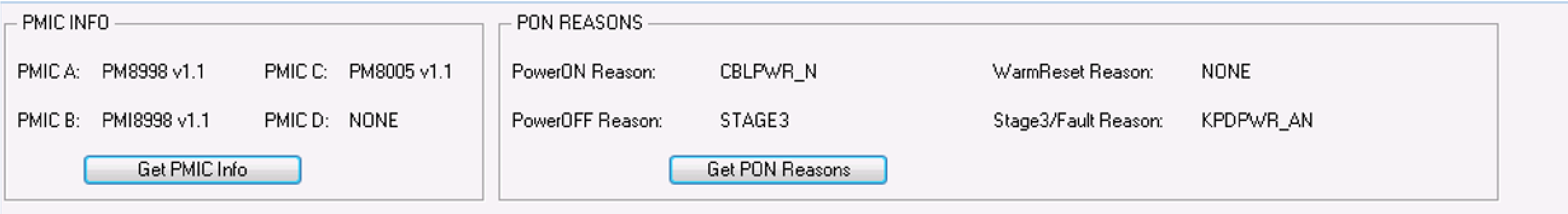
- To run the tool, type the following command on any TRACE32 window (Apps or AOP TRACE32 are recommended):

```
"do PMICDashBoard.cmm"
```

- GLUE build location:
  - \Common\Core\tools\tools\systemdrivers\pmic\hoya

Note: To run the script from the AOP TRACE32 window, first attach to the Apps TRACE32 window via sys.m.a command.

- Click **Get PMIC Info** to get the PMIC information on the target.
- Click **Get PON Reasons** to get PON reasons for the current boot.
- Select a PMIC from the pull-down menu and click **Get PON Reasons** to get PON reasons for the intended boot.



PMIC INFO

PMIC A: PM8998 v1.1   PMIC C: PM8005 v1.1

PMIC B: PMI8998 v1.1   PMIC D: NONE

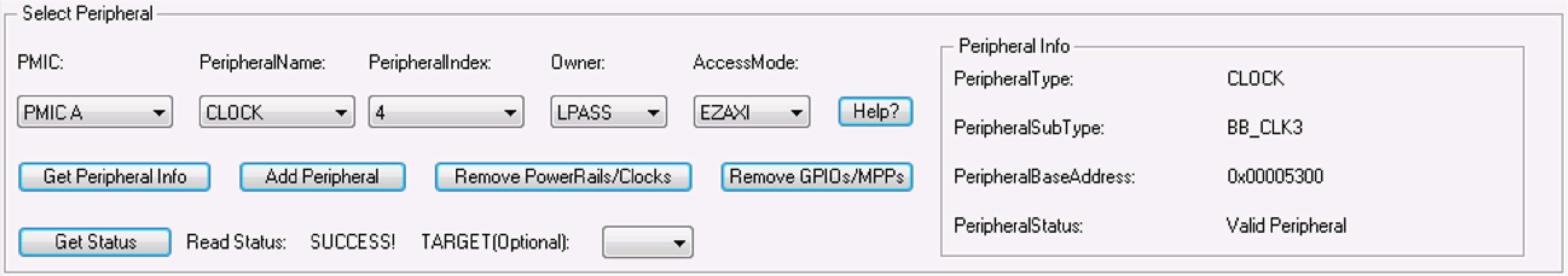
PON REASONS

PowerON Reason: CBLPWR_N   WarmReset Reason: NONE

PowerOFF Reason: STAGE3   Stage3/Fault Reason: KPDPWR_AN

PMIC Dashboard (cont.)

- Select the **PMIC**, peripheral from **PeripheralName**, and index number from **PeripheralIndex**
- It is unnecessary to modify the following settings irrespective of the TRACE32 window script is run from:
  - Owner – "LPASS“ or "DEFAULT"
  - AccessMode – "EZAXI“
- If CX is collapsed or in retention during sleep, modify the following:
  - Owner – "AOP"
  - AccessMode – "A"
- Click **Help?** to reveal the mapping of the selected peripheral, for example, CLOCK2 → LNBBCLK1
- Click **Get Peripheral Info** to display selected peripheral information under PeripheralInfo
- Click **Add Peripheral** to add a peripheral to the table and show status (once per peripheral)
- Click **Remove PowerRail/Clocks** to remove the power rail or clock
- Click **Remove GPIOs/MPPs** to remove GPIOs or MPPs



Select Peripheral

PMIC: PMIC A
PeripheralName: CLOCK
PeripheralIndex: 4
Owner: LPASS
AccessMode: EZAXI

Read Status: SUCCESS! TARGET(Optional):

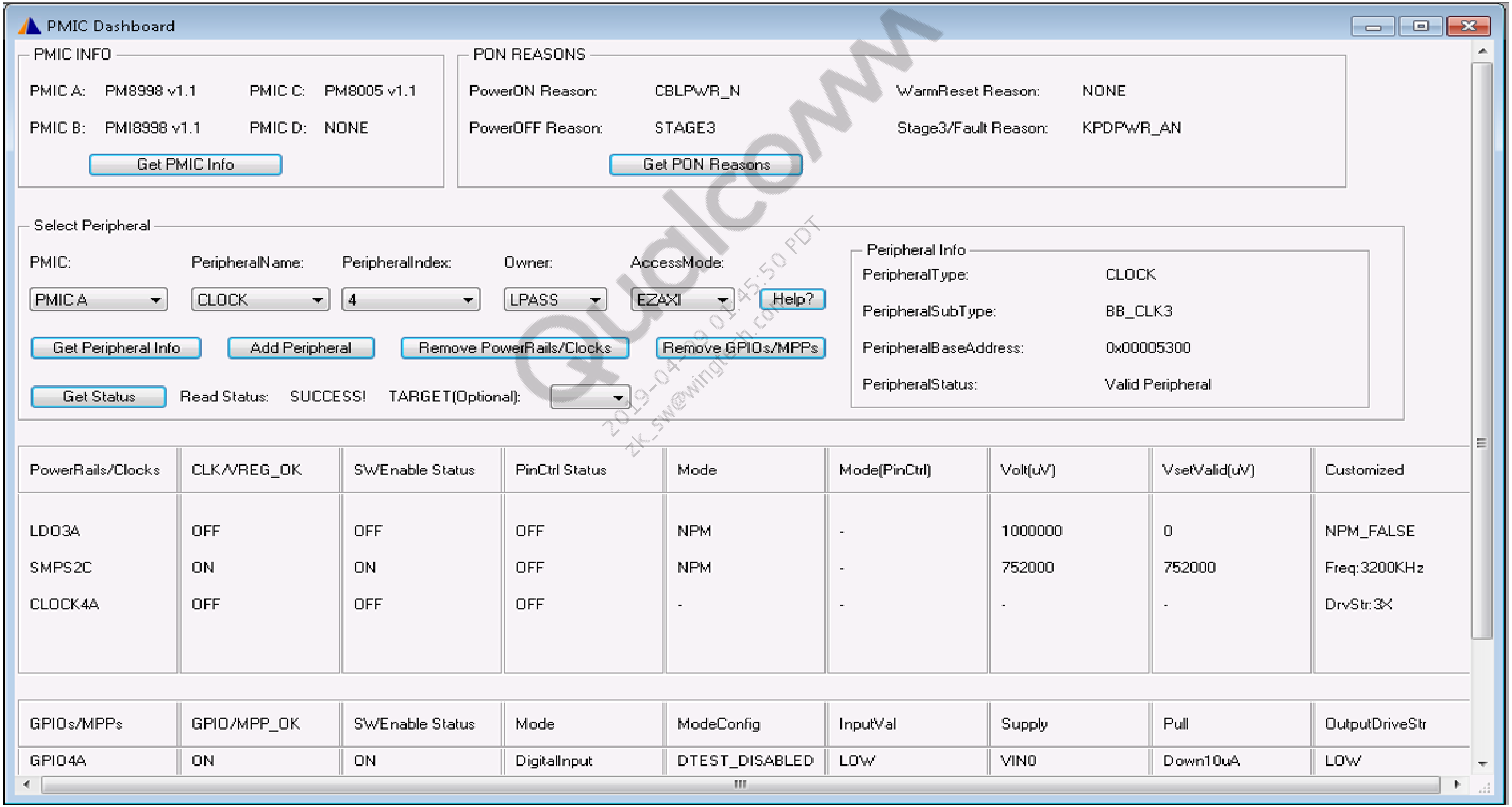
Peripheral Info

PeripheralType:	CLOCK
PeripheralSubType:	BB_CLK3
PeripheralBaseAddress:	0x00005300
PeripheralStatus:	Valid Peripheral

PMIC Dashboard (cont.)

- Click **Get Status** to get the status of each peripheral in the table
- If the SPMI is not initialized, select TARGET to initialize the SPMI
- Example – To select LDO3A:
  - From the PMIC menu, select A
  - From the PeripheralName menu, select LDO
  - From the PeripheralIndex menu, select 3
- Adding peripherals
  - If the peripheral is a power rail or clock, it is added to the top PowerRails/Clocks table, otherwise it is added to the bottom GPIOs/MPSS table
  - Add as many peripherals as necessary based on the table size limit
- Removing peripherals
  - Power rail and clock peripherals are removed from the bottom of the table
  - GPIOs and MPPs are removed from the top of the table

PMIC Dashboard (cont.)



PMIC Dashboard

PMIC INFO

PMIC A: PM8998 v1.1 PMIC C: PM8005 v1.1

PMIC B: PMI8998 v1.1 PMIC D: NONE

PON REASONS

PowerON Reason: CBLPWR_N WarmReset Reason: NONE

PowerOFF Reason: STAGE3 Stage3/Fault Reason: KPDPWR_AN

Select Peripheral

PMIC:	PeripheralName:	PeripheralIndex:	Owner:	AccessMode:
PMIC A	CLOCK	4	LPASS	EZAXI

Read Status: SUCCESS! TARGET(Optional):

Peripheral Info

PeripheralType: CLOCK

PeripheralSubType: BB_CLK3

PeripheralBaseAddress: 0x00005300

PeripheralStatus: Valid Peripheral

PowerRails/Clocks	CLK/VREG_OK	SWEnable Status	PinCtrl Status	Mode	Mode(PinCtrl)	Volt(uV)	VsetValid(uV)	Customized
LDO3A	OFF	OFF	OFF	NPM	-	1000000	0	NPM_FALSE
SMPS2C	ON	ON	OFF	NPM	-	752000	752000	Freq:3200KHz
CLOCK4A	OFF	OFF	OFF	-	-	-	-	DrvStr:3X

GPIOs/MPPs	GPIO/MPP_OK	SWEnable Status	Mode	ModeConfig	InputVal	Supply	Pull	OutputDriveStr
GPIO4A	ON	ON	DigitalInput	DTEST_DISABLED	LOW	VIN0	Down10uA	LOW

PMIC VRM Tools

- vrm.cmm script – Prints different DRV client votes on VRM resources and VRM resource status (including PMIC status)
- To run the tool, type the following command on any TRACE32 window (Apps or AOP TRACE32 are recommended):

```
"do vrm.cmm <target> <rsrc_names> file=<dump_file>"
```

  - <target> – Target on which the script is executed
  - <rsrc_names> – Resource names for which status is printed
  - <dump_files> – Optional dump filename with location

- GLUE build location:
  - \Common\Core\tools\tools\systemdrivers\pmic\hoya

Note: To run the script from the AOP TRACE32 window, first attach to the Apps TRACE32 window via sys.m.a command.

PMIC VRM Tools (cont.)

Example vrm.cmm command	Description
`do vrm.cmm 845`	Prints all VRM resource votes and status
`do vrm.cmm 845 *`	Prints all VRM resource votes and status
`do vrm.cmm 845 l5a` (or ldo5a, ldoa5)	Prints LDO5A VRM votes and status
`do vrm.cmm 845 s3a` (or smp3a, smpa3)	Prints SMPS3A VRM votes and status
`do vrm.cmm 845 rf3a` (or rfclk3a, rfclka3)	Prints RFCLK3A VRM votes and status
`do vrm.cmm 845 s2c` (or smp2c, smpc2)	Prints SMPS2C VRM votes and status
`do vrm.cmm 845 bb1a` (or bbclk1a, lnbbclk1a)	Prints LNBBCLK1A VRM votes and status
`do vrm.cmm 845 "s3a bob1b l5a rf2a"`	Prints SMPS3A, BOB1B, LDO5A, RFCLK2A VRM votes, and status
`do vrm.cmm 845 * file=default`	Prints and copies all the VRM resource votes and status to default file

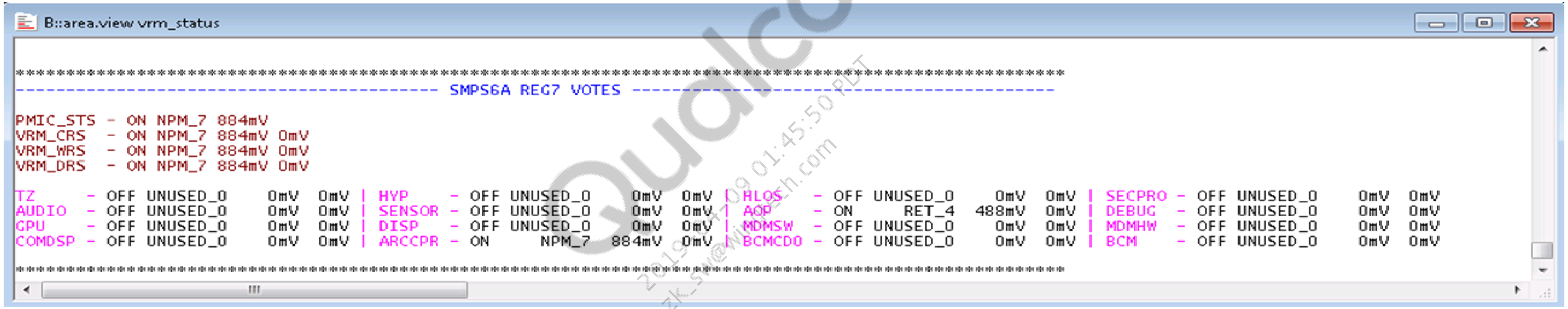
`do vrm.cmm 845 *`
`file=\\<user>\dropbox\demo.txt`
	Prints and copies all the VRM resource votes and status to demo.txt

`do vrm.cmm 845 l5a access=a`	Use access=a in sleep on AOP TRACE32 if CX is collapsed or in retention

PMIC VRM Tools (cont.)

- Example output running the following command:

```
do vrm.cmm 670 s5a
```



B::area.view vrm_status

```
********************************************************************************************************
------------------------------------------ SMPS6A REG7 VOTES ------------------------------------------

PMIC_STS - ON NPM_7 884mV
VRM_CRS  - ON NPM_7 884mV 0mV
VRM_WRS  - ON NPM_7 884mV 0mV
VRM_DRS  - ON NPM_7 884mV 0mV

TZ     - OFF UNUSED_0    0mV  0mV | HYP    - OFF UNUSED_0    0mV  0mV | HLOS   - OFF UNUSED_0    0mV  0mV | SECPRO - OFF UNUSED_0    0mV  0mV
AUDIO  - OFF UNUSED_0    0mV  0mV | SENSOR - OFF UNUSED_0    0mV  0mV | AOP    - ON      RET_4  488mV  0mV | DEBUG  - OFF UNUSED_0    0mV  0mV
GPU    - OFF UNUSED_0    0mV  0mV | DISP   - OFF UNUSED_0    0mV  0mV | MDMSW  - OFF UNUSED_0    0mV  0mV | MDMHW  - OFF UNUSED_0    0mV  0mV
COMDSP - OFF UNUSED_0    0mV  0mV | ARCCPR - ON      NPM_7  884mV  0mV | BCMCD0 - OFF UNUSED_0    0mV  0mV | BCM    - OFF UNUSED_0    0mV  0mV

********************************************************************************************************
```

PMIC VRM Tools (cont.)

- vrm_dump.cmm script – Collects the VRM register dump
- To run the tool, type the following command on any TRACE32 window (Apps or AOP TRACE32 are recommended):

```
"do vrm_dump.cmm <target> <register_filter> file=<dump_file>"
```

  - <target> – Target on which the script is executed
  - <register_filter> – Optional register filter name for debugging
  - <dump_files> – Optional dump filename with location
- GLUE build location:
  - \Common\Core\tools\tools\systemdrivers\pmic\hoya

Note: To run the script from the AOP TRACE32 window, first attach to the Apps TRACE32 window via sys.m.a command.

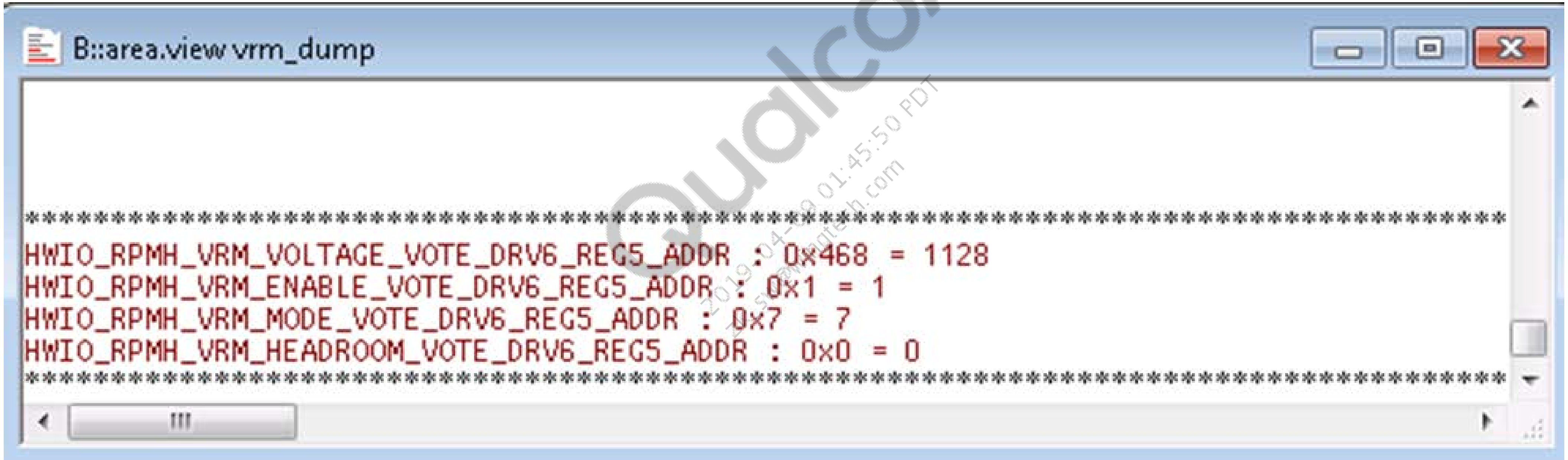
PMIC VRM Tools (cont.)

Example vrm_dump.cmm command	Description
do vrm_dump.cmm 670	Collects full VRM register dump and copies to default file
do vrm_dump.cmm 670 reg10	Collects and prints all the registers dump with "reg10" in the name (regulator 10-related register dumps)
do vrm_dump.cmm 670 vote_drv10_xob5	Collects and prints all the registers dump with "reg10" in the name (vote table dump for DRV 10, XOB 5)
do vrm_dump.cmm 670 * file=default	Prints and copies all the VRM resource votes and status to the default file
do vrm_dump.cmm 670 * file=\\<user>\dropbox\demo.txt	Collects a full VRM dump, prints, and copies it to demo.txt
do vrm_dump.cmm 670 reg10 access=a	Use access=a in sleep on AOP TRACE32 if CX is collapsed or in retention

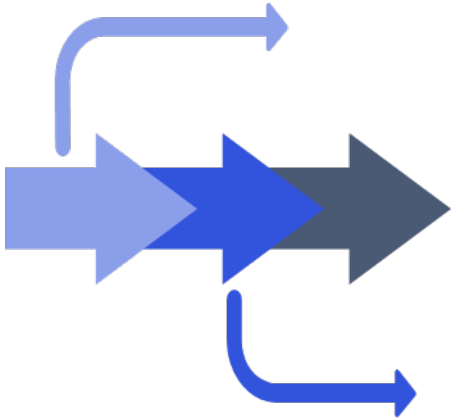
PMIC VRM Tools (cont.)

- Example output running the following command:

```
do vrm_dump.cmm 670 vote_drv6_reg5
```



```
**************************************************************************************
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV6_REG5_ADDR : 0x468 = 1128
HWIO_RPMH_VRM_ENABLE_VOTE_DRV6_REG5_ADDR : 0x1 = 1
HWIO_RPMH_VRM_MODE_VOTE_DRV6_REG5_ADDR : 0x7 = 7
HWIO_RPMH_VRM_HEADROOM_VOTE_DRV6_REG5_ADDR : 0x0 = 0
**************************************************************************************
```



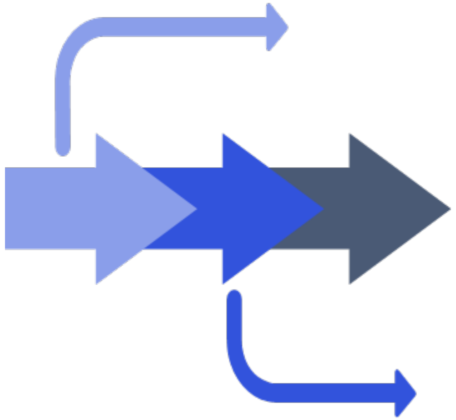
References

References

Documents	
Title	**Number**
Qualcomm Technologies, Inc.	
SDM670/SDM710/SDM712 Linux Android PMIC Software Drivers Overview	80-PD126-4
PM670 and PM670A/PM670L Power Management ICs	80-PD119-5A
UEFI PMIC Software User Guide	80-P2484-42
RPM Hardening Overview and Debug	80-P9301-16
Non-HLOS Voltage Regulator and Clock PMIC Software User Guide	80-P9301-78
System Drivers PMIC SPMI IRQ Overview	80-NN989-1
Linux PMIC SPMI IRQ Overview	80-NN990-1
Linux Android PMIC GPIO Software User Guide	80-P9301-88
Non-HLOS PMIC ADC Software User Guide	80-P2484-71

References (cont.)

Acronyms	
Acronym or term	**Definition**
AOP	Always on processor
AMC	Active mode configuration
ARC	Aggregated resource controller
BUA	Battery UICC alarm
DRV	Direct resource voter
PBS	Programmable boot sequencer
PON	Power on
POFF	Power off
PRM	PMIC resource manager
RPM	Resource power manager
RPMh	RPM hardening
RSC	Resource state coordinator
TCS	Trigger command set
VRM	Voltage regulator manager
WD	Watchdog



Questions?

https://createpoint.qti.qualcomm.com