UPOWERFWUG

uPower Firmware User's Guide

Rev. 0 — 30 November 2022 **User guide**

Document information

Information	Content
Keywords	uPower, i.MX 8ULP, Linux
Abstract	This document describes the i.MX 8ULP power management subsystem features, power domains, uPower interface, RTD and APD power mode transitions, PMIC driver, and APIs provided by uPower for Arm Cortex-M33/ A35 runtime call.

1 Overview

This document describes the i.MX 8ULP power management subsystem features, power domains, uPower interface, RTD and APD power mode transitions, PMIC driver, and APIs provided by uPower for Arm Cortex-M33/A35 runtime call.

2 Acronyms and Abbreviations

Table 1. Acronyms and abbreviations

Name	Description	Comment
AD	Application Domain	-
AFBB	Asymmetric Forward Body Bias	Used for high-performance modes. Increases transistor switching speed.
AHB	Arm Advanced High Performance Bus	-
AMBA	Arm Advanced Microcontroller Bus Architecture	-
AOGPOR	General Purpose Output Register	Always on domain.
APB	Arm Advanced Peripheral Bus	-
APD	Application Domain	-
AVD	Audio Video Domain	-
CMC	Core Mode Controller	Refer to the i.MX 8ULP Reference Manual.
DGO	Always ON domain	-
Dombias	Domain bias	
HVD	High Voltage Detector	-
MU	Message Unit	-
P-Channel	Arm P-Channel Interface	-
PMC	Power Management Controller	-
PMIC	Power Management IC	-
PS	Power Switch	-
RBB	Reverse Body Bias	Used for low-power modes. Reduces leakage.
RTD	Real Time Domain	-
SIC	System Interface Control Module	-
LVD	Low Voltage Detector	-

3 uPower Subsystem and Firmware Introduction

The uPower is a programmable subsystem based on a RISC-V CPU compatible with the RV32EMC instruction set. The uPower runs a dedicated firmware provided by NXP. It controls the i.MX 8ULP device power-related functions and provides power-related services to its clients. Its main functions notably include:

- Running pre-programmed control functions during device initialization.
- Abstracting power control services for its clients (application software).
- Managing device various power mode entry and exit, and wakeup sources.
- Exposing a set of information services, such as device temperature or power consumption measurement of a specific power domain.

Complex power control services may be provided, such as the determination of the optimal.

The uPower has a dedicated power supply, so it can stay operational even if all the other device power domains are powered off. The power supply has a low-power mode that is used when the uPower is in low-power state with its clock gated off. The subsystem has a dedicated 16/64 MHz configurable oscillator controlled by the uPower CPU. If required, the oscillator is switched to 64 MHz mode. For standard applications, the frequency of the uPower subsystem is 16 MHz. The oscillator is turned off in the uPower low-power mode.

The following figure describes the main modules of the uPower firmware.

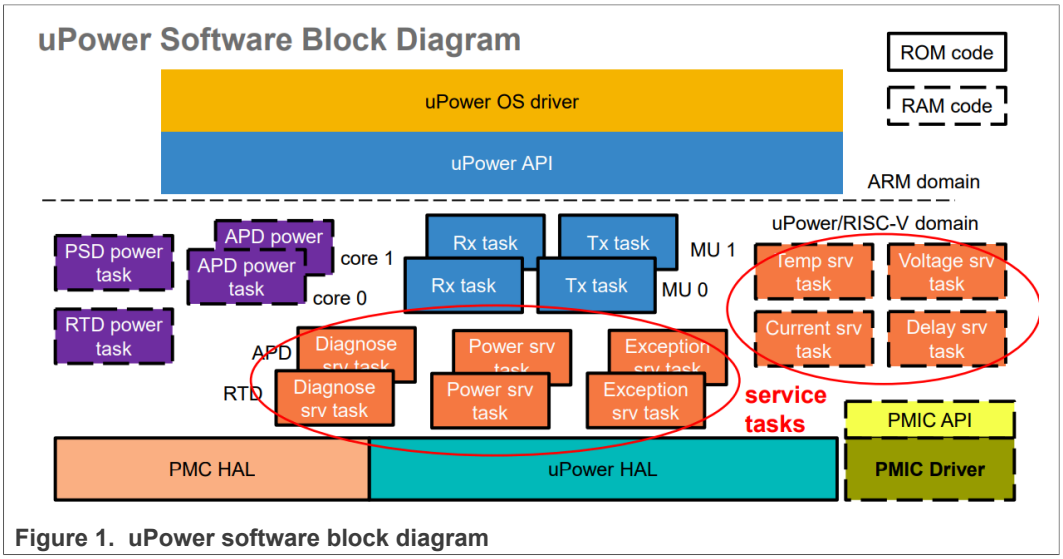

Figure 1. uPower software block diagram

The call sequence flow for each API is similar. Take “turn on power switches” as an example.

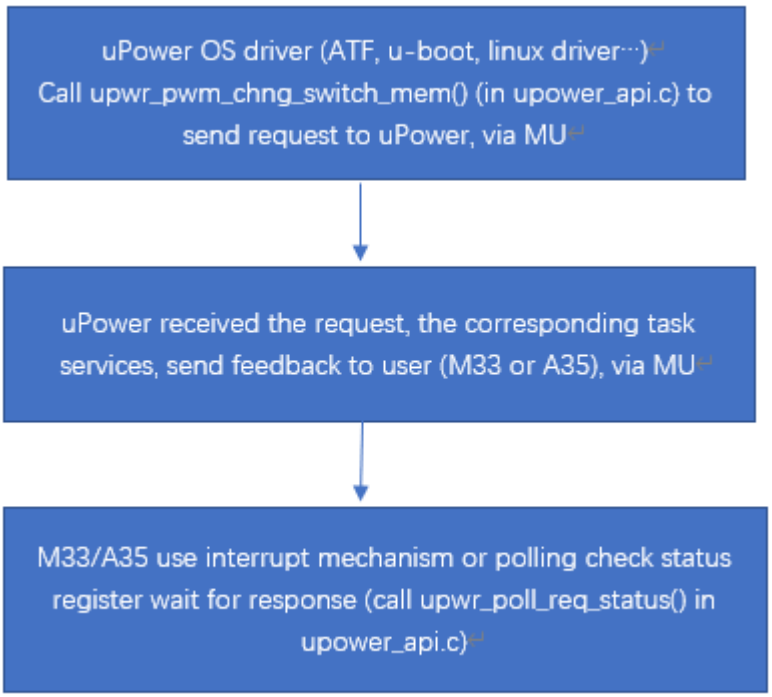

Figure 2. Call sequence flow

4 Feature List

The uPower Firmware includes the following features:

- Process Monitor for device process corner evaluation
- Power Meter for device power domains consumption measurement
- Temperature Sensor for device temperature measurement
- Critical Path delay meter for critical path delay measurement
- Messaging Units for communication with on-chip processors
- I2C for communication with off-chip devices especially power sources
- Internal Voltage Meter (VMeter)

5 Release Package

Table 2. NXP uPower software release package

Category	File List	Description
uPower_api_files	`upower_defs.h`	The two header files contain the necessary data structures for uPower API calls and power mode transitions.
	`upower_soc_defs.h`	
	`upower_api.h`	uPower API header file, listing all the supported APIs.
	`upower_api.c`	uPower API C language file.

Table 2. NXP uPower software release package*...continued*

Category	File List	Description
uPower_A1_firmware_binary	`upower_fw.bin`	uPower official release firmware binary: `upower_fw.bin`, users need to integrate this uPower firmware binary into OEM container, same as the Cortex-M33 or Cortex-A35 image, using different core IDs. The uPower firmware image core ID is 4. (The Cortex-M33 core ID is 1, and the Cortex-A35 core ID is 2). For how to build an OEM container, see the *i.MX Linux User's Guide* (IMXLUG).
uPower_firmware_porting_kit_A1	`pmicdrv`	PMIC driver directory, including PCA9460 PMIC driver source code and necessary uPower header files.
	`lib_upower_fw.a`	uPower firmware static library.
	`Release_note.txt`	Release Notes.
	`nxp_official_upower_firmware_binary/upower_fw.bin`	NXP official release upower firmware binary.
	`a1_rom_rc3_3.sym`	uPower ROM A1 symbol table.
	`linker_ram_final.ld`	uPower firmware linker file.
	`Makefile`	Build command: `make clean;make`
	`Makefile_combine_upower_fw_pmic`	Use this Makefile to combine the uPower firmware static library and PMIC driver static library to generate the uPower firmware binary.
	`Makefile_pmic_lib`	Use this Makefile to generate the PMIC driver static library.

6 Power Domains

Power domains

- RealTime domain (RTD)
  - Fusion DSP
    - Fusion Always-on
  - Fuses
  - Cortex-M33
- Application domain (AD)
  - Fast NIC
  - Cortex-A35[0]
  - Cortex-A35[1]
- Low Power Audio-Video domain (LPAVD)
  - HiFi4
  - DDR/DDR PHY
  - PXP/EPDC
  - GPU3D
  - MIPI DSI

  - MIPI CSI
- Always ON domain (DGO)
- VBAT domain

7 uPower Subsystem Block Diagram

The following figure shows the uPower interface with emphasis on the system interface (AHB) and always on domain interface signals. It also shows the interrupt signals from various sources, internal to the subsystem and external.

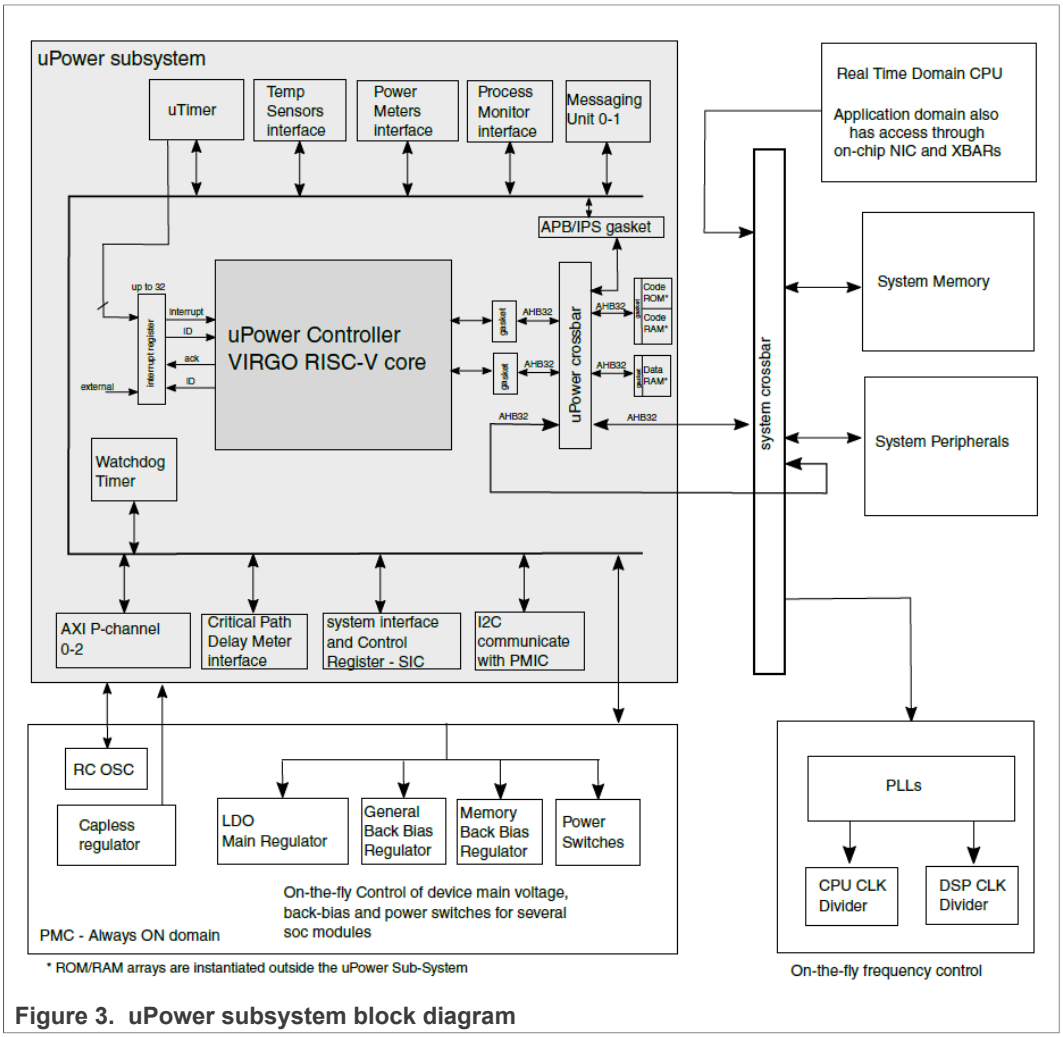

Figure 3. uPower subsystem block diagram

The following figure shows the uPower interfaces with other modules and power domains. Those interfaces are with the Real Time Domain (RTD), where the P-Channel, Messaging Unit (MU), and AHB interfaces are used. With the Application Domain (APD), the P Channel interface and MU interface are used. uPower interfaces with the Power Mode Controller (PMC) through an IPS interface and with an external PMIC through an I2C interface.

From the uPower subsystem block diagram, we can see that:

- uPower has shared system memory (memory address: 0x28330000) to exchange data with Arm Cortex-M33/A35.

- uPower has a temperature sensor module to provide the measure temperature service.
- uPower has a Power meter module to provide the measure power consumption service.
- uPower has a VMeter module to provide measure voltage service.
- uPower has a process monitor module to provide the process monitor service.
- uPower communicates with the PMIC chip through I2C.
- uPower controls Power switches.
- uPower can write AOGPOR (General Purpose Output Register – Always on domain).

The following figure shows the connection between uPower and RTD/APD.

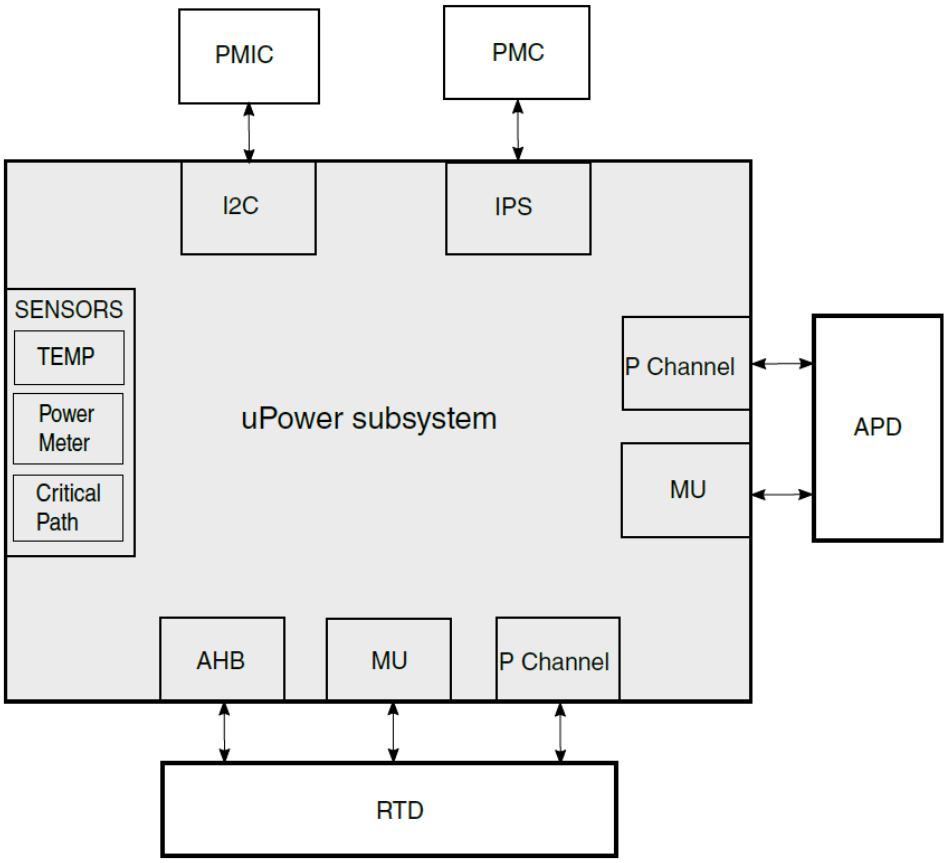

Figure 4. Connection between uPower and RTD/APD

The MU module registers A and B sides are represented in the uPower subsystem, even though they are in the RTD and APD power domains. The uPower interface implements the MU IPS slave bus and other side band signals for both RTD and APD domains.

From the perspective of users (Arm Cortex-M33/A35), uPower mainly provides two functions:

- Provides APIs for Arm Cortex-M33/A35 runtime call.
- Assists Arm Cortex-M33/A35 to complete power mode transition tasks.

8 uPower API

uPower service requests are classified in Service Groups. Each Service Group has a set of related functions. The service groups are as follows:

- Exception Service Group – `upwr_xcp_*`
  Gathers functions that deal with errors and other processes outside the functional scope.
- Power Management Services Group – `upwr_pwm_*`
  Functions to control switchs, configure power modes, and set internal voltages, etc.
- Delay Measurement Service Group – `upwr_dlm_*`
  Delay measurement functions using the process monitor and delay meter.
- Voltage Measurement Servcie Group – `upwr_vtm_*`
  Functions for voltage measurements, power meter, and set/get PMIC rail voltage, etc
- Temperature Measurement Service Group – `upwr_tpm_*`
  Functions for temperature measurements.

For detailed function purposes and argument descriptions, see the *uPower API Reference Manual* (UPOWERAPIRM).

For the sample code of how to call the uPower API, see the SDK or BSP code.

For example, the sample code to initialize, see Section 11.1.

9 Power Mode Transition

i.MX 8ULP supports several different power modes for RTD and APD, including Active, Sleep, Deep Sleep, Power Down, Deep Power Down, and HOLD. These power mode transitions are initiated and controlled by Arm Cortex-M33/A35. Some transitions require the participation of uPower. Arm Cortex-M33/A35 can configure power switches, memory partitions, dombias mode, AFBB/RBB, PADs and PMIC under different power modes by configuring the power mode data structure stored in shared memory (memory address: 0x28330000).

This chapter describes the power modes, the connection between RTD/APD and uPower, and the meaning of the power mode data structure.

For how to do power mode transition on Arm Cortex-M33/A35 side, see the SDK/BSP source code and user guides.

9.1 RTD power mode transition

9.1.1 RTD connection

The following figure shows the connection between RTD and uPower, CMC, CGC, PCC, Interrupts WKUP.

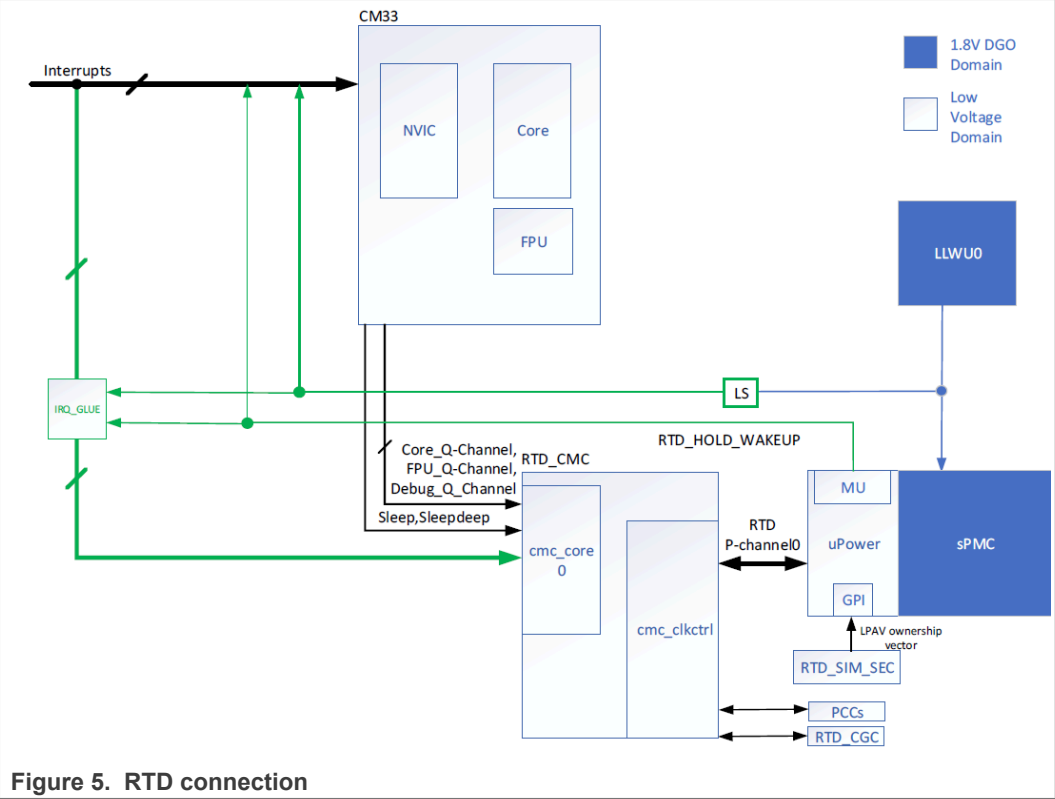

Figure 5. RTD connection

9.1.2 RTD clock mode

CKMODE	Core	Platform/AHB		Bus/Slow
		Master	Slave	Periph
0	Clocking	Clocking	NA	Clocking
1	Gated	Clocking	NA	Clocking
3	Gated	Gated	NA	Clocking
7	Gated	Gated	NA	Gated

Note: *The clock mode is controlled by the CKMODE field of CMC register. See the Cortex-M33 SDK source code for details on how to program it.*

9.1.3 RTD power modes

LPMODE[4]	LPMODE[3]	LPMODE[2]	LPMODE[1]	LPMODE[0]	SoC Mode
0	0	0	0	0	Active
0	0	0	0	1	Sleep
0	0	0	1	1	Deep Sleep
0	0	1	1	1	Power Down
0	1	1	1	1	Deep Power Down
1	1	1	1	1	HOLD

- Active
  - Normal RUN mode
  - All functional
  - Low-power configuration using FW services
- Sleep
  - Clock gating mode only
  - No uPower engagement
  - Low-power configuration using FW services before transitioning
  - Fast entry and exit
- Deep Sleep
  - uPower is engaged
  - Low-power configuration using FW data structs (applied by FW)
- HOLD
  - Transient state (automatic wakeup by uPower after configuration is applied)
  - Used to safely apply low-power configuration
- Power Down
  - Power gating mode
  - All power switches are open
  - Memories can be retained
- Deep Power Down
  - Power gating mode
  - All power switches are open
  - External supplies are shut down

The following figure shows the power mode state transition diagrams for Cortex-M33.

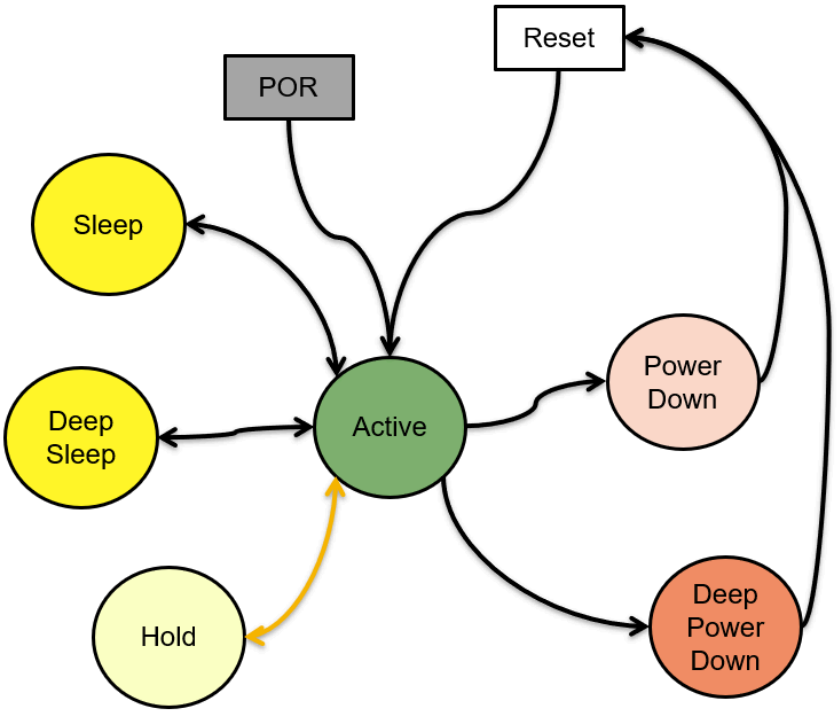

Figure 6. Power mode state transition

9.1.4 Example of RTD low-power mode entry sequence

For example, the steps for RTD to enter lower-power mode are as follows:

1. Initialize uPower configuration data (RTD part), which is stored at (SSRAM) memory.
   - See definition of `struct ps_pwr_mode_cfg_t` in the `upower_soc_defs.h` file for details.
2. Program CMC0 to notify the system (uPower) to enter Power Down mode.
   - Write register `CMC_RTD.CKCTRL` to control if the gate system clocks in low power mode.
   - Write register `CMC_RTD.RTD_PMPROT` to remove write protection.
   - Write register `CMC_RTD.DBGCTL.CMC_DBGCTL_SOD_MASK` to control whether to enable debug feature in low power mode.
   - Write register `CMC_RTD.RTD_PMCTRL.RTD_LPMODE` to configure which low power mode plan to enter.
   - Write register `SCB.SCR.SCB_SCR_SLEEPDEEP_Msk` to configure core SLEEPDEEP bit.
   - `__DSB();`
   - `__WFI();` /* RTD M33 pends here in low power mode
   - `__ISB();`

Note:

When Cortex-M33/A35 is triggered (by CMC) to enter low power mode, uPower starts the power mode transition task and the task loads the configuration data from SSRAM to set corresponding modules, e.g., power switches, memory partitions, PMIC registers, etc.

The global configuration data is initialized by uPower and users do not need to initialize it.

9.2 APD power mode transition

9.2.1 APD connection

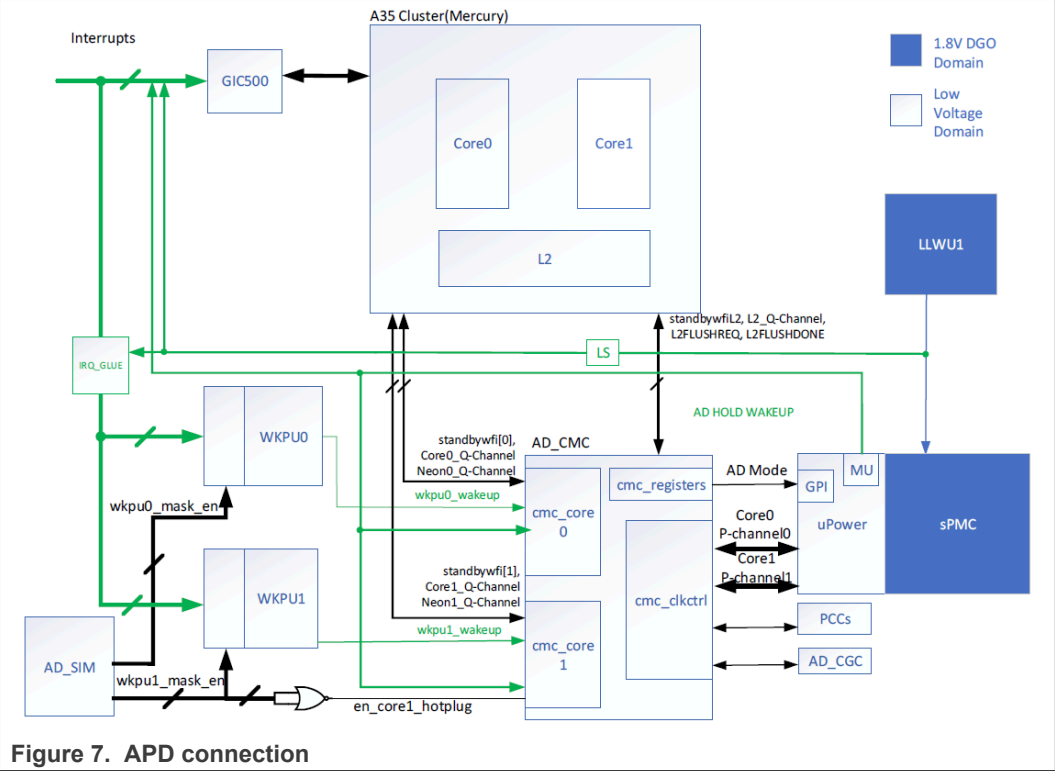

Figure 7. APD connection

9.2.2 APD clock modes

CKMODE	Core	Platform/AHB		Bus/Slow
		Master	Slave	Periph
0	Clocking	Clocking	NA	Clocking
1	Gated	Clocking	NA	Clocking
3	Gated	Gated	NA	Clocking
7	Gated	Gated	NA	Gated

Note: *The clock mode is controlled by the CKMODE field of CMC register. See the BSP source code for details on how to program it.*

9.2.3 APD power modes

Note: *Although ACTIVE DMA mode is listed here, but actually i.MX 8ULP does not support ACTIVE DMA mode at present. In addition, there is a bug in the uPower ROM, it uses ACTIVE MODE index instead of ACTIVE MODE. Therefore, as a workaround, when users need to use and configure ACTIVE mode, configure the shared memory pointed by the ACTIVE MODE index.*

- Active
  - Normal RUN mode
  - All functional

  - Low-power configuration using FW services
- Sleep
  - Clock gating mode only
  - No uPower engagement
  - Low-power configuration using FW services before transitioning
  - Fast entry and exit
- Deep Sleep
  - uPower is engaged
  - Low-power configuration using FW data structs (applied by FW)
- Hold
  - Temporary mode (automatic wakeup by uPower after configuration is applied)
  - Used to safely apply low-power configuration
- Active DMA
  - Transient state (automatic re-enter on previous low-power mode after handling the DMA)
- Partial Active
  - Power gating mode
  - Cortex-A35 complex and fast NIC domains are powered off
  - Used to allow RTD using AD and AVD peripherals
- Power Down
  - Power gating mode
  - All power switches are open
  - Memories can be retained
- Deep Power Down
  - Power gating mode
  - All power switches are open
  - External supplies are shut down

The following figure shows the power mode state transition diagrams for Cortext-A35 domain.

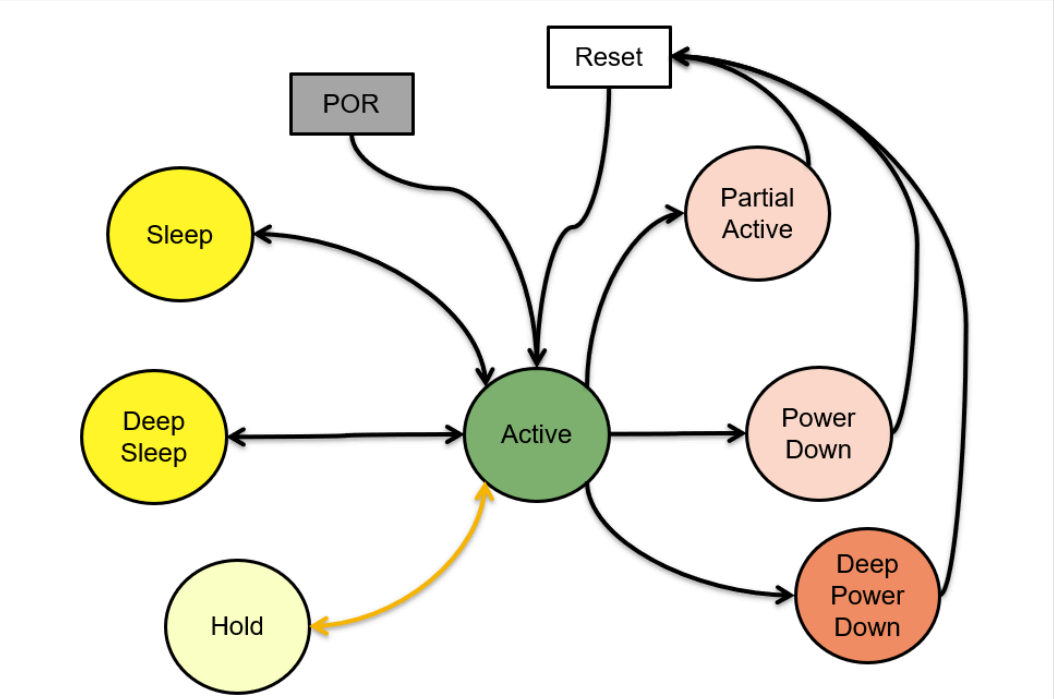

Figure 8. Power mode state transition

9.2.4 Example of APD low-power mode entry sequence

The steps for APD to enter lower-power mode are as follows:

- Initialize uPower configuration data (APD part), which is stored at (SSRAM) memory.
  - See definition of `struct ps_pwr_mode_cfg_t` in the `upower_soc_defs.h` file for details.
- Core sleep procedure:
  1. Enable CA35 retention.
  2. Disable data cache.
  3. Clean and invalidate all data from L1 Data cache.
  4. Disable data coherency with other cores in the cluster.
  5. Mask interrupts on the core.
  6. Execute an ISB instruction to ensure that all of the register changes from the previous steps have been committed.
  7. Execute a DSB SY instruction to ensure that all cache, TLB, and branch predictor maintenance operations issued by any core in the cluster device before the SMPEN bit is cleared have completed.
  8. Execute a WFI instruction and wait until the STANDBYWFI output is asserted to indicate that the core is in idle and low-power state.

9.3 Domain power mode combinations

NR = Not recommended
No = Not supported in hardware

CM33 / CA35[1]	*Active*[2]	*Sleep*	*Deep Sleep*	*Power Down*	*Deep Power Down*
Active[3]	YES	NR[3]	NR[3]	No[5]	NO[4,5]
Partial Active	YES	YES	YES	No	No
Sleep	YES	YES	NR[3]	No[5]	NO[5]
Deep Sleep	YES	YES	YES	No[5]	NO[5]
Power Down	YES	YES	YES	YES[6]	YES
Deep Power Down	YES	YES	YES	YES[6]	YES

NOTES:

1. *CA35 only supports static (w/o clock) RBB.*
2. *Active mode will also allow to enable RBB but will limit the operation frequency to 40Mhz.*
3. *No power advantage to keep CM33 in low power modes while CA35 is in active mode, though hardware supports this option*
4. *No clock available for CA35 when CM33 is in "Deep Power Down" mode. There are also no power advantages using this mode.*
5. *8ULP does not support any isolation from CM33 to CA35 domain so there is no possibility for CM33 to be kept in Power Down modes while CA35 in higher power modes.*
6. *CM33 domain can only enter Power Down mode once CA35 is in Power Down mode. CM33 must be active to be able to wake CA35 from Power Down mode. For the case where both CA35 and CM33 are in Deep Power Down and Power Down mode respectively, CM33 must wake-up first before CA35 can exit Power Down mode.*

9.4 Power mode transition data structure

The following figure and tables describe the power mode configuration data structure, users (Arm Cortex-M33/A35) need to configure the power mode transition data located in shared memory. The memory address is 0x28330000. See `struct ps_pwr_mode_cfg` in `upower_soc_defs.h`.

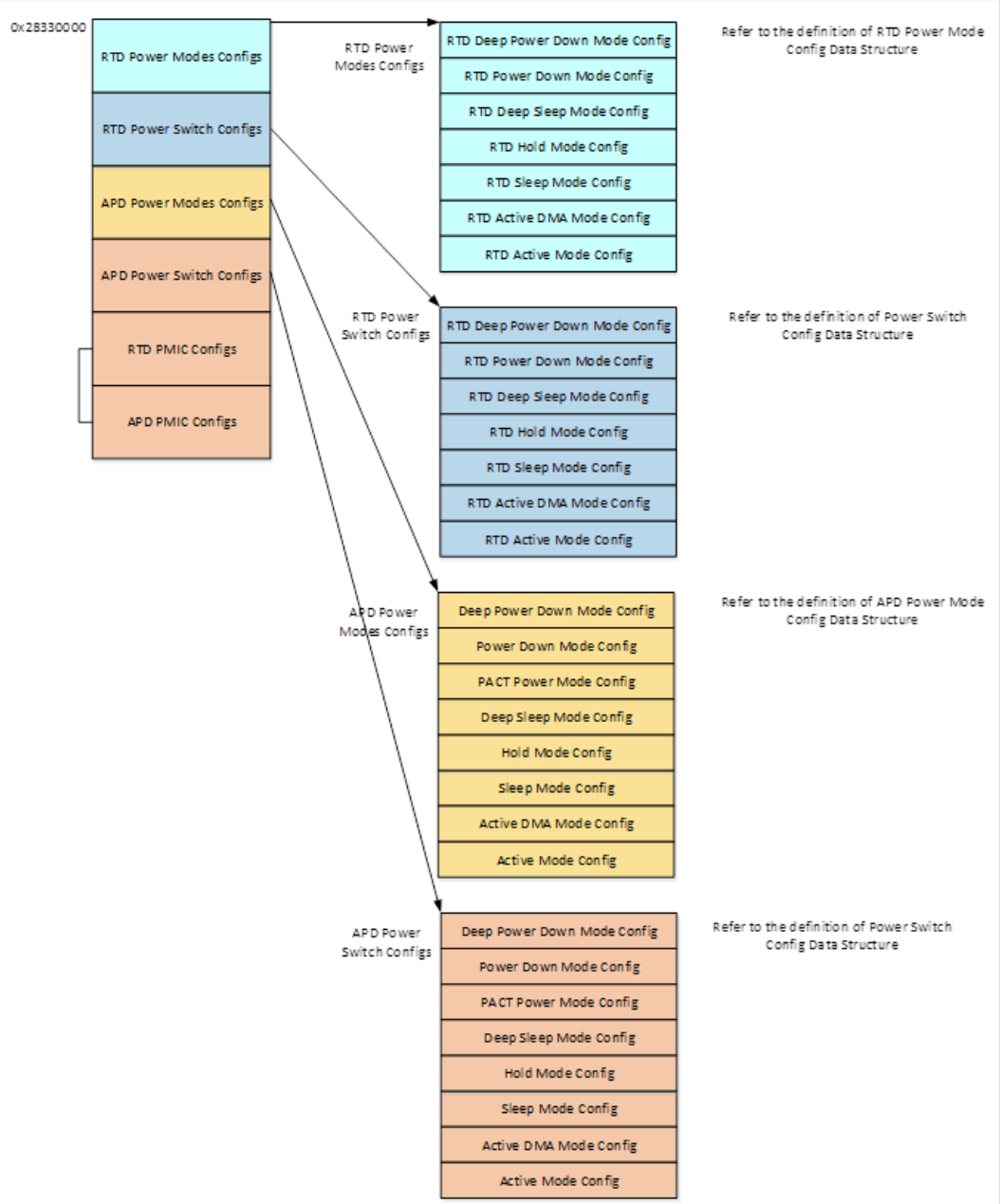

Figure 9. Power mode transition data structure

For RTD power mode configuration, see `upower_soc_defs.h`, struct `ps_pwr_mode_cfg_t->ps_rtd_swt_cfgs_t`.

Table 3. RTD power mode configuration

Field	Item	Definition	Size (byte)	Comment
`swt_board`	-	`swt_board` configuration pointer	4	The memory address of `ps_pwr_mode_cfg_t->ps_apd_pwr_mode_cfgs_t` (refer to Power Switch/Mem configuration)

Table 3. RTD power mode configuration*...continued*

Field	Item	Definition	Size (byte)	Comment
swt_mem	-	swt-mem configuration pointer	4	The memory address of ps_pwr_mode_cfg_t->ps_apd_pwr_mode_cfgs_t ->swt_mem (refer to Power Switch/Mem configuration)
in_reg_cfg	volt	Regulator voltage configuration	4	• Internal regulator configuration. • Refer to the definition of data structure upwr_reg_cfg_t. HW register: PMC.COREREG_CTRL.B.COREREGVL PMC.COREREG_CTRL.B.COREREGM
	mode	Regulator mode configuration	4	
pmic_cfg	volt	Regulator voltage configuration	4	• External regulator configuration. • Refer to the definition of data structure upwr_reg_cfg_t. The Mode[3:0] field in this register controls the PMIC signals: PMIC Stand-by Request, PMIC Mode[2], PMIC Mode[1], and PMIC Mode[0] respectively. HW register: SIC.AOPMICCR.R
	mode	Regulator mode configuration	4	
pad_cfg	pad_close	PMC PAD close configuration	4	• Pad configuration for power transision. • Refer to the definition of data structure upwr_pmc_pad_cfg_t. Refer to “PAD Configuration”. HW register: PMC.SYS_CTRL_PAD_0.PADRESET (4b) PMC.SYS_CTRL_PAD_0.PADCLOSE (4b) PMC.SYS_CTRL_PAD_1.TQ_SLEEP (2b)
	pad_reset	PMC PAD reset configuration	4	
	pad_tqsleep	PMC PAD TQ Sleep configuration	4	
mon_cfg	mon_hvd_en	PMC mon HVD	4	• Monitor configuration. • Refer to the definition of data structure upwr_pmc_mon_rtd_cfg_t. Actually, just use mon_hvd_en to represent the entire power monitor configuration. Do not use mon_lvd_en or mon_lvdlvl. Set them to **0**. Refer to “Power Monitor Configuration”. HW register: PMC.MON_CTRL_1 PMC.MON_INTC
	mon_lvd_en	PMC mon LVD	4	
	mon_lvdlvl	PMC mon LVDLVL	4	

Table 3. RTD power mode configuration*...continued*

Field	Item	Definition	Size (byte)	Comment
dombias_cfg	mode	Domain bias mode configuration	4	• Domain Bias configuration. • Refer to the definition of data structure UPWR_DOM_BIAS_CFG_T. HW register: PMC.DOMBIAS_DCTRL_RTD.MODE 2b (byte 0) PMC.DOMBIAS_DCTRL_AVD.MODE 2b (byte 2) PMC.DOMBIAS_DCTRL_RTD.RBBNWVL 5b (byte 0) PMC.DOMBIAS_DCTRL_AVD.RBBNWVL 1b (byte 2) PMC.DOMBIAS_DCTRL_RTD.RBBPWVL 5b (byte 0) PMC.DOMBIAS_DCTRL_AVD.RBBPWVL 1b (byte 2) (refer to Dombias Mode)
	rbbn	Reverse back bias N well	4	
	rbbp	Reverse back bias P well	4	
membias_cfg	en	Memory bias enable configuration	4	• Memory Bias configuration. • Refer to the definition of data structure UPWR_MEM_BIAS_CFG_T. HW register: PMC.MEMBIAS_DCTRL_0.MEMBIASEN (1b)
pwrsys_lpm_cfg	lpm_mode	Powersys low-power mode	4	• pwrsys low power configuration. • Refer to the definition of data structure upwr_powersys_cfg_t. Actually, do not use this item in the current i.MX 8ULP uPower.

For APD power mode configuration, see upower_soc_defs.h, struct ps_pwr_mode_cfg_t->ps_apd_pwr_mode_cfgs_t.

Table 4. APD power mode configuration

Field	Item	Definition	Size (byte)	Comment
swt_board_offs	-	swt_board configuration offset	4	The offset of ps_apd_swt_cfgs_t (refer to Power Switch/Mem configuration)
swt_mem_offs	-	swt-mem configuration offset	4	The offset of ps_apd_swt_cfgs_t->swt_mem (refer to Power Switch/Mem configuration)

Table 4. APD power mode configuration*...continued*

Field	Item	Definition	Size (byte)	Comment
pmic_cfg	volt	Voltage configuration	4	• External regulator configuration. • Refer to the definition of data structure upwr_pmic_cfg_t. The Mode[3:0] field in this register controls the PMIC signals: PMIC Stand-by Request, PMIC Mode[2], PMIC Mode[1], and PMIC Mode[0] respectively. Refer to vdetlvl in RM, SIC. AOPMICCR and PMIC IC SPEC.
	mode	Mode configuration	4	
	mode_msk	Mode mask	4	
pad_cfg	pad_close	PMC PAD close configuration	4	• Pad configuration for power transition. • Refer to the definition of data structure upwr_pmc_pad_cfg_t. HW register: PMC.SYS_CTRL_PAD_0.PADRESET (4b) PMC.SYS_CTRL_PAD_0.PADCLOSE (4b) Note: pad_tqsleep is reused for power monitor.
	pad_reset	PMC PAD reset configuration	4	
	pad_tqsleep	PMC PAD TQ Sleep configuration Actually, this field is reused to present power monitor.	4	
dombias_cfg	mode	Domain bias mode configuration	4	• Domain Bias configuration. • Refer to the definition of data structure UPWR_DOM_BIAS_CFG_T. HW register: PMC.DOMBIAS_DCTRL_APD.MODE 2b (byte 0) PMC.DOMBIAS_DCTRL_AVD.MODE 2b (byte 2) PMC.DOMBIAS_DCTRL_APD.RBBNWVL 1b (byte 0) PMC.DOMBIAS_DCTRL_APD.RBBNWVL 1b (byte 2) PMC.DOMBIAS_DCTRL_APD.RBBPWVL 1b (byte 0) PMC.DOMBIAS_DCTRL_APD.RBBPWVL 1b (byte 2) (refer to Dombias Mode)
	rbbn	Reverse back bias N well	4	
	rbbp	Reverse back bias P well	4	
membias_cfg	en	Memory bias enable configuration	4	• Memory Bias configuration. • Refer to the definition of data structure UPWR_MEM_BIAS_CFG_T. HW register: PMC.MEMBIAS_DCTRL_0.MEMBIASEN (1b)

9.4.1 Power Switch/Mem configuration

Refer to `upower_soc_defs.h`, struct `ps_pwr_mode_cfg_t->ps_rtd_swt_cfgs_t` and struct `ps_pwr_mode_cfg_t->ps_apd_swt_cfgs_t`, and refer to power switch and memory partition tables.

Table 5. Power Switch/Mem configuration

Field	Item	Definition	Size (byte)	Comment
`swt_board`	on	Switch on state, 1 bit per instance	4	• There is 1 instance of switch board. 1 instance is a set of 32 switches. See Section 9.4.4. • See data structure `upwr_switch_board_t`. • bit = 1 applies on bit HW register: `PMC.PSW_CTRL_1`
	mask	Actuation mask, 1 bit per instance	4	

`swt_mem`	array	RAM/ROM array state, 1 bit per instance	4	• There are 2 instances of switch memory. 1 instance is a set of 32 switches. See Section 9.4.5.
				• See data structure `upwr_mem_switches_t`.
				• bit = 1 applies on bit
				HW register:
				`PMC.SRAM_CTRL_ARRAY_0`
				`PMC.SRAM_CTRL_PERI_0`
				`PMC.SRAM_CTRL_ARRAY_1`
				`PMC.SRAM_CTRL_PERI_1`

	perif	RAM/ROM peripheral state, 1 bit per instance	4	
	mask	Actuation mask, 1 bit per instance	4	
	array	RAM/ROM array state, 1 bit per instance	4	
	perif	RAM/ROM peripheral state, 1 bit per instance	4	
	mask	Actuation mask, 1 bit per instance	4	

9.4.2 PAD configuration

Table 6. Pad configuration

Pad signal	Description	Usage	Bit correspondence
PADRESET	`PADRESETn` This register describes the behavior from PAD signal the PADRESET. 0000b – Pad reset n is released 0001b – Pad reset n is active.	`pad_reset`, when asserted, resets all this latching logic within the pad.	Bit 0: PTA, PTA_JTAG Bit 1: PTB Bit 2: PTE Bit 3: PTF

Table 6. Pad configuration*...continued*

Pad signal	Description	Usage	Bit correspondence
PADCLOSE	`PADCLOSEn` This register describes the behavior from PAD signal the PADCLOSE. 0000b – Pad n is not isolated 0001b – Pad n is isolated	`pad_close` helps to latch the IBE value in the pad – this is the one enabling the PAD.	Bit 0: PTA, PTA_ JTAG Bit 1: PTB Bit 2: PTE Bit 3: PTF
TQ_SLEEP	TQ_`SLEEPn` This register describes the behavior from PAD signal the TQ_SLEEP. 00b – Pad no change 01b – Pad n is in low power condition	`pad_tqsleep` should have a similar implementation for HSGPIOs as `pad_close`.	Bit 0: PTC Bit 1: PTD

Table 7. Pad reset

POR	Asserted with 1.8v Por and de-asserted after 1.1v > LVD.
Power Down	If configured, asserts before domain switch is opened. De-asserts after domain switch is closed.
Deep Power Down	If configured, asserts before domain switch is opened. De-asserts after domain switch is closed.

Table 8. Pad close

POR	Asserted with 1.8v Por and de-asserted after 1.1v > LVD.
Deep Sleep	If configured, asserts with mode entry. Exits with mode exit.
Power Down	• If `padreset` is configured: asserts beforemode enters. De-assests after mode exits. (For APD/AVD: If an indication that VDD_DIG1 is off, it is preferrable to use it). • If `padreset` is not configured: asserts before mode enters. De-assests after mode exits. (For APD/AVD: If an indication that VDD_ DIG1 is off, it is preferrable to use it) and application software writes to ISO_ACK. RTD: padreset[0] affects padclose[0]; padreset[1] affects padclose[1]. APD/AVD: padreset[2] affects padclose[2]; padreset[3] affects padclose[3].
Deep Power Down	• If padreset is configured: asserts beforemode enters. De-assests after mode exits. (For APD/AVD: If an indication that VDD_DIG1 is off, it is preferrable to use it). • If padreset is not configured: asserts beforemode enters. De-assests after mode exits. (For APD/AVD: If an indication that VDD_DIG1 is off, it is preferrable to use it) and application software writes to ISO_ACK. RTD: padreset[0] affects padclose[0]; padreset[1] affects padclose[1]. APD/AVD: padreset[2] affects padclose[2]; padreset[3] affects padclose[3].

Table 9. TQ_Sleep

Sleep	De-asserted.
Deep Sleep	If configured, asserts with mode entry. De-asserts with mode exit.
Power Down	If configured, asserts with mode entry. De-asserts with mode exit.
Deep Power Down	If configured, asserts with mode entry. De-asserts with mode exit.

9.4.3 Power monitor configuration

RTD: LVD1, HVD1

APD: LVD2, HVD2

AVD: LVD3, HVD3

LVD and HVD register description table

Table 10. Power monitor configuration

Item	Description	Value
HVD1_EN	RTD High Voltage Detector Enable	0b - HVD1 disabled 1b - HVD1 enabled
LVD1_EN	RTD Low Voltage Detector Enable	0b – LVD1 disabled 1b – LVD1 enabled
LVD1LVL	LVD1 Failing Trip Voltage	0b0000 - 0.720V <value> - 0.650V + <value>x25mV 0b1111 - 0.945V
HVD1IE	HVD1 Interrupt Enable	0b - Interrupt disabled 1b - Interrupt enabled
LVD1IE	LVD1 Interrupt Enable	0b - Interrupt disabled 1b - Interrupt enabled
HVD1RE	HVD1 Reset Enable	0b - Reset disabled 1b - Reset enabled
HVD1RE	LVD1 Reset Enable	0b - Reset disabled 1b - Reset enabled
HVD2_EN	APD High Voltage Detector Enable	0b – HVD2 disabled 1b – HVD2 enabled
LVD2_EN	APD Low Voltage Detector Enable	0b - LVD2 disabled 1b - LVD2 enabled
LVD2LVL	LVD2 Failing Trip Voltage	0b0000 - 0.720V <value> - 0.650V + <value>x25mV 0b1111 - 0.945V
HVD2IE	HVD2 Interrupt Enable	0b - Interrupt disabled 1b - Interrupt enabled
LVD2IE	LVD2 Interrupt Enable	0b - Interrupt disabled 1b - Interrupt enabled
LVD3LVL	LVD3 Failing Trip Voltage	0b0000 - 0.720V <value> - 0.650V + <value>x25mV 0b1111 - 0.945V

Table 10. Power monitor configuration*...continued*

Item	Description	Value
HVD3_EN	AVD High Voltage Detector Enable	0b – HVD2 disabled 1b – HVD2 enabled
LVD3_EN	AVD Low Voltage Detector Enable	0b - LVD3 disabled 1b - LVD3 enabled
HVD3IE	HVD3 Interrupt Enable	0b - Interrupt disabled 1b - Interrupt enabled
LVD3IE	LVD3 Interrupt Enable	0b - Interrupt disabled 1b - Interrupt enabled
HVD3RE	HVD3 Reset Enable	0b - Reset disabled 1b - Reset enabled
LVD3RE	LVD3 Reset Enable	0b - Reset disabled 1b - Reset enabled

The comment and code in `upower_soc_defs.h`:

```
/* LVD/HVD monitor config for a single domain */

/* Domain + AVD monitor config
 * For RTD, mapped in mon_cfg.mon_hvd_en
 * For APD, mapped temporarily in pad_cfg.pad_tqsleep
 */
typedef union upwr_mon_cfg_union_t {
  volatile uint32_t         R;
  struct {
  /* Original config, not change */
  volatile uint32_t          rsrv_1          : 8;
  /* DOM */
  volatile uint32_t          dom_lvd_irq_ena : 1;
  volatile uint32_t          dom_lvd_rst_ena : 1;
  volatile uint32_t          dom_hvd_irq_ena : 1;
  volatile uint32_t          dom_hvd_rst_ena : 1;
  volatile uint32_t          dom_lvd_lvl     : 4;
  volatile uint32_t          dom_lvd_ena     : 1;
  volatile uint32_t          dom_hvd_ena     : 1;
  /* AVD */
  volatile uint32_t          avd_lvd_irq_ena : 1;
  volatile uint32_t          avd_lvd_rst_ena : 1;
  volatile uint32_t          avd_hvd_irq_ena : 1;
  volatile uint32_t          avd_hvd_rst_ena : 1;
  volatile uint32_t          avd_lvd_lvl     : 4;
  volatile uint32_t          avd_lvd_ena     : 1;
  volatile uint32_t          avd_hvd_ena     : 1;
 }                          B;
} upwr_mon_cfg_t;
```

If Cortex-M33 configures power monitor during RTD power mode transition, `dom_lvd*` and `dom_hvd*` represent RTD (LVD1 and HVD1), and `avd_lvd*` and `avd_hvd*` represent AVD (LVD3 and HVD3). Only when RTD owns AVD, it takes effect.

If Cortex-A35 configures power monitor during APD power mode transition, `dom_lvd*` and `dom_hvd*` represent APD (LVD2 and HVD2), and `avd_lvd*` and `avd_hvd*` represent AVD (LVD3 and HVD3). Only when APD owns AVD, it takes effect.

For example, in RTD active mode configuration, configure the value to **0xdeb3a00**. The following table lists the meanings of the binary values: 1101 1110 1011 0011 1010 0000 0000.

Table 11. Binary values

Structure Field	Value	Description
rsrv_1	0000 0000	Reserved
dom_lvd_irq_ena	0	Disables RTD LVD interrupt
dom_lvd_rst_ena	1	Enables RTD LVD reset
dom_hvd_irq_ena	0	Disables RTD HVD interrupt
dom_hvd_rst_ena	1	Enables RTD HVD reset
dom_lvd_lvl	0011	RTD LVD1 Failing Trip Voltage
dom_lvd_ena	1	Enables HVD for RTD
dom_hvd_ena	1	Enables LVD for RTD
If RTD owns AVD, apply the following AVD configuration		
avd_lvd_irq_ena	0	Disables AVD LVD interrupt
avd_lvd_rst_ena	1	Enables AVD LVD reset
avd_hvd_irq_ena	0	Disables AVD HVD interrupt
avd_hvd_rst_ena	1	Enables AVD HVD reset
avd_lvd_lvl	0111	AVD LVD3 Failing Trip Voltage
avd_lvd_ena	1	Enables LVD for AVD
avd_hvd_ena	1	Enables HVD for AVD

9.4.4 Power switches

This power switches table is also included in the i.MX 8ULP Reference Manual.

Table 12. Power switches

Function	Logical Power Switch	Domain	POR
RTD (A)	PS0	RTD	Enabled
RTD (B)			
Fusion	PS1		Disabled
A35[0] Core	PS2	APD	Disabled
A35[1] Core	PS3		
Mercury L2 Cache [1]	PS4		
Fast NIC / Mercury	PS5		
APD Periph.	PS6		
GPU3D	PS7	AVD	Disabled
HiFi4	PS8		
DDR Controller	PS9,10,11,12		

Table 12. Power switches*...continued*

Function	Logical Power Switch	Domain	POR
PXP, EPDC	PS13		
MIPI DSI	PS14		
MIPI CSI	PS15		
NIC AV / Periph.	PS16		
Fusion AO	PS17	RTD	Disabled
FUSE	PS18		Enabled
uPower	PS19	RTD/uPower	Disabled

9.4.5 Memory partitions

This memory partitions table is also included in the i.MX 8ULP Reference Manual.

Table 13. Memory partitions

Partition #	IOs Supply Source	Memory Cell Suply Source	Memories controlled	Domain	POR
0	PS2	PS2	CA35 Core 0 L1 cache	Application	Disabled
1	PS3	PS3	CA35 Core 1 L1 cache		
2	PS5	PS5	L2 Cache 0		
3	PS4	PS4	L2 Cache 1		
4	PS5	PS5	L2 Cache victim/tag		
5	PS5	PS5	CAAM Secure RAM		
6	PS6	PS6	DMA1 RAM		
7	PS6	PS6	FlexSPI2 FiFO, Buffer		
8	PS6	PS6	SRAM0		
9	PS6	PS6	AD ROM		
10	PS5	VDD_DIG1	USB0 TX/RX RAM		
11	PS6	VDD_DIG1	uSDHC0 FIFO RAM		
12	PS6	PS6	uSDHC1 FIFO RAM		
13	PS5	PS5	uSDHC2 FIFO and USB1 TX/RX RAM		
14	PS6	PS6	GIC RAM		
15	PS5	PS5	ENET TX FIXO		
16	PS6	VDD_DIG1	Brainshift		
17	PS13	PS13	DCNano Tile2Linear and RGB Correction	Audio-Video	Disabled
18	PS13	PS13	DCNano Cursor and FIFO		
19	PS10	PS10	EPDC LUT		
20	PS10	PS10	EPDC FIFO		

Table 13. Memory partitions*...continued*

Partition #	IOs Supply Source	Memory Cell Suply Source	Memories controlled	Domain	POR
21	PS13	PS13	DMA2 RAM		
22	PS13	PS13	GPU2D RAM Group 1		
23	PS13	PS13	GPU2D RAM Group 2		
24	PS7	PS7	GPU3D RAM Group 1		
25	PS7	PS7	GPU3D RAM Group 2		
26	PS8	PS8	HiFi4 Caches, IRAM, DRAM		
27	PS13	PS13	ISI Buffers		
28	PS13	PS13	MIPI-CSI FIFO		
29	PS13	PS13	MIPI-DSI FIFO		
30	PS10	PS10	PXP Caches, Buffers		
31	PS13	PS13	SRAM1		

Table 14. Memory partitions

Partition #	IOs Supply Source	Memory Cell Suply Source	Memories controlled	Domain	POR
32	PS0	PS0	Casper RAM	Real-Time	Disabled
33	PS0	PS0	DMA0 RAM		
34	PS0	PS0	FlexCAN RAM		
35	PS0	PS0	FlexSPI0 FIFO, Buffer		Enabled
36	PS0	PS0	FlexSPI1 FIFO, Buffer		Disabled
37	PS0	PS0	CM33 Cache Cache		Enabled
38	PS0	PS0	PowerQuad RAM		Disabled
39	PS0	PS0	M33-ETF RAM		
40	PS0	PS0	Sentinel PKC, Data RAM1, Inst RAM 0/1		Enabled
41	PS0	PS0	Sentinel ROM		
42	PS16/uPower	PS16/uPower	uPower IRAM/DRAM		Enabled
43	PS16/uPower	PS16/uPower	uPower ROM		
44	PS0	PS0	CM33 ROM		Enabled
45	PS0	VDD_DIG0	SSRAM Partition 0		
46	PS0	VDD_DIG0	SSRAM Partition 1		
47	PS0	VDD_DIG0	SSRAM Partition 2,3,4		
48	PS0	VDD_DIG0	SSRAM Partition 5		
49	PS0	VDD_DIG0	SSRAM Partition 6		

Table 14. Memory partitions*...continued*

Partition #	IOs Supply Source	Memory Cell Suply Source	Memories controlled	Domain	POR
50	PS0	VDD_DIG0	SSRAM Partition 7_a (128KB)		
51	PS0	VDD_DIG0	SSRAM Partition 7_b (64KB)		
52	PS0	VDD_DIG0	SSRAM Partition 7_c (64KB)		
53	PS0	VDD_DIG0	Sentinel Data RAM0, Inst RAM2		

9.4.6 RTD and APD power mode PMIC configuration

User (Arm Cortex-M33/A35) can configure PMIC IC during power mode transition. For example, turn on/off one regulator, adjust the voltage of one regulator…

There are two situations:

1. RTD/APD enters lower power mode: uPower first operates power switches, memory partitions, pads, etc., and then configures PMIC.
2. RTD/APD exits lower power mode: uPower first configures PMIC, and then operates power switches, memory partitions, pads, etc.

For the sample code, see Section 11.3.

9.4.7 DOMBIAS mode

Users can configure RBB or AFBB dombias mode during power mode transition.

9.4.7.1 RTD/APD configuration RBB mode

Users can configure power mode data to enable/disable RBB mode during power mode transition. See `upower_soc_defs.h` and read the comments.

- RTD power mode transition: users can configure `ps_pwr_mode_cfg_t->ps_rtd_pwr_mode_cfgs_t->upwr_pmc_bias_cfg_t->UPWR_DOM_BIAS_CFG_T->mode` to set/disable RBB mode. uPower sets/disables RTD dombias to RBB mode. If AVD is owned by RTD, uPower also sets/disables AVD dombias to RBB mode. Configure `ps_pwr_mode_cfg_t->ps_rtd_pwr_mode_cfgs_t->upwr_pmc_bias_cfg_t->UPWR_DOM_BIAS_CFG_T->rbbn` to adjust RBBN (reverse back bias N well mV) or `ps_pwr_mode_cfg_t->ps_rtd_pwr_mode_cfgs_t->upwr_pmc_bias_cfg_t->UPWR_DOM_BIAS_CFG_T->rbbp` to adjust RBBP (reverse back bias P well mV).
- APD power mode transition: users can configure `ps_pwr_mode_cfg_t->ps_apd_pwr_mode_cfgs_t->upwr_pmc_bias_cfg_t->UPWR_DOM_BIAS_CFG_T->mode` to set/disable RBB mode. uPower sets/disables APD dombias to RBB mode. If AVD is owned by APD, uPower also sets/disables AVD dombias to RBB mode. Configure `ps_pwr_mode_cfg_t->ps_apd_pwr_mode_cfgs_t->upwr_pmc_bias_cfg_t->UPWR_DOM_BIAS_CFG_T->rbbn` to adjust RBBN (reverse back bias N well mV) or `ps_pwr_mode_cfg_t->ps_apd_pwr_mode_cfgs_t->upwr_pmc_bias_cfg_t->UPWR_DOM_BIAS_CFG_T->rbbp` to adjust RBBP (reverse back bias P well mV).

Table 15. Valid biasing options on i.MX 8ULP

Cortex-A35/LPAV Domain	Cortex-M33 Domain	Valid i.MX 8ULP use-case	Application Power mode
No Biasing	No Biasing	YES	No Restriction
AFBB	No Biasing	YES	Cortex-A35 and Cortex-M33 Active modes
AFBB	AFBB	YES	Cortex-A35 and Cortex-M33 Active modes
AFBB	RBB	NO	N/A
Power Gated	AFBB	YES	Cortex-M33 (Active)
Power Gated	RBB	YES	Cortex-M33 (Active, Sleep, Deep Sleep, Power Down)
RBB	No Biasing	YES	Cortex-A35 (Sleep/Deep Sleep), Cortex-M33 (Active)
RBB	AFBB	NO	N/A
RBB	RBB	YES	Cortex-A35 (Sleep, Deep Sleep), Cortex-M33 (All modes)

9.4.7.2 RTD/APD configuration AFBB mode

Users can configure power mode data to enable/disable AFBB mode during power mode transition. See `upower_soc_defs.h` and read the comments.

- RTD power mode transition: users can configure `ps_pwr_mode_cfg_t->ps_rtd_pwr_mode_cfgs_t->upwr_pmc_bias_cfg_t->UPWR_DOM_BIAS_CFG_T->mode` to set/disable AFBB mode. uPower sets/disables RTD dombias to AFBB mode and sets/disables RTD SRAM AFBB. If AVD is owned by RTD, uPower also sets/disables AVD dombias to AFBB mode and sets/disables AVD SRAM AFBB.
- APD power mode transition: users can configure `ps_pwr_mode_cfg_t->ps_apd_pwr_mode_cfgs_t->upwr_pmc_bias_cfg_t->UPWR_DOM_BIAS_CFG_T->mode` to set/disable AFBB mode. uPower sets/disables APD dombias to AFBB mode and sets/disables APD SRAM AFBB. If AVD is owned by APD, uPower also sets/disables AVD dombias to AFBB mode and sets/disables AVD SRAM AFBB.

For detailed SRAM memory partitions, see Section 9.4.5.

- To enable RTD AFBB mode, uPower sets RTD SRAM_AFBB value to **0x3FE120**. To disable, it sets RTD SRAM_AFBB value to **0x0**.
- To enable APD AFBB mode, uPower sets APD SRAM_AFBB value to **0x1012C**. To disable, it sets RTD SRAM_AFBB value to **0x0**.
- To enable AVD AFBB mode, uPower sets AVD SRAM_AFBB value to **0x80080000**. To disable, it sets RTD SRAM_AFBB value to **0x0**.

9.4.7.3 Default SRAM AFBB values on the A1 chip

- The default SRAM AFBB values for RTD: **0x3FE120**
- The default SRAM AFBB values for APD: **0x1012C**
- The default SRAM AFBB values for AVD: **0x80080000**

10 PMIC

On i.MX 8ULP, Cortex-M33/A35 makes the uPower core control the PMIC by calling the uPower API or configuring the PMIC register data for power mode transition. This chapter describes the APIs provided by uPower to the Cortex-M33/A35, and how to migrate the PMIC driver if customers want to replace another PMIC IC.

10.1 PCA9460 on the EVK board

PMIC rails	max current	voltage	power on seq.	power on seq.	power off seq.	power off seq.	8ULP power rails	Peripherals power rails
LDO_SNVS	10mA	3.0V	0	NA	0	NA	VDD_VBAT42	
BUCK1	1000mA	1.8V	1	T0+18mS	1	T1+120mS	VDD_PMC18, VDD_PLL18, VDD_FUSE18, VDD18_IOREF, VDD_PTB, VDD_ANA18, VREF_ANA18, VDD_PMC18_DIG0 (default Not Connected)	LPDDR4(x).VDD1 SPI Nor.VCC(FlexSPI0) SPI Nor/Nand.VCC (FlexSPI2) pSRAM.VCC(FlexSPI1) EMMC.VCCQ
LDO2	250mA	3.3V	2	T0+20mS	2	T1+112mS	VDD_PTA, VDD_ANA33, VDD_USB_33	
LSW4	100mA	1.8V	2	T0+20mS	2	T1+112mS	VDD_PTE/F	
BUCK2	1000mA	1.0V	3	T0+22mS	3	T1+104mS	VDD_DIG0	
LSW2	100mA	1.8V	3	T0+22mS	3	T1+104mS	VDD_USB_18	
BUCK3	1000mA	1.0V	4	T0+24mS	4	T1+96mS	VDD_DIG1, VDD_DIG2, VDD_DDR_PLL, MIPI_DSI_1V1, CSI_1V1	
BUCK4	1000mA	1.1V	4	T0+24mS	4	T1+96mS	VDDQX_AO	LPDDR4(x).VDD2
LDO3	250mA	3.3V	5	T0+26mS	5	T1+88mS		EMMC.VCC
LDO4	250mA	1.8V	5	T0+26mS	5	T1+88mS	VDD_PTD	
LSW1	100mA	1.8V	6	T0+28mS	6	T1+80mS	VDD_PTC	
EX_LDO	500mA	0.6V	7	T0+30mS	7	T1+72mS	VDDQ (LPDDR4x)	LPDDR4x.VDDQ
LDO1	250mA	1.1V	7	T0+30mS	7	T1+72mS	VDDQ (LPDDR4), VDDQX	LPDDR4.VDDQ
LSW3	100mA	1.8V	8	T0+32mS	8	T1+64mS	MIPI_DSI/CSI_1V8	

For detailed information about PCA9460, see PCA9460 SPEC.

The rail ID definition of PCA9460 in the uPower firmware driver:

```
#define PMIC_BUCK1 0U
#define PMIC_BUCK2 1U
#define PMIC_BUCK3 2U
#define PMIC_BUCK4 3U
#define PMIC_LDO1 4U
#define PMIC_LDO2 5U
#define PMIC_LDO3 6U
#define PMIC_LDO4 7U
#define PMIC_LSW1 8U
#define PMIC_LSW2 9U
#define PMIC_LSW3 10U
#define PMIC_LSW4 11U
```

User Cortex-M33/A35 application must use the same ID.

10.2 PMIC API for user Cortex-M33/A35

There are several uPower APIs for user Cortex-M33/A35 to configure or control PMIC.

Table 16. PMIC API for user Cortex-M33/A35

API prototype	Description
`int upwr_vtm_pmic_cold_reset(upwr_callb callb)`	Cold reset the PMIC. PMIC will power cycle all the regulators.

Table 16. PMIC API for user Cortex-M33/A35*...continued*

API prototype	Description
`int upwr_vtm_set_pmic_mode(uint32_t pmic_mode, upwr_callb callb)`	i.MX 8ULP SOC has four mode pins, and PCA9460 can support 8 normal modes and standby mode. For details, see the i.MX 8ULP Reference Manual and PCA9460 SPEC.
`int upwr_vtm_get_pmic_voltage( uint32_t rail, upwr_callb callb)`	Gets the voltage of the given regulator.
`int upwr_vtm_chng_pmic_voltage( uint32_t rail, uint32_t volt, upwr_ callb callb)`	Adjusts the voltage of the given regulator.
`int upwr_vtm_pmic_config(const void* config, uint32_t size, upwr_ callb callb)`	Configures PMIC IC in runtime. See Section 11.2.

10.3 Porting PMIC driver

10.3.1 Preparing the uPower software build environment

The porting kit package builds on a Linux host machine. Users need to install the RISC-v GCC (8.3.0) toolchain.

Users can download opensource code from GitHub and build toolchain. The steps are as follows:

```
#!/bin/sh

# on ubuntu, executing the following command should suffice
sudo apt-get install autoconf automake autotools-dev curl
 python3 libmpc-dev libmpfr-dev libgmp-dev gawk build-essential
 bison flex texinfo gperf libtool patchutils bc zlib1g-dev
 libexpat-dev

# On Fedra/CentOS/RHEL OS, executing the following command
 should suffice:
sudo yum install autoconf automake python3 libmpc-devel mpfr-
devel gmp-devel gawk  bison flex texinfo patchutils gcc gcc-c++
 zlib-devel expat-devel

# On Arch Linux, executing the following command should
 suffice:
sudo pacman -Syyu autoconf automake curl python3 libmpc
 mpfr gmp gawk base-devel bison flex texinfo gperf libtool
 patchutils bc zlib expat


git clone https://github.com/riscv/riscv-gnu-toolchain
cd riscv-gnu-toolchain

git clone --recursive https://github.com/riscv/riscv-qemu.git
git clone --recursive https://github.com/riscv/riscv-newlib.git
git clone --recursive https://github.com/riscv/riscv-binutils-
gdb.git
git clone --recursive https://github.com/riscv/riscv-
dejagnu.git
```

```
git clone --recursive https://github.com/riscv/riscv-glibc.git
git clone --recursive https://github.com/riscv/riscv-gcc.git

rm qemu -rf
rm newlib -rf
rm riscv-binutils -rf
rm riscv-gdb -rf
rm glibc -rf

mv riscv-qemu qemu
mv riscv-newlib newlib
tar zcvf riscv-binutils-gdb.tar.gz riscv-binutils-gdb
mv riscv-binutils-gdb riscv-binutils
tar zxvf riscv-binutils-gdb.tar.gz
mv riscv-binutils-gdb riscv-gdb
rm riscv-binutils-gdb.tar.gz
mv riscv-glibc glibc

cd glibc
git checkout riscv-glibc-2.31
cd ..

cd riscv-gcc
git checkout riscv-gcc-8.3.0
cd ..

# please checkout to these commit
# riscv-gnu-toolchain commit id:
 409b951ba6621f2f115aebddfb15ce2dd78ec24f
# newlib commit id: f289cef6be67da67b2d97a47d6576fa7e6b4c858
# glibc commit id: 07305b3f41effdba4a1bdcae50af17188aae3fa2
# riscv-gcc commit id: bdf3ad8996cb305a822feb1eb11235e08fe4b974
# riscv-binutils commit id:
 20756b0fbe065a84710aa38f2457563b57546440
# risv-gdb commit id: 20756b0fbe065a84710aa38f2457563b57546440
# riscv-dejagnu commit id:
 4ea498a8e1fafeb568530d84db1880066478c86b

./configure --prefix=/home/nxf55768/riscv --with-arch=rv32emc
 --with-abi=ilp32e
cd riscv-gnu-toolchain
make
```

After the user downloads the source code and builds the toolchain, this method can be used to verify whether the built toolchain is the same as that used by NXP: build uPower firmware (with default NXP P9460 PMIC driver), and compare the MD5 value with the uPower firmware image of NXP official release.

NXP release a package includes the PCA9460 driver source code and basic uPower firmware static library. If the user uses other PMIC chips, it is not difficult to port the PMIC driver.

Here are the steps:

1. Modify Makefile, `Makefile_combine_upower_fw_pmic`, `Makefile_pmic_lib`, and `pmicdrv/Makefile`, change `CROSS_COMPILE ?= /pkg/OSS-riscv-/rv32emc/x86_64-linux/bin/riscv32-unknown-elf-` to the correct RISC-v GCC compiler toolchain path.

2. Write your own PMIC drivers, and replace PMICDRV source code file. Users must implement the APIs in `pmic_model.h` so that uPower firmware calls these APIs.
3. Run build command: `make clean;make`, and then generate uPower firmware image binary `upower_fw.bin`.

Implement APIs in `pmic_model.h`.

Table 17. APIs in pmic_model.h

API prototype	Description
`void pmic_init(void)`	Do necessary initialization work, and assign I2C address to global variable `upwr_pmic_cfg.i2c_addr`. uPower firmware calls this API to initialize PMIC driver.
`int pmic_get_rail_voltage(uint8_t rail, uint8_t mode)`	Gets voltage of the given rail at given mode.
`int pmic_set_rail_voltage(uint8_t rail, uint8_t mode, int voltage)`	Sets voltage of the given rail at given mode.
`void configure_pmic_ic_iomux(void)`	Configures PMIC chip IOMUX.
`int pmic_config(void *pmic_config_data)`	Configures PMIC chip. See the sample code of PCA9460 driver.
`int pmic_cold_reset(void)`	Triggers PMIC cold reset.

Note: *Rail and mode are from user's PMIC specifications. The uPower firwmare passes the rail and mode parameters from uPower API call to get/set rail functions.*

See the NXP PCA9460 driver.

uPower firmware static library provides two interfaces for porting the PMIC driver.

Table 18. Interfaces for porting the PMIC driver

API prototype	Description
`int upwr_pmic_read_register_buf(uint32_t reg, uint8_t* buf, size_t len)`	Reads bytes following a register from PMIC and stores them into buffer.
`int upwr_pmic_write_register_buf(uint32_t reg, uint8_t* buf, size_t len)`	Writes a register address and bytes in buf to PMIC.

11 Appendix

11.1 uPower API initialization sample code

```
#if (defined(__ARM_FEATURE_CMSE) && (__ARM_FEATURE_CMSE & 0x2))
#define POWERSYS_MUA_RTD_BASE (0x38029000u)
#else
#define POWERSYS_MUA_RTD_BASE (0x28029000u)
#endif

#define POWERSYS_MUA_RTD ((MU_Type *)(POWERSYS_MUA_RTD_BASE))

#define UPOWER_MU ((struct MU_tag *)(POWERSYS_MUA_RTD_BASE))
```

```
/************************ Variables
 *****************************/
static upwr_isr_callb s_muTxRxHandler;
static upwr_isr_callb s_muNmiHandler;

static volatile bool callbackStatus = false;
/************************ Prototypes
 *****************************/

/************************ Code
 ***********************************/
void uPower_IRQHandler(void)
{
    if ((POWERSYS_MUA_RTD->CSSR0 & MU_CSSR0_NMIC_MASK) != 0U)
    {
        /* Clear NMI */
        POWERSYS_MUA_RTD->CSSR0 = MU_CSSR0_NMIC_MASK;
        if (s_muNmiHandler != NULL)
        {
            s_muNmiHandler();
        }
        else
        {
        }
    }
    else
    {
        assert(s_muTxRxHandler);
        s_muTxRxHandler();
    }
}

static void UPOWER_DummyInstallISR(upwr_isr_callb txrx,
 upwr_isr_callb excp)
{
    s_muTxRxHandler = txrx;
    s_muNmiHandler  = excp;
}

static void UPOWER_LockMuInt(int lock)
{
    if (lock != 0)
    {
        NVIC_DisableIRQ(uPower_IRQn);
    }
    else
    {
        NVIC_EnableIRQ(uPower_IRQn);
    }
}

static void UPOWER_Ready(uint32_t soc, uint32_t vmajor,
 uint32_t vminor)
{
    callbackStatus = true;

    (void)soc;
    (void)vmajor;
    (void)vminor;
```

```
}

static void UPOWER_Callback(upwr_sg_t sg, uint32_t func,
 upwr_resp_t errCode, int ret)
{
    callbackStatus = true;
}

/*!
 * @brief Check status of current request to uPower.
 *
 * Will block till related callback function has been called
 *
 * @sg Indicate which Service Group that request point to
 */
void UPOWER_CheckReq(upwr_sg_t sg)
{
    upwr_req_status_t reqStatus;

    /* wait callback */
    while (!callbackStatus)
    {
    }

    callbackStatus = false;

    /* Get reply from upower */
    reqStatus = upwr_poll_req_status(sg, NULL, NULL, NULL, 0U);
    if (reqStatus != UPWR_REQ_OK)
    {
        assert(false);
    }
}

/*!
 * @brief Initialize MU interface for uPower access.
 *
 * @param pVersion Pointer to the structure to save uPower ROM
 and RAM versions
 */
void UPOWER_Init(upower_version_t *pVersion)
{
    int status;
    uint32_t soc;
    uint32_t major, minor, fixes;

    CLOCK_EnableClock(kCLOCK_UpowerMuARtd);

    status = upwr_init(RTD_DOMAIN, UPOWER_MU, NULL, NULL,
 UPOWER_DummyInstallISR, UPOWER_LockMuInt);
    if (status != 0)
    {
        assert(false);
    }

    NVIC_EnableIRQ(uPower_IRQn);

    soc = upwr_rom_version(&major, &minor, &fixes);
    if (soc == 0U)
    {
```

```
        assert(false);
    }

    if (pVersion != NULL)
    {
        pVersion->romMajor = major;
        pVersion->romMinor = minor;
        pVersion->romFixes = fixes;
    }

    status = upwr_start(1U, UPOWER_Ready);
    if (status != 0)
    {
        assert(false);
    }

    UPOWER_CheckReq(UPWR_SG_EXCEPT);

    major = upwr_ram_version(&minor, &fixes);

    if (pVersion != NULL)
    {
        pVersion->ramMajor = major;
        pVersion->ramMinor = minor;
        pVersion->ramFixes = fixes;
    }
}
```

11.2 PMIC IC configuration sample code

```
int upower_pmic_config(void)
{
    struct pmic_config_struct pmic_config_struct_data;

    /* The TAG indicate it is valid configuration data */
    pmic_config_struct_data.cfg_tag = PMIC_CONFIG_TAG;
    /* Configure 5 register-data mapping */
    pmic_config_struct_data.cfg_reg_size = 5;

    /* the register value and data value, please refer to PMIC
 SPEC */
    pmic_config_struct_data.reg_addr_data_array[0].reg = 0x31 ;
    pmic_config_struct_data.reg_addr_data_array[0].data = 0x83;
    pmic_config_struct_data.reg_addr_data_array[1].reg = 0x36;
    pmic_config_struct_data.reg_addr_data_array[1].data = 0x31;
    pmic_config_struct_data.reg_addr_data_array[2].reg = 0x38;
    pmic_config_struct_data.reg_addr_data_array[2].data = 0x12;
    pmic_config_struct_data.reg_addr_data_array[3].reg = 0x43;
    pmic_config_struct_data.reg_addr_data_array[3].data = 0x10;
    pmic_config_struct_data.reg_addr_data_array[4].reg = 0x40;
    pmic_config_struct_data.reg_addr_data_array[4].data = 0x10;

    /* calculate the total size (Bytes) of configuration data
 */
    int size = 4 + 4 + pmic_config_struct_data.cfg_reg_size * 2
 * 4;

    return upower_pwm_chng_pmic_config((void
 *)&pmic_config_struct_data, size);
```

```
}
```

11.3 Power mode PMIC configuration sample code

The configuration data definition, in `upower_soc_defs.h`:

```
#define PMIC_REG_VALID_TAG 0xAAU

/**
* limit the max pmic register->value count to 8
* each data cost 4 Bytes, totally 32 Bytes
*/
#define MAX_PMIC_REG_COUNT 0x8U

/**
* the configuration structure for PMIC register setting
*
* @ tag: The TAG number to judge if the data is valid or not,
 valid tag is PMIC_REG_VALID_TAG
* @ power_mode : corresponding to each domain's power mode
* RTD refer to upwr_ps_rtd_pwr_mode_t
* APD refer to abs_pwr_mode_t
* @ i2c_addr : i2c address
* @ i2c_data : i2c data value
*/
struct ps_pmic_reg_data_cfg_t{
    uint32_t tag : 8;
    uint32_t power_mode : 8;
    uint32_t i2c_addr : 8;
    uint32_t i2c_data : 8;
};

typedef struct ps_rtd_pwr_mode_cfg_t
 ps_rtd_pwr_mode_cfgs_t[NUM_RTD_PWR_MODES];
typedef struct ps_apd_pwr_mode_cfg_t
 ps_apd_pwr_mode_cfgs_t[NUM_APD_PWR_MODES];
typedef struct ps_pmic_reg_data_cfg_t
 ps_rtd_pmic_reg_data_cfgs_t[MAX_PMIC_REG_COUNT];
typedef struct ps_pmic_reg_data_cfg_t
 ps_apd_pmic_reg_data_cfgs_t[MAX_PMIC_REG_COUNT];

struct ps_pwr_mode_cfg_t {
         ps_rtd_pwr_mode_cfgs_t  ps_rtd_pwr_mode_cfg;
         ps_rtd_swt_cfgs_t       ps_rtd_swt_cfg;
         ps_apd_pwr_mode_cfgs_t  ps_apd_pwr_mode_cfg ;
         ps_apd_swt_cfgs_t       ps_apd_swt_cfg;
    ps_rtd_pmic_reg_data_cfgs_t ps_rtd_pmic_reg_data_cfg;
    ps_apd_pmic_reg_data_cfgs_t ps_apd_pmic_reg_data_cfg;
};
```

The sample code to configure PMIC in RTD/APD power mode data are as follows. These sample code are for PCA9460 PMIC IC. For the definition of I2C address and data, see PCA9460 SPEC.

```
void set_rtd_pmic_register(void)
{
    volatile struct ps_pwr_mode_cfg_t *p_ps_pwr_mode_cfg =
 (volatile struct ps_pwr_mode_cfg_t *)0x28330000U;
```

```
    /* configure BUCK2 power off for power down */
    p_ps_pwr_mode_cfg->ps_rtd_pmic_reg_data_cfg[0].tag =
 PMIC_REG_VALID_TAG;
    p_ps_pwr_mode_cfg->ps_rtd_pmic_reg_data_cfg[0].power_mode =
 DPD_RTD_PWR_MODE;
    p_ps_pwr_mode_cfg->ps_rtd_pmic_reg_data_cfg[0].i2c_addr =
 0x37;
    p_ps_pwr_mode_cfg->ps_rtd_pmic_reg_data_cfg[0].i2c_data =
 0x10;
}

void set_apd_pmic_register(void)
{
    volatile struct ps_pwr_mode_cfg_t *p_ps_pwr_mode_cfg =
 (volatile struct ps_pwr_mode_cfg_t *)0x28330000U;

    /* configure LDO1 power off for power down */
    p_ps_pwr_mode_cfg->ps_apd_pmic_reg_data_cfg[0].tag =
 PMIC_REG_VALID_TAG;
    p_ps_pwr_mode_cfg->ps_apd_pmic_reg_data_cfg[0].power_mode =
 PD_PWR_MODE;

    /* LDO1_CFG (0x30) */
    p_ps_pwr_mode_cfg->ps_apd_pmic_reg_data_cfg[0].i2c_addr =
 0x30;

    /**
LDO1 Enable mode
00b = OFF
01b = ON at RUN State
10b = ON at ACTIVE mode or STANDBY mode, OFF at DPSTANDBY
11b = ON at ACTIVE mode, OFF at STANDBY or DPSTANDBY
     */
    p_ps_pwr_mode_cfg->ps_apd_pmic_reg_data_cfg[0].i2c_data =
 0x9C;
```

12 Revision History

Table 19. Revision history

Revision number	Date	Substantive changes
0	11/2022	Initial release

13 Legal information

13.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

13.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

13.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com

Date of release: 30 November 2022
Document identifier: UPOWERUG