life.augmented

AN5726

Application note

STM32MP2 MPUs using low-power modes

Introduction

The STM32MP25x microprocessor devices are built on an Arm® Cortex®-A35 with single or dual-core CPU subsystem, called CPU1 combined with an Arm® Cortex®-M33 MCU, called CPU2.

The STM32MP25x microprocessor devices are referred to as STM32MP2 devices from now on.

The STM32MP2 devices are dynamically configurable to adapt the power consumption to the most appropriate power mode. The objective is to maintain the power consumption to the most efficient level.

This application note explains:

- The various low-power modes of STM32MP2 devices
- How to configure them
- How to exit from them.

This document gives guidelines on how to use low-power modes at the system level. It also presents guidelines when using an external STPMIC2 power-regulator component.

The full list of applicable products is given in Table 2.

1 General information

This document applies to the STM32MP2 Lines dual-core Arm®-based Series microprocesor.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

1.1 Glossary

The following table contains a non-exhaustive list of terms used in this document.

Table 1. Glossary

Term	Meaning
AHB	Advanced high-performance bus
AVD	Analog voltage detector
BKPSRAM	Backup SRAM
BOR	Brownout reset
BUCK	Step-down switched-mode power-supply converter
CSS	Clock security system
DDR	Double data rate (SRAM)
DDRCTRL	DDR controller
DDRPHYC	DDR physical interface control
ETH	Ethernet controller
EMMC	Embedded multi-media card
EXTI	Extended interrupt
FDCAN	Controller area network with flexible data rate. Could also support time-triggered CAN (TT)
GPIO	General-purpose input/output
GPU	Graphics processing unit
HDP	Hardware debug port
IRQ	Interrupt request
IWDG	Independent watchdog
LDO	Low dropout regulator
LpDDR	Low-power DDR
LPSRAM	Low-power SRAM
LSE	Low-speed external quartz oscillator
LSI	Low-speed internal oscillator
LVDS	Low Voltage Differential Signaling
MCU	Microcontroller
MLAHB	Multilayer AHB/AHB-based interconnect
MPU	Microprocessor
OP-TEE	Open portable trusted execution environment
PLL	Phase-locked loop
PVD	Programmable voltage detector
PWR	Power control block

Term	Meaning
PSCI	Power state and coordination interface
OCTOSPI	Octal data lanes serial peripheral interface
RCC	Reset and clock control
RETRAM	Retention RAM
RIF	Resource isolation framework
RTC/TAMP	Real time clock/tamper management
SDMMC	Secure digital and multimedia card interface. Supports SD, MMC, eMMC, and SDIO protocols
SMPS	Switched mode power supply converter
SRAM	Static random-access memory
STPMIC2	Power management-integrated circuit. An external circuit that provides various platform power supplies with large controllability through signals and serial interface. STPMIC25A (for 3.3 V on V_{DD} application) and STPMIC25B (for 1.8 V on V_{DD} application) are available on STM32MP25x devices.
SYSRAM	System SRAM
SW	Software
TEMP	Temperature sensor
TEMPH-L	Temperature sensor high-low monitoring
TF-A	Trusted firmware for Cortex®-A
USART	Universal synchronous/asynchronous receiver transmitter
USB OTG	Universal serial bus (USB) on-the-go (OTG). A standard USB interface able to become a host or device
VBATH-L	VBAT high-low monitoring
VTT	DDR termination resistance power-supply
WFE	Wait for event
WFI	Wait for interrupt

1.2 Reference documents

For further information on STM32MP2 MPUs, refer to the following documents and deliverables available on www.st.com.

- STM32MP2 product line reference manuals (see Table 2 for details)
- STM32MP2 product line datasheets
- STPMIC2 datasheet
- *Getting Started with STM32MP25x lines hardware development* (AN5489)
- *How to use STPMIC2 for hardware and software integration on STM32MP2 MPUs* (AN5725)
- *How to use STPMIC25 for a wall adapter powered application on STM32MP25x MPUs* (AN5727)

2 Overview

This application note is applicable to all the devices that are defined in the introduction. The tables below describe their main characteristics.

Depending on the device part number, the system includes a Cortex®-M33, and a Cortex®-A35 with either a single-core or a dual-core. In the whole document:

- The Cortex®-A35 is also called CPU1.
- The Cortex®-M33 is also called CPU2 or MCU.
- The Cortex®-M0+ is also called CPU3.
- The full featured system, as defined in the table below, is partitioned into:
- One CPU1 subsystem: single or dual Cortex®-A35 with L2 cache
- One CPU2 subsystem: Cortex®-M33 with associated peripherals clocked according to CPU activity

The present document assumes a full featured device, for example the STM32MP257.

Table 2. STM32MP25x device configuration

Lines	Reference manual	Cortex-A35 configuration	Cortex-M33/ Cortex-M0+	GPU	LVDS	FDCAN	ETH
STM32MP251	RM0457	Single-core	Yes/Yes	No	No	No	x1
STM32MP253		Dual-core	Yes/Yes	No	Yes	Yes	x2
STM32MP255		Dual-core	Yes/Yes	Yes	Yes	Yes	x2
STM32MP257		Dual-core	Yes/Yes	Yes	Yes	Yes	x3

The STPMIC2 has several part numbers. The part numbers appropriate for STM32MP25x devices are given below:

- STPMIC25APQR for application using V_{DD} = 3.3 V
- STPMIC25BPQR for application using V_{DD} = 1.8 V

The generic part number " STPMIC2" is used in the rest of the document. For a specific part number, refer to the above paragraph.

3 Power management concept

This section describes the high-level power management concept of STM32MP2 devices. Refer to the corresponding reference manual for more details.

3.1 STM32MP2 device system architecture

The STM32MP2 devices are based on:

- a main single or dual-core CPU1 subsystem
- a CPU2 subsystem.
- an optional CPU3 subsystem having mostly system independent power modes.

The multiple core architecture requires specific power modes specific power modes both at system and at individual subsystem level.

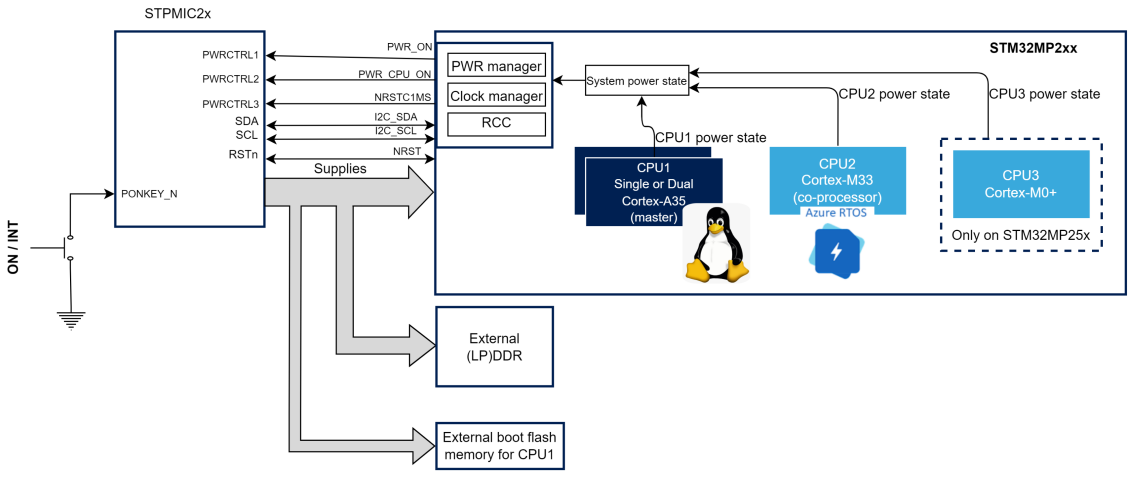
Internal digital logic is supplied by the STPMIC2 dedicated for the STM32MP2 devices. The latter has an internal regulator (such as an LDO or SMPS) for such purpose. This key difference results in a different power-supply management for the STM32MP2 devices which requires the use of external signals and STPMIC2 to control the desired voltage supplies.

The power management features are spread between the reset clock control (RCC), and the power (PWR) blocks of the STM32MP2 devices.

- The RCC block ensures the clock tree handling, such as PLLs, multiplexers, dividers, clock gating are reset.
- The RCC block manages the resets: the local peripheral, the Cortex®-A35, the Cortex®-M33, and the platform.
- The RCC and PWR blocks allow the power mode selection based on the respective power states of the CPU1 and CPU2 subsystems.
- The PWR block is responsible for low-power entry and exit. It drives the control pins (PWR_ON, PWR_CPU_ON, PWR_LP) to the external regulator based on the power mode.

The figure below shows the high-level system architecture of STM32MP2 devices.

Figure 1. STM32MP2 device high-level system architecture



3.2 System supplies (V_{DD}, V_{DDCPU}, and V_{DDCORE})

The STM32MP2 devices require several power supplies. Among these power supplies, V_{DD}, $V_{DDA18AON}$, V_{DDCPU}, and V_{DDCORE} play a key role in the low-power mode configuration.

- V_{DD}, $V_{DDA18AON}$: power supply input for I/Os and system analog such as reset, power management oscillators, and PLLs.
- V_{DDCPU}: CPU1 subsystem digital supply
- V_{DDCORE}: digital core domain supply.

Note: V_{DD}, and $V_{DDA18AON}$ must be present before V_{DDCPU} and V_{DDCORE}.

The various power pins of STM32MP2 devices can be supplied by using STPMIC2. The STPMIC2 usage for this purpose is detailed in the reference design section of the application note *How to use STPMIC25 for a wall adapter powered application on STM32MP25x MPUs* (AN5727).

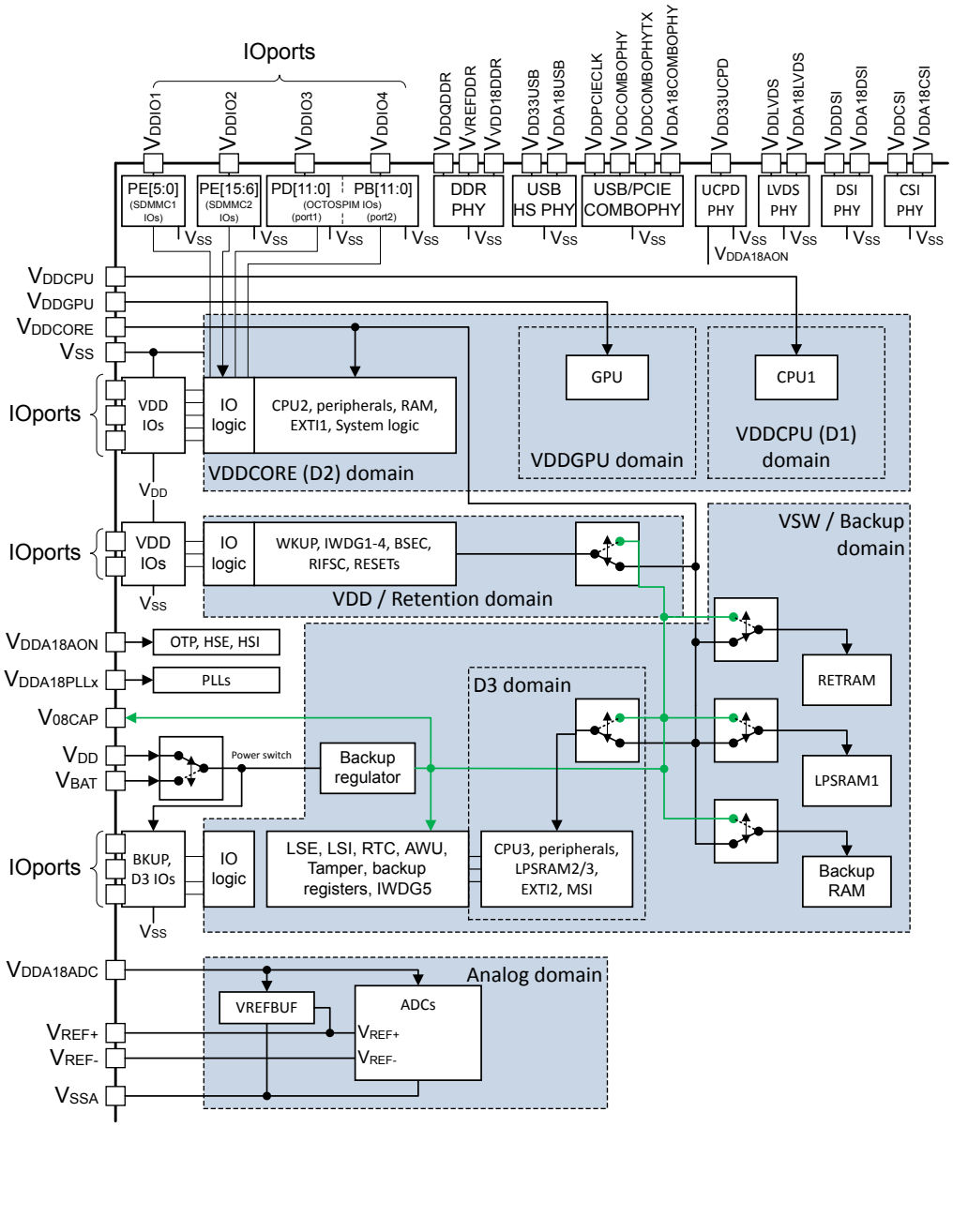
3.3 Power management description

The power management controls V_{DDCPU} and V_{DDCORE} supplies in accordance with the system operation modes (see Section 4: Operating modes). The system is split into three domains:

- D1 domain supplied by V_{DDCPU}, containing the Cortex®-A35 (CPU1).
- D2 system domain, which is the system domain supplied by V_{DDCORE}, contains the Cortex®-M33 (CPU2), a large part of the peripherals and the system control.
- D3 domain supplied by V_{DDCORE} or an internal backup regulator, containing the Cortex®-M0+ (CPU3), some peripherals able to work in autonomous mode, and a small part of the system control.

The power supply scheme is given in the illustration below.

Figure 2. STM32MP25x power supply scheme



4 Operating modes

This section describes the STM32MP2 device power modes and the ways of activating them.

4.1 Operating modes description

The operating modes control the clock distribution to the different parts of the system and the system power. The system operating mode is driven by the CPU1 subsystem and CPU2 subsystem, and by D3 domain in Standby mode. A CPU subsystem can include multiple domains depending on its peripheral allocation.

The table below presents the operating modes for the different systems.

Table 3. Operating modes

Domain	Power mode	D1 domain	D3 domain	Description
D2 system domain	Run1	DRun	SRun1	V_{DDCORE}, V_{DDCPU} power on, clock on
		DStop1		
	Run2	DStandby		V_{DDCORE} power on, V_{DDCPU} power off, clock on
	Stop1	DStop1	SRun2 or SStop1	V_{DDCORE}, V_{DDCPU} power on, clock off
	Stop2	DStandby		V_{DDCORE} power on, V_{DDCPU} power off, clock off
	LP-Stop1(1)(2)	DStop2	SRun3 orSStop2	V_{DDCORE}, V_{DDCPU} power on, clock off
	LP-Stop2(1)(2)	DStandby		V_{DDCORE} power on, V_{DDCPU} power off, clock off
	LPLV-Stop1(1)	DStop3		V_{DDCORE} and V_{DDCPU} reduced power-level, and supply load, clock off
	LPLV-Stop2(1)	DStandby		V_{DDCORE} reduced power level, and supply load, V_{DDCPU} power off, clock off
	Standby1			V_{DDCORE}, V_{DDCPU} power off, clock off
	Standby2	DStandby	SStandby	V_{DDCORE}, V_{DDCPU} power off, clock off
	VBAT(3)	DStandby	SRun3 orSStop2 or SStandby	VSW is supplied by battery. All power supplies are off except V_{BAT}. RTC/TAMP is still active clocked by LSE crystal
	Power off	DStandby	SStandby	All power supplies off
D1 domain	DRun	-	-	V_{DDCPU} power on, CPU1 clock on, CPU1 subsystem is in CRun or CSleep mode
	DStop1/ DStop2	-	-	V_{DDCPU} power on, CPU1 clock off, CPU1 subsystem is in CStop mode
	DStop3	-	-	V_{DDCPU} reduced power-level, CPU1 clock off, CPU1 subsystem is in CStop mode
	DStandby	-	-	V_{DDCPU} power off, CPU1 clock off, CPU1 subsystem is in CStop mode
D3 domain	SRun1	-	-	V_{DDCORE} power on, the domain bus matrix is clocked by system clocks, CPU3 subsystem is in CRun or CSleep mode, and the D2 system domain is in Run1 or Run2 mode
	SRun2	-	-	V_{DDCORE} power on, the domain bus matrix is clocked by a local clock, CPU3 subsystem is in CRun or CSleep mode and the D2 system domain is in Stop1 or Stop2 mode
	SRun3	-	-	V_{DDCORE} power on/reduced power-level, the domain bus matrix is clocked by a local clock, CPU3 subsystem is in CRun or CSleep mode and the D2 system domain is in LP-Stop1 or LP-Stop2, LPLV-Stop1 or LPLV-Stop2, or Standby mode

Domain	Power mode	D1 domain	D3 domain	Description
D3 domain	SStop1	-	-	V_{DDCORE} power on, the domain bus matrix may be stalled if clock is not requested by autonomous peripherals, CPU3 subsystem is in CStop mode and the D2 system domain is in Stop1 or Stop2 mode
	SStop2	-	-	V_{DDCORE} power on/reduced power-level, the domain bus matrix may be stalled if clock is not requested by autonomous peripherals, CPU3 subsystem is in CStop mode and the system D2 domain is in LP-Stop1 or LP-Stop 2, LPLV-Stop1 or LPLV-Stop2, or Standby mode
	SStandby	-	-	V_{DDCORE} power off, CPU3 clock off, All CPUs subsystems are in Cstop, the D2 system domain is in Standby, and the PDDS_D3 in PWR_D3CR selects SStandby
CPU1, CPU2 and CPU3	CRun	-	-	CPU and CPU subsystem peripherals clock on
	CSleep(4)	-	-	CPU clock OFF and CPU subsystem peripheral clock on/off
	CStop	-	-	CPU and CPU subsystem peripherals clock off
CPU1	eCSleep(5)	-	-	CPU clock OFF and CPU subsystem peripheral clock on/off

1. *There is no difference in the PWR_ON, PWR_CPU_ON and PWR_LP output control pins between LP-Stop1/2 and LPLV-Stop1/2 modes (see Section 4.3: External control signals PWR_ON, PWR_CPU_ON, PWR_LP pins). The user must ensure the STPMIC2 is correctly configured before entering any low power mode.*
2. *The main difference between Stop1 and Stop2, and LP-Stop1 and LP-Stop2 modes is that in LP-Stop1 and LP-Stop2 modes, the PWR_ON output signal is toggling to 0 when using STPMIC2 external power regulator. This means that when in LP-Stop1 or LP-Stop2 mode, STPMIC2 actions such as powering off DDR termination resistance power supply are possible.*
3. *To retain the content of the VSW domain when V_{DD} is turned off, the VBAT pin can be connected to an optional standby voltage supplied from a battery or from another source. The VSW domain is composed of:*
   - *RTC/TAMP*
   - *Backup registers*
   - *BKPSRAM*
   - *RETRAM*
4. *The CPU subsystems allocated peripheral clock operates according to RCC PERxLPEN. When the CPU1 enters CSleep with wait for event (WFE), the CPU1 subsystem allocated peripheral clock operates as in CPU1 CRun mode, irrespective of RCC PERxLPEN.*
5. *The CPU1 can go in an enhanced CSleep when the ESLPREQ is set in RCC_C1SREQSETR register. When this bit is set and STPREQ_Px (x = 0, 1) bits combination is 0, PLL1 is powered down and bypassed.*

Important: *The condition where V_{DDCORE} is OFF while V_{DDCPU} is ON is forbidden and not supported.*

The choice of which low-power mode to use dependends on the power saving target and it must be chosen according to which wakeup interrupts are needed and available in the chosen low-power mode and the expected wakeup time duration.

The user must always make sure that the chosen low-power mode is consistent with the available wakeup sources. For instance, Standby1 and Standby2 modes have very limited wakeup source capabilities.

Also, the wakeup duration is longer if power supply voltages have been reduced or switched off by reducing V_{DDCORE}, switching off DDR resistance termination supply with DDR in self-refresh or completely switching off the DDR. This last action may require reloading the firmware from the flash memory. The system low-power mode wakeup capabilities are presented in the table below.

Table 4. Low-power mode wake-up capabilities of the system

System power mode	Wake-up sources
Stop1/Stop2 LP-Stop1/LP-Stop2	CPU3, DBG,PVD, PVM, LPDMA, HPDMAx (x = 1, 2, 3), HSI frequency monitoring, USBH, USB3DRD, UCPD1, ETHx (x = 1,2), USARTx (x = 1 to 9), LPUART1, I2Cx (x = 1 to 8), I3Cx (x = 1 to 4), SPIx (x = 1 to 8), ADF1, DTS, LPTIMx (x = 1, 2), LPTIMy(y = 3, 4, 5), WWDG2, MBOX2, HSEM, GPIOs
LPLV-Stop1/2	CPU3, DBG, PVD, PVM, LPDMA, LPUART1, I2C8, I3C4, SPI8, ADF1, LPTIMy (y = 3, 4, 5), WWDG2, MBOX2, HSEM, GPIOs
Standby1	CPU3, DBG, LPDMA, LPUART1, I2C8, I3C4, SPI8, ADF1, LPTIMy (y = 3, 4, 5), WWDG2, MBOX2, HSEM, GPIOZ + 6 x WKUP pins
Standby2	6 x WKUP pins
All modes	BOR, VBATH/VBATL, TEMPH/TEMPL, LSE CSS, RTC/auto wake-up, tamper pins, IWDGx

4.2 Low-power mode control

4.2.1 Low-power mode control registers

The control register bits presented below are related to the low-power modes. For more information on low-power modes see Section 4.1: Operating modes description.

The low-power modes control register bits are presented in the table below.

Table 5. Low-power modes control register bits low-power modes control register bits

Register bit setting	Bit value	Description
PDDS_D1 bit of the PWR_CPU1CR register(1)	0: DStop mode kept when CPU1 enters CStop	This bit allows CPU1 to define the Deepsleep mode for the D1 domain.
	1: DStandby mode allowed when CPU1 enters CStop	
PDDS_D2 bit of the PWR_CPU1CR register(1)	0: Stop mode kept when CPU2 enters CStop	This bit allows CPU1 to define the Deepsleep mode for the D2 domain/ system.
	1: Standby mode allowed when CPU2 enters CStop	
PDDS_D2 bit of the PWR_CPU2CR register (2)	0: Stop mode kept when CPU2 enters CStop	This bit allows CPU2 to define the Deepsleep mode for the D2 domain/ system.
	1: Standby mode allowed when CPU2 enters Cstop	
PDDS_D3 bit of the PWR_D3CR register (3)	0: SStop mode kept when system enters CStop	This bit allows CPU3 to define the Deepsleep mode for the D3 domain.
	1: SStandby mode allowed when system enters CStop	
STPREQ_P0 bit of the RCC_C1SREQSETR register(4)	0: Writing 0 has no effect. Reading 0 means that the CPU1 processor core 0 does not allow the CPU1 subsystem to go to CStop	RCC CPU1 stop request set register defines whether the CPU1 subsystem is allowed to enter CStop.
	1: Writing 1 sets the STPREQ_P0 bit. Reading 1 means that the CPU1 processor core 0 allows the CPU1 subsystem to go to CStop	
STPREQ_P1 bit of the RCC_C1SREQSETR register(4)	0: Writing 0 has no effect. Reading 0 means that the CPU1 processor core 1 does not allow the CPU1 subsystem to go to CStop	RCC CPU1 stop request set register defines whether the CPU1 subsystem is allowed to enter CStop.
	1: Writing 1 sets the STPREQ_P0 bit. Reading 1 means that the CPU1 processor core 1 allows the CPU1 subsystem to go to Cstop	
ESLPREQ bit of the RCC_C1SREQSETR(4)	0: Writing 0 has no effect. Reading 0 means that enhanced CSleep for CPU1 subsystem is not requested	RCC CPU1 stop request set register defines whether the enhanced CSleep has been requested for CPU1.
	1: Writing 1 sets the ESLPREQ bit. Reading 1 means that enhanced CSleep for CPU1 subsystem is requested	

1. *PWR CPU1 control register*
2. *PWR CPU2 control register*
3. *PWR D3 control register*
4. *RCC CPU1 stop request set register*

The table below summarizes the relationship between the system and the CPUx (where x = 1, 2, or 3) low-power modes

Table 6. System low-power modes summary

System	CPU1	CPU2	CPU3	System clock	D3 oscillator	V_{DDCPU}	V_{DDCORE}
Run1	CRun or Csleep or eCSleep ESLPREQ = 1(1)	CRun, CSleep, or Cstop	CRun, CSleep, or Cstop	ON	ON	ON	ON (nominal)
	Cstop	CRun or Csleep					
Run2	CStop and PDDS_D1=1	CRun or Csleep				OFF	
Stop1	Cstop	Cstop		OFF	ON/OFF	ON	
Stop2	CStop and PDDS_D1 = 1					OFF	
LP-Stop1(2)	CStop and LPDS_D1=1(3)	CStop and LPDS_D2 = 1(4)				ON	
LP-Stop2	CStop and PDDS_D1=1					OFF	
LPLV-Stop1(2)	CStop and LPDS_D1=1(3) and LVDS_D1=1(5)	CStop and LPDS_D2=1(3) and LVDS_D2=1(6)				ON (reduced)	ON (reduced)
LPLV-Stop2	CStop and PDDS_D1=1					OFF	
Standby1		CStop and PDDS_D2=1					OFF
Standby2			CStop and PDDS_D3=1		OFF		

1. *ESLPREQ in RCC_C1SREQSETR selects enhanced CSleep for CPU1.*
2. *At least one PDDS_Dn (n = 1, 2) in PWR_CPU1CR and PWR_CPU2CR selects Stop.*
3. *LPDS_D1 in PWR_CPU1CR selects LP-Stop.*
4. *LPDS_D2 in PWR_CPU2CR selects LP-Stop.*
5. *LVDS_D1 in PWR_CPU1CR selects LPLV-Stop.*
6. *LVDS_D2 in PWR_CPU2CR selects LPLV-Stop.*

4.3 External control signals PWR_ON, PWR_CPU_ON, PWR_LP pins

The three output pins PWR_ON, PWR_CPU_ON, and PWR_LP are related to the V_{DDCORE} and V_{DDCPU} supplies. They are used by external components or regulators to identify which voltage level must be applied on V_{DDCORE} and V_{DDCPU}.

- PWR_ON: V_{DDCORE} supply request (active high):
  It is automatically generated by the hardware depending on the state of the STM32MP2 device and on the value of LPCFG_D2 (PWR_ON pin configuration) bit of the PWR D2 control register (PWR_D2CR).
- PWR_CPU_ON: V_{DDCPU} supply request (active high):
  It is automatically generated by the hardware depending on the state of the STM32MP2xx device and on the value of LPCFG_D1 (PWR_ CPU_ON pin configuration) bit of the PWR D1 control register (PWR_D1CR).
- PWR_LP: V_{DDCORE} low-power mode control (active low):
  It is automatically generated by the hardware depending on the state of the STM32MP2 device and the value of LPDS_D1, LPDS_D2, LVDS_D1 and LVDS_D2 bits of the PWR CPU1 and CPU2 control register (PWR_ CPU1CR) and (PWR_ CPU2CR). This pin is not used with STPMIC2.

Table 7. Register bit settings

Register bit setting	Bit value	Description
LPDS_D1	0: Stop mode selected, external regulator kept in main power mode.	This is bit 16 of the PWR CPU1 control register (PWR_CPU1CR. It controls Low-Power Deep Sleep Stop mode selection for the D1 domain.(1)
	1: LP/LPLV-Stop mode selected, the external regulator may enter low-power mode.	
LPDS_D2	0: Stop mode selected, external regulator kept in main power mode (pwr_lp = 1).	This is bit 16 of the PWR CPU2 control register (PWR_CPU2CR). It controls Low-Power Deep Sleep Stop mode selection for the D2 domain/system.(2)
	1: LP/LPLV- top mode selected, the external regulator may enter low-power mode (pwr_lp = 0).	
LVDS_D1	0: LP-Stop mode V_{DDCPU} domain supply reset level at same level as Run mode. V_{DDCPU} domain supply level in LP-Stop mode must be kept at same level as Run mode.	This is bit 17 of the PWR CPU1 control register (PWR_CPU1CR). It controls Low-Voltage Deep Sleep LPLV-Stop mode selection for the D1 domain. VDDCPU supply can be lowered in LPLV-Stop mode (see datasheet for voltage levels).
	1: LPLV-Stop mode V_{DDCPU} domain supply reset level at lower level than Run mode.	
LVDS_D2	0: LP-Stop mode V_{DDCORE} domain supply reset level at same level as Run mode. V_{DDCORE} domain supply level in LP-Stop mode must be kept at same level as Run mode.	This is bit 17 of the CPU2 control register (PWR_CPU2CR). It controls Low-Voltage Deep Sleep LPLV-Stop mode selection for the D2 domain/system. VDDCORE supply can be lowered in LPLV-Stop mode (see datasheet for voltage levels).
	1: LPLV-Stop mode V_{DDCORE} domain supply reset level at lower level than Run mode.	
LPCFG_D1	0: PWR_CPU_ON pin signals DStandby mode (PWR_CPU_ON = 1 in DRun, DStop1, DStop2, DStop3, and = 0 in DStandby).	This is bit 0 of the PWR D1 control register (PWR_D1CR) It controls the PWR_ON pin configuration
	1: PWR_CPU_ON pin signals DStandby and DStop2, DStop3 modes (PWR_CPU_ON = 1 in DRun, DStop1, and = 0 in DStop2, DStop3, and DStandby).	
LPCFG_D2	0: PWR_ON pin signals Standby (PWR_ON = 1 in Run, Stop1, Stop2, LP-Stop1, LP-Stop2, LPLV-Stop1, LPLV-Stop2, and = 0 in Standby1, and Standby2).	This is bit 0 of the PWR D2 control register (PWR_D2CR) It controls the PWR_ON pin configuration
	1: PWR_ON pin signals Standby, LP-Stop, and LPLV-Stop modes (PWR_ON = 1 in Run, Stop1, Stop2, and = 0 in LP-Stop1, LP-Stop2, LPLV-Stop1, LPLV-Stop2 and Standby1, and Standby2).	

1. *Further low-power mode selection is provided by LVDS_D1.*
2. *Further low-power mode selection is provided by LVDS_D2.*

The table below indicates:

- The PWR_ON, PWR_CPU_ON, and PWR_LP output pin values. These values depend on the various power mode configurations and LPDS_Dn, LPCFG_Dn, and LVDS_Dn bits (n= 1, 2).
- Shows the different way of using pins:
  - PWR_CPU_ON
  - PWR_ON
  - PWR_LP

This is determined by how the STM32MP2 device is used with STPMIC2. The power modes are detailed in the PWR section of the appropriate reference manual.

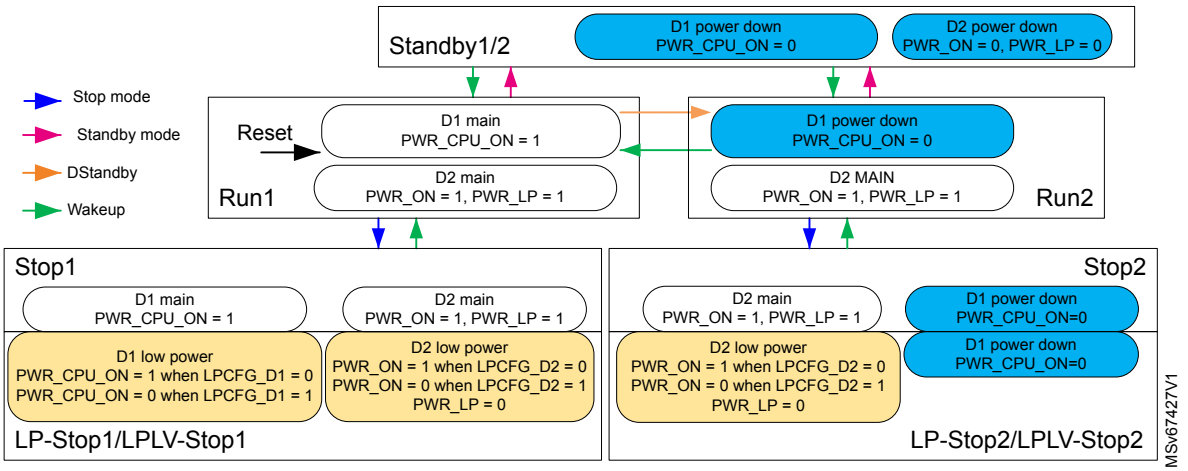
Table 8. PWR_LP, PWR_ON, PWR_CPU_ON levels according to power modes, LPDS, LVDS, and LPCFG bits

System	LPDS_D1/ LVDS_D1 bits	LPDS_D2/ LVDS_D2 bits	PWR_LP	PWR_ON		PWR__CPU_ON	
				LPCFG_D2 =0	LPCFG_D2[1] =1	LPCFG_D1[2] =0	LPCFG_D1 =1
Startup (until V_{DD} reaches POR threshold level)	X/X	X/X	0	0	0	0	0
Run1	X/X	0/X	1	1	1	1	1
	0/X	X/X					
Run2	0/X	X/X				0	0
Stop1	0/X	0/X				1	1
Stop2	0/X	0/X				0	0
LP-Stop1	1/0	1/0	0[3]		0[3]	1	
LP-Stop2	1/0	1/0				0	
LPLV-Stop1	1/1	1/1				1	
LPLV-Stop2	1/1	1/1				0	
Standby1	X/X	X/X		0			
Standby2	X/X	X/X					
V_{BAT}(V_{DD} powered down)	X/X	X/X	High-Z	High-Z	High-Z	High-Z	High-Z

1. *Configuration used with STPMIC25x PWRCTRL1 pin connected to PWR_ON.*
2. *Configuration used with STPMIC25x PWRCTRL2 pin connected to PWR_CPU_ON.*
3. *There is no difference between LP-Stop1/2, LPLV-Stop1/2, and Standby1/2 mode on the PWR_ON, PWR_LP output values '00' with LPCFG_D2 bit=1.*

The figure below illustrates the relation between the system states and the PWR_ON, PWR_CPU_ON, PWR_LP output pins of regulator control for STM32MP2 devices.

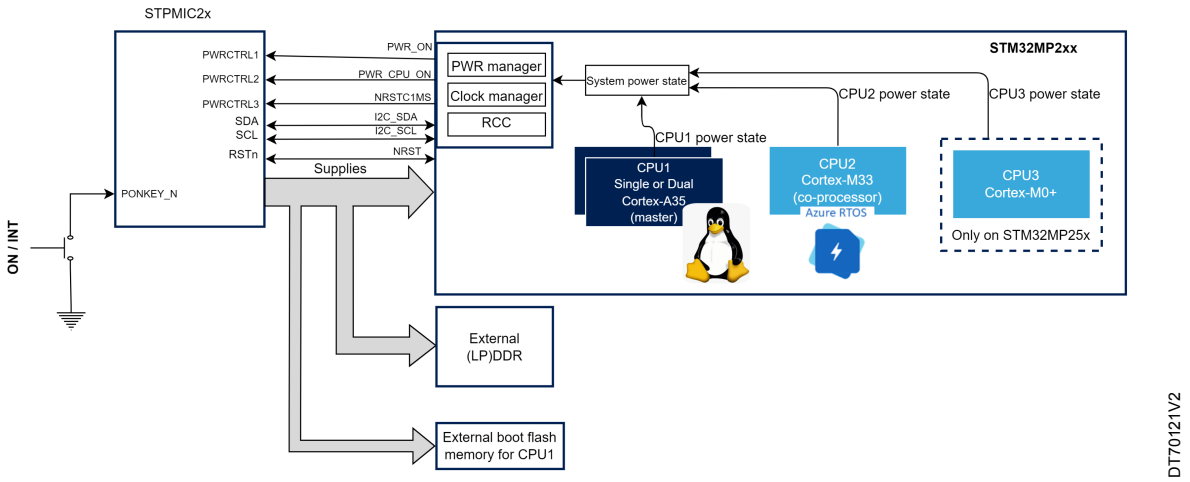
Figure 3. Power states and external regulator control



4.3.1 Using the STPMIC2 power regulator

The figure below shows the STM32MP2 product lines high-level system architecture when using the STPMIC2 power regulator.

Figure 4. STM32MP2 product lines high-level system architecture using STPMIC2



The following pins from the STM32MP2 are needed to interface and control the STPMIC2 power modes:

Table 9. STM32MP2 / STPMIC2 connections

STM32MP2 pins	STPMIC2 pins
PWR_ON	PWRCTRL1
PWR_CPU_ON	PWRCTRL2
NRSTC1MS	PWRCTRL3

In that case, the user must set the LPCFG_D1 bit of the PWR_D1CR register to 0, and LPCFG_D2 bit of the PWR_D2CR to 1.

Table 8 shows that there is no difference between the LP-Stop1 and LP-Stop2, LPLV-Stop1 and LPLV-Stop 2, and Standby1 and Standby2 modes on the output values of the control signals PWR_ON and PWR_LP which are 0 if the bit LPCFG_D2 is set to 1.

The STPMIC2 differentiates between the LP-Stop1 and LP-Stop2, LPLV-Stop1 and LPLV-Stop2, and Standby1/2 low-power modes and applies the correct V_{DDCORE} and V_{DDCPU} value thanks to the STPMIC2 internal registers. The programming of these registers is done via the I2C interface and not through the PWR_ON, PWR_CPU_ON, nor PWR_LP pins values.

When using the STM32MPU OpenSTLinux distribution, the secure monitor TF-A or OP-TEE handles this programming. Before entering into LP-Stop1 and LP-Stop2, LPLV-Stop1 and LPLV-Stop2, or Standby1/2 mode, the application must configure the STPMIC2 internal registers to the correct system power-supplies level. These are V_{DDCORE}, V_{DDCPU}, V_{DD}, VDDR, and others. When entering LP-Stop1, and LP-Stop2 mode, it is still possible to program the STPMIC2 even though the V_{DDCORE} level is not decreased. This applies only in LPLV-Stop1, and LPLV-Stop2. This results in other power supplies being shut down if not needed. This is, for example, the case for the power supply of the DDR termination resistances.

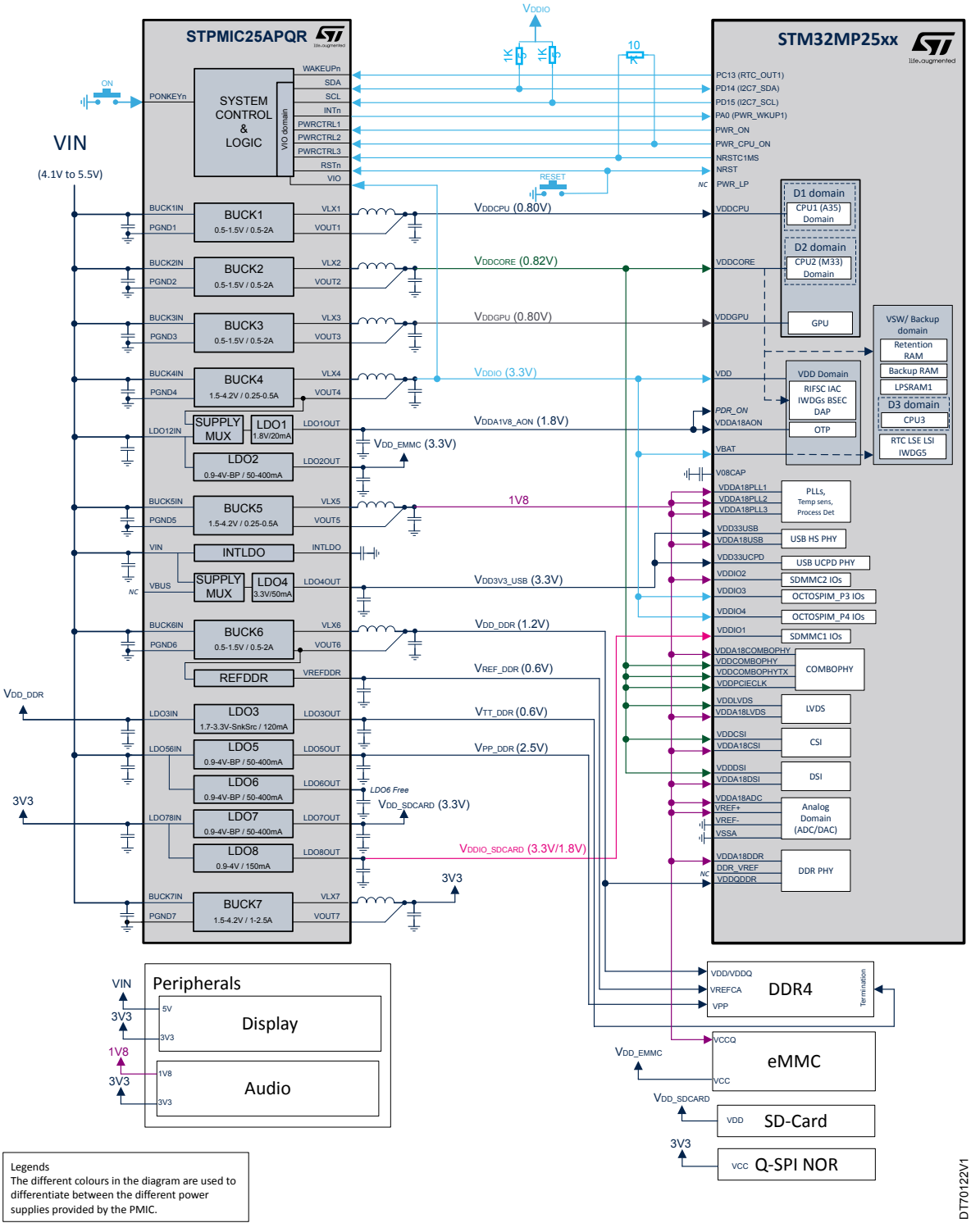
- Control signals between STPMIC2 and STM32MP25x (see Table 2 for more details of the devices).
  This paragraph describes how the STM32MP25x microprocessor communicates with the STPMIC2 device. There are several interface choices that can be used depending on the application requirements. some interface is described below. Refer to see AN5727 for more information.
  - I²C Interface:
    The STPMIC2 can be controlled by the STM32MP2 via the I²C interface. This interface controls advanced low power features such as:
    - Enable or disable a regulator
    - Set a regulator voltage and mode (low-power mode)
    - Switch between nominal mode and overdrive mode.
  - ON / INT push button:
    STPMIC2 PONKEY_N digital input pin (active low) is connected to the user “ON / INT” push button on the application. This button has three actions:
    - Powers up the STPMIC2
    - Sends an interrupt to the STM32MP2
    - Forces a Turn on/off condition (see section Application turn-on / turn-off conditions on AN5727 for more information).
  - NRST signal
    The STM32MP2 NRST pin is linked to the STPMIC2 RSTn digital input active low pin. It can also be connected to a “RESET” push button. The RSTn signal is a bidirectional reset pin for the STPMIC2:
    - When the STPMIC2 asserts an RSTn, such as during the power-up or the power-down sequence, it drives the NRST signal low: the STM32MP2 is forced into a reset status until the STPMIC2 releases the RSTn.
    - When the STM32MP2 asserts an NRST signal, such as a watchdog event, low voltage on V_{DD} and so on, or on a press of the “RESET” button, the STPMIC2 immediately asserts a reset with the RSTn pin and performs a non-interruptible power cycle: the STPMIC2 performs a power down sequence followed by a power up sequence and finally releases the RSTn.
- At the end of the power-cycle sequence, the STPMIC2 waits for the STM32MP2 NRST signal to go high before rearming the reset to avoid an infinite reset loop.
  - PWRCTRL1, PWRCTRL2, PWRCTRL3
    Power controls are STPMIC2 digital input signal controlled from the STM32MP2, all STPMIC2 power control pins are connected to a power control pin of the STM32MP2 as described below:
    - The PWR_ON signal is driven by the STM32MP2 PWR_ON pin to control the PWRCTRL1 pin of STPMIC2. The PWR_ON signal is also used to control the V_{DDCORE} power supply for the STM32MP2 D2 domain. When the PWR_ON signal is low, it enables the STPMIC2 to switch to a low-power mode, such as LP-Stop1, LP-Stop2, LPLV-Stop1, LPLV-Stop2, or Standby. This reduces the power consumption of the system by turning off or reducing the power to non-essential components.
    - The PWR_CPU_ON signal is driven by the STM32MP2 PWR_CPU_ON pin to control the PWRCTRL2 pin of STPMIC2. The PWR_CPU_ON signal is also used to control the V_{DDCPU} power supply for the STM32MP2 D1 domain. When the PWR_CPU_ON signal is low, it enables the STPMIC2 to switch to a low-power mode, such as LP-Stop1, LP-Stop2, LPLV-Stop1, LPLV-Stop2, or Standby. This reduces the power consumption of the system by turning off or reducing the power to non-essential components.
    - The NRSTC1MS signal is driven by the STM32MP2 NRSTC1MS pin to control PWRCTRL3 pin of the STPMIC2. The NRSTC1MS pin is active when a system reset is generated. The NRSTC1MS pin can be used to control supplies of external flash required for first level boot of CPU1 and which needs a power cycle to ensure a platform reboot (eMMC, SD-Card).

4.3.2 STPMIC2 supplies example 3.3 V I/Os with DDR4

This reference design example targets a complex 3.3 V I/Os platform with DDR4 and high integration with the STPMIC25APQR. The design example is illustrated in Figure 5.

Usually, all platform components can be powered by the STPMIC2. Full power supply control is supported with the STPMIC25APQR I²C and side band signals. All low power modes are supported. See the STPMIC2 documentation for details of STPMIC25APQR components.

Figure 5. STPMIC25APQR example 3.3 V I/Os with DDR4



Low power modes are managed by the STM32MP25x. The external STM32MP25x control signals controls the STPMIC2 PWRCTRLx signals assigned to manage the STPMIC2 regulators behavior. Depending on PWRCTRLx status, the STM32MP25x enters into one low power mode or another (see Table 8).

Before entering in any low-power mode, the STM32MP25x must set the STPMIC2 PWRCTRL xxxx_ALT_CR register in line with the expected STPMIC2 regulator settings for low-power mode behavior. If needed, the STM32MP25x must set the STPMIC2 PWRCTRL xxxx_MAIN_CR control registers to guarantee that the application leaves the low-power mode.

The following table shows how the STPMIC2 can be programmed in Run1 mode and before entering any LP-Stop, any LPLV-Stop, and any Standby modes respectively. The table below lists all the power supplies and not only those relates to the V_{DDCORE}. The settings used on them are in line with the example presented in Figure 5.

Table 10. STPMIC25 configuration depending on operating modes

Supply name	STPMIC25 regulators	PWRCTRLx assignement	Configuration				
			RUN1/STOP1	RUN2/STOP2	LP-STOP1/ LPLV-STOP1	LP-STOP2/ LPLV-STOP2	STANDBY1/ STANDBY2
V_{DDCPU}	BUCK1	PWR_CPU_ON	ON HP(1) (xxxx_MAIN_CR)	OFF (xxxx_ALT_CR)	ON HP(1) (xxxx_MAIN_CR)	OFF (xxxx_ALT_CR)	OFF (xxxx_ALT_CR)
V_{DDCORE}	BUCK2	PWR_ON	ON HP(1) (xxxx_MAIN_CR)	ON HP(1) (xxxx_MAIN_CR)	ON HP(1) (xxxx_ALT_CR) (2)	ON HP(1) (xxxx_ALT_CR)(2)	OFF (xxxx_ALT_CR)
V_{DDA18}	BUCK5	PWR_ON	ON (HP(1) for BUCKs) (xxxx_MAIN_CR)	ON (HP(1) for BUCKs) (xxxx_MAIN_CR)	ON HP(1) (xxxx_ALT_CR)	ON HP(1) (xxxx_ALT_CR)	OFF (xxxx_ALT_CR)
V_{DD_DDR}	BUCK6						
V_{REF_DDR}	VREFDDR						
V_{PP_DDR}	LDO5						
V_{DD_EMMC}	LDO2	NRSTC1MS	ON (xxxx_MAIN_CR)	ON (xxxx_MAIN_CR)	ON (xxxx_MAIN_CR)	OFF (RESET)	OFF (RESET)
V_{DD_SDCARD}	LDO7						
V_{DDIO_SDCARD}	LDO8						
V_{TT_DDR}	LDO3	PWR_ON	ON (xxxx_MAIN_CR)	ON (xxxx_MAIN_CR)	OFF (xxxx_ALT_CR)	OFF (xxxx_ALT_CR)	OFF (xxxx_ALT_CR)
V_{DD3V3_USB}	LDO4	-	ON or OFF (xxxx_MAIN_CR)	OFF (xxxx_MAIN_CR)	ON or OFF (xxxx_MAIN_CR)	OFF (xxxx_MAIN_CR)	OFF (xxxx_MAIN_CR)

1. *HP stands for high power. Ref to Section 4.6.6: Exit from LP-Stop2 and LPLV-Stop2 mode*
2. *VDDCORE supply lowered to 670mV (for LPLV-Stop1 or LPLV-Stop2 mode) (xxxx_ALT_CR)*

BUCK4 (V_{DDIO}) and LDO1 (V_{DDA1V8_AON}) are always ON.

Application note AN5727 provides more details on how to use STPMIC25x.

4.4 Low-power mode entry sequence

The STM32MP2 devices have dedicated power-modes at subsystem level (CPU1, CPU2 and CPU3), at D1 domain level and at system level (see Section 4: Operating modes). This section details how to enter those power modes at subsystem and at system level.

Table 11. Subsystem low-power mode entry sequence definition

System level definition	Description
Subsystem	
CPU1 subsystem	The CPU1 subsystem low-power modes (CSleep and CStop) are entered by the CPU1 when executing the wait for interrupt (WFI) or wait for event (WFE) instructions. WFE being available only when entering CSleep. In order for the CPU1 to enter the subsystem low-power modes, the RCC and PWR registers must already be correctly programmed (refer to Section 5.1: Peripheral assignment and allocation). When using STM32 MPU OpenSTLinux Distributions those mechanisms are handled by the first stage bootloader (refer to Section 4.5: Power management under Linux®).
CPU2 subsystem	The CPU2 subsystem low-power modes (CSleep and CStop) are entered by the CPU2 when executing the WFI, or WFE instructions, or when the SLEEPONEXIT bit in the Cortex®-M33 system control register is set on return from ISR.
CPU3 subsystem	The CPU3 subsystem low-power modes (CSleep and CStop) are entered by the CPU3 when executing the WFI, or WFE instructions, or when the SLEEPONEXIT bit in the Cortex®-M0+ system control register is set on Return from ISR.
Domain	
D1 domain	The D1 domain enters D1 DStop1 mode when all EXTI wake-up sources assigned to CPU1 are cleared, and CPU1 is in CStop mode and the PDDS_D1 and LPDS_D1 bits in the PWR_CPU1CR register selects DStop for the D1 domain. The D1 domain enters D1 DStandby mode when all EXTI wake-up sources assigned to CPU1 are cleared, and CPU1 is in CStop mode and the PDDS_D1 bit in the PWR_CPU1CR register select DStandby for the D1 domain.
System	
System	The system enters Stop1, Stop2, LP-Stop1, LP-Stop2, LPLV-Stop1, LPLV-Stop2 or Standby1 when all the EXTI wakeup sources are cleared and when both the CPU1 and CPU2 are in CStop. The system enters Standby2 when all the EXTI wakeup sources are cleared, and all CPUs are in CStop.

Note: The system low-power mode entry sequence are defined in Table 13.

When using STM32CubeMP2 package, several functions are defined for low-power modes. The table below describes each function defined in STM32CubeMP2 package according to its dedicated low-power mode.

Table 12. Low-power modes functions used on CPU within STM32CubeMP2 package

Low-power mode	Function	Argument	Description
Csleep	HAL_PWR_EnterSLEEPMode	SLEEPEntry: • PWR_SLEEPENTRY_WFI • PWR_SLEEPENTRY_WFE	Specifies if CSleep mode is entered with WFI or WFE instruction.
	HAL_PWR_EnableSleepOnExit	None	Set SLEEPONEXIT bit of SCR register. When this bit is set, the processor reenters SLEEP mode when an interruption handling is over.
	HAL_PWR_DisableSleepOnExit	None	Clears SLEEPONEXIT bit of SCR register.
CstopPDDS_Dx=0	HAL_PWR_EnterSTOPMode	Regulator: • PWR_REGULATOR_LP_OFF • PWR_REGULATOR_LP_ON_LV_OFF • PWR_REGULATOR_LP_ON_LV_ON	Specifies which power state the regulator supplying current CPU domain can reach.
		STOPEntry: • PWR_STOPENTRY_WFI • PWR_STOPENTRY_WFE	Specifies if CStop mode is entered with WFI or WFE instruction.
CstopPDDS_Dx=1	HAL_PWR_EnterSTANDBYMode	STANDBYType: • PWR_STANDBY_1(1) • PWR_STANDBY_2(2)	This function puts the CPU in CStop (WFI) and sets the CPUx PDDS_Dx bit to 1, allowing the system to enter Standby mode. The `STANDBYType` parameter specifies the Standby mode that can be reached.

1. *Standby1 mode: D3 still On, but D1 and D2 Off*
2. *Standby2 mode: D3, D2 and D1 Off*

Table 13. System low-power mode entry sequence

System low-power mode	Mode entry	Comments
Stop1	When CPU1and CPU2 subsystems are in CStop and there is no active EXTI1-2 wake-up source: • PDDS_D1 = 0 in PWR_CPU1CR • PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR • LPDS_D1 = 0 in PWR_CPU1CR or LPDS_D2 = 0 in PWR_CPU2CR	The system clock including any PLL, and the system bus matrix clocks are stopped.
Stop2	When CPU1 and CPU2 subsystems are in CStop and there is no active EXTI1-2 wake-up source: • PDDS_D1 = 1 in PWR_CPU1CR • PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR • LPDS_D2 = 0 in PWR_CPU2CR	The Stop2 mode is equivalent to Stop1 mode in term of available features, but with the D1 in DStandby mode (V_{DDCPU} supply is powered down).
LP-Stop1	When CPU1and CPU2 subsystems are in CStop and there is no active EXTI1-2 wake-up source: • PDDS_D1 = 0 in PWR_CPU1CR • PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR • LPDS_D1 = 1 in PWR_CPU1CR and LPDS_D2 = 1 in PWR_CPU2CR	The LP-Stop1 mode is handled in the same way as Stop1 mode, during LP-Stop1 mode V_{DDCPU} and V_{DDCORE} are kept at Run mode level, but with a limited power capability (1).

System low-power mode	Mode entry	Comments
LPLV-Stop1	When CPU1and CPU2 subsystems are in CStop and there is no active EXTI1-2 wake-up source: • PDDS_D1 = 0 in PWR_CPU1CR • PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR • LPDS_D1 = 1 in PWR_CPU1CR and LPDS_D2 = 1 in PWR_CPU2CR • LVDS_D1 = 1 in PWR_CPU1CR and LVDS_D2 = 1 in PWR_CPU2CR	The LPLV-Stop1 mode is handled in the same way as Stop1 mode, but V_{DDCPU} and V_{DDCORE} supplies are reduced, and the power capability (1) is reduced as well. (2)
LP-Stop2	When CPU1and CPU2 subsystems are in CStop and there is no active EXTI1-2 wake-up source: • PDDS_D1 = 1 in PWR_CPU1CR • PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CR • LPDS_D2 = 1 in PWR_CPU2CR	The LP-Stop1 mode is handled in the same way as Stop2 mode, during LP-Stop2 mode V_{DDCORE} is kept at Run mode level, but with a limited power capability(1), the D1 domain is in DStandby mode (V_{DDCPU} supply is powered down).
LPLV-Stop2	When CPU1and CPU2 subsystems are in CStop and there is no active EXTI1-2 wake-up source: • PDDS_D1 = 1 in PWR_CPU1CR • PDDS_D2 = 0 in PWR_CPU1CR or PWR_CPU2CRLPDS_D2 = 1 in PWR_CPU2CR • LVDS_D2 = 1 in PWR_CPU2CR	The LPLV-Stop2 mode is handled in the same way as Stop2 mode, but V_{DDCORE} supply is reduced, and the power capability(1) is reduced as well, the D1 domain is in DStandby mode (V_{DDCPU} supply is powered down).(2)
Standby1	When CPU1and CPU2 subsystems are in CStop and there is no active EXTI1-2 wake-up source: • All WKUPF bits in PWR_WKUPCRx are cleared • All RTC wake-up events are cleared • PDDS_D1 = 1 in PWR_CPU1CR • PDDS_D2 = 1 in both PWR_CPU1CR and PWR_CPU2CR • PDDS_D3 = 0 in PWR_D3CR	The Standby1 mode is used to achieve a lower power consumption. It is based on CPU1 and CPU2 subsystem CStop mode, as for Stop modes, but with V_{DDCORE} and V_{DDCPU} supply regulators powered off. The D3 domain is still powered from the internal backup regulator.(3)
Standby2	All CPU subsystems are in CStop and there is no active EXTI1/2 wake-up source: • All WKUPF bits in PWR_WKUPCRx are cleared • All RTC wake-up events are cleared • PDDS_D1 = 1 in PWR_CPU1CR • PDDS_D2 = 1 in both PWR_CPU1CR and PWR_CPU2CR • PDDS_D3 = 1 in PWR_D3CR	The Standby2 mode is used to achieve the lowest power consumption. It is handled in the same way as Standby1, with CPU3 in CStop, and PDDS_D3 in PWR_D3CR selecting SStandby mode.(3)

1. *Limited power capability: this is usually achieved using an external STPMIC2 having two power modes:*
   - *A high-power mode (HP) for nominal cases (Run1 and Stop1)*
   - *A low-power mode (LP) limiting the power capability for LP-Stop1, LP-Stop 2 and LPLV-Stop1, LPLV-Stop2.*
2. *In LP-Stop 2 and LPLV-Stop1, LPLV-Stop2 with low-voltage mode, the peripherals in D2 domain that can wake up from Stop1 and Stop2 (such as TIM, UART, I2C, SPI), must all be disabled. The DDR retention must be enabled by DDRRETEN in PWR_CR11. The D3 domain is supplied from the internal backup regulator (SStop2) to operate even if the D2 domain is in low-voltage mode.*
3. *When the system enters Standby1 or Standby2, PWR_ON, PWR_CPU_ON and PWR_LP pins are set low. The external voltage regulators that provide V_{DDCORE} and V_{DDCPU}, are disabled. V_{DDCORE} and V_{DDCPU} domains are completely powered off. PLLs, HSI and HSE oscillators are also switched off. SRAM and register contents are lost except for V_{DD}, VSW, backup domain, and retention domain (RTC/TAMP registers, TAMP backup register, LSE, BKPSRAM and RETRAM), and Standby circuitry (D3 domain in Standby1 mode).*

Note: To avoid over consumption on V_{DDQ_DDR}, when entering LP-Stop1 and LP-Stop2 or LPLV-Stop1 and LPLV-Stop 2 if V_{DDQ_DDR} is not shut down, the DDR memory must be put in self-refresh. Furthermore, DDR PHY must be set in retention mode. This is done by setting the DDR retention-enable bit which is the DDRRETEN bit of the PWR control register 11 (PWR_CR11) register.

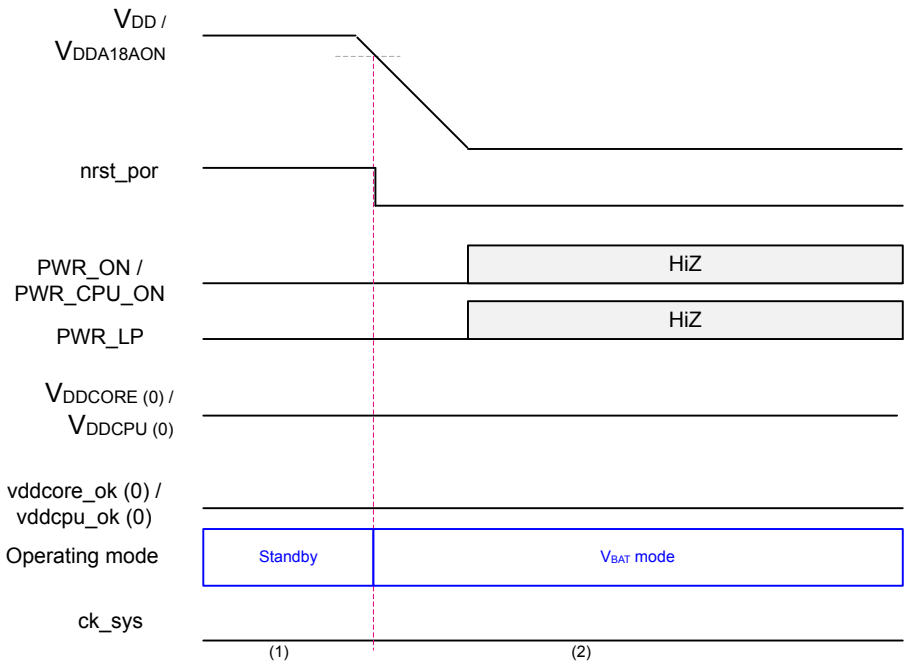
4.4.1 VBAT mode

To enter VBAT mode from Standby mode, the V_{DD} and $V_{DDA18AON}$ supplies must be powered down.

The PWR_ON and PWR_CPU_ON signals continue to request for V_{DDCORE} and V_{DDCPU} supplies to be powered off.

The device enters VBAT mode when V_{DD} is powered down. The PWR_ON, PWR_CPU_ON, and PWR_LP signals then become high-impedance (HiZ).

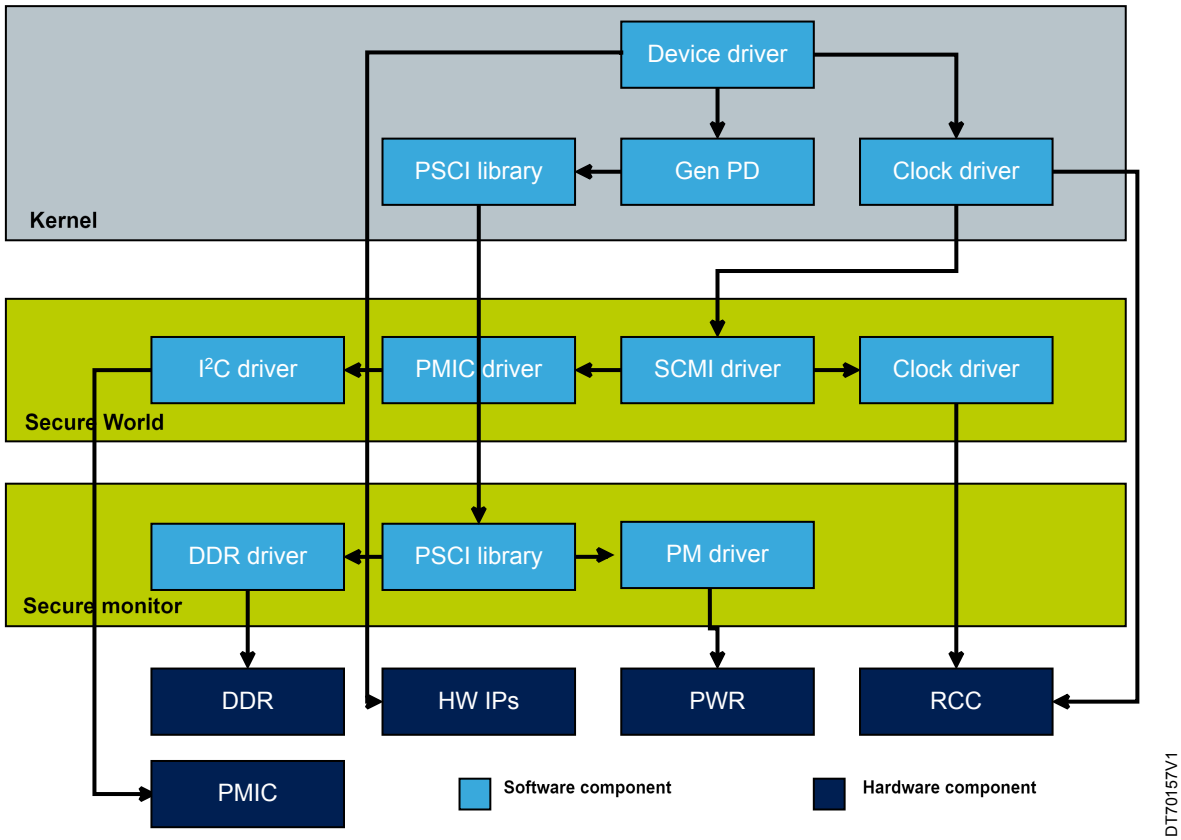
Figure 6. Device VBAT mode power control



4.5 Power management under Linux®

Linux power management is done through the software framework shown in the figure below.

Figure 7. Linux power management software framework



The Linux suspend framework is used to enter low power modes. It relies on:

- genPD (generic power-domain) framework which allows the definition of power domains and allow the mapping of IPs on those power domains
- PSCI standard to perform the low-level power management process
- SCMI handles wakeup sources by providing a platform-specific interface for the OS to register, deregister, query and enable/disable wakeup sources. Additional information can be found on the Power overview wiki article available at:
  http://wiki.st.com/stm32mpu/ wiki/Power_overview .
  The secure world framework is used to configure the external power-regulator voltages for each power mode. It relies on an ST STPMIC2x (PMIC) driver and an I²C driver.

4.5.1 Linux® power commands mapping to STM32MP25x lines power modes

When using Linux operating system, power mode commands are predefined and this section explains how they are mapped to the STM32MP25x lines hardware low-power modes. It also explains the low-power mode control using the MCU coprocessor.

Under Linux, the user can ask the system to enter into low-power mode with the following direct input command: To enter in Standby2 mode with DDR off:

```
'shutdown -h 0'
```

Or to reach any low-power mode among Stop1 variants withDDR in self-refresh:

```
echo 'freeze' > /sys/power/state
```

Or to reach any low-power mode among Stop2 variants and Standby1, with DDR in self-refresh:

```
echo 'mem' > /sys/power/state
```

The `'disk'` and `'standby'`commands are not supported on STM32MP2 devices.

The Cortex®-A35 is put in wait for interrupt (WFI) when entering a low-power mode.

However, the system only reaches the targeted low-power mode after the MCU subsystem has been previously configured in the expected low-power mode.

For instance, the Linux `mem` command puts the system in Standby1 mode only if the MCU has been previously set to CStop mode allowing Standby (for example using `HAL_PWR_EnterStandbyMode()` function).

The list of available wakeup sources is not the same for all low-power modes, so a software mechanism is implemented through the genPD framework to ensure that the low-power mode and the activated wakeup source are consistent.

Important: *The strategy is to allow the MPU to enter the deepest low-power mode available according to the currently activated wakeup source(s).*

Example 1

The user activates the UART4 as wakeup source. The UART is powered by V_{DDCORE}, so all low-power modes that are modifying V_{DDCORE} are automatically forbidden.

Then, calling: `echo mem > /sys/power/state` results in the MPU entering CStandby allowing Stop2 or LP-Stop2.

Example 2

The user selects the RTC as wake-up source. The RTC is always powered, whatever the low-power mode status. Then, calling `echo mem > /sys/power/state`results in the MPU entering CStop allowing Standby1.

This mechanism ensures that a blocking situation such as activating UART4 as wakeup source and requesting

SoC Standby1 can never happen.

The system power mode is the result of both MPU and MCU power states.

The table 13 below lists the deepest possible power mode according to wakeup source groups and lists the equivalence between Linux standard low-power modes and STM32MP25x lines system low-power modes, including the external (Lp)DDR power state (on, off, self-refresh (SR)) as it is implemented in the STM32MPU OpenSTLinux Distributions BSP provided environment.

Systems using 32-bit DDR4 interface most likely use external resistance termination on address and command signals. Those systems shall be powered by a resistance termination power supply (VTT) that can be a significant contributor to the overall system consumption during low-power modes.

Using STPMIC2x, it is possible to switch off this power supply when PWR_ON signals are pulled down (LP-Stop1, LPLV-Stop1, LP-Stop2, LPLV-Stop2 and Standby1).

Table 14. Deepest power mode per wakeup source group and equivalence between Linux and STM32MP25x lines

Wakeup source	Linux command	STM32MP2x device system deepest power mode	System DDR	Linux kernel state	Power consuming	Wake-up time	Comment/Application guideline
Group1 : HPDMA, HSI, USB, UCPD1, ETH, USART, I2C, I3C, SPI, DTS, LPTIMx	"freeze"	Stop1 or LP-Stop1	SR (VTT on/off)*	"Suspend-to-idle "	Medium	Low	LP-Stop1: driving external PWR_LP and PWR_ON permits to design custom strategy for external regulator. Typical application is to switch-off DDR4 termination supply (VTT) in DDR4 32-bit design
Group2 : PVD, PVM, GPIOs	"freeze"	LPLV-Stop1	SR (VTT off)	"Suspend-to-idle "	Medium	Medium	LPLV-Stop1: save power thanks to power retention. Can be suitable for applications without aggressive power constraints and tolerant with limitations of wakeup source
Group1 : HPDMA, HSI, USB, UCPD1, ETH, USART, I2C, I3C, SPI, DTS, LPTIMx	"mem"	Stop2 or LP-Stop2	SR (VTT off)	"Suspend-to-ram "	Low	Medium	Stop2 or LP-Stop2: save power thanks to CPU1 power off. Can be suitable for applications with aggressive power constraints and without limitations of wakeup source
Group2 : PVD, PVM, GPIOs	"mem"	LPLV-Stop2	SR (VTT off)	"Suspend-to-ram "	Low	Medium	LPLV-Stop2: save power thanks to power retention. Can be suitable for applications with aggressive power constraints and tolerant with limitations of wakeup source
Group3 : CPU3, DBG, LPDMA, LPUART1, I2C8, I3C4, SPI8, ADF1, LPTIMy, WWDG2, MBOX2, HSEM, GPIOZ	"mem"	Standby1	SR (VTT off)	"Suspend-to-ram "	Low	High	Standby1 saves more power at the expense of wake-up time
Group4 : 6 x WKUP pins	"shutdown"	Standby2	SR (VTT off)	"Shutdown"	Low	High	Standby2 saves more power at the expense of wake-up time and limitations of wakeup source
Group5 :	"shutdown"	Off/VBAT	Off	"Shutdown"	Very low	Very High	-

Wakeup source	Linux command	STM32MP2x device system deepest power mode	System DDR	Linux kernel state	Power consuming	Wake-up time	Comment/Application guideline
BOR, VBATH/ VBATL, TEMPH/ TEMPL, LSE CSS, RTC/auto wake-up, tamper pins, IWDGx							

Note: This table assumes that Cortex®-M33 is in CStop mode for all expected Stop1, LP-Stop1, LPLV-Stop1, Stop2, LP-Stop2, LPLV-Stop2 system power modes and in CStop with PDDS_D2=1 for Standby1, Standby2 system power mode.

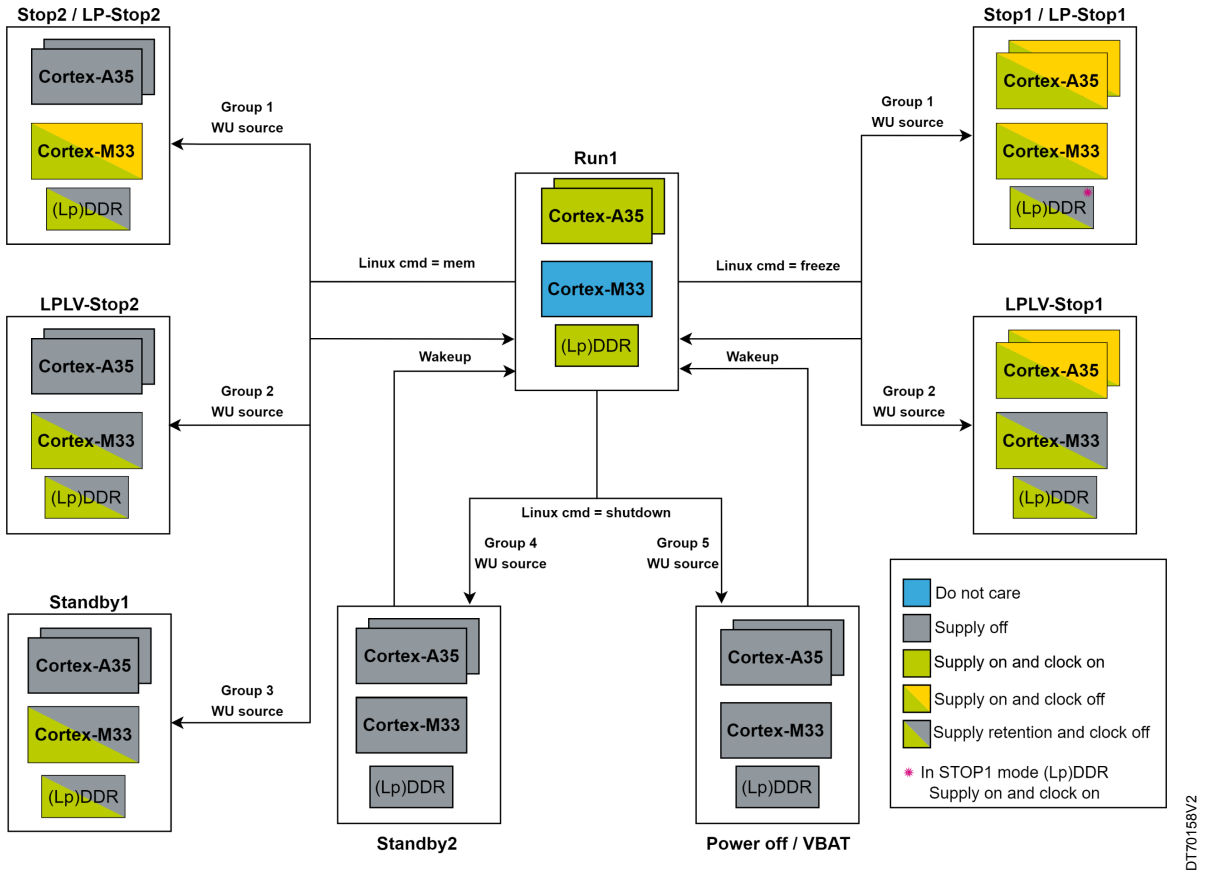
The following figure shows the power state transitions that are available in the system. The same definitions for each wakeup source group used in the table above are considered in this figure.

The wakeup source group with a lower index includes all other groups with an index higher than its own. (refer to Table 4).

If several wakeup sources are activated in different groups, the low-power mode is chosen by respecting the hierarchy stated in the table above.

The choice between Stop1/2 or LP-Stop1/2 is done through settings in the tree of the secure monitor device tree.

Figure 8. Available power state transitions



4.5.2 Controlling MCU power modes under Linux®

Independent clusters

The CPU2 may manage its low-power mode states in its software code.

4.6 Low-power mode exit sequence

For details on how to exit from low-power modes, refer to the corresponding product reference manual sections:

PWR exiting from low-power modes.

When the STM32MP2 device exits from any low-power mode, all the output voltages of the STPMIC2 must be set up to the appropriate level. Moreover, all the output voltages of the STPMIC2 must be stable before the Run mode is applied.

4.6.1 Exit from system power-reset

The external flash memory power supply must be ready and stable before the STM32MP2 device enters the Run mode and reads data from the flash memory to boot.

Note: This condition is automatically handled by the STPMIC2. By definition, the STPMIC2 does not release the STM32MP2 device NRST pin until all supplies have reached their expected value.

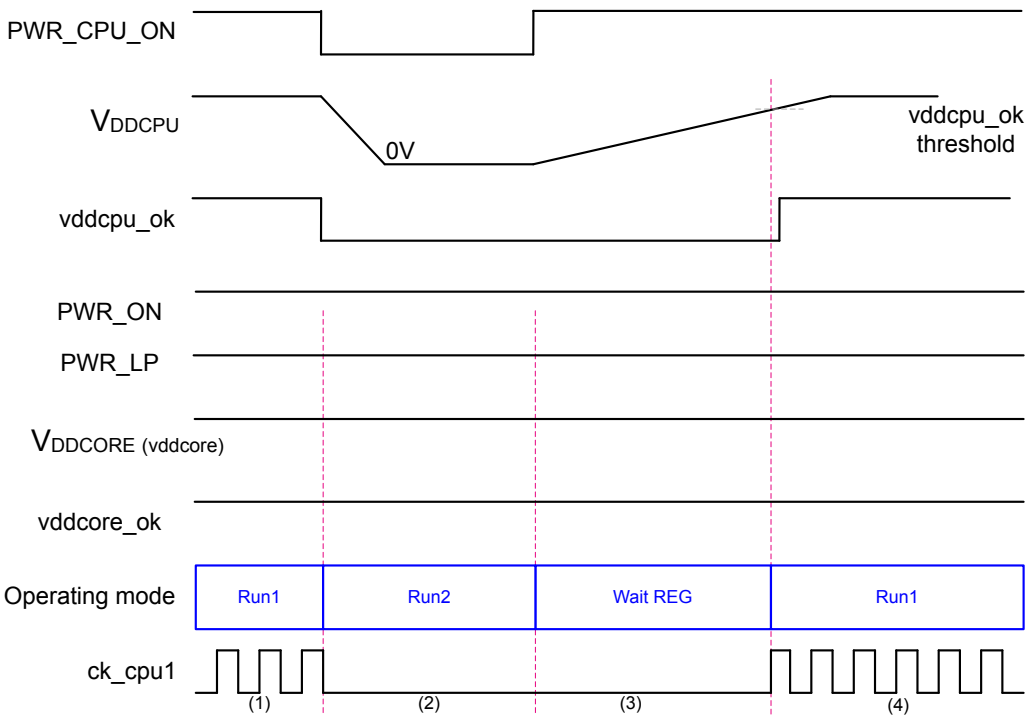
4.6.2 Exit from Run2 mode

When in Run2 mode, V_{DDCORE} is not changed. However, V_{DDCPU} is switched off and needs to be set back to the Run1 mode value.

- The PWR_CPU_ON signal request the V_{DDCPU} supply to be powered off. If needed, a guaranteed minimum PWR_CPU_ON pulse low time can be defined by POPL_D1 bit of the PWR_D1CR register.
- The PWR_ON and PWR_LP are kept high. The STPMIC2 supplying the V_{DDCORE} remains in Main mode.
- As long as V_{DDCPU} is below the vddcpu_ok threshold, the system is kept in Run2 mode. When a wake-up event occurs before the guaranteed minimum PWR_CPU_ON pulse low time, the STPMIC2 instructs the activation of the the V_{DDCPU} supply only after the minimum PWR_CPU_ON pulse low time has elapsed.
- Once V_{DDCPU} supply is above the vddcpu_ok threshold level, the CPU1 clocks are enabled, and the system enters Run1 mode.

The following figure illustrates the wake-up sequence

Figure 9. Wake-up sequence from Run2



4.6.3 Exit from Stop1 mode

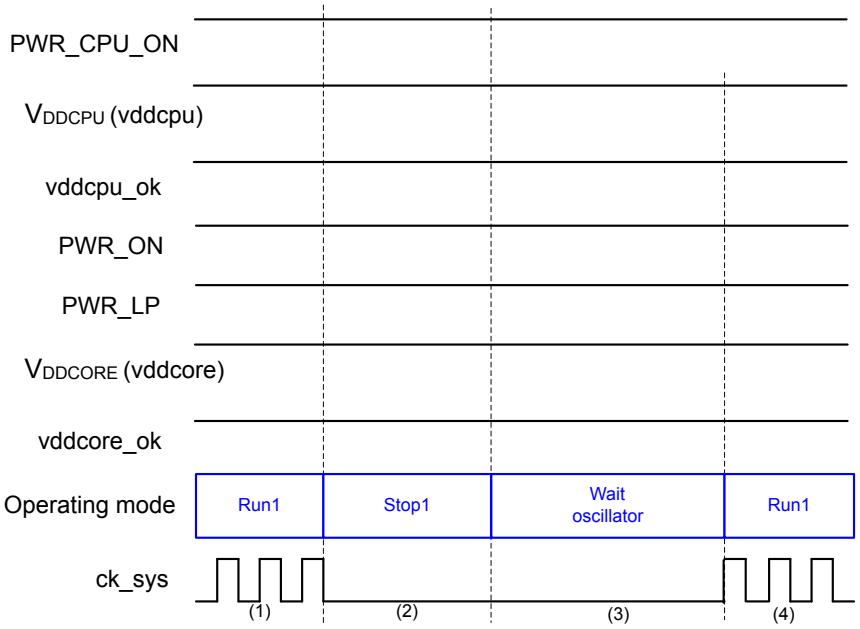
When exiting from Stop1 mode, the voltages are not changed. The RCC re-enables the necessary oscillators and PLLs to reestablish the subsystem clocks by employing the clock restore sequence in the RCC.

This also applies to all other Stop modes: Stop2, LP-Stop1, LP-Stop2, LPLV-Stop1, and LPLV-Stop2.

- In Stop1 mode, the PWR_ON, PWR_CPU_ON, and PWR_LP are kept high. The STPMIC2 supplying the V_{DDCORE} and V_{DDCPU} remain in Main mode.
- On a wake-up event, the selected oscillators are enabled. Once the oscillator is stable, the system enters Run1 mode.

The following figure illustrates the wake-up sequence

Figure 10. Wake-up sequence from Stop1



4.6.4 Exit from Stop2 mode

When the STM32MP2 device is in Stop2 mode:

V_{DDCORE} is not changed

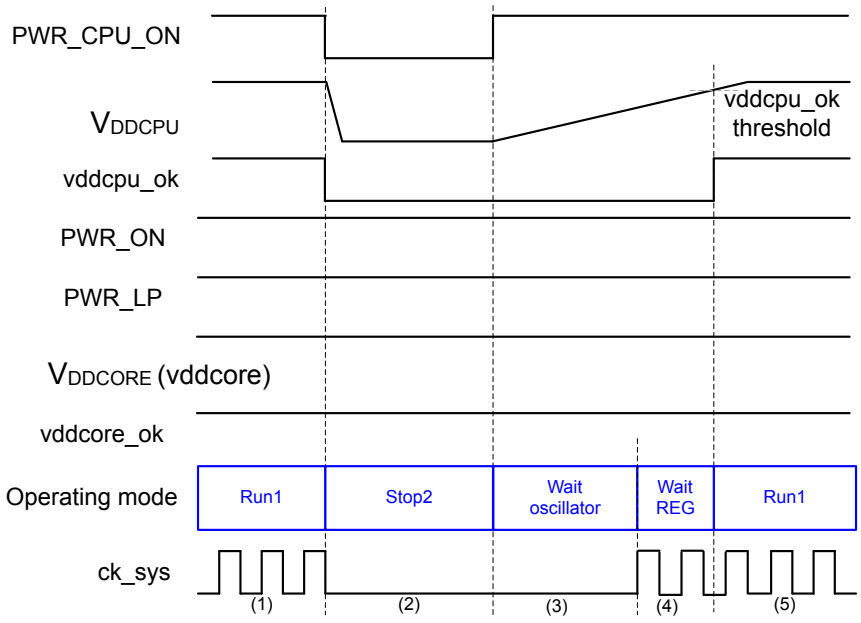
V_{DDCPU} is switched off.

To exit Stop2 mode, V_{DDCPU} is switched off, and must be set back to the Run1 mode value.

- In Stop2 mode the PWR_CPU_ON signal requests the V_{DDCPU} supply to be powered off. If needed, a guaranteed minimum PWR_CPU_ON pulse low time can be defined by POPL_D1 bit of the PWR_D1CR register.
- On a CPU1 wake-up event, the external STPMIC2, providing the V_{DDCPU} supply, is enabled via the PWR_CPU_ON signal, and the selected oscillators are enabled.
- Once the oscillator is stable, and as long as the V_{DDCPU} is below the vddcpu_ok threshold, the CORE domain is kept under system reset.
- Once the V_{DDCPU} is above the vddcpu_ok threshold level, the system enters Run1 mode.

The following figure illustrates the wake-up sequence

Figure 11. Wake-up sequence from Stop2



4.6.5 Exit from LP-Stop1/LPLV-Stop1 mode

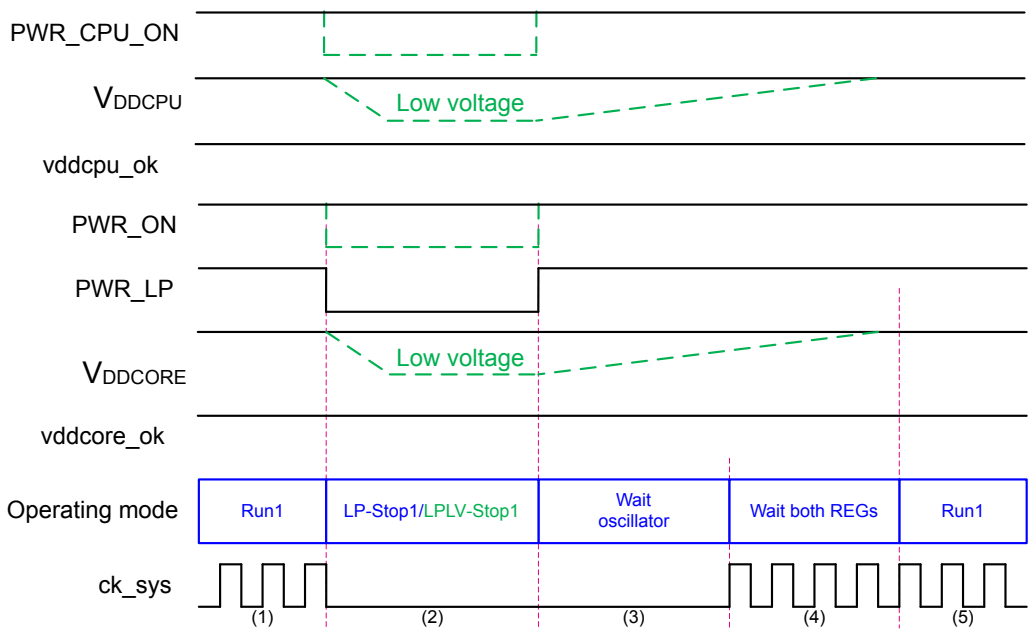
To exit from LP-Stop1, and LPLV-Stop1 mode, V_{DDCORE}, V_{DDCPU} and other voltages at system level must be set back to their Run1 mode value. These voltages may be switched off or reduced.

For example, the DDR4/DDR4L (x32-bit bus width) termination-resistance power supply VTT can be switched off during LP-Stop1, and LPLV-Stop1.

- In LP-Stop1 mode, PWR_CPU_ON and PWR_ON are either kept high or follow the PWR_LP as selected by the LPCFG_D1 bit of the PWR_CPU1CR register and the LPCFG_D2 bit of the in PWR_CPU2CR register. The PWR_LP is de-asserted, signaling LP-Stop1.
  In LPLV-Stop1 where the LVDS bits are set, V_{DDCPU} and V_{DDCORE} supplies can be lowered.
- On a wake-up event, the STPMIC2 is taken out of low-power mode using PWR_LP pin, and the selected oscillators are enabled. In LPLV-Stop1 mode, a $t_{\_LPLVDLY}$, delay as defined by LPLVDLY_D2 bit field of the PWR_D2CR, is started to allow V_{DDCPU} and V_{DDCORE} to reach the Run1 mode operating supply level. After this delay, the oscillators are enabled.
- Once the oscillator is stable, a PWRLP_TEMPO delay to allow the external STPMIC2 to switch from low-power (LP) to high-power (HP) mode, before enabling power-consuming parts. The application controls the delay value via RCC_PWRLPDLYCR register. After the delay timeout, the system enters Run1 mode.

The following figure illustrates the wake-up sequence

Figure 12. Wake-up sequence from LP-Stop1/LPLV-Stop1



Note: In LPLV-Stop1 (green dashed lines) mode, to be able to lower V_{DDCORE} and V_{DDCPU} supplies, the LVDS bits must be set.

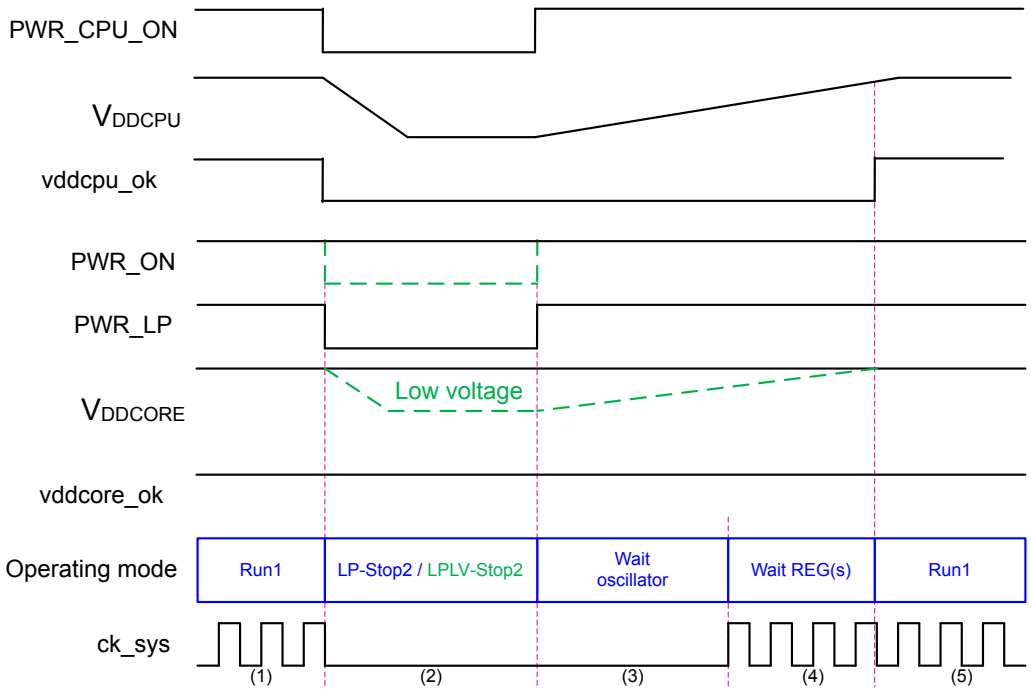
4.6.6 Exit from LP-Stop2 and LPLV-Stop2 mode

The same conditions for the exit from LP-Stop1 and LPLV-Stop1 apply to exit from LP-Stop2 and LPLV-Stop2 mode. Additionally, V_{DDCPU} must be set back to the Run1 mode level.

- The PWR_CPU_ON signal requests the V_{DDCPU} supply to be powered off. The PWR_ON is either kept high or follows the PWR_LP as selected using the LPCFG_D2 bit of the PWR_CPU2CR register. The PWR_LP is asserted, signaling an LP-Stop2.In LPLV-Stop2 (LVDS bits are set), V_{DDCORE} supply can now be lowered.
- On a CPU1 wake-up event, the STPMIC2 providing the V_{DDCPU} supply is enabled via the PWR_CPU_ON signal. The STPMIC2 supplying V_{DDCORE} is taken out of low-power mode via PWR_LP signal. The selected oscillators are enabled. In LPLV-Stop2 mode, a $t_{\_LPLVDLY}$ delay is started to allow V_{DDCPU} to reach the Run mode operating supply level. This is triggered by setting the LPLVDLY_D2 bit of the PWR_D2CR register. After this delay, the oscillators are enabled.
- Once the oscillator is stable, PWRLP_TEMPO is used to allow the external STPMIC2 to switch from low-power (LP) to high-power (HP) mode, before enabling power-consuming parts. The application can control the delay value via RCC_PWRLPDLYCR register.
  - After the delay timeout, and once V_{DDCPU} is above the vddcpu_ok threshold, the system enters Run1 mode.
  - After the delay timeout, and once V_{DDCPU} is above the vddcpu_ok threshold, the system enters Run1 mode.

The following figure illustrates the wake-up sequence

Figure 13. Wake-up sequence from LP-Stop2/LPLV-Stop2



Note: In LPLV-Stop2 (green dashed lines) mode, to be able to lower V_{DDCORE} and V_{DDCPU} supplies, the LVDS bits must be set.

4.6.7 Exit from Standby1 and Standby2 mode

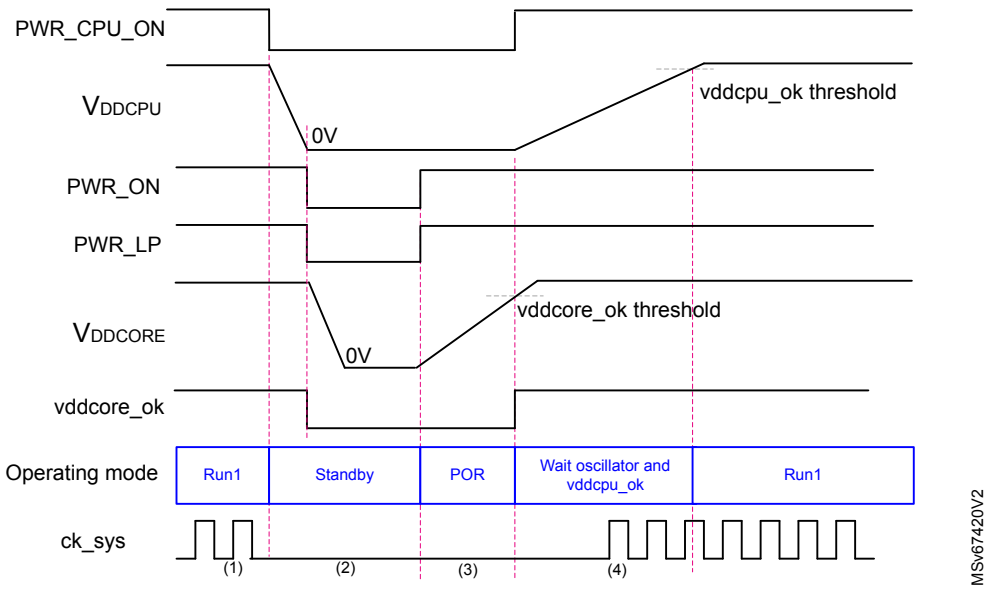
On exit from Standby mode, V_{DDCORE}, and V_{DDCPU} voltages must be switched on and set to their Run mode values. These were turned off in Standby.

- When entering Standby mode, the PWR_CPU_ON signal requests the V_{DDCPU} supply to be powered off, after a delay defined by PODH_D2 in PWR_D2CR. If needed, a guaranteed minimum PWR_ON/ PWR_CPU_ON pulse low time can be defined by POPL_D1/2 in PWR_D1CR/PWR_D2CR.
- On a wake-up event, a request is sent to the STPMIC2 through the PWR_ON signals to supply the V_{DDCORE}. As long as V_{DDCORE} is below the vddcore_ok threshold, the system is kept in Standby mode, and V_{DDCPU} is not requested. When a wake-up event occurs before the guaranteed minimum PWR_ON pulse low time, the STPMIC2 only requested to switch on the V_{DDCORE} supply for the duration of PWR_ON pulse low time.
- Once V_{DDCORE} is above the vddcore_ok threshold level, the HSI oscillator is enabled. The PWR_CPU_ON signal requests the V_{DDCPU} supply to be powered on.

The system does not enter Run1 mode, as long as the V_{DDCPU} is below the vddcpu_ok threshold. This is the case, even when the oscillator is stable.

The following figure illustrates the wake-up sequence from Standby1 and Standby2.

Figure 14. Wake-up sequence from Standby1 and Standby2



The system exits Standby1and Standby2 when at least one of the following events is detected:

- an application reset (such as NRST, IWDG)
- a WKUP pin event
- an EXTI2 wake-up event

The PWR registers are reset after wake-up from Standby1 or Standby2, except for power control and status registers. The reset condition is defined for each PWR register.

A wake-up event source such as an EXTI2 wake-up event on the WKUP pins can be assigned to CPU1 and CPU2 independently. The external regulators that provide V_{DDCPU} and V_{DDCORE} are enabled according to the CPU to wake up but also according to the first CPU which is allowed to boot. Refer to RCC processor state after Standby exit for more details about which CPU is allowed to boot first.

The selection of the CPU to wake-up is notified to the RCC through PWR CPU1_HW_BEN and CPU2_HW_BEN signals. Once the wake-up source is cleared, CPU1_HW_BEN and CPU2_HW_BEN are cleared to 0.

Table 15. **CPU1_HW_BEN and CPU2_HW_BEN management**

Standby exit event	First CPU allowed to boot	CPU1_HW_BEN	CPU2_HW_BEN	PWR_CPU_ON	PWR_ON
CPU1 wake-up event	CPU1	1	0	1	1
	CPU2	1	0	1	1
CPU2 wake-up event	CPU1	0	1	1	1
	CPU2	0	1	0	1
CPU1 and CPU2 wake-up event	CPU1	1	1	1	1
	CPU2	1	1	1	1

4.6.8 Exit from VBAT1/2 mode

When in Standby1 or Standby2 mode, the device can be set in VBAT1 or VBAT2 mode by powering down V_{DD} and $V_{DDA18AON}$ supplies.

The PWR_ON and PWR_CPU_ON signals maintain V_{DDCORE} and V_{DDCPU} supplies powered off.

When powering down V_{DD}, the device enters VBAT mode. PWR_ON, PWR_CPU_ON, and PWR_LP signals become HiZ.

4.6.9 Exit from D1 DStop1 and DStandby

The D1 DStop1 and DStandby have different exit modes.

- When exiting from D1 DStandby with the system in Stop2, LP-Stop2, LPLV-Stop2, program execution restarts with a local CPU1 reset. The boot ROM does not load anything; it calls back an address stored in a backup register (TAMP_BKP11R).
- When exiting from D1 DStandby with the system in Standby1 or Standby2, the program execution restarts similarly to a power-on reset: option bytes loading, reset vector fetched and so on.
- When exiting from D1 DStop1 with the system in Stop1, LP-Stop1, or LPLV-Stop1 mode, the program execution restarts through the interrupt handler. This occurs if the WFI instruction or return from ISR was used to enter into low-power mode.

5 STM32MP2 device peripherals configuration for low-power modes

STM32MP2 devices implement many peripherals. Their configuration plays an important role in power consumption. This section describes how to configure the relevant peripherals for efficient power consumption.

5.1 Peripheral assignment and allocation

There are four available contexts during runtime:

- Cortex®-A35 secure
- Cortex®-A35 non-secure
- Cortex®-M33 secure
- Cortex®-M33 non-secure

Each peripheral can be “assigned” to one or several (shared) of those contexts. This is done by the Extended TrustZone protection controller (RIF).

Each peripheral is assigned at reset time (Cortex®-A35 secure, Cortex®-A35 non-secure, Cortex®-M33 secure, Cortex®-M33 non-secure or shared) according to the Reference manual register content, or during boot time when using STM32MPU OpenSTLinux Distributions illustrated in Figure 16. STM32MP2 lines peripheral assignment using OpentSTLinux distribution. The software can later update this default configuration during runtime.

When one of the contexts wants to use a peripheral assigned to it, it must ensure the peripheral is clocked. So, before using a peripheral, the MPU or MCU must enable it, also it can define if this peripheral remains active in CSleep mode. This clock enabling is done by the RCC and is called “allocation” of peripheral to a processor. The list below defines how to allocate and release a peripheral:

- A peripheral is allocated when dedicated PERxEN is set in RCC_PERxCFGR (PERx is the peripheral x).
- A peripheral is deallocated when dedicated PERxEN is reset in RCC_PERxCFGR. register.
- Activating a peripheral during CSleep is done by setting the dedicated PERxLPEN bit on the RCC_PERxCFGR register.
- Disabling a peripheral during CSleep is done by resetting the dedicated PERxLPEN bit on the RCC_PERxCFGR register.

Refer to the Peripheral allocation inside RCC and General clock concept overview sections of the corresponding reference manual for more details.

When using STM32Cube_FW_MP2, the user can allocate the peripheral and control the peripheral clocks in CSleep mode through the dedicated function defined in STM32MP2xx_hal_rcc.h:

- `__HAL_RCC_xxxx_CLK_ENABLE();`
- `__HAL_RCC_xxxx_CLK_SLEEP_ENABLE();`

The peripheral allocation is used by the RCC to automatically control the clock gating according to the processor modes.

Depending on the operating power-supply range, some peripherals may be used with limited functionality and performance. For more details, refer to the section “General operating conditions” in the corresponding device datasheet.

According to the product RIF policy, each CPU peripheral domain is created based on the following criteria:

- RIF peripheral configuration
- Filters which CPU is allowed to allocate or release a peripheral.
- RCC RIF resource configuration

This concerns peripherals that can be used simultaneously by both CPUs and are mainly the RIF-aware peripherals and the internal memories (see Table 16. Shareable peripherals list).

The RCC RIF resource configuration must be set in “Shared between different masters with semaphore protection” to allow these peripherals to be shared by both CPUs.

Nevertheless, any peripheral listed in the table below can be allocated by one CPU when the related RCC RIF resource configuration is set in “single-access” mode. The application must program the RIF peripheral configuration accordingly.

Note: Any peripheral configured in single-access mode must be released to be available for any other CPU. If not, any RIF configuration change does not work for any new allocation.

Example: If the application wants HPDMA1 to be used only by CPU1, the following steps are needed:

1. *Set HPDMA1 RIF configuration in CPU1 single-access mode.*
2. *Set RCC HPDMA1 RIF configuration in CPU1 single-access mode as per the following example using RCC_R83CIDCFGR:*
   - *CFEN = 1*
   - *SEM_EN = 0*
   - *SCID = CPU1 ID.*

Table 16. Shareable peripherals list

Category	List
Peripherals	• BSEC • DDR(1) • ETR • EXTI1, EXTI2 • FMC • GPIOA, GPIOB, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH, GPIOI, GPIOJ, GPIOK, GPIOZ • HPDMA1, HPDMA2, HPDMA3 • HSEM • IPCC1, IPCC2 • LPDMA1 • PWR(2) • RCC • SYSCFG
Memories	• SYSRAMOCTOSPI1 • OCTOSPI2 • VDERAM • SRAM1, SRAM2 • LPSRAM1, LPSRAM2, LPSRAM3 • RETRAM • BKPSRAM

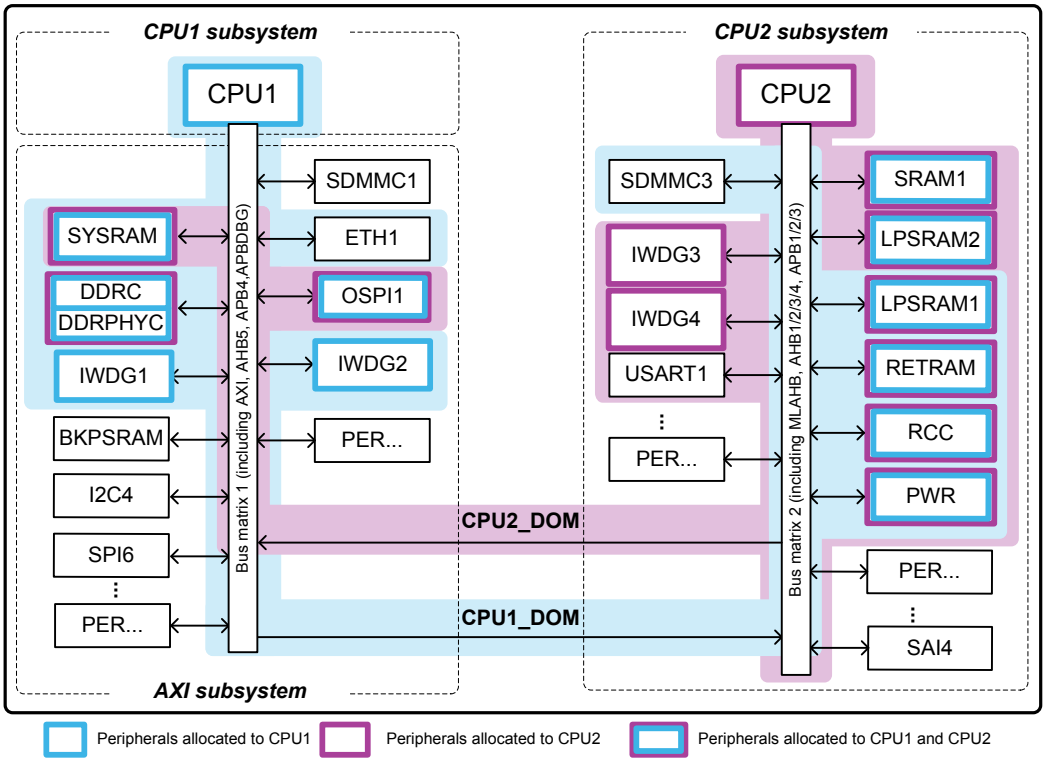
1. *DDR can be set shareable according to DDRSHR in RCC_DDRITFCFGR.*
2. *D2 domain (D3 domain instance is not shareable).*

Figure 15 gives an example of peripheral allocation with the following considerations:

- CPU1 enabled ETH1, SDMMC3, RETRAM, and LPSRAM1. The group is composed of the CPU1, bus matrix 1, bus matrix 2, and peripherals allocated via RCC_PERxCFGR form the CPU1 peripheral domain (CPU1_DOM).
- CPU2 enabled SYSRAM, OSPI1, SRAM1, LPSRAM2 and USART1. The group composed of CPU2, bus matrix 1, bus matrix 2, and peripherals allocated via RCC_PERxCFGR form the CPU2 peripheral domain (CPU2_DOM).

Note: IWDG1 and IWDG2 are only available to CPU1. IWDG3 and IWDG4 are only available to CPU2. PWR and RCC are common resources and allocated to both CPU1 and CPU2. Memories are allocated to both CPUs.

Figure 15. Peripheral allocation example

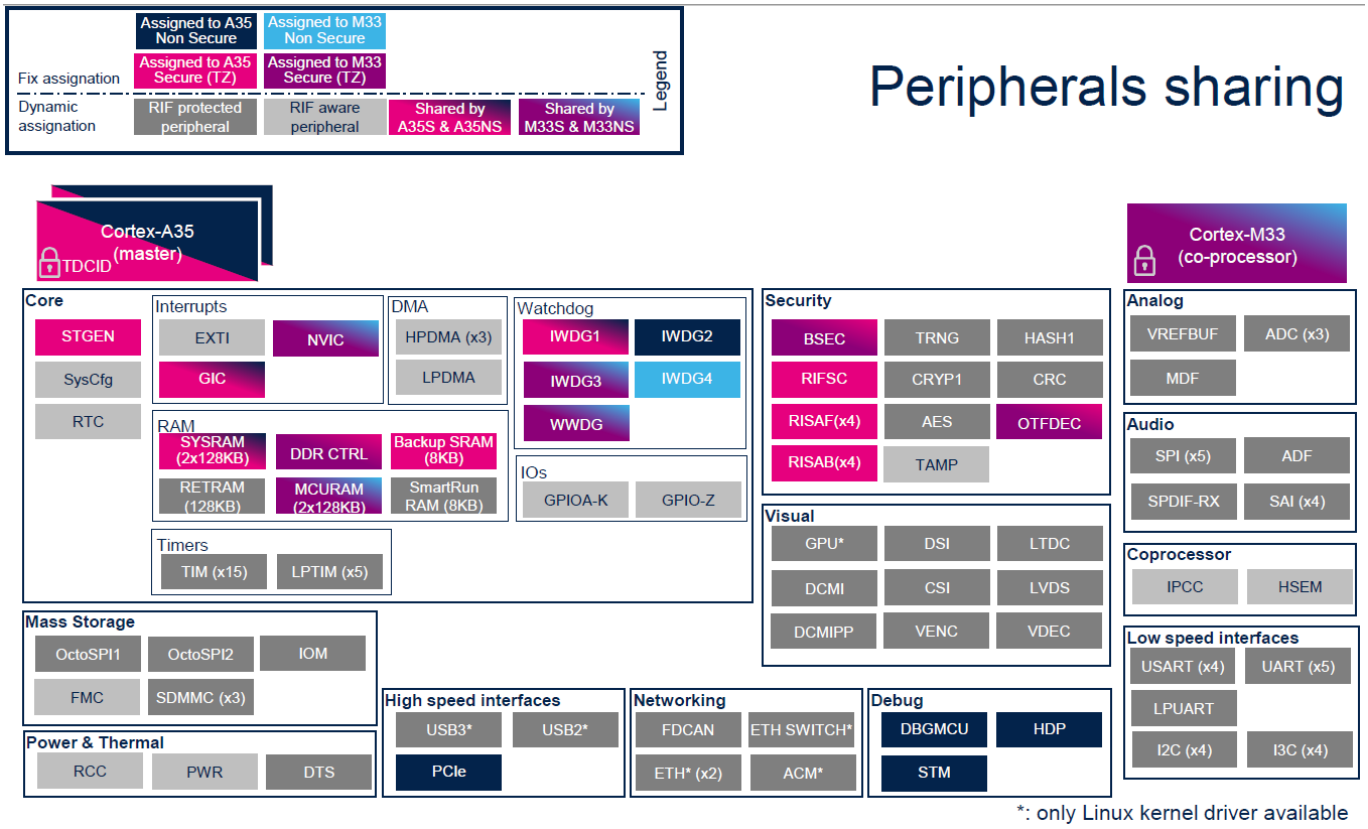


Note: CPU1_DOM and CPU2_DOM can be spread over the two domains.

For D3 peripherals allocated to CPU1_DOM or CPU2_DOM, the CPU can put these peripherals in autonomous mode by setting PERxAMEN in RCC_PERxCFGR. This mode is used for peripherals that must be kept clocked when the system goes in LPLV-Stop1, LPLV-Stop2, or Standby1 mode.

If using ST Linux distribution, Figure 16 shows the list of peripherals and their possible assignment to Cortex®-A35 secure, Cortex®-A35 non-secure, Cortex®-M33 secure, Cortex®-M33 non-secure, Cortex®-M0+ non-secure or Shared context.

Figure 16. **STM32MP2 lines peripheral assignment using OpenSTLinux distribution**



5.2 Peripheral clock distribution

The peripheral clocks are the clocks provided by the RCC to the peripherals. Two kinds of clocks are available:

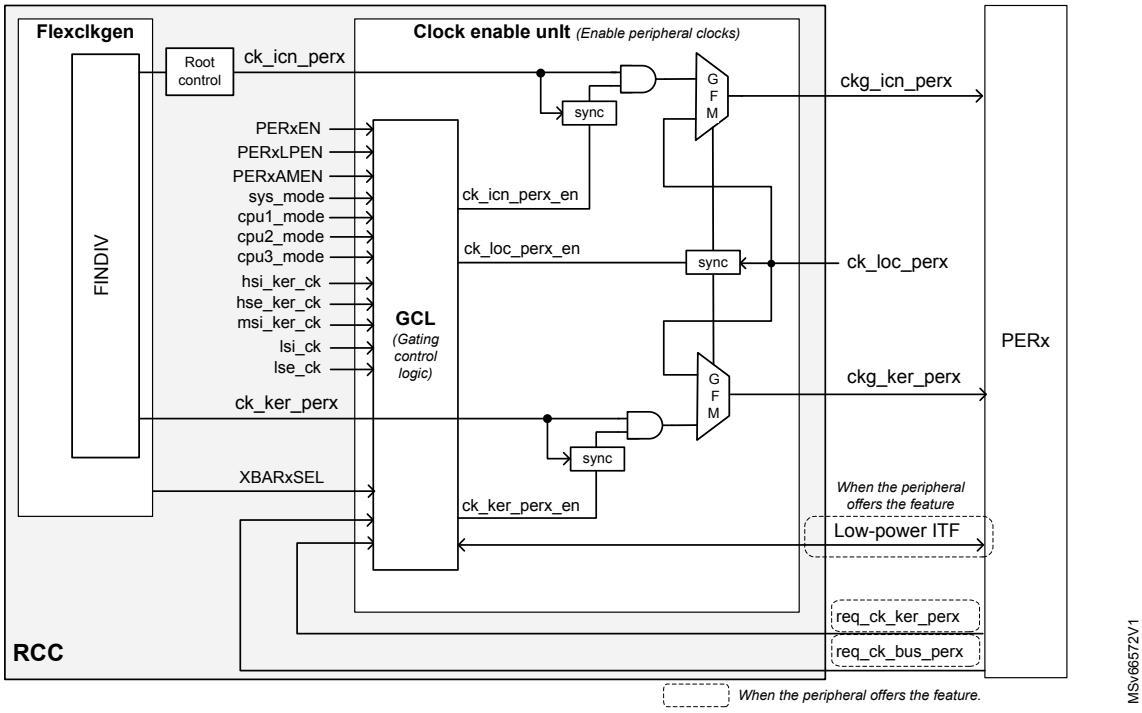
- Bus interface clocks.
- Kernel clocks.

Bus interface and kernel clocks can both be gated according to several conditions detailed hereafter.

As shown in the Figure 17, the enabling of the kernel and bus interface clocks of each peripheral depends on several input signals:

- PERxEN, PERxLPEN, PERxAMEN bits of the RCC_PERxCFGR register
  - PERxEN represents the peripheral enable (allocation) bit for the CPU.
  - PERxLPEN represents the peripheral low-power enable bit.
  - PERxAMEN represents the peripheral autonomous enable bit.
- cpu1_mode, cpu2_mode, cpu3_mode
  - CPU1 state (cpu1_mode) is a logic combination of CPUx states (WFI or WFE) and STPREQ_P[1:0] bits.
  - CPU2 and CPU3 (cpu2_mode, cpu3_mode), their state depends on the low-power core setting.
- sys_mode
  - System mode (sys_mode) is used to filter the clock request of autonomous peripherals.
- Kernel clock request (req_ck_ker_perx) of the peripheral itself (when feature available)
- Bus clock request (req_ck_bus_perx) of the peripheral itself (when feature available) (autonomous peripherals)
- flexclkgen cross bar channel selection (XBARxSEL in RCC_XBARxCFGR)

Figure 17. Peripheral clock enable logic details



A kernel clock allows the peripheral to use a different clock frequency for the peripheral function as compared to the peripheral bus clock.

The Linux clock driver framework offers a 'clk_set_parent' service able to select the kernel clock.

For more details about kernel clock distribution and peripheral clock gating, refer to the section “Peripheral clock gating control” of the reference manual.

The bus interface clock is provided to the peripherals in the following conditions:

- When the CPU to which the peripheral is allocated is in CRun
- When the CPU is in CSleep, with PERxLPEN = 1
- When the CPU is in CStop, system in Run, Stop1, Stop2, LP-Stop1or, LP-Stop2 mode, with PERxLPEN = 1, and the peripheral generates a bus clock request
- When the CPU is in CStop, system in LPLV-Stop1, LPLV-Stop2 or Standby1 mode, with PERxLPEN = PERxAMEN = 1, and the peripheral generates a bus clock request

The kernel clock is provided to the peripherals in the following conditions:

- When the CPU to which the peripheral is allocated is in Crun
- When the CPU is in CSleep, with PERxLPEN = 1
- When the CPU is in CStop, with PERxLPEN = 1, the peripheral generates a kernel clock request, and the selected clock is hsi_ker_ck or msi_ker_ck or hse_ker_ck
- When the CPU is in CStop, with PERxLPEN = 1, and the kernel source clock of the peripheral is lse_ck or lsi_ck
- When the CPU is in CStop, with PERxLPEN = PERxAMEN = 1, the peripheral generates a kernel clock request, and the selected clock is lsi_ck, lse_ck or msi_ker_ck

Note: Only some peripherals can request the kernel clock: I2C, I3C, SPI, U(S)ART, UCPD.

This feature enables the peripheral to wait for an event with optimal power consumption.

6 Debug tips

This section describes how to use the debug features in low-power modes.

6.1 Enabling debugging in low-power mode: low-power mode emulation

The device includes power saving features which allow the various domain clocks and even the core power to be switched off when not required. If the core power is switched off (Standby mode), or the system debug domain is not clocked (Stop mode), none of the debug components are accessible to the debugger. To avoid this, power saving mode emulation is implemented. If emulation is enabled, the device still enters the programmed low-power mode, but its clock and power are maintained. In other words, the domain behaves as if it is in low-power mode, but the debugger does not lose the connection.

The Emulation mode is programmed in the MCU debug (DBGMCU) unit.

When entering a low-power mode, the clock to the processor, the system clock, or the core power supply (only in Standby), can be switched off. This prevents any debug accesses to the unclocked domains.

To simplify debugging of applications which use these power saving features, the clock and power can be maintained or not according to the setting of DBG_STANDBY, DBG_STOP, and DBG_SLEEP in DBGMCU_CR.

Note: The naming DBGMCU is kept for consistency with legacy STM32 MCU products.

Comprehensive information can be found in the following parts of the corresponding STM32MP2 device reference manual:

- *Low-power emulation mode in the RCC chapter*
- *Microcontroller debug unit (DBGMCU) in Debug support chapter*

Note: Using low-power emulation modes has a significant impact on power savings where various clocks and power supplies are kept on. Such modes should only be used for debugging purposes. Power consumption measurements are not relevant while using low-power mode emulation.

6.2 Using HDP for debugging the low-power modes

The hardware debug port (HDP) allows the observation of internal signals used to debug the entry in and the exit from low-power modes. These signals can be the routed to pins of the STM32MP2 device and they can be easily monitored using an osciloscope or a logic analyzer. See the hardware debug port section in the corresponding STM32MP2 series reference manual for the comprehensive list of signals and their mixing configuration.

HDP and associated GPIOs must be disabled in the final application to reach a lower power consumption. The most important signals for power-mode debugging are described in the tables below.

Table 17. Monitoring the CSleep modes

Signal	Description
CPU1 STANDBYWFI0 (HDP7, HDP_MUX=0) CPU1 STANDBYWFE0 (HDP7, HDP_MUX=1)	The CPU1 STANDBYWFI0 in case of CPU1 core 0 WFI, or CPU1 STANDBYWFE0 in the case of CPU1 core 0 WFE signals are used to monitor the CPU1 core 0 CSleep mode. These signals indicate whether a core is in WFI or WFE state.

Table 18. Monitoring the entry in CStop and Stop mode

Signal	Description
RCC pwrds_cpu1 (HDP2, HDP_MUX=6)	The RCC pwrds_cpu1 signal can be used to monitor the CPU1 CStop mode.
RCC pwrds_cpu2 (HDP1, HDP_MUX=6)	The RCC pwrds_cpu2 signal can be used to monitor the CPU2 CStop mode.
RCC pwrds_sys (HDP0, HDP_MUX=6)	The RCC pwrds_sys signal can be used to monitor the STM32MP2Xx device s in one of the Stop modes, and no longer clocked.

Table 19. **Monitoring the exit from the Stop mode**

Signal	Description
PWR pwrwake_sys (HDP0, HDP_MUX=0)	The PWR pwrwake_sys signal can be used to monitor that the EXTI has received a wake-up interruption from a peripheral.
PWR pwrwake_cpu1 (HDP0, HDP_MUX=2)	The PWR pwrwake_mpu signal can be used to monitor that the CPU1 subsystem is requested to return to CRun mode.
PWR pwrwake_cpu2 (HDP0, HDP_MUX=1)	The PWR pwrwake_mpu signal can be used to monitor that the CPU2 subsystem is requested to return to CRun mode.

Refer to https://wiki.st.com/stm32mpu/wiki/HDP_Linux_driver for an example of HDP usage.

6.3 Linux debug hints

Checking clock summary using: `cat /sys/kernel/debug/clk/clk_summary` displays the clock tree in a hierarchical way with frequency and state.

If the system does not wake up, check that at least one wakeup source is enabled, that wakeup peripherals are clocked from either the HSI or the HSE clock and that wakeup pin polarity and pull up/pull down resistors have been set correctly.

If the system does not enter low-power:

- Check that a wakeup event is not already pending when calling the mode.
- `cat /proc/interrupts` can give an indication of such events.
- Check EXTI setup.
  If overall power consumption is too high, check that clocks are released when a peripheral is not used. This can be checked using the `clk_summary` command.

Revision history

Table 20. **Document revision history**

Date	Version	Changes
06-Jun-2024	1	Initial release.

Contents

List of figures

List of tables