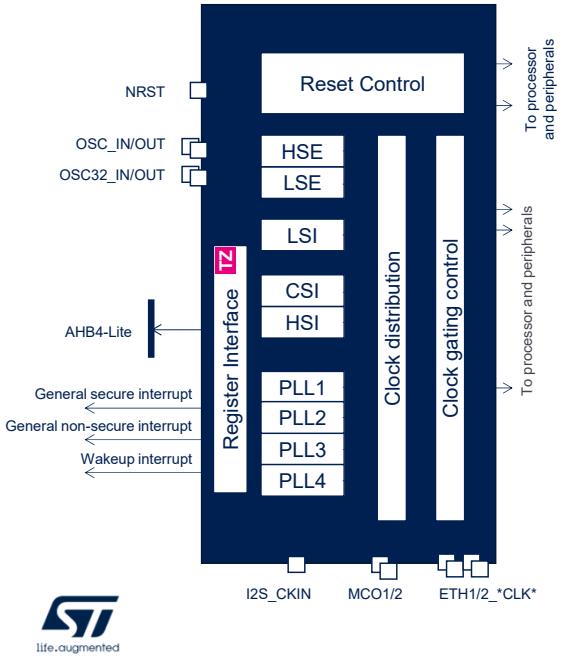
Hello, and welcome to this presentation of the STM32MP13x reset and clock controller (RCC).
The RCC is a complex block working closely with the Power Controller block.
So, before going through this slide set, please have a look at the Power Control module.

Overview

life.augmented

Overview



- The Reset and Clock Controller (RCC) features:
  - Generation and distribution of the clocks for the complete device, based on:
    - 2 external oscillators and 3 internal oscillators
    - 4 PLLs
  - Control of the system reset, application reset and any local reset

Application benefits

- High flexibility of the clock sources choice to meet consumption and accuracy requirements.
- Safe and flexible reset management

The RCC offers several clock sources:

- 3 internal RC oscillators,
- 2 oscillators using an external crystal or resonator,
- And 4 PLLs (phase-locked loop).

The clock distribution block uses all of these clock sources to provide a flexible set of clocks for peripherals, bus interconnects, and the processor.

The clock gating control block is in charge of gating of clocks in order to optimize power consumption.
Many peripherals have their own kernel clock, independent from the bus interface clock, which allows the user to dynamically change the bus interface clock without impacting the device peripheral interface speeds.

The STM32MP13 RCC provides high flexibility in the choice of

clock sources, allowing the application to meet both power consumption and accuracy requirements.

In addition, the RCC also embeds a reset control block, which generates both the local reset for each module of the device, plus system reset and application reset, and may also be used for external devices of the platform.

Finally, the RCC is aware of the secure state of any module of the device so as to control and partition the different resets and clocking as secure or non-secure.
The RCC isolates the secure resources from a non-secure application, via the TrustZone support of its AHB register interface.

Security

Security

Secure partitioning of RCC resources

- The RCC is TrustZone Aware: it enables to control and partition its internal resources as secure or non-secure.
- A read access is always allowed, independently of the secure state of the resource.
- When a resource/function is configured as secure by a secure software, the following access rules are then applied to a RCC register field which impacts this secure resource/function:
  - A write access must be secure
  - A non-secure write access has no impact on a secure register, and if all its secure fields are driven from a single security attribute then a hard fault is generated
  - A non-secure write access to a register combining different secure attributes at a bit field level, has an impact only on the non-secure fields and a hard fault is not generated

life.augmented

The RCC is TrustZone Aware: it enables control and partitioning of its internal resources as secure or non-secure.

A read access is always allowed, independently of security.

When a resource/function is configured as secure by a secure software, the following access rules are then applied to an RCC register field which impacts this secure resource/function:

A write access must be secure

A non-secure write access has no impact on a secure register and if all of its secure fields are driven from a single security attribute, a hard fault is generated.

A non-secure write access to a register that combines different secure attributes at a bit field level has an impact only on the non-secure fields and a hard fault is not generated.

Security

- A RCC register bit/field is either:
  - (Always) secure
    - For clocking/reset of BSEC, DMA3, DMAMUX2, ETZPC, SYSCFG, and TZC.
  - Securable. Its secure state is driven by:
    - A register bit in the TrustZone-aware module itself: for RTC, for RCC
    - A register bit in another TrustZone-aware module: RCC_SECGFR.PWRSEC for PWR
    - A register bit (DECPROT[1]) in the centralized ETZPC module: for most of securable peripherals and SRAMs (c.f. next slide)
  - (Always) non-secure

Security Policy	Peripheral/Memory
Always secure	BSEC
	DMA3
	DMAMUX2
	ETZPC
	SYSCFG
	TZC
TrustZone-aware	RCC
	RTC
RCC-driven securable	PWR

life.augmented

6

The RCC can be fully controlled via registers.

The RCC offers TrustZone capabilities, and manages 3 types of security behavior at the register level or register bit level:

- Always-secure register bits can only be modified by performing write-secure accesses.
- Securable register bits can be switched to secure or non-secure mode.
  Secure configuration bits enable the application to control the security perimeter. They are located either in the TrustZone-aware module itself, as for RCC and RTC, or in the centralized ETZPC module, as we will detail in the next slide.
- And non-secure register bits can be modified by secure and non-secure write accesses.

Security

Security Policy	Peripheral/Memory
Centralized ETZPC-driven securable	ADCx x=1,2
	BKPSRAM
	CRYP
	DCMIPP
	DDRC and DDRPHYC
	ETHx x=1,2
	FMC
	HASH
	I2Cx x=3,4,5
	IWDG1
	LPTIMx x=2,3
	LTDC
	DDRMCE

Security Policy	Peripheral/Memory
Centralized ETZPC-driven securable	PKA
	QSPI
	RNG1
	SAES
	SDMMCx x=1,2
	SPI/I2S4
	SPI5
	SRAMx x=1,2,3
	STGENC
	TIMx x=12,13,…,17
	USARTx x=1,2
	USBO
	USBPHY

life.augmented

This is the list of securable peripherals and SRAMs. The secure state is driven from the corresponding secure configuration bit DECPROT in the centralized ETZPC module.

Security

- The RCC_SECCFGR register allows to allocate and partition separately a sub-function/resource as secure or non-secure, at a field level:
  - Oscillators and source clocks:
    - HSI clock (via HSISEC bit)
    - CSI clock (via CSISEC bit)
    - HSE clock (via HSESEC bit)
    - LSI clock (via LSISEC bit)
    - LES clock (via LSESEC bit)
  - Intermediate clocks:
    - PLL1 and PLL2 clocks (via PLL12SEC bit)
    - PLL3 clock (via PLL3SEC bit)
    - PLL4 clock (via PLL4SEC bit)
    - MPU sub-system clock (via MPUSEC bit)
    - AXI sub-system clock (via AXISEC bit)
    - MLAHB sub-system clock (via MLAHBSEC bit
  - Sink clocks prescalers :
    - APBx divider clock (via APBxDIVSEC x=4,4,5,6 bit)
    - Timers group3 prescaler (via TIMG3SEC bit)
  - Sink/output clocks:
    - Common peripheral clock (via CPERSEC bitPLL1 and PLL2 clocks (via PLL12SEC bit)
    - MCO1 clock (via MCO1SEC bit)
  - Other/miscellaneous features:
    - Reset register-based control (via RSTSEC bit)
    - Stop modes register-based control (via STPSEC bit)
    - PWR block (via PWRSEC bit)

life.augmented

8

The RCC_SECCFGR register allows sub-functions/resources to be allocated and partitioned separately as secure or non-secure, at a bit field level.
The secure perimeter of the RCC can be defined with the following granularity :

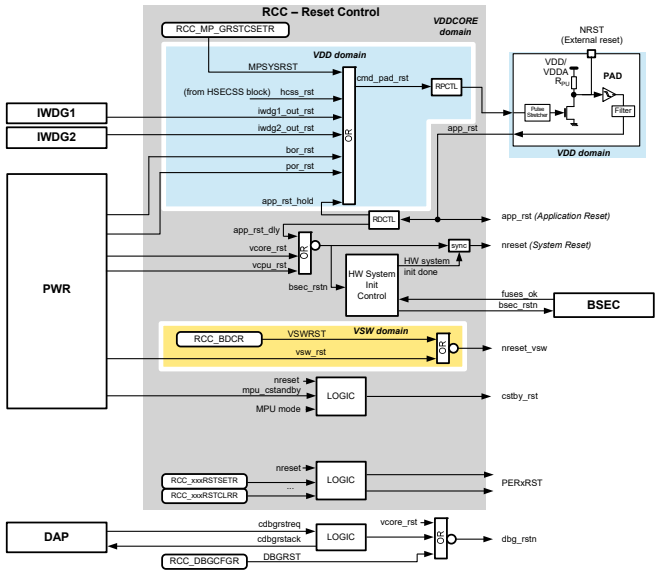
- At oscillator and source clock generation level
- At intermediate clock generation level
- At common clock prescaler level
- At sink and output clock level
- At miscellaneous features level (global reset control, low power stop mode)
- For driving the PWR security state

Reset

Reset Structure

Safe and flexible reset management without external components

- No external components are needed due to internal filter and power monitoring
- Application reset sources can reset external components

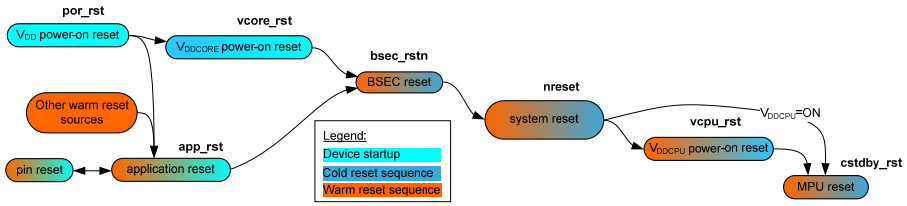


Thanks to voltage monitoring included in the PWR block, the filters embedded in the NRST PAD and RCC reset control, the number of external components is significantly reduced.
Note that the application reset asserts the PAD NRST, allowing reset of external components.

The block named RPCTL (Reset Pulse Control) allows the application to define the minimum activation time of the NRST pad.
The block named RDCTL (Reset Delay Control) allows the application to delay the activation of the system reset until the DDR is in self-refresh mode.
These two blocks will be explained later.

The Reset control part of the RCC interacts with other modules as shown in this block diagram. The slides that follow detail the generation of the global and local reset signals.

System Reset



- A **system reset** (nreset) is generated after a warm reset or a soft reset.
  - A **warm reset** occurs from either:
    - A **pin reset** (NRST)
    - An **application reset** (app_rst) which is not caused by a V_{DD} power-on reset (por_rst).
  - A **cold reset** occurs after a V_{DDCORE} power-on reset (vcore_rst) and the BSEC reset (bsec_rstn), that is to say after option byte loading and hardware system initialization.
    - When the cold reset sequence is initiated by the main V_{DD} power-on reset (por_rst), the reset sequence is also known as the device startup.
- The system reset is followed by a V_{DDCPU} power-on reset (vcpu_rst) and a MPU reset (cstdby_rst).

life.augmented

11

A **system reset** (nreset) is generated after a warm reset or a soft reset.

A **warm reset** results from either:

A **pin reset** (NRST)

Or an **application reset** (app_rst) which is not caused by a V_{DD} power-on reset (por_rst).

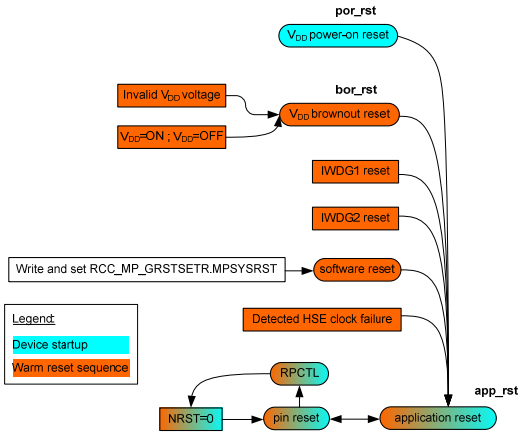
A **cold reset** occurs after a V_{DDCORE} power-on reset (vcore_rst) and the BSEC reset (bsec_rstn), that is to say after option byte loading and hardware system initialization.

When the cold reset sequence is initiated by the main V_{DD} power-on reset (por_rst), the reset sequence is also known as the device startup.

The system reset is followed by a V_{DDCPU} power-on reset (vcpu_rst) and an MPU reset (cstdby_rst).

Application Reset

- An application reset is generated from one of the following sources:
  - a V_{DD} power-on reset (**por_rst**)
  - a V_{DD} brownout reset (**bor_rst**)
  - a reset from the independent watchdog 1
  - a reset from the independent watchdog 2
  - a software reset when the Cortex-A7 (MPU) writes and sets the RCC_MP_GRSTECSETR.MPSYSRST
  - a detected clock failure on HSE
  - a pin reset (NRST)
- An application reset can cause a pin reset for external device(s), via a programmable reset pulse control (RPCTL).



life.augmented

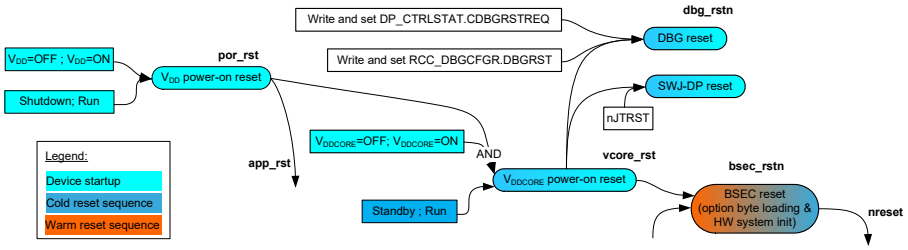
An application reset is generated from one of the following sources:

a V_{DD} power-on reset (**por_rst**)
a V_{DD} brownout reset (**bor_rst**)
a reset from the independent watchdog 1
a reset from the independent watchdog 2
a software reset when the Cortex-A7 (MPU) writes and sets the RCC_MP_GRSTECSETR.MPSYSRST
a detected clock failure on HSE
Or a pin reset (NRST)

An application reset can cause a pin reset for external device(s), via a programmable reset pulse control (RPCTL).

V_{DDCORE} power-on reset

- A V_{DDCORE} power-on reset causes also
  - reset of DBG module
  - reset of SWJ-DP JTAG port
    - possibly externally reset from nJTRST pad.



- When the system is in Standby mode, the V_{DDCORE} and the V_{DDCPU} are switched off, but V_{DD} is still present.
  - When the system exits from **Standby to Run mode** via WKUPx pin(s) or via a RTC event or via an application reset resulting from a V_{DD} brownout reset, an IWDG1/2 reset or a pin reset, the cold boot sequence occurs: V_{DDCORE} power-on reset, DBG/JTAG reset, BSEC reset, system reset, V_{DDCPU} power-on reset and MPU reset.

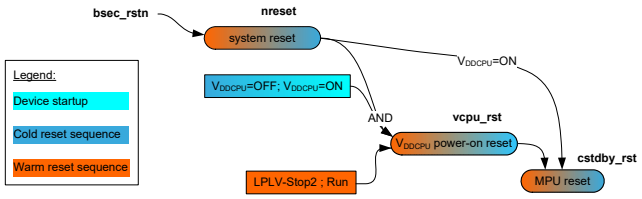
life.augmented

13

A V_{DDCORE} power-on reset also causes
reset of the DBG module
reset of the SWJ-DP JTAG port
And possibly external reset from nJTRST pad.

When the system is in Standby mode, the V_{DDCORE} and the V_{DDCPU} are switched off, but V_{DD} is still present.
When the system exits from **Standby to Run mode** via WKUPx pin(s) or via an RTC event or via an application reset resulting from a V_{DD} brownout reset, an IWDG1/2 reset or a pin reset, the cold boot sequence occurs: V_{DDCORE} power-on reset, DBG/JTAG reset, BSEC reset, system reset, V_{DDCPU} power-on reset and MPU reset.

V_{DDCPU} power-on reset



- On a system reset
  - If cold reset, V_{DDCPU} power-on reset occurs after that the supply V_{DDCPU} is established.
  - If warm reset, there is no V_{DDCPU} power-on reset, but MPU reset.
- When the system is in LPLV-Stop2 mode, V_{DDCPU} is switched off, but V_{DD} and V_{DDCORE} are still present.
  - When the system exits from LPLV-Stop2, only the V_{DDCPU} power-on reset and the MPU reset are generated.

life.augmented

On a system reset

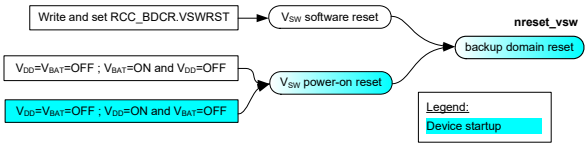
If it is a cold reset, V_{DDCPU} power-on reset occurs, after the supply V_{DDCPU} is established.

If it is a warm reset, there is no V_{DDCPU} power-on reset, but there is an MPU reset.

When the system is in LPLV-Stop2 mode, V_{DDCPU} is switched off, but V_{DD} and V_{DDCORE} are still present.

When the system exits from LPLV-Stop2, only the V_{DDCPU} power-on reset and the MPU reset are generated.

Backup Domain Reset



- A backup domain reset is generated following either:
  - a power-on reset of the V_{SW} domain:
    - driven by the sub-part of the PWR controller in the V_{SW} domain.
    - a V_{SW} power-on reset occurs on a V_{DD} or V_{BAT} power-on, provided that both supplies have been previously powered off.
    - It does reset the V_{SW} domain. Regarding RCC, it means it does reset the clock data path from the LSE source clock to the RTC kernel clock, and does reset the RCC register in the V_{SW} domain, i.e. the RCC_BDCR register. The BKPSRAM is not impacted.
  - a backup domain software reset: when is written and set RCC_BDCR.VSWRST=1.
    - It does reset the V_{SW} domain. Regarding RCC, it means it does reset the clock data path from the LSE source clock to the RTC kernel clock, and does reset the RCC_BDCR register, excepted the VSWRST bit. The BKPSRAM content is no longer valid.

life.augmented

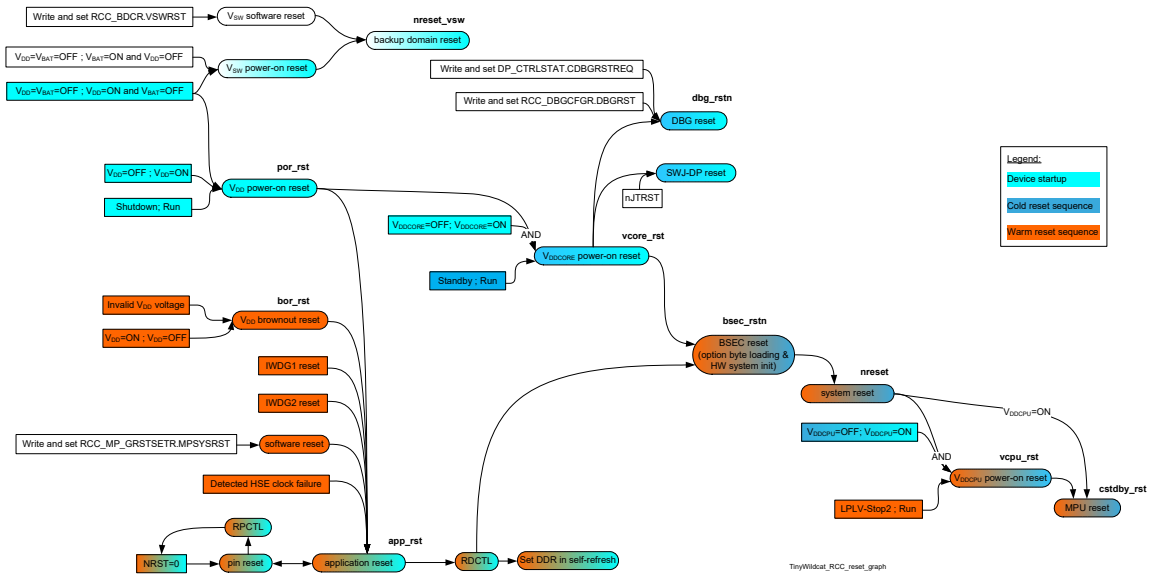
15

A backup domain reset is generated following:

- Either a power-on reset of the V_{SW} domain:
  driven by the sub-part of the PWR controller in the V_{SW} domain.
  A V_{SW} power-on reset occurs on a V_{DD} or V_{BAT} power-on, provided that both supplies have been previously powered off.
  It resets the V_{SW} domain. Regarding RCC, this means that it resets the clock data path from the LSE source clock to the RTC kernel clock, and resets the RCC register in the V_{SW} domain, i.e. the RCC_BDCR register. The BKPSRAM is not impacted.
- Or a backup domain software reset: when written and set, RCC_BDCR.VSWRST=1.
  It resets the V_{SW} domain. Regarding RCC, this means that it resets the clock data path from the

LSE source clock to the RTC kernel clock, and resets the RCC_BDCR register, except for the VSWRST bit. The BKPSRAM content is no longer valid.

Full Reset Sequence



Here is the full reset sequence with all the previous parts in the full reset sequence diagram.

It includes device startup, cold reset, warm reset, application reset, system reset, and different local resets.
Following the different user-actions, including device changes at the platform level.

Power Reset

- The PWR block provides to the RCC via its supply monitoring capabilities:
  - The **por_rst** signal for when $V_{DD} < V_{POR}$ threshold
  - The **bor_rst** signal for when $V_{DD} < V_{BOR}$ threshold
  - The **vcore_rst** signal for when $V_{DDCORE} < V_{POR}$ threshold
  - The **vsw_rst** signal for when $V_{SW} < V_{SW}$ threshold
  - The **vcpu_rst** signal for when $V_{DDCPU} < V_{DDCPU}$ threshold

life.augmented

17

The PWR block provides certain reset signals to the RCC:
 The **por_rst** signal, which is asserted when the VDD supply is below the VPOR
threshold.
 The **bor_rst** signal, which is asserted when the VDD supply is below the VBOR
threshold.
 The **vcore_rst** signal, which is asserted when the VDDCORE supply is below the
VDDCORE threshold.
 The **vsw_rst** signal, which is asserted when the VSW supply is below the expected
threshold.
 And the **vcpu_rst** signal, which is asserted when the VDDCPU supply is below the
VDDCPU threshold.

Reset Coverage

Impacted functions by the reset	por_rst	vcore_rst	app_rst	nreset	vcpu_rst	cstby_rst	dbg_rstn	bsec_rstn	PERxRST	nreset_vsw
VDD domain	✓	-	-	-	-	-	-	-	-	-
VDDCORE domain (including AXI/AHB interconnect)	✓	✓	-	-	-	-	-	-	-	-
DDR Interface	✓	✓	✓	✓	-	-	-	-	-	-
VDDCPU domain	✓	✓	✓	✓	✓	✓	-	-	-	-
Debug components (DBGMCU, MPU debug)	✓	✓	-	-	-	-	✓	-	-	-
SWJ-DP	✓	✓	-	-	-	-	✓	-	-	-
IWDGx (x= 1,2)	✓	-	✓	-	-	-	-	-	-	-
Hardware system initialization and loading of option bytes	✓	✓	✓	-	-	-	-	✓	-	-
RCC registers (at least some)	✓	✓	✓	✓	-	-	-	-	-	✓
PWR registers (at least some)	✓	✓	✓	✓	-	-	-	-	-	✓
RTC and BKPSRAM	-	-	-	-	-	-	-	-	-	✓
Peripherals	✓	✓	✓	✓	-	-	-	-	✓	-

life.augmented

This table is a simplified view of the parts reset by the most important reset sources.

The power-on reset (por_rst) is the reset with the widest coverage.
It resets all of the logic located in the VDD domain, and most of the logic of VDDCORE.
The logic located in the VSW domain is not reset.
As seen in the reset diagram, the power-on reset includes the main VDD power-on reset, the VDDCORE power-on reset and the VDDCPU power-on reset. It also triggers the system reset and app_rst, and the NRST PAD is asserted during the power-on reset, as well as the MPU reset.

The application reset (app_rst) resets most of the logic located in the VDDCORE domain except for certain resources located in the RCC and PWR. The backup and VSW domains are not

affected by this reset.

The system reset (nreset) resets most of the logic located in the VDDCORE domain except for certain resources located in the RCC and PWR.
Debug components, IWDG[2:1] and the backup and VSW domains are not affected by this reset.

The vcore reset (vcore_rst) resets most of the logic located in the VDDCORE domain except for certain resources located in the RCC and PWR.
The IWDG[2:1] and the backup and VSW domains are not affected by this reset.

The backup domain reset (nreset_vsw) resets all the components located in the Backup VSW domain which contains the RTC and the external low-speed oscillator, the backup RAM, and the retention RAM.

Reset of the RCC registers

RCC register	Voltage domain	Reset condition	Comment
RCC_RDLSICR	V_{DD}	por_rst	Control / status of the reset and the LSI clock. Reset by main V_{DD} power-on reset. It initiates the device startup and includes a system reset.
RCC_BR_RSTSCLRR			
RCC_BDCR but VSWRST	V_{SW}	nreset_vsw	Control / status of the backup domain. Reset by a backup domain reset. It is initiated by a V_{SW} power-on reset or a write and set RCC_BDCR.VSW=1
RCC_BDCR.VSWRST		RCC_BDCR.VSWRST=0	Reset by a write and clear RCC_BDCR.VSW=0
Others	V_{DDCORE}	nreset	System reset

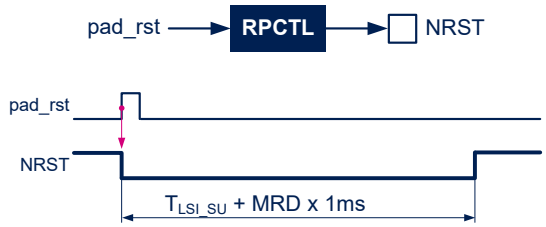
life.augmented

19

This table lists the specific registers which are not inside the VDDCORE domain and consequently are not reset by a system reset.
In the VDD domain, there are the RCC_RDLSICR and RCC_BR_RSTCLRR registers for the control and status of the reset and the LSI clock.
In the VSW domain, there is the RCC_BDCR register for the control and status of the RTC clock, the LSE clock, and the backup domain reset

Reset Pulse Control

Reset duration can be adapted to external component constraints



- Adjustable minimum pulse reset duration:
  - From 1 to 31 ms
  - RPCTL block can bypassed
  - Accuracy given by LSI oscillator

life.augmented

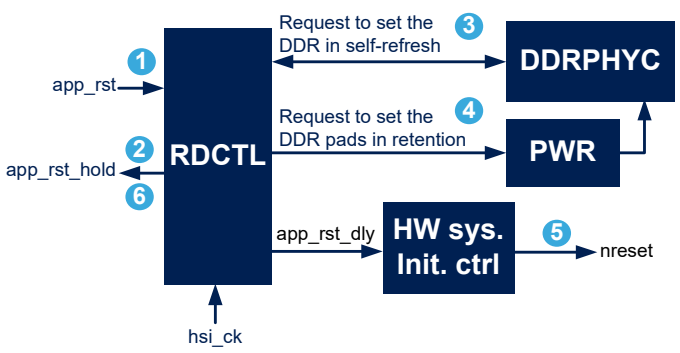
The reset pulse control block allows the application to control the minimum activation time of the NRST pad.
This feature is particularly helpful because some external devices may request a minimum reset duration for the activation of the NRST.

The reset pulse duration can be adjusted from 1 to 31 ms with the accuracy of the LSI oscillator.
If the LSI is switched off when a reset occurs, the LSI is switched on as long as it is needed by the RPCTL control register.

Reset Delay Control

Reset can be delayed until the DDR is in Self-Refresh

- Possibility to set the DDR in self-refresh mode when an application reset occurs



- The RDCTL delay the propagation of the application reset
- The RDCTL requests to switch the DDR into self-refresh mode
- Once it is done, the RDCTL propagates the application reset
- In case of timeout, the application reset is propagated.

life.augmented

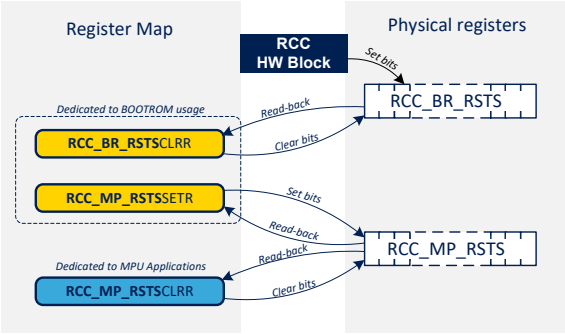
21

The reset delay control block allows the application to set the external DDR device in the self-refresh mode before generating a system reset (nreset) to the circuit.

When an application reset occurs (see item 1), the RDCTL maintains this reset active, and prevents the propagation of the system reset (see item 2).
The RDCTL then requests the DDR PHY to switch the DDR into self-refresh (see item 3).
Once this has been done, the RDCTL asks the PWR to switch the DDR pads to retention (see item 4),
Finally, the system reset is propagated (item 5), and the application reset is no longer maintained (see item 6),
If the DDRPHY does not reply to the self-refresh request, the system reset is propagated to the circuit after a user-progammable timeout delay.

The RDCTL block can also be bypassed if required.

Reset Source Identification



1. The RCC hardware updates the reset status physical register RCC_BR_RSTS identifying the reset source, for the BOOTROM.
2. The BOOTROM can
   - Read the reset status physical register via a read to RCC_BR_RSTSCLR
   - Update the reset status physical register (RCC_MP_RSTS) for the MPU application. By setting bits via writing 1 to respectively RCC_MP_RSTSETR bits.
   - Clear the asserted bits of the physical register RCC_BR_RSTS by writing 1 into RCC_BR_RSTS_CLRR bits.
3. The MPU application can
   - Read the reset status physical register RCC_MP_RSTS (via RCC_MP_RSTSCLRR or RCC_MP_RSTSSETR)
   - Clear the asserted bits of the physical register RCC_MP_RSTS by writing 1 into RCC_MP_RSTCLRR bits.

life.augmented

22

The RCC stores the reset root cause in the physical register RCC_BR_RSTS.

The BOOTROM can check and acknowledge the reset root cause via the RCC_BR_RSTCLRR register.
In addition, the BOOTROM updates the physical register RCC_MP_RSTS via the RCC_MP_RSTSETR register.

The MPU can check and acknowledge the reset root cause via the RCC_MP_RSTCLRR register.

Clocking

life.augmented

Clock Key Features

Choice of clock sources for low-power, accuracy, and performance

- Three internal clock sources
  - High-speed internal (up to) 64 MHz RC oscillator (HSI)
  - Low-Power internal 4 MHz RC oscillator (CSI)
  - Low-speed internal 32 kHz RC oscillator (LSI)
- Two external oscillators
  - High-speed external 8 to 48 MHz oscillator (HSE) with clock security system
  - Low-speed external 32.768 kHz oscillator (LSE) with clock security system
- 4 PLLs, each with three independent outputs

life.augmented

Focusing on the clocking part, the RCC offers a wide choice of clock sources, which can be selected according to low-power, accuracy, and performance requirements.
The STM32MP13 microprocessor embeds 3 internal RC oscillators:

- a high-speed internal RC oscillator (HSI) which can operate at 64, 32, 16 or 8 MHz,
- a low-power internal RC oscillator (CSI), operating at 4 MHz,
- And a low-speed internal 32 kHz RC oscillator (LSI).

It also embeds 2 oscillators working with external crystals or resonators:

- a high-speed external 8 to 48 MHz oscillator (HSE) with a clock security system and
- a low-speed external 32.768 kHz oscillator (LSE) also with a clock security system.

4 PLLs are available as well, each with three independent outputs for clocking different peripherals at different frequencies.
Note that the USB has its own PLL.

High-Speed Internal (HSI) Clock

+/-1% accuracy, high-speed and fast wakeup time (few µs)

- HSI clock is factory trimmed and can be user-trimmed.
- HSI clock can be 64, 32, 16 or 8 MHz.
- HSI clock is the default system clock after system reset
- In any Stop mode
  - HSI clock can be selected as wakeup clock.
  - Most of peripherals can enable the HSI clock when they need it as kernel clock.
  - HSI oscillator can remain active to avoid the start-up delay

life.augmented

25

The high-speed internal oscillator is a 64 MHz RC oscillator with a typical accuracy in the order of +/-1% over temperature, and a fast wakeup time (of a few microsecs at most).
With a typical dynamic consumption around 320 uA.
The HSI clock is trimmed during production testing, and can also be user-trimmed.
A dedicated divider allows the generation of a 64, 32, 16 or 8 MHz clock.

The HSI clock is the default system clock after system reset.

The HSI clock is selected as the system clock for the MCU after the system reset.

In any Stop mode :
The HSI clock can be selected as wakeup clock.
Most of the peripherals can request activation of the HSI clock

for their own processing when needed:
- The HSI oscillator can remain active when the system goes into Stop mode in order to avoid oscillator wake-up time and provide the source clock as soon as possible as a peripheral kernel clock.
- Or else, if the HSI oscillator was configured as OFF, the RCC turns the HSI oscillator ON and provides the HSI clock to the peripheral when requested.

Low-Power Internal (CSI) Clock

+/-5% accuracy, low-power and fast wakeup time (few µs)

- CSI clock is factory trimmed and can be user-trimmed.
- In any Stop mode
  - CSI clock can be selected as wakeup clock.
  - Most of peripherals can enable the CSI clock when they need it as kernel clock.
  - CSI oscillator can remain active to avoid the start-up delay

life.augmented

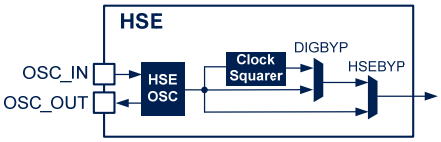
The low-power internal oscillator is a 4 MHz RC oscillator with a typical accuracy of +/-5% over temperature and supply voltage, and a fast wakeup time (of a few microsecs maximum).
With a typical dynamic consumption around 25 µA.
The CSI clock is trimmed during production testing, and can also be user-trimmed.

In any Stop mode:
The CSI clock can be selected as wakeup clock.
Most of the peripherals can request activation of the CSI clock for their own processing when required:
- The CSI oscillator can remain active when the system goes into Stop mode in order to speed up the oscillator wake-up time and provide the source clock as soon as possible as a peripheral kernel clock.
- Or else if the CSI oscillator was configured as OFF, the RCC

turns the CSI oscillator ON and provides the HSI clock to the peripheral when requested.

High-Speed External (HSE) clock

Safe crystal system clock (few ms wakeup time)



- HSE 8-48MHz, can generate a clock from:
  - External digital source
  - External analog source
    - Down to 200 mVpp amplitude
  - External crystal/ceramic resonator (4 - 48 MHz)
- Clock Security System (CSS)
  - Automatic detection of HSE clock failure. Then:
    - An application reset is generated
    - A failure event is generated to protect backup registers and BKPSRAM
- In any Stop mode
  - HSE clock can be selected as wakeup clock.
  - Most of peripherals can enable the HSE clock when they need it as kernel clock.
  - HSE oscillator can remain active to avoid the start-up delay

life.augmented

27

A crystal or ceramic resonator working in the range of 8 to 48 MHz can be connected to the HSE clock.
The HSE can also be set in digital bypass mode, and receives a digital clock on the OSC_IN input.
In analog bypass mode, a clock with a reduced amplitude, down to 200 millivolts peak to peak, can be used.

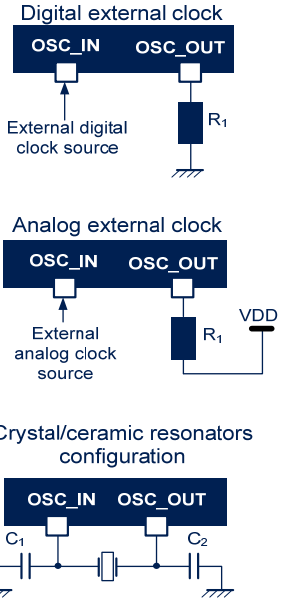
HSE features a maximum critical gm of 1.5m A/V, a wake-up time typically of a few milliseconds, and a typical consumption of 0.5 mA with a 24 MHz crystal.

A clock security system allows automatic detection of any HSE failure.
When this is the case, an application reset is generated, and a failure event is provided to the TAMP block, allowing the protection of backup registers and the backup RAM.

In any Stop mode:
The HSE clock can be selected as the wake-up clock.
Most of the peripherals can request activation of the HSE clock for their own processing when required:
- The HSE oscillator can remain active when the system goes into Stop mode in order to speed up the oscillator wake-up time and provide the source clock as soon as possible as a peripheral kernel clock.
- Or else, if the HSE oscillator was configured as OFF, the RCC turns the HSE oscillator ON and provides the HSE clock to the peripheral when requested.

High-Speed External (HSE) clock

Smart detection of HSE mode



- The BOOTROM detects the HSE configuration:
  - If a pull-down is connected to OSC_OUT, then the BOOTROM sets the HSE clock in digital bypass mode
  - If a pull-up is connected to OSC_OUT, then the BOOTROM sets the HSE clock in analog bypass mode
  - Otherwise the BOOTROM sets the HSE in oscillator mode

life.augmented

28

The BOOTROM must use the HSE oscillator when a boot is requested via the USB port.
For that purpose, the BOOTROM needs to know how the HSE clock is used in the product so as to configure it properly.

The OSC_OUT must be connected to ground via a resistor in order to inform the BOOTROM that an external digital clock is provided as the HSE clock.
The OSC_OUT must be connected to VDD via a resistor in order to inform the BOOTROM that an external analog clock is provided as the HSE clock.
If pull-up or pull-down are not detected, the BOOTROM configures the HSE in oscillator mode.

Low-Speed Internal (LSI) clock

+/-5% accuracy, ultra-low power internal 32 kHz oscillator (few 100 µs wakeup time)

- The LSI clock is used for some internal functions, and can be used for peripherals such as RTC, LPTIMs, IWDGs, RNGs and SAES.
- LSI clock is available in all modes except VBAT mode and can remain active during Stop and Standby modes.

life.augmented

29

The STM32MP13 microprocessor embeds an ultra-low-power 32 kHz RC oscillator, which is available in all modes except in VBAT mode.

LSI features a typical accuracy over temperature and VDD of +/- 5%, a maximum startup time of 170 µs, and a typical dynamic consumption of 130 nA.

The LSI clock can be used to clock the RTC, LPTIMs, IWDGs, RNGs and SAES.

Low-Speed External (LSE) clock

32.768 kHz configurable for low-power or high-drive (few sec wakeup time)

- LSE clock can be generated from:
  - An external digital source
  - An external analog source
  - An external crystal/ceramic resonator
- Selectable driving capability according to the external crystal characteristics
- Available in all power modes and in VBAT mode
- Clock Security System on LSE is available in all modes except VBAT.
- The LSE clock can be used for RTC, LPTIMs, RNG1 and DTS.

life.augmented

30

The 32.768 kHz low-speed external oscillator can be used with external crystal or resonator, or with an external clock source in bypass mode.
The oscillator driving capability is selectable in order to adapt the oscillator characteristics to the external crystal device.

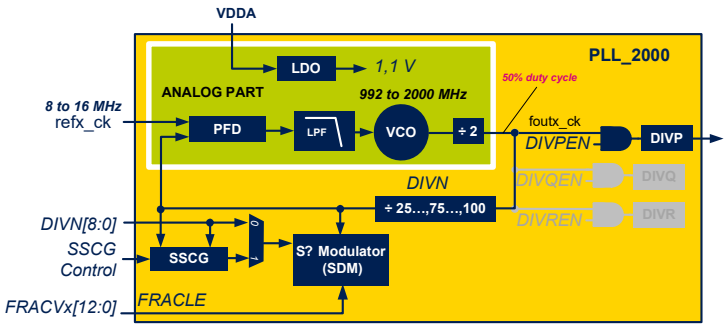
A clock security system monitors the failures of the LSE oscillator.
In case of failure, the application can switch the RTC clock to the LSI clock.
The clock security system is functional in all modes except VBAT mode.
The LSE clock can be used to clock the RTC, the low-power timers, the RNG1 and the DTS.

The LSE features a maximum critical crystal gm between 0.5 uA/V and 2.7 uA/V depending on the driving mode.

A maximum startup time of 2 sec, and a typical dynamic consumption of 290 nA -900 nA depending on the driving mode.

PLL_2000

High frequency, fractional, with spread spectrum possibilities



- 1x PLL_2000 (PLL1):
  - Used for A7 MPU subsystem
  - Input frequency range: 8 to 16 MHz
  - VCO frequency range: 992 to 2000 MHz
  - 13-bit fractional multiplication factor
    - Programmable on-the-fly
  - Integrated LDO
  - Spread Spectrum in order to reduce EMI
  - One used output
    - With post-divider range from 1 to 128

life.augmented

31

The PLLs embedded into the device provide a flexible way to generate the desired frequency for the system or peripheral clocks.
The STM32MP13 microprocessor integrates one PLL2000 dedicated to the Cortex-A7 MPU subsystem..

The input frequency must be between 8 and 16 MHz.
The VCO frequency must be in the frequency range of 992 to 2000 MHz.

The PLLs also provide 3 different outputs which are all derived from the VCO frequency divided by two, via post-dividers (DIVP, DIVQ and DIVR).
Only the DIVP is used for this PLL1.
In addition it is possible to change the values of the post-divider without disabling the PLLs. The application just needs to disable the corresponding post-divider, change the division

ratio, and re-enable the post-divider.

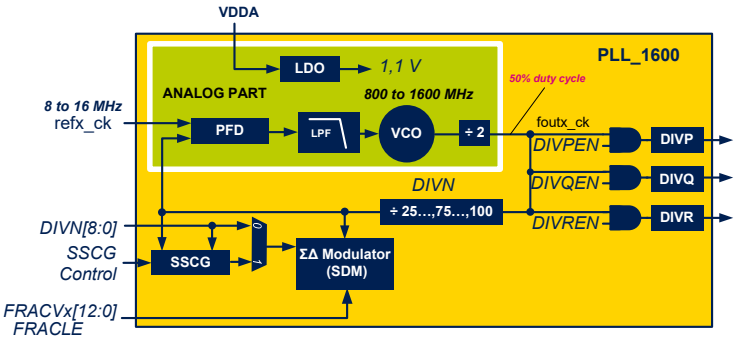
In order to get a duty cycle close to 50%, the application has to program the post-divider to even values.

The PLLs can be switched in fractional mode, allowing a high precision in the VCO frequency.
The 13-bit fractional divider can be changed without disabling the PLLs. This feature can be useful to perform accurate clock drift compensation.

Finally, the PLLs integrate a spread spectrum mechanism in order to reduce the amount of EMI, especially for the DDR interface.

PLL_1600

High frequency, fractional, with spread spectrum possibilities



- 1x PLL_1600 (PLL2):
  - Can be used for AXI subsystem, for the DCMIPP, and for the DDR interface.
  - Input frequency range: 8 to 16 MHz
  - VCO frequency range: 800 to 1600 MHz
  - 13-bit fractional multiplication factor
    - Programmable on-the-fly
  - Integrated LDO
  - Spread Spectrum in order to reduce EMI
  - 3 Outputs per PLL
    - With post-divider range from 1 to 128

life.augmented

32

The STM32MP13 microprocessor integrates one PLL1600. It can be used for the clock generation of the AXI subsystem, for the DCMIPP and for the DDR interface.
The input frequency must be between 8 and 16 MHz.
The VCO frequency must be in the frequency range of 800 to 1600 MHz.

The PLLs also provide 3 different outputs which are all derived from the VCO frequency divided by two, via post-dividers (DIVP, DIVQ and DIVR).
In addition it is possible to change the values of the post-divider without disabling the PLLs. The application just needs to disable the corresponding post-divider, change the division ratio, and re-enable the post-divider.

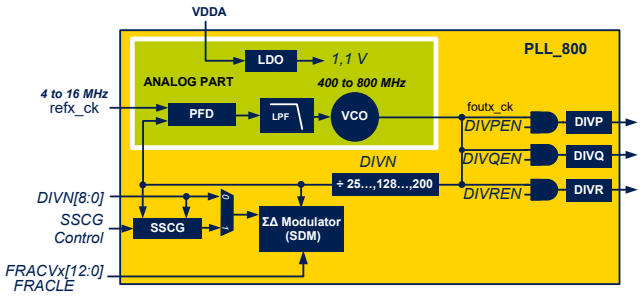
In order to get a duty cycle close to 50%, the application has to program the post-dividers to even values.

The PLLs can be switched in fractional mode, allowing a high precision in the VCO frequency.
The 13-bit fractional divider can be changed without disabling the PLLs. This feature can be useful to perform accurate clock drift compensation.

Finally, the PLLs integrate a spread spectrum mechanism in order to reduce the amount of EMI, more especially for DDR interface.

PLL_800

Medium frequency, fractional, with spread spectrum possibilities



- 2x PLL_800 (PLL3, PLL4):
  - Can be used for the MLAHB subsystem (PLL3) and for the peripherals kernel clock
  - Input frequency range: 4 to 16 MHz
  - VCO frequency range: 400 to 800 MHz
  - 13-bit fractional multiplication factor
    - Programmable on-the-fly
  - Integrated LDO
  - Spread Spectrum in order to reduce EMI
  - 3 Outputs per PLL
    - With post-divider range from 1 to 128

life.augmented

33

The STM32MP13 microprocessor also provides two PLL800s. The architecture is very similar to the previous ones, those PLLs are more dedicated for the generation of the medium-speed clocks.
They can be used for the clock generation of the MLAHB subsystem and for the kernel clock generation of some peripherals.

The input frequency must be between 4 and 16 MHz.
The VCO frequency must be in the frequency range of 400 to 800 MHz.

Other functionalities are similar to the PLL1600.

Integer & Fractional Mode

- In integer mode, the frequency at one of the 3 outputs is given by:

$$F_{pll\_y\_ck} = \frac{F_{ref\_ck} \cdot (DIVN + 1)}{DIVy + 1}$$

$$F_{VCO\_PLL\_800} = F_{ref\_ck} \cdot (DIVN + 1)$$

$$F_{VCO\_PLL\_1600} = 2 \cdot F_{ref\_ck} \cdot (DIVN + 1)$$

- In fractional mode, the frequency at one of the 3 outputs is given by:

$$F_{pll\_y\_ck} = \frac{F_{ref\_ck} \cdot \left((DIVN + 1) + \frac{FRACV}{2^{13}} \right)}{DIVy + 1}$$

y can be P, Q or R
F_{ref_ck} is the frequency provided to the PLL
$F_{pll\_y\_ck}$ is one of the three output clocks generated by the PLL

life.augmented

The slide shows the relation between the input and the output frequency of the PLL, for integer and fractional modes.
The application shall take care to respect the frequency range for the VCO, and for the clock frequency of the signal provided to the PLLs.

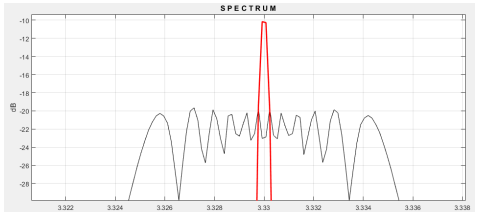
Spread Spectrum Block (SSCG)

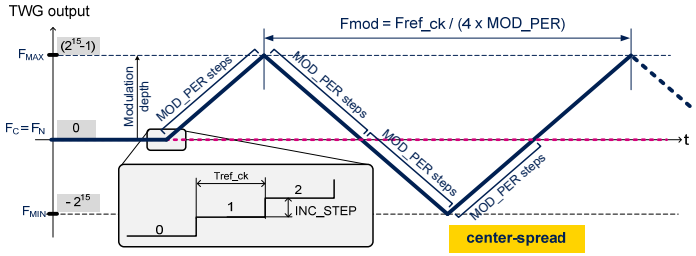
- The spread spectrum block performs a small triangular modulation of the VCO frequency.
- The following parameters can be adjusted
  - The modulation period (MOD_PER)
  - The modulation depth (MOD_INC)
  - The modulation mode (SSCG_MODE)

$$M_D\,(\%) = \frac{MOD\_PER \times INC\_STEP \times 100 \times 5}{(2^{15} - 1) \times (DIVN + 1)}$$

$$F_{mod} = \frac{F_{ck\_ref}}{4 \times MOD\_PER}$$

- The expression of the modulation depth (M_D) and modulation frequency (F_{mod}) are given here.
- An example of EMI reduction (black curve) is given as example





The spread spectrum clock generator (SSCG), performs a triangular frequency modulation of the VCO frequency, using a 16-bit fractional divider.

The application can adjust:

- The modulation period via MOD_PER field
- The modulation depth via MOD_INC field
- The modulation mode via SSCG_MODE field. This option allows the SSCG to generate a modulation either centered on the VCO frequency or below the VCO frequency.

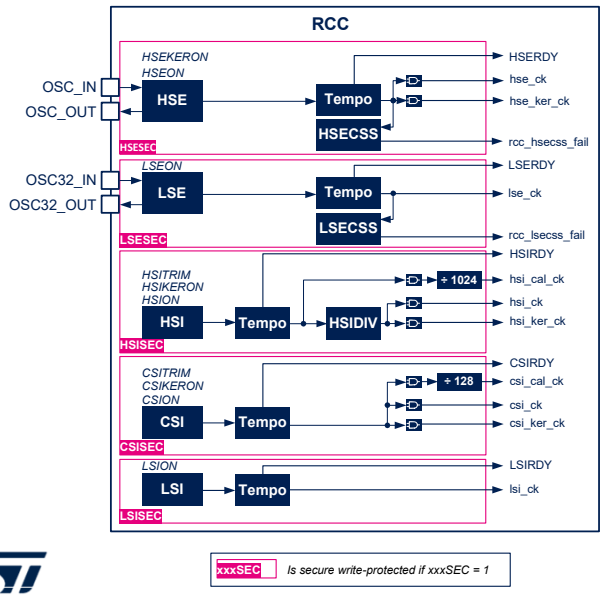
The SSCG is mainly used for the DDR interface as it is an important contributor to EMI interferences.
This is due to the amount of IOs toggling at high frequency with steep edges.

The SSCG does not affect significantly the jitter because the

modulation frequency is very low compared to the speed of DDR interface.
The modulation frequency is generally in the range of 20 to 50 kHz, and the modulation depth is around 1%.

The small plot, shows an example of the effect of the clock spreading: Spreading the frequency over a limited frequency band spreads the power of the signal over this band, reducing the peak of power. The red signal is the VCO output without SSCG, and the black signal is the VCO output with SSCG.

Oscillators Source Clock



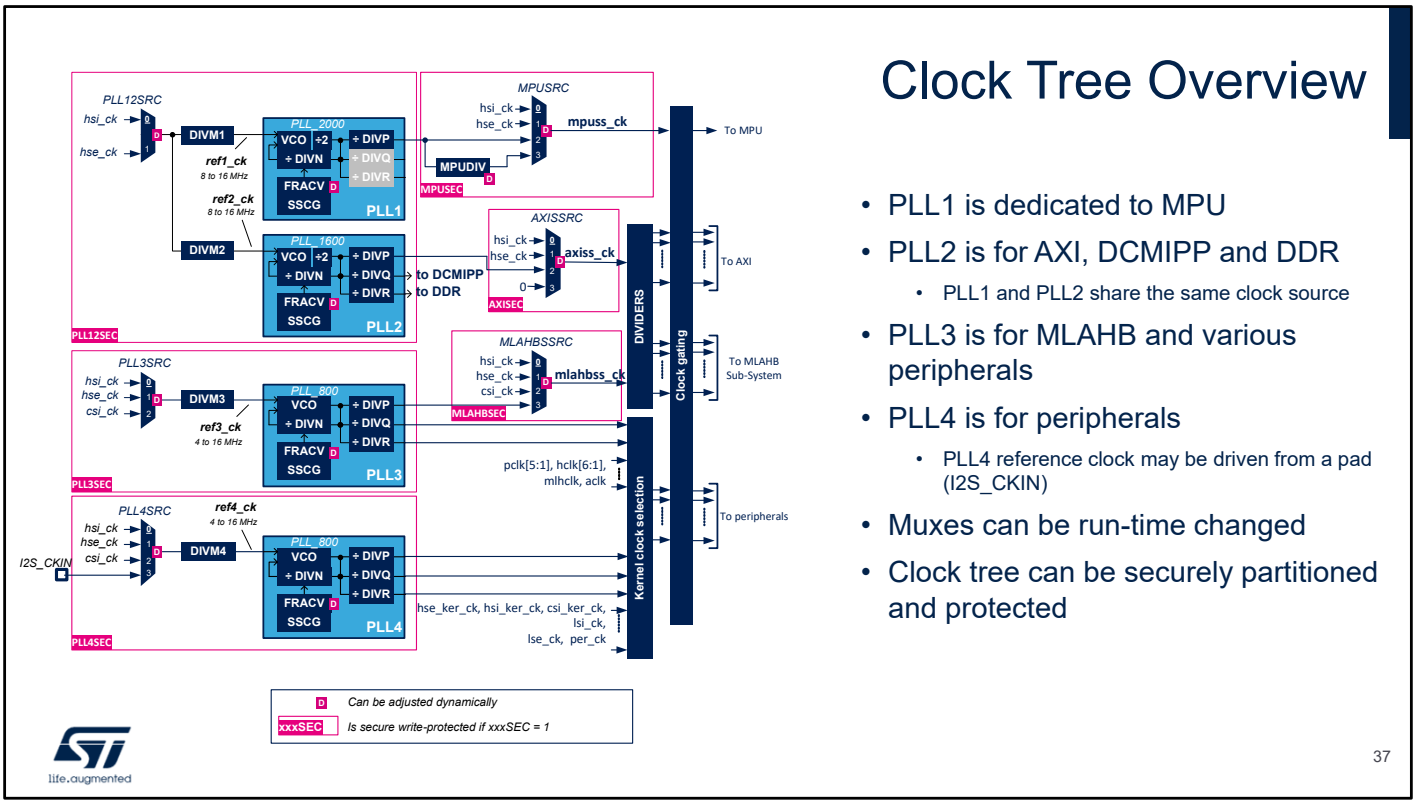
- Each oscillator has a ready logic
- The control of the oscillators can be securely protected.
- HSE and HSI source clocks can be used for kernel clock and a system clock
- HSE, HSI and CSI source clocks are used to generate the sub-system clocks (MPU, AXI, MLAHB) and peripheral kernel clocks.

life.augmented

The oscillators don’t deliver a clock if they are not ready.
The ready flag is based on a temporization.
This temporization is bypassed for the LSE clock, when the LSE is in bypass mode.

The control of each oscillator is securely write protected if the corresponding bit is set in RCC_SECCFGR register.

The HSE, HSI, and CSI oscillators provide a HSE, HIS and CSI source clock which is then used to generate sub-systems clocks and peripherals kernel clocks.

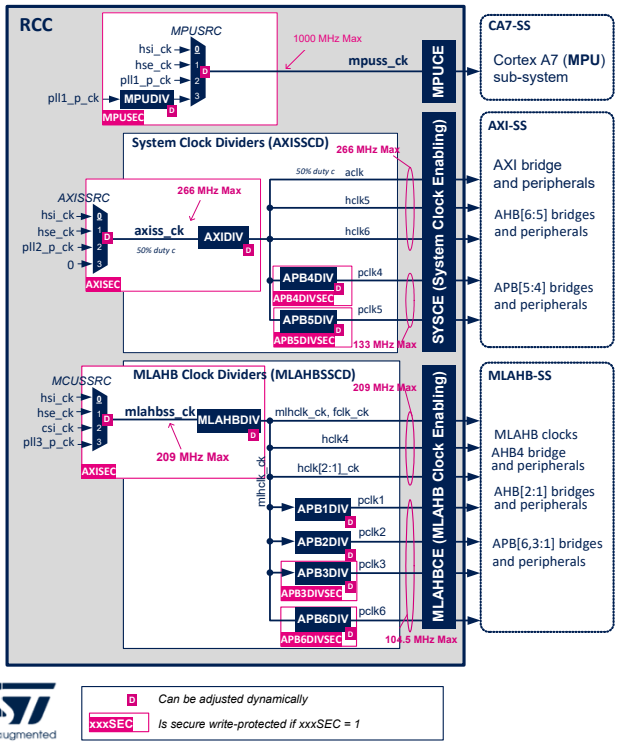


The PLL1 and PLL2 can use either HSE or HSI as reference clock.
The PLL3 can use either HSE or HSI or CSI as reference clock, and the PLL4 can use either HSE or HSI or CSI or I2S_CKIN as reference clock.

The PLL1 is dedicated to the MPU.
The PLL2 is dedicated to the AXI sub-system, to the DCMIPP and to the DDR.
The PLL3 is dedicated to the MLAHB sub-system, and to peripherals kernel clocks, finally the PLL4 is dedicated to peripherals kernel clocks.

Each PLL has a dedicated pre-divider in order to adjust the reference clock for each PLL.
Note that the PLLs clock source must not be changed if one of the PLL is enabled.

In addition, the generation of the MPU, AXI, MLAHB and DDR clocks can be switched to secure mode.



Clock Tree Overview

- Dynamic frequency dividers allow easy frequency adjustment. Can be changed dynamically:
  - MPU clock
  - AXI subsystem clock
  - MLAHB clock
  - APBx clocks
- Clock tree can be securely partitioned and protected

life.augmented

38

There are mainly three sub-systems in the STM32MP1 microprocessor:

- The MPU sub-system,
- The AXI sub-system,
- The MCU sub-system.

The clock provided to the MPU can be changed dynamically via the switch MPUSRC, or by changing MPUDIV.

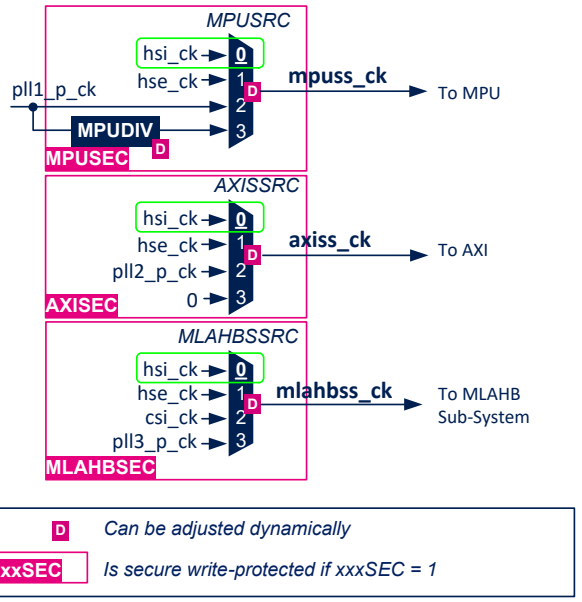
The clock provided to the AXI sub-system, can also be changed dynamically via the switch AXISSRC, or by changing AXIDIV.
In addition the frequency of APB4 and APB5 can be changed dynamically.

The clock provided to the MLAHB sub-system, can also be changed dynamically via the switch MLAHBSSRC, or by

changing MLAHBDIV.
In addition the frequency of APB1,2 and 3 can be changed dynamically.

The MPU can be clocked to up to 1 GHz.
The multi-layer AHB can be clocked up to 209 MHz.
The AXI clock shall not exceed 266 MHz,
Other limitations are shown in the figure.

Selected Clock After Reset



- On a system reset
  - Whatever is the source
    - A V_{DD} power-on reset
    - An exit from Standby
    - Any warm/application reset
  - MPU starts with HSI clock
  - AXI starts with HSI clock
  - MLAHB starts with HSI clock

life.augmented

39

On a system reset

Whatever is the source

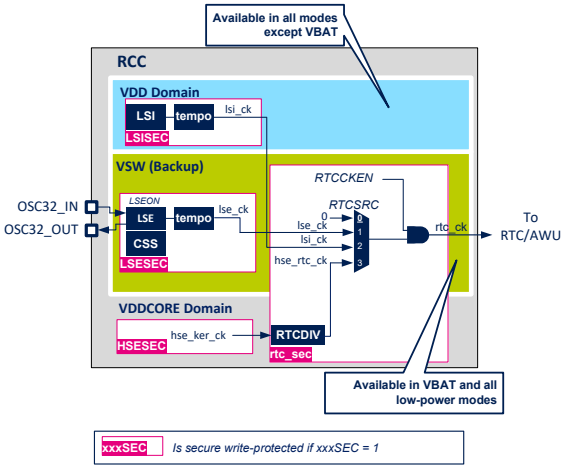
A V_{DD} power-on reset

An exit from Standby

Any warm/application reset

, the MPU, the AXI sub-system and the MLAHB sub-system are clocked with the HSI clock.

RTC Clocking



- Clock source for RTC/AWU can be:
  - LSI, LSE or divided HSE clock
- RTCSRC can be modified only once after a reset of the backup domain, or after a LSE failure.
  - If LSE clock is selected, the RTC works in all modes, including VBAT
  - If LSI clock is selected, the RTC works in all modes, except VBAT
  - If HSE clock is selected, the RTC works in Run mode, and if HSEKERON=1 in any Stop mode
- RTC clock is secure write-protected as defined in the RTC register RTC_SECCFGR
  - If at least one RTC function is secure, RTC clocking is secure.

life.augmented

40

The clock for the RTC/AWU can be selected among: LSE, LSI or divided HSE clock.
Note that when the divided HSE clock is used as kernel clock for the RTC/AWU, the frequency shall not exceed 4 MHz.

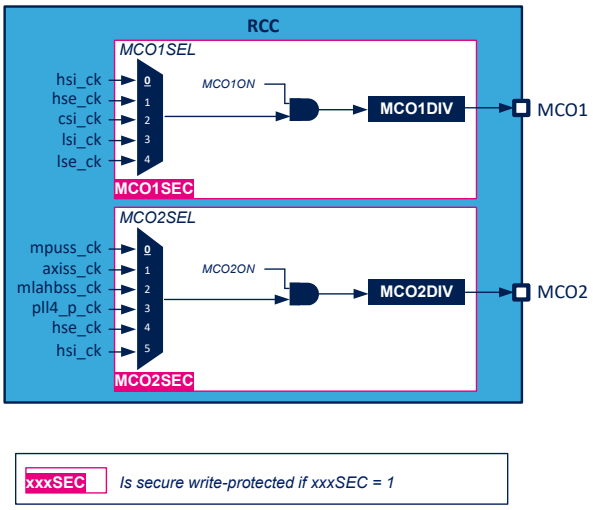
The RTCSRC switch can be modified only once after a reset of the backup domain, or after a LSE failure.

Note as well that if the LSI clock is used as kernel clock for the RTC/AWU, the RTC/AWU will no longer be clocked in VBAT mode.
If the divided HSE clock is used as kernel clock for the RTC/AWU, the RTC/AWU will no longer be clocked in STANDBY or VBAT mode, it can be clocked is STOP mode if HSEKERON is set to ‘1’.

The LSE clock remains enabled in all low-power modes,

including VBAT mode.

MCO Clock Outputs



xxxSEC *Is secure write-protected if xxxSEC = 1*

- Two clock outputs: MCO1 and MCO2 offering a big choice of clock sources.
- The control of the MCO1 and MCO2 clocks can be securely protected.

Application benefits

- MCO1 or MCO2 can be distributed to external devices.

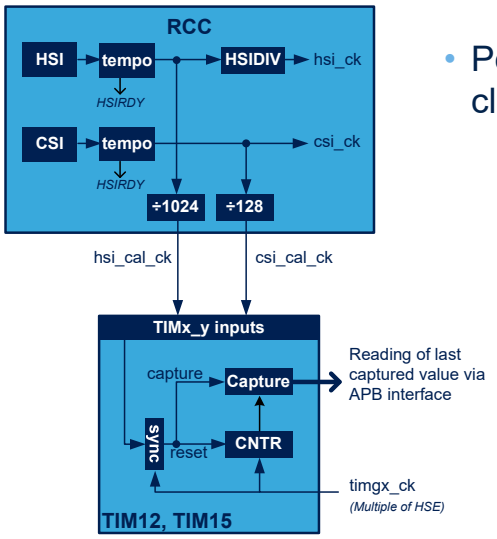
life.augmented

41

The RCC also provides two clock output signals: MCO1 and MCO2.
A prescaler allows the application to adapt the frequency to the PADs capability.

This feature allows the circuit to distribute its internal clocks to external devices.

Clocks Calibration



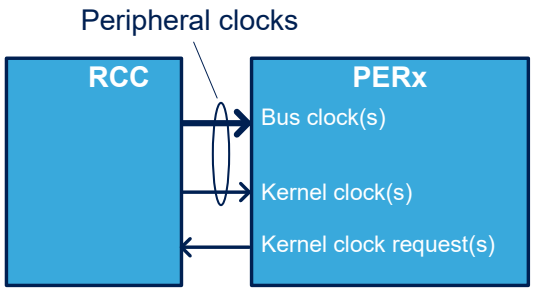
- Possibility to estimate the frequency of CSI and HSI clocks with respect to HSE clock.
  - Accuracy is better than 0.3%
  - The timer is counting with a clock derived from HSE clock.
  - The timer is counting during 1 period of one of the calibration clocks.
  - The capture registers of timers keep the last completed calibration value
  - TIM interrupt services can be used

life.augmented

The RCC offers the possibility to estimate the frequency of CSI and HSI clocks with respect to HSE clock, with an accuracy better than 0.3 %.
Timers 12 and 15 receive the calibration clock signal for that purpose.
The timers count the amount of timer clock periods within a period of the calibration clock signal.
The capture registers of the timers keep the last completed calibration value.
Interrupt services are also available.

Peripheral Clock Distribution

Bus speed can be changed without affecting peripherals



- Peripherals generally receive:
  - One or several bus clocks
  - One or several kernel clocks
- The bus clocks allow registers access and data transfer to memories.
- The kernel clocks are generally used by peripherals to handle external interface: UART, I2S, SPI, USB, Ethernet…

life.augmented

43

Many peripherals in the STM32MP13 have different clocks for the data and control flows (via the processor bus interface), and for the peripheral specific interface.

Generally the clocks for the data and control flows via the processor bus interface, are named ‘Bus clocks’, and the clocks for the peripheral specific interface are named ‘kernel clocks’.
The peripheral clocks represent the clocks received by the peripheral: ‘bus clocks’ and ‘kernel clocks’.

Having a separate bus clock and kernel clock allows the application to change the interconnect and processor working frequency without affecting the peripheral.
For some peripherals, it is also possible to disabled the bus clock as long as the peripheral does not need to transfer data to the system.

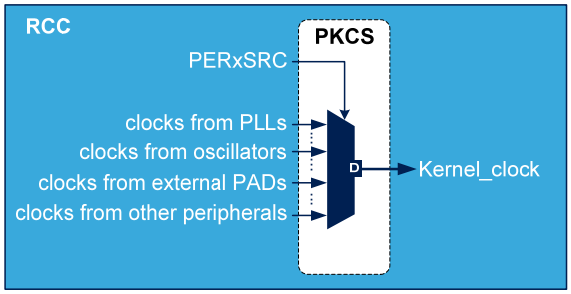
So it gives a good flexibility on the frequency selection for the bus processor and memories, and for the real need of the peripheral interface.

For example: The UARTs have a kernel clock which is used, among other things, by the baud rate generator for the serial interface communication, and an APB clock for the register interface.

In addition some peripherals are able to request the kernel clock when they detected specific events.

Peripheral Clock Distribution

Kernel clock switches are dynamic and glitch-free



- Most of peripherals using a kernel clock have a clock switch to select the optimal kernel clock
  - The clock switch is dynamic and glitch free.
  - Several clock sources are proposed depending on the peripheral needs.
- The kernel clock switch is configured via a RCC register field
  - The switch control inherits from the secure/unsecure state of the peripheral.
  - If peripheral is secure, the switch is securely write-protected.

life.augmented

44

The distribution of the kernel clocks is not detailed in this presentation, please refer to the reference manual for specific details.

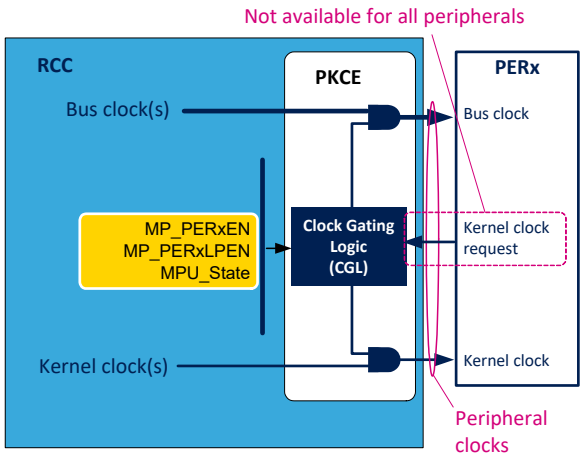
Most of the peripherals using a kernel clock have a dynamic clock switch to select the optimal clock source.
The proposed clock sources generally come:
From one of the PLL outputs,
From internal or external oscillators: This is mandatory for peripherals requiring a kernel clock when the system is in STOP mode,
From PADs: For example on peripherals using an external PHY, but also for audio as well, in order to use a clock reference from an external device,
From other internal peripherals: For synchronization between blocks.

The dynamic switching eases the transition between one source to another.
RCC registers allow the configuration of the kernel clocks for all peripherals.

The kernel clock switch is configured via a RCC register field

- The switch control inherits from the secure/unsecure state of the peripheral.
- If peripheral is secure, the switch is securely write-protected.

Peripheral Clock Gating



- The peripheral clock(s) gating is generally controlled by several parameters:
  - Enable register bit (MP_PERxEN)
  - Low-power enable register bit (MP_PERxLPEN)
  - System operating state (Run and Stop modes) and MPU processor operating states (CRun, CSleep, CStop, and CStandby modes)
- Setting the bit MP_PERxEN to '1' indicates that the peripheral PERx is allocated and enabled for the MPU
- The control of the peripheral clock gating inherits from the secure/unsecure state of the peripheral
  - If peripheral is secure, the RCC enable register bits are securely write-protected.

life.augmented

45

Peripherals generally receive:
One or more bus clocks
One or more kernel clocks

The MPU can control clock gating of peripheral clocks via dedicated registers located in the RCC.

The gating of peripheral clocks depends on several parameters:
-The clock enable bit, named MP_PERxEN
-The low-power clock enable bit, named MP_PERxLPEN
-and the system states (RUN and STOP modes).

Setting the bit MP_PERxEN to '1' indicates that the peripheral PERx is enabled and allocated to the MPU.
This operation is referred to as "allocating" a peripheral.

The control of peripheral clock gating inherits from the secure/non-secure state of the peripheral.

If the peripheral is secure, the RCC enable register bits are securely write-protected.

The next slides gives additional information on clock gating conditions, after presenting the system and MPU operating states.

The following table is a simplified view of the system state and MPU processor state.

The system is in RUN mode when the MPU is in CRUN or CSLEEP mode.
The system is in STOP, LP-Stop or LPLV-Stop mode when the MPU is in CSTOP mode.
The system is in LPLV-STop2 or in Standby mode when the MPU is in Cstandby. The VDDCPU is switched OFF.
In system STANDBY mode, VDDCORE is switched off.

For more details on system states, please refer to the PWR training slides.

Entering a Stop mode

Going to (LP)(LV)(-)Stop(2) mode

The system can go to Stop, LP-Stop, LPLV-Stop or LPLV-Stop2 mode when the MPU is in CStop or respectively CStandby. Then:

1. The RCC waits that DDRC sets the DDR into retention/self-refresh.
2. The RCC stops all the clocks.
3. The RCC informs the PWR that all clocks are gated.

Then PWR can go to the requested Stop mode.

During (LP)(LV)(-) Stop(2) mode

All clocks are disabled except:

1. LSE and/or LSI clock(s) remaining enabled, if enabled
2. HSE, HSI and CSI oscillators may remain active (if their bit xxxKERON is set to '1')
3. HSE, HSI and CSI clocks can be temporary enabled on peripheral requests

life.augmented

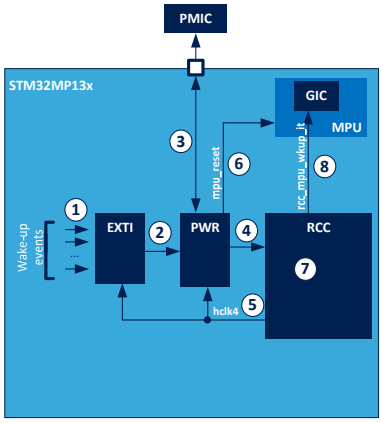
47

When requested to enter any STOP mode, the RCC performs the following operations:
The RCC waits for the DDRC to enter self-refresh mode,
The RCC stops all clocks, in particular, the PLLs are disabled, and the oscillators are disabled unless they are requested to remain active.
Finally, the RCC informs the PWR block that all clocks are gated. The PWR block can then either reduce the VDDCORE voltage, or maintain the current VDDCORE voltage according to the way it is configured.

When the system is in STOP mode, most of the clocks are disabled except:
LSE and LSI can still be running if enabled.
HSE, HSI or CSI oscillators can remain activated in STOP mode if their KERON bit is set to '1'. This feature allows very fast reaction of peripherals in STOP mode, and also a faster

exit from STOP mode.
HSE, HSI and CSI oscillators can also be activated on peripheral requests.

Exiting from a Stop mode (1/2)



- The system exits from a Stop mode via an event generated by a peripheral being able to work in a Stop mode, as follows:
  1. A peripheral generates a wake-up event to the EXTI.
  2. The EXTI informs the PWR.
  3. The PWR requests a valid V_{DDCORE} and a valid V_{DDCPU} to the PMIC external regulators, and waits for them.
  4. The PWR informs the RCC that exit from Stop mode can be performed.
  5. The RCC activates the HSI oscillator (if not already ON) and provides the HSI clock for the MPU, AXI, and MLAHB subsystems and for the hclk4 clock to the EXTI and PWR.
  6. If exiting from LPLV-Stop2 mode, the PWR generates a MPU reset and BOOTROM code is executed
  7. The RCC restores the oscillator(s), the source clock(s), the PLL1&2, the clocks to the MPU, AXI and MLAHB subsystems, all as they were configured before going to Stop mode.
  8. If not exiting from LPLV-Stop2 mode, the RCC generates a wake-up interrupt to the MPU.

life.augmented

48

The system exits from STOP modes thanks to events generated by peripherals able to work in a STOP mode.
Before entering a STOP mode, the application programs the EXTI to define which event will exit the system from the STOP mode.
When a peripheral generates the expected wake-up event (step 1 in the drawing):

The EXTI informs the PWR block that a wake-up event requests the exit from STOP mode (step 2 in drawing).
Then Step 3: The PWR block sends a request for a valid VDDCORE and a valid VDDCPU to the PMIC external regulators, and waits for them to be established.
Step 4: The PWR informs the RCC that exit from STOP mode can be performed.
Step 5: The RCC enables the HSI oscillator (if not already ON) and provides the HSI clock for the MPU, AXI and MLAHB

subsystems, and the hclk4 clock to the EXTI and PWR.
Conditional Step 6: If exit is from LPLV-Stop 2 mode, the PWR generates a MPU reset and BOOTROM code is executed.
Step 7: The RCC restores the oscillator(s), source clocks, PLLs1&2, and the clocks to the MPU, AXI and MLAHB subsystems as they were configured before going into Stop mode.
And conditional Step 8): The RCC generates the corresponding wake-up interrupt, if not exiting from LPLV-Stop2 mode.

Exiting a Stop mode (2/2)

- On wakeup from any system Stop mode, the automatic clock restore sequence is performed before giving back control to the application.
- Compared to Stop mode, a wakeup from a LP-Stop mode can be added with a programmable delay (PWRLP_TEMPO, based on HSI clock), in order to
  - Enable the external regulator to switch from a low-power mode to the main mode
  - Wait before that RCC enables/restores PLL1 & PLL2

life.augmented

On wakeup from any system Stop mode, the automatic clock restore sequence is performed before control is returned to the application.
Compared to Stop mode, a wakeup from a LP-Stop mode can be added with a programmable delay (PWRLP_TEMPO, based on HSI clock), in order to

Enable the external regulator to switch from a low-power mode to the main mode

Wait before that RCC enables/restores PLL1 & PLL2.

Kernel Clock Requests

- Some peripherals such as the UARTs and I2Cs are able to detect asynchronous events requesting a kernel clock for the processing.
- The RCC can provide on demand a kernel clock to those peripherals, when the system is in a Stop mode.
- The kernel clock request will only work if the selected kernel clock is directly driven from an oscillator: HSE, HSI or CSI. Not from a PLL.
- In order to speed-up the activation time, the HSE, HSI or CSI can be maintained activated during a Stop mode.
- In system Stop mode, the LSI and LSE clocks are available for peripherals

life.augmented

50

Some peripherals such as the UARTs and I2Cs are able to asynchronously detect events requesting a kernel clock for processing.
The RCC can provide a kernel clock to thse peripherals on demand, when the system is in a STOP mode.
The kernel clock request only works if the selected kernel clock source is an oscillator: HSE, HSI or CSI. Not if driven by a PLL.
In order to speed up the activation time when a peripheral requests a kernel clock, the HSE, HSI or CSI oscillators can be maintained activated during the system STOP mode.
In system STOP mode, if enabled, the LSI and LSE clocks are always available for peripherals.

Peripheral Clock gating

- A **kernel clock** is provided to the peripherals PERx if one of the following conditions is met:
  1. When the peripheral is allocated (i.e. if PERxEN='1') **and** the MPU processor is in CRun mode.
  2. When the peripheral is allocated (i.e. if PERxEN='1') **and** the MPU processor is in CSleep mode **and** PERxLPEN = '1'.
  3. When the peripheral is allocated (i.e. if PERxEN='1') **and** the system is in a Stop mode (MPU processor is in CStop/CStandby mode) **and** PERxLPEN = '1 **and** the peripheral is generating a kernel clock request **and** the kernel clock source is HSE, HSI, CSI, LSE or LSI.
  4. When the peripheral is allocated (i.e. if PERxEN='1') **and** the system is in a Stop mode (MPU processor is in CStop/CStandby mode) **and** PERxLPEN = '1', **and** the kernel clock source of the peripheral is LSE or LSI.
- The **bus interface** clock is provided to the peripherals only when condition 1 or condition 2 is met.

life.augmented

51

A kernel clock is provided to the peripherals if one of the following conditions is met:

1. When the peripheral is allocated (i.e. if PERxEN='1') **and** the MPU processor is in CRun mode.
2. When the peripheral is allocated (i.e. if PERxEN='1') **and** the MPU processor is in CSleep mode **and** PERxLPEN = '1'.
3. When the peripheral is allocated (i.e. if PERxEN='1') **and** the system is in a Stop mode (MPU processor is in CStop/CStandby mode) **and** PERxLPEN = '1 **and** the peripheral is generating a kernel clock request **and** the kernel clock source is HSE, HSI, CSI, LSE or LSI.
4. When the peripheral is allocated (i.e. if PERxEN='1') **and** the system is in a Stop mode (MPU processor is in CStop/CStandby mode) **and** PERxLPEN = '1', **and** the kernel clock source of the peripheral is LSE or

LSI.

The bus interface clock is provided to the peripherals only when condition 1 or 2 is met.

DDRC Self-Refresh mode control (1/2)

- Software Self-Refresh (SSR):
  - The application has to program directly the DDRC in order to manage each transition of the external DDR device from and out of the self-refresh.
- Automatic Self-Refresh (ASR1 and ASR2):
  - The application does not need to program the DDRC every time the DDRC must be set into self-refresh.
  - The DDRC goes automatically into self-refresh mode when the DDRC is inactive **for an amount of time defined into the DDRC**.
  - A new transaction on the DDRC AXI port exits the DDR from self refresh after some extra microseconds.
  - In ASR1, the RCC dynamically controls the gating of the DDRC independently of the processor state.
  - In ASR2, the RCC dynamically controls the gating of the DDRC and DDRPHYC blocks independently of the processor state.

life.augmented

52

The RCC integrates three main modes in order to control the low-power transitions of the DDR controller (DDRC) and DDR PHY.
Software self-refresh mode (SSR)
Automatic self-refresh mode (ASR)
And Hardware self-refresh mode (HSR)

In Software self-refresh mode, the application has to program the DDRC directly in order to manage each transition of the external DDR device from and out of the self-refresh.
The RCC simply performs the gating of the DDRC and DDRPHYC clocks, as performed for other peripherals.

In Automatic self-refresh mode, the software does not need to program the DDRC every time the DDRC must
be set into self-refresh. The DDRC goes automatically into self-refresh mode when the DDRC is inactive for a length of

time defined in the DDRC.
That is to say if the DDRC does not receive any transactions for a length of time configured by the application.
When the DDRC receives a new transaction, the DDRC requests the DDR to exit from self-refresh mode. The counterpart is a few microseconds more to exit from the self-refresh mode.
In this mode, the RCC has two levels of clock gating:
The RCC dynamically controls gating of the DDRC clock, according to the DDR state, and independently of the processor state.
The RCC dynamically controls gating of the DDRC and DDRPHYC clocks, according to the DDR state, and independently of the processor state. This mode can only be used when the DLLs of the DDRPHYC are bypassed. For example, it can be used for an LPDDR2 working in the range of 125 MHz.

DDRC Self-Refresh mode control (2/2)

- Hardware Self-Refresh (HSRx):
  - The DDRC automatically goes into self-refresh mode in the following cases:
    - When the interface is disabled via DDRCxEN bits,
    - When the MPU goes from CRun to CSleep mode,
    - When the MPU goes from CRun to CStop mode.
  - The DDRC automatically exits from self-refresh mode in the following cases:
    - When the interface is enabled via DDRCxEN bits,
    - When the MPU exits from CSleep or CStop mode,
    - When the DDRC requests to exit from self-refresh. This could happen if another master is accessing the DDRC AXI port.
  - This mode can be used to set automatically the DDR in self-refresh when the MPU is in low-power mode, and other masters such as DMA can exit the DDR from self-refresh mode when needed.
  - Via the DDRC AXI low-power interface
    - In HSR1, the RCC dynamically controls the gating of the DDRC.
    - In HSR2, the RCC also controls the gating of the DDRPHYC.

life.augmented

53

The Hardware Self-Refresh mode uses the AXI low-power interface to control DDRC and DDRPHYC clock gating.

In Hardware Self-Refresh mode, the DDR automatically goes into self-refresh mode:
When the interface is disabled via DDRCxEN bits,
When the MPU goes from CRUN to CSLEEP mode,
When the MPU goes from CRUN to CSTOP mode.

The DDR automatically exits from self-refresh mode:
When the interface is enabled via DDRCxEN bits,
When the MPU exits from CSleep or Cstop mode,
When the DDRC requests to exit from self-refresh mode. This could happen if another master is accessing the DDRC AXI port.

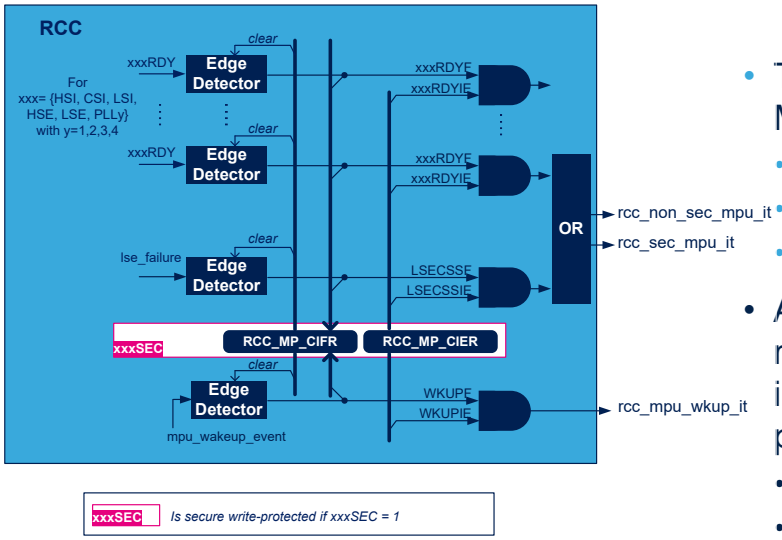
This mode tends to set the DDR in self-refresh mode when the

MPU is in low-power mode, but other masters such as DMA can exit the DDR from self-refresh mode if needed.

In this mode, the RCC also has two levels of clock gating:
The RCC dynamically controls gating of the DDRC clock,
The RCC dynamically controls gating of the DDRC and DDRPHYC clocks, This mode can only be used when the DLLs of the DDRPHYC are bypassed. For example, it can be used for an LPDDR2 working in the range of 125 MHz.

Interrupts

Interrupts



- Three RCC interrupt lines to the MPU:
  - A general non-secure interrupt
  - A general secure interrupt
  - A dedicated wakeup interrupt
- As defined by RCC_SECCFGR register, any interrupt event can be independently and securely write-protected, for control of the:
  - Interrupt flag clear (RCC_MP_CIFR)
  - Interrupt enable (RCC_MP_CIER)

life.augmented

55

The RCC provides three interrupt lines to the MPU:
 A general secure interrupt line (**rcc_sec_mpu_it**), gathering any secure interrupt event
 A general non-secure interrupt line (**rcc_nonsec_mpu_it**), gathering any non-secure
interrupt event
 and A dedicated interrupt line for wakeup of the MPU

Any interrupt event is enabled or disabled independently via a separate interrupt enable control bit in the *RCC_MP_CIER register*.
Any interrupt flag can be checked and cleared via the *RCC_MP_CIFR register*. Any interrupt flag can be read by a secure or non-secure software agent.
Any interrupt flag can be separately and securelywrite-protected at a field level.
Secure write-protection of the interrupt flag clear and the

interrupt enable is defined by the corresponding RCC_SECCCFGR register bit.

Interrupts

Interrupt line	Interrupt event	Description
rcc_sec_mpu_it, rcc_non_sec_mpu_it	LSI ready	Ready LSI clock and oscillator
	LSE ready	Ready LSE clock and oscillator
	HSI ready	Ready HSI clock and oscillator
	HSE ready	Ready HSE clock and oscillator
	CSI ready	Ready CSI clock and oscillator
	PLL2 ready	Ready and locked PLL1 clock
	PLL2 ready	Ready and locked PLL2 clock
	PLL3 ready	Ready and locked PLL3 clock
	PLL4 ready	Ready and locked PLL4 clock
	LSE clock failure	Detected LSE clock failure by LSE clock security system
rcc_mpu_wkup_it	MPU wakeup	Set when the MPU needs to be wakeup

life.augmented

There are mainly 3 kinds of event:

- Oscillators and PLLs ready events
- LSE clock failure event
- And Wakeup event

Related peripherals

- Refer to these trainings linked to this peripheral, if any
  - STM32MP13x Power control (PWR)
  - STM32MP13x Asynchronous Interrupts and Event Controller (AIEC)

life.augmented

In addition to this training, you may find the Power Control and Interrupt Controller trainings useful.
Thanks a lot for your attention !

Thank you

life.augmented